

R. Ciocârîie

P. Éleş

J. Bérka
(RPU)

S. Kerezzély
(RPU)

M. Sirbu

A. Davidoviciu

Z. Benyó
(RPU)

L. Orăşanu

Fl. Filip

M. Murateea

A. Varga

M. Dumitru

A. Moangă

A. Donciulescu

C. Vasiliu

A. Alexandru

L. Iacob

P. Rădulescu

O. Gheorghiu

T. Călin

I. Dima

V. Vişan

V. Stănescu

T. Geber

I. Dobre

C. Busvioc

D. Horhoianu

M. Nişă

G. Olaru

M. Stănculescu

N. Patrubany

N. Pop

A. Kiss

N. Socaciu

D. Szász

V. Sima

Th. Popescu

Th. Popescu

V. Sima

A. Varga

N. Ivan

PASCAL ŞI PASCAL CONCURRENT (Ciclul „Limbaie de programare“)

„O PUNTE ÎNTRE ŞTIINŢA ŞI TEHNOLOGIE“

(Congresul Mondial al Federaţiei Inter-
naţionale de Automatizare IFAC '84)

- Conducerea cu microcalculatoare
- Sisteme mari • PAC • Educaţie
- Sisteme expert şi om-maşină
- Decizii ierarhice şi multicriteriale
- Transporturi • Aplicaţii spaţiale
- Hidrometeorologie, calitatea aeru-
lui, gospodărirea apelor

TESTĂRI AUTOMATE PENTRU MINICALCULATOARELE CORAL ŞI INDEPENDENT

(Ciclul „Service pentru calculatoare“)

MICROCALCULATOARELE PERSONALE PRAE ŞI LIMBAJUL LOR BASIC

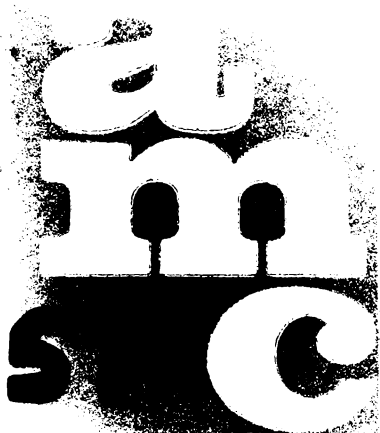
(Ciclul „Calculatoare personale“)

PORTABILITATEA PROGRAMELOR MODELAREA SERILOR DE TIMP

(Ciclul „Algoritmi şi pachete de
programe“)

PROIECTAREA ASISTATĂ DE CALCULATOR (PAC)

- Estimarea recursivă
- Pachete de programe PAC





Vol. 51

AUTOMATICĂ MANAGEMENT CALCULATOARE

- **Ciclul special
„Congresul Mondial IFAC '84”**
- **Limbajele PASCAL și PASCAL
CONCURRENT**
 - **Testări automate pentru
minicalculatoare**
 - **Microcalculatoarele personale
PRAE și limbajul lor BASIC**
- **Algoritmi și pachete de programe**
- **Proiectarea asistată de calculator**



EDITURA TEHNICĂ

BUCUREȘTI - 1985

**a
m
c**

Vol. 51

Traduceri, adaptări : ARISTIDE PREDOI, ADRIAN DAVIDOVICIU, MARCEL SÎRBU, DAN DOBRESU, SANDU LAZĂR, ALEXANDRA TATU, MATEI CONSTANTIN, VASILE SIMA, RADU MAGDA, DOINA TUDOR, NICOLAE TUDORANDESCU, ILIE PAIUȘI, CAMELIA BASARAB, DAN RĂDULESCU, VIRGIL NICULA, ȘERBAN SPOREA, RADU MANOLESCU, NICOLAE ALGIU, LEONARD STOIAN ȞOROBET, GABRIEL SPIRIDON, EUST. STANCIU, ȘTEFAN LUPESCU, OLEG CRĂMĂRĂSCU, AURELIAN SIMIONESCU.

Recenzii : SANDU LAZĂR, ADRIAN DAVIDOVICIU, LIVIU DUMITRAȘCU, GORUN MANOLESCU, TOMA GEBER, ADRIAN PETRESCU, HORIA CIOCĂRLIE, PETRU ELEȘ, PAUL CONSTANTINESCU.

Colectivul de traducere, adaptare, recenzare menționat, a lucrat la toate volumele AMC 48—51.

Redactori : ing. **PAUL ZAMFIRESCU**
ing. **MIRCEA GROSU**

Tehnoredactor : OLIMPIADA NISTOR
Coperta : SIMONA DUMITRESCU

Bun de tipar : 16. sept. 1985. Coli tipar 22,5

Tiparul executat la
Intreprinderea Poligrafică „Banatul”
Timișoara



CUPRINS

Ciclul
„LIMBAJE DE
PROGRAMARE“

<i>H. Ciocărlie, P. Eleş</i>	Limbajele PASCAL și PASCAL CONCURENT.	
	Sinteză	5—16 354—359

Ciclul
„O PUNTE
INTRE
ȘTIINȚA ȘI
TEHNOLOGIE“
— Congresul
IFAC —
Budapesta '84

<i>J. Börka, S. Keresztély</i>	Sistem de conducere cu microcalculatoare în centrale nucleare	17—30
<i>M. Sirbu, A. Davidoviciu, Z. Benyó</i>	Educația în automatizări (R. P. Chineză, Austria, S.U.A., Japonia, Europa de Vest, R. P. Ungară)	31—42
<i>L. Orășanu, Fl. Filip, M. Muratcea</i>	Sisteme complexe și mari	43—66
<i>A. Varga, M. Dumitru, A. Moangă</i>	PAC pentru sisteme automate	67—81
<i>A. Donciulescu</i>	Ingineria sistemelor industriale	82—87
<i>C. Vasiliu</i>	Sisteme expert	88—94
<i>A. Alexandru, L. Iacob, P. Rădulescu</i>	Conducerea de către operator uman a vehiculelor și telemanipulatoarelor	95—111
<i>O. Gheorghiu, T. Călin</i>	Sisteme decizionale ierarhice și metode de decizie multicriteriale	112—125
<i>M. Sirbu</i>	Optimizări în transporturi	126—132
<i>I. Dima, V. Vișan, V. Stănescu</i>	Hidrologia, protecția calității aerului și a apei, gospodărirea resurselor de apă	132—150
<i>M. Sirbu</i>	Aplicații spațiale	151—158



Vol. 51

Ciclul
„SERVICE
PENTRU
CALCULATOARE“

T. Geber, I. Dobre, Testări automate pentru
C. Busuioc, minicalculatoarele CORAL
D. Horhoianu, M. Niță, și INDEPENDENT 159—184
G. Olaru,
M. Stănciulescu

Ciclul
„CALCULATOARE
PERSONALE“

N. Patrubany, N. Pop, Familia de calculatoare
A. Kiss, N. Socaciu, personale românești PRAE
D. Szász și limbajul său BASIC ... 185—220

Ciclul
„ALGORITMI
ȘI PACHETE“
DE PROGRAME“

V. Sima Portabilitatea programelor
de calcule tehnico-științifice 221—240

Th. Popescu Pachet de programe pentru
modelarea seriilor de timp 241—263

Ciclul
„PROIECTARE
ASISTATĂ
DE CALCUL-
LATOR (PAC)“

V. Sima Tehnici de factorizare în al-
goritmi de conducere adap-
tivă 265—289

● Sisteme automate
● Construcții
de mașini

Th. Popescu Metode de estimare recur-
sivă a parametrilor sisteme-
lor 290—319

A. Varga BIMASC — Pachet de sub-
programe FORTRAN pentru
proiectarea și simularea
sistemelor automate ... 320—338 290—

N. Ivan PAC pentru procese tehnolo-
gice în construcții de ma-
șini. Sistemul PAPT-1. ... 339—353 339—

În paginile 264 și 360 sînt cuprinse informații asupra unor mani-
festări din țară și străinătate, ca și recomandări de cărți.

Ciclul „Limbaje de programare“

LIMBAJELE PASCAL ȘI PASCAL CONCURENT. SINTEZĂ

Horia Ciocârlie, Petru Eles
Institutul Politehnic Timișoara

Conceptul de programare structurată a constituit punctul de plecare al dezvoltării programării calculatoarelor pe fundamente științifice. Pe baza lui, N. Wirth a definit în anul 1971 limbajul PASCAL [1]. Pornind de la PASCAL P.B. Hansen a reușit să introducă într-un limbaj evoluat conceptele programării concurente, definind în anul 1975 PASCALUL CONCURENT [2].

În lucrare se face o prezentare a limbajelor de programare PASCAL și PASCAL CONCURENT, din punctul de vedere al conceptelor fundamentale care au stat la baza definiției lor, și anume:

- structurarea și modularizarea acțiunilor și datelor unui program;
- programarea concurentă în limbaje de nivel înalt.

Maniera de prezentare se bazează pe întrepătrunderea unor noțiuni mai generale de programare, cu aspecte practice specifice realizării lor în cele două limbaje.

1. Structurare și concurență în programare

Limbajul PASCAL este un rezultat al evoluției conceptelor apărute ca urmare a crizei ce caracteriza domeniul programării calculatoarelor la sfârșitul anilor '60. În această perioadă răspîndirea pe plan mondial a prelucrării datelor cu calculatorul a cunoscut un salt deosebit. S-au putut astfel aborda și rezolva probleme din ce în ce mai complexe. Programele mari corespunzătoare acestor probleme s-au complicat în așa măsură încît au devenit o „junglă“ de nepătruns, chiar și pentru autorii lor. Înțelegerea, depanarea și modificarea unor astfel de programe prezenta adesea dificultăți deosebite. Limbajele de programare existente [3] și absența unor principii clare care să impună o „disciplină a programării“, au favorizat în mare măsură această stare a lucrurilor. A apărut în mod natural necesitatea elaborării unei metodologii generale care să permită scrierea în mod sistematic a unor programe eficiente în realizare, exploatare și întreținere.

Ca urmare a acestei necesități s-a cristalizat *metoda proiectării și programării structurate* [4, 5]. Un program structurat este construit din unități funcționale

bine conturate, ierarhizate conform naturii intrinseci a problemei. În interiorul unor astfel de unități structurarea se manifestă atât la nivelul acțiunilor (instrucțiunilor), cât și al datelor prelucrate. Rezultă programe clare, ordonate, inteligibile, fără salturi și reveniri.

Programarea structurată este o metodă independentă de limbajul de programare, ea acționând la nivelul stilului de lucru. Totuși, practica a demonstrat că limbajul poate înlesni în mod hotărâtor realizarea dezideratelor enunțate mai sus. Aceasta este valabil prin excelență pentru limbajul PASCAL, proiectat în mod sistematic, conform conceptelor științifice ale programării structurate. În ideea simplificării și ordonării activității de programare, PASCAL facilitează și chiar impune scrierea unor programe clare, ușor de întreținut, depanat și modificat [6]. Datorită acestor calități, limbajul s-a răspândit rapid în toate domeniile în care se folosește calculatorul, fiind astăzi unul dintre cele mai cunoscute și utilizate limbaje pe plan mondial.

În domeniul special în care practica a demonstrat necesitatea introducerii unor facilități specifice pentru ca un limbaj de programare să devină util, PASCAL a constituit un punct de plecare preferat pentru definirea unor noi limbaje.

Astfel, în domeniul programării de sistem, PASCAL nu a permis abordarea unor categorii de probleme specifice: multiprogramarea, sincronizarea unor procese care folosesc resurse comune, protecția resurselor și datelor nepartajabile etc. A rămas deschisă problema găsirii unei metodologii cu ajutorul căreia să se scrie în mod eficient programe concurente inteligibile, modificabile și fiabile. Prin definirea PASCALULUI CONCURENT s-a demonstrat că un limbaj de nivel înalt adecvat poate să constituie unealta corespunzătoare pentru scrierea unor astfel de programe. PASCAL CONCURENT păstrează spiritul și forma limbajului PASCAL [6]. Pe această bază s-au introdus însă noi concepte menite să rezolve probleme specifice programelor de tip sistem și care au dat limbajului un caracter cu totul deosebit. Domeniul exclusiv al programării de sistem a devenit astfel accesibil unui cerc tot mai larg de programatori, permițând totodată creșterea spectaculoasă a productivității elaborării programelor concurente.

Problemele prezentate mai sus au constituit motivația principală datorită căreia un colectiv de specialitate de la Institutul Politehnic „Traian Vuia” Timișoara s-a preocupat în mod constant în ultimii ani de diferite aspecte teoretice și practice vizând limbajele PASCAL și PASCAL CONCURENT. S-au realizat astfel, compilatoare PASCAL și PASCAL CONCURENT pentru calculatoarele Felix C (256, 512, 1024), ca și implementări ale celor două limbaje pe calculatoarele din familiile Independent și Felix M-18 [7, 8, 9, 10].

2. Structuri de acțiuni și de date

2.1. Instrucțiuni

Gruparea acțiunilor unui program în structuri [11] conferă acestuia suplețe, claritate și flexibilitate. Limbajul PASCAL permite descrierea acțiunilor ca niște *instrucțiuni simple*, asignarea (atribuirea) și apelul de procedură, sau sub

forma unor structuri de acțiuni cu o intrare și o ieșire numite *instrucțiuni structurale*. Ele provoacă executarea secvențială, repetată sau condiționată, a uneia sau mai multor instrucțiuni și se împart în mod corespunzător în trei categorii :

a) **Instrucțiunea compusă**, de forma :

begin instrucțiune 1 ; instrucțiune 2 ; ... instrucțiune n *end*

Efectul ei este executarea instrucțiunilor componente (instrucțiune 1 ... instrucțiune n) în ordinea strictă în care au fost scrise.

b) **Instrucțiuni repetitive**

— Instrucțiunea *while*, de forma :

while condiție *do* instrucțiune

Se realizează executarea repetată a instrucțiunii care urmează cuvintului cheie *do*, atâta timp cât condiția este adevărată.

— Instrucțiunea *repeat*, de forma :

repeat instrucțiune 1 ; instrucțiune 2 ; ... instrucțiune n *until* condiție

Instrucțiunea provoacă repetarea executării listei de instrucțiuni dintre cuvintele cheie *repeat* și *until*, pînă cînd condiția devine adevărată.

— Instrucțiunea *for* cu următoarele două forme posibile :

for contor : = expresie1 *to* expresie2 *do* instrucțiune

for contor : = expresie1 *downto* expresie2 *do* instrucțiune

Instrucțiunea *for* provoacă executarea instrucțiunii care urmează cuvintului cheie *do*, în mod repetat pentru fiecare valoare a contorului. Acesta primește inițial valoarea obținută prin calcularea expresiei 1 și se incrementează (forma cu *to*) respectiv se decrementează (forma cu *downto*) cu pasul 1, la fiecare iterație pînă la valoarea expresiei 2.

c) **Instrucțiuni condiționale**

— Instrucțiunea *if* cu următoarele două forme posibile :

if condiție *then* instrucțiune 1 *else* instrucțiune 2

if condiție *then* instrucțiune

În prima formă, în funcție de condiție se selectează o instrucțiune din cele două posibile. Forma fără *else* provoacă executarea condiționată a instrucțiunii care urmează cuvintului cheie *then*.

— Instrucțiunea *case*, de forma :

case expresie *of*

constantă 1 : instrucțiune 1 ;

constantă 2, constantă 3 : instrucțiune 2 ;

.....

constantă n : instrucțiune m

end

Instrucțiunea are ca efect evaluarea expresiei care urmează cuvintului cheie *case* și executarea doar a instrucțiunii a cărei etichetă coincide cu valoarea obținută.

2.2 Date

2.2.1. Noțiunea de tip

Instrucțiunile unui program acționează asupra datelor sale conform algoritmului corespunzător problemei ce urmează a fi rezolvată. În ceea ce privește modalitățile de limbaj create pentru descrierea datelor, PASCAL a prezentat un salt cu totul excepțional, care a influențat în mod decisiv evoluția ulterioară a limbajelor de programare.

La baza descrierii datelor în PASCAL stă noțiunea de *tip*. Tipul unei mărimi (constantă, variabilă, valoarea unei expresii etc.) determină mulțimea valorilor pe care aceasta le poate lua, precum și setul de operații ce i se pot aplica.

Descrierea datelor în PASCAL presupune definirea de către programator a unor tipuri de forma dorită. Limbajul permite descrierea în primă instanță a unor tipuri simple (nestructurate). Acestea constituie cărămizile de bază, pornind de la care se descriu structuri formate din componente de diferite tipuri, legate într-o construcție de tip nou. Mijloacele de structurare ale limbajului (tabloul, articolul, mulțimea etc.), permit astfel descrierea unor tipuri de date de orice complexitate [12].

Tipul, după cum s-a arătat, determină o mulțime de *valori posibile*. Entitățile care pe parcursul executării programului pot lua diferite valori de un anumit tip se numesc *variabile*. O variabilă se declară, înainte de a fi utilizată, ocazie cu care i se asociază un identificator și i se specifică tipul, ca în exemplul de mai jos :

var identificator : tip

2.2.2. Tipuri simple

În limbajul PASCAL există patru tipuri simple standard care pot fi utilizate în program fără a fi definite. De asemenea, programatorul poate defini tipuri simple proprii. O proprietate comună a tipurilor simple este ordonarea mulțimii valorilor corespunzătoarelor (ele se mai numesc și tipuri scalare).

2.2.2.1. *Tipurile standard*. a) Tipul `boolean` este format din mulțimea valorilor logice false și true între care există relația de ordine `false < true`.

b) Tipul `integer` este o submulțime a mulțimii numerelor întregi, care depinde de implementare.

c) Tipul `real` este format dintr-o submulțime a numerelor raționale, dependentă de implementare, reprezentând aproximații ale numerelor reale.

d) Tipul `char` este format din mulțimea caracterelor tipăritabile (deci este și ea dependentă de implementare). Constantele de tip caracter se reprezintă prin caracterul respectiv cuprins între apostrofuli: 'A', 'x'.

2.2.2.2. *Tipul enumerare*. Mulțimea valorilor unui tip enumerare se definește prin enumerarea identificatorilor ce reprezintă aceste valori, ca în exemplele de mai jos :

```

type zilucru = (luni, marți, miercuri, joi, vineri);
   culori   = (roșu, galben, albastru);
   grad     = (soldat, fruntaș, caporal, sergent);
   boolean  = (false, true);

```

Ordonarea în mulțimea valorilor unui tip enumerare este determinată de poziția identificatorilor în listă [7].

Identificatorii reprezentând valori ale mărimilor de tip enumerare („luni“, „roșu“, „frunțaș“) pot fi folosiți în calitate de constante, ceea ce mărește considerabil claritatea programului.

Exemplu :

```
var zi: zilucru ; în loc de      var zi: integer ;
.....                          .....
zi: = marți ;                  zi: =1 ;
.....                          .....
f zi < > vineri then .....    if zi < > 4 then .....
```

2.2.2.3. *Tipul subdomeniu.* În anumite aplicații se cunosc limitele între care o variabilă de tip scalar poate lua valori. În acest caz tipul variabilei se poate defini ca un subdomeniu al tipului scalar respectiv. În definiția tipului se indică limitele inferioară și superioară ale intervalului, ca mai jos :

```
type gradat = frunțaș .. sergent ;
   ti       = 20..115 ;
   cifre    = '0'...'9' ;
```

Nu se definesc subdomenii ale tipului real.

2.2.3. Tipuri structurate

2.2.3.1. *Tabloul.* Tabloul este o structură formată dintr-un număr fix de componente, toate de același tip. Componentele unui tablou se numesc elemente ; selectarea unui element de tablou se face pornind de la numele variabilei tablou și indicând poziția elementului în cadrul structurii prin așa-numitul indice [7].

La definirea unui tablou se precizează atât tipul elementelor (tipul de bază al tabloului), cât și al indicelui. Prin acesta din urmă se fixează implicit și numărul componentelor tabloului.

Exemple :

```
type t = array (.ti.) of tb
   ti: tipul indicelui, care poate fi orice tip scalar, mai puțin real ;
   tb: tipul de bază, cu privire la care nu există nici o restricție.
type orelucru = array (.zilucru.) of real ;
   text      = array (.1..20.) of char ;
   matrice   = array (.1..10.) of (.3..15). of integer ;
sau          matrice = array (.1..10,3..15.) of integer ;
var ore: orelucru ; m: matrice ; i: integer ;
.....
   ore (.marți.) = 8.25 ;
   m (.I+1,4.) = 7 ;
```

Indicele este o expresie a cărui tip trebuie să corespundă cu cel indicat la definirea tabloului (ceea ce presupune implicit și încadrarea valorii expresiei în limitele impuse).

2.2.3.2. *Articolul*. Articolul reprezintă o reuniune a unui număr fix de componente, care pot fi de tip diferit. El este astfel o metodă de structurare mai generală decât tabloul. Componentele articolului se numesc cîmpuri. La definirea articolului se specifică atât numele cîmpurilor, cît și tipul lor [7].

Exemple :

```

type data = record
    zi : 1..31 ;
    luna : 1..12 ;
    an : 1900..2000
end ;
persoana = record
    nume : array (1..10.) of c h a r ;
    datan : data
end ;

```

Selectarea cîmpurilor se realizează indicînd atât identificatorul variabilei articol, cît și al cîmpului, printr-un procedeu numit calificare, ca în exemplul de mai jos :

```

var pers : persoana ;
.....
pers. nume : = 'popescu' ;
pers. datan.an : = 1950 ;

```

Structuri de articol diferite, dar care prezintă componente cu semnificație înrudită, sau chiar porțiuni identice, pot fi descrise ca variante ale aceluiași tip articol. În acest scop, la definirea tipului se va preciza un cîmp, numit cîmp selector, a cărui valoare momentană va indica structura curentă a articolului. Pentru diferitele valori posibile ale cîmpului selector se vor descrie cîmpurile corespunzătoare variantelor.

Exemplu :

```

type sex = (m, f) ;
persoana2 = record
    nume : array (1..10.) of c h a r ;
    case sexul : sex of
        m : (datan : data ; greutate : r e a l) ;
        f : (nrcopii : i n t e g e r)
end ;

```

Cîmpul selector se numește „sexul” și este de tip „sex”. Cele două structuri posibile ale unui articol de tip „persoana2” sînt prezentate în fig. 1.

2.2.3.3. *Mulțimea*. În limbajul PASCAL este posibil ca elementele mulțimii de valori, corespunzătoare unui tip să fie la rîndul lor mulțimi. Acest caz corespunde definirii unui așa-numit tip mulțime, ca în exemplul de mai jos :

```

type t = set of t0

```


Nume	POPESCU			Nume	IONESCU	
Sexul	M			Sexul	F	
Data	2	11	1980	Nr copii	5	
Greutate	80					

Fig. 1. Variantele articolului de tip „persoana 2”.

Valorile posibile pentru o variabilă de tipul „t” sînt toate submulțimile mulțimii valorilor corespunzătoare tipului „t0”, inclusiv mulțimea vidă.

Dacă definim $t0 = (a, b, c)$, o variabilă de tipul „t” de mai sus ar putea lua ca valori una din următoarele mulțimi:

$\{ \}; \{ a \}; \{ b \}; \{ c \}; \{ a, b \}; \{ a, c \}; \{ b, c \}; \{ a, b, c \}$.

Între mulțimi astfel definite se pot executa, în sensul matematic cunoscut, operațiile de reuniune, intersecție, scădere (*or*, *and*, *—*) și se pot aplica operatorii relaționali egal, diferit, incluziune în ambele sensuri ($=$, $<$, $>$, $<=$, $>=$). De asemenea se poate verifica apartenența unui element la o mulțime cu ajutorul operatorului relațional *in* [1].

2.2.3.4. *Fișierul* Fișierul este o mulțime organizată de informații. Aceste informații sînt grupate sub formă de înregistrări, care în marea majoritate a cazurilor sînt stocate pe un suport extern. Numărul înregistrărilor este neprecizat și din acest punct de vedere fișierul poate fi considerat ca o structură dinamică (v. § 2.2.4).

Una din caracteristicile esențiale ale unui fișier este tipul înregistrărilor sale (aceiași pentru întregul fișier). Acesta se numește tipul de bază al fișierului și se indică la definirea tipului fișier, astfel:

type tipfiș = *file of* tipbază

O variabilă de tip fișier reprezintă o structură organizată conform tipului de bază, prin intermediul căreia se execută prelucrările asupra fișierului (este practic o zonă tampon). Ea conține la un moment dat o înregistrare a fișierului.

Definiția de tip *file* nu descrie caracteristicile dependente de suport și de organizare ale fișierului propriu-zis. Acestea se indică prin alte construcții de limbaj, ale căror formă și semnificație depind de implementarea pe un anumit sistem de calcul [8].

Prelucrarea fișierelor se execută prin rutine standard pentru deschidere, închidere, citire, scriere, punere la zi etc.

2.2.4. Structuri de date dinamice. Tipul pointer

Tipurile tablou, articol, mulțime descriu, așa cum s-a arătat, structuri de formă și dimensiune fixă cunoscute la compilarea programului. Ele reprezintă din acest motiv structuri statice. Fișierul, chiar dacă în ansamblu este o structură de dimensiuni nelimitate, poate fi privit ca structură statică, prin prisma semnificației asociate unei variabile de tip fișier.

Deseori devine însă necesară utilizarea unor structuri a căror formă sau/și dimensiune nu este previzibilă la scrierea programului. Numărul componentelor unei astfel de structuri și legăturile între ele se modifică pe parcursul execuției. Aceste structuri se numesc dinamice.

În limbajul PASCAL componentele (nodurile) unei structuri dinamice se pot crea sau distruge în timpul execuției, conform necesităților exprimate dinamic în program. În acest scop există posibilitatea de a se defini un tip special, denumit pointer, ca în exemplul de mai jos :

```
type tp = ^t ;
```

Valorile corespunzătoare tipului „tp” sînt adrese care permit accesul la variabile alocate dinamic, de tipul „t”. În fig. 2 se prezintă două structuri dinamice, o listă liniară și respectiv un arbore. Descrierile în PASCAL ale nodurilor corespunzătoare acestor structuri sînt :

```
type ptlista = ^ nodlista ;
   nodlista = record
       info : tipinfo ;
       urm : ptlista
   end ;
```

```
type ptarbore = ^ nodarbore ;
   nodarbore = record
       info : tipinfo ;
       stîng, drept : ptarbore
   end ;
```

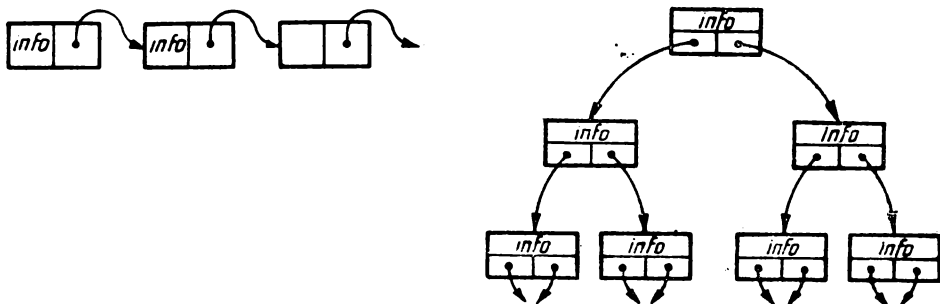


Fig. 2. Exemple de structuri dinamice.

Alocarea dinamică a unei variabile de tip t se realizează la cererea explicită din program, prin apelul procedurii standard `new` sub forma : `new (p)` ; „p” este o variabilă de tipul `tp = ^t`.

În afara alocării zonei de memorie corespunzătoare structurii de tip „t”, apelul lui `new` efectuează și asignarea variabilei „p” cu valoarea adresei structurii respective. Toate referirile la variabila alocată dinamic se realizează prin pointerul „p”, sub forma : `p`.

Procesul invers, de eliberare a unei zone alocate dinamic, referite prin pointerul `p`, se execută ca urmare a apelului procedurii standard `dispose` sub forma : `dispose (p)`.

În continuare se prezintă secvența PASCAL de generare și introducere a unui nou nod (cu pointerul „q”) într-o listă liniară de forma din fig. 2, după nodul cu pointerul „p”:

```
var p, q: plista ;
.....
new (q); q∂.info := ...
q∂.urm := p∂.urm ; p∂.urm := q ;
```

2.3. Structura unui program PASCAL. Blocul

Unitatea de bază a programului este blocul. El cuprinde, pe de o parte, descrierea datelor și pe de altă parte, acțiunile executate asupra acestora. Descrierea datelor se face prin declarații de constante, tipuri și variabile. Acțiunile sînt cuprinse în corpul de instrucțiuni al blocului. Un bloc poate să conțină la rîndul lui descrierea altor unități de program reprezentate de asemenea prin blocuri. Acestea sînt procedurile și funcțiile (rutinele limbajului). Prin urmare un program PASCAL prezintă o structură ierarhizată de blocuri suprapuse.

Această structură poate fi urmărită și în programul 1 (fig. 3). Întregul program este un bloc cu partea de declarații cuprinsă între liniile 2—17 și corpul de instrucțiuni între 18—22. Acest bloc conține blocul corespunzător procedurii „rezolvă” (declarații 4—9, corp de instrucțiuni 10—17). În cadrul procedurii se declără încă două blocuri și anume funcțiile „y” și „yd”, a căror parte de declarații este vidă. Executarea programului începe cu corpul de instrucțiuni al blocului exterior (programul principal) — linia 18.

```
1 PROGRAM NEWTON;
2 VAR R, START: REAL;
3 PROCEDURE REZOLVA (VAR RAD: REAL; X1: REAL);
4 CONST EPS = 0.0001;
5 VAR X0, X2: REAL;
6 FUNCTION Y(X: REAL): REAL;
7 BEGIN Y := COS(X) - X END;
8 FUNCTION YD(X: REAL): REAL;
9 BEGIN YD := -SIN(X) - 1 END;
10 BEGIN
11 X1 := X1;
12 REPEAT
13 X0 := X1;
14 X1 := X0 - Y(X0) / YD(X0)
15 UNTIL (ABS(X1 - X0) < EPS);
16 RAD := X1
17 END;
18 BEGIN
19 READ (START);
20 REZOLVA (R, START);
21 WRITE ('SOLUTIA = ', R)
22 END.
```

Fig. 3. (Programul 1). Determinarea rădăcinii ecuației $\cos(x) = x$ prin metoda Newton.

Programul calculează rădăcina ecuației $\cos(x) - x = 0$, cu aproximația „eps” = 0.0001, pornind de la valoarea inițială „start”, care se citește.

Pentru determinarea aproximativă a rădăcinii se folosește metoda lui Newton. Pe baza formulei $\operatorname{tg} \alpha = y(x_0) / (x_0 - x_1)$, unde $\operatorname{tg} \alpha$ este valoarea derivatei

funcției în x_0 , se calculează valori succesive x_1 în funcție de valorile anterioare x_0 . Calculul se încheie cînd aproximațiile succesive sînt suficient de apropiate.

Procedurile și funcțiile reprezintă blocuri cărora li se asociază la declarare un nume și o listă de parametri formali. La declararea unei funcții se precizează în plus și tipul rezultatului. Procedurile și funcțiile sînt modulele din care se construiește în mod ierarhizat un program PASCAL.

Activarea unei proceduri se face prin apelarea ei în cadrul unei instrucțiuni de apel (linia 20). O funcție se activează la evaluarea unei expresii în care apare numele ei (linia 14). În ambele cazuri se va executa corpul de instrucțiuni al blocului corespunzător rutinei. Argumentele (parametrii actuali) transmiși la apel trebuie să corespundă ca număr și tip cu parametrii formali specificați la declararea rutinei.

2.4. Tipuri sistem. Modularizarea unui program PASCAL CONCURENT

S-a arătat anterior că prin noțiunea de tip se realizează asocierea între o mulțime de valori și un set de operații permise între valorile respective. În acest sens, tipurilor standard le sînt asociate anumite operații elementare definite prin limbaj. Descrierea operațiilor între structuri rămîne în sarcina programatorului și se realizează în general prin proceduri și funcții adecvate. Prin urmare s-a ivit necesitatea de a introduce construcții de limbaj prin care să se permită asocierea în mod explicit a unor structuri de date cu operațiile aplicabile lor. Se asigură astfel legătura date-acțiunii, în sensul că unor structuri li se pot aplica doar anumite operații bine determinate.

Pornind de la aceste considerente, în limbajul PASCAL CONCURENT s-a extins noțiunea de tip prin introducerea tipurilor sistem *proces*, *monitor* și *class*. O definiție de tip sistem are sintaxa unui bloc și conține prin urmare descrierea unor date (declarații de constante, tipuri, variabile) și a acțiunilor prin care se operează asupra acestora (declarații de proceduri și funcții, corp de instrucțiuni).

În programul 2 (fig. 4) se definesc tipurile clasă „ie-terminal“ (liniile 22—75) „ie-banda“ (77—94), „ie-consola“ (96—127), tipul monitor „acces-banda“ (129—151) și tipurile proces „central“ (153—183) și „operator“ (185—207).

Asupra datelor descrise în interiorul unui tip sistem se poate acționa doar prin intermediul acțiunilor descrise în aceste tipuri. În cazul în care accesul la datele tipului sistem este permis din afara acestuia, prelucrările din exterior se execută apelînd rutinele descrise în tipul sistem și declarate ca externe (declarația conținînd cuvîntul cheie *entry*).

Modulele din care este compus un program PASCAL CONCURENT sînt elemente de tip *proces*, *monitor*, *class*. Programul concurrent se descrie ca o interconexiune a unor variabile de aceste tipuri sistem.

```

1  CONST NRPROCESE=3;
2  CONST EM='(:25:)' ;CRK='(:13:)' ;LF='(:21:)' ;NL='(:21:)' ;TRM='(:11:)' ;
3  TYPE OPERATIE=(CITIRE,SCRIERE,CONTROL,EDI,C,WASP,C,C,REBOBINARE,
4  STEGERE,SALT_BLOC,SALT_FISIER,SCR_EOF,POZITIONARE,
5  CIT_PUZITIE,ARR_ATT_BRK,ATT_BRK,DEZACTIV);
6  TYPE PARAMIE=RECORD
7  OPERATIE:OPERATIE;
8  STARE,LUNGIME:INTEGER;
9  VS:ARRAY(.1..6.) OF CHAR;
10 DV:ARRAY(.1..4.) OF CHAR;
11 ABDKT:BOULEAN
12 END;
13 TYPE APARATIE=(LECTUR,PERF_CART,IMPRIMANTA,CONSOLE,BANDA,DISC,TERMINAL);
14 CONST LUNGLIN=89;
15 TYPE LINIE = ARRAY(.1..LUNGLIN.) OF CHAR;
16 DATA=RECORD AN,LUNA,ZI:INTEGER END;
17 TYPE DOCUMENT=RECORD
18 C1,C2,C3:LINIE
19 END;
20 CONST LGART=48;
21 [#####]
22 TYPE IE_TERMINAL=CLASS(NNAPARAT:CHAR);
23 [#####]
24 VAR PARAMI,PARAME,PARAMBRK:PARAMIE;
25 CONV:CONVERSII; SPECIALE:SET OF CHAR;
26 SPECIALE:SET OF CHAR;
27 PROCEDURE SCRMESAJ(LIN:LINIE); [ PROCEDURA INTERNA ]
28 VAR ZONA:LINIE; I:INTEGER; SFR:BOOLEAN;
29 BEGIN
30 SFR:=FALSE; I:=0;
31 WHILE NOT SFR DO BEGIN
32 I:=I+1; ZONA(.1.):=LIN(I.);
33 SFR:=(I=LUNGLIN) OR (LIN(I.)='(:0:)' );
34 END; [S-A DETERMINAT LUNGIMEA MESAJULUI CARE SE TERMINA CU (:0:)]
35 PARAME.LUNGIME:=I;
36 IO(ZONA,PARAME,TERMINAL);
37 END;
38 PROCEDURE ENTRY CITMESAJ(VAR LIN:LINIE);
39 VAR ZONA:LINIE; CCONTOR:INTEGER; SFR:BOOLEAN;
40 BEGIN
41 IO(ZONA,PARAMI,TERMINAL);
42 IF PARAMI.STARE<>0 THEN BEGIN [ TRATARE EROARE ]
43 CCONTOR:=1;
44 REPEAT
45 SCRMESAJ('(:13:)(:21:)(REPETATI MESAJUL(:13:)(:21:)(:0:)' );
46 IO(ZONA,PARAMI,TERMINAL);
47 CCONTOR:=CCONTOR+1;
48 IF CCONTOR=10 THEN IO(ZONA,PARAMBRK,TERMINAL);
49 UNTIL PARAMI.STARE=0; \ [ SF. TRATARE EROARE ]
50 END;
51 SFR:=FALSE; CCONTOR:=0;
52 WHILE NOT SFR DO BEGIN
53 CCONTOR:=CCONTOR+1;
54 LIN(CCONTOR.):=ZONA(CCONTOR.);
55 SFR:=(CCONTOR=LUNGLIN) OR (LIN(CCONTOR.) IN SPECIALE) ;
56 END;
57 IF CCONTOR<LUNGLIN THEN FOR CCONTOR:=CCONTOR TO LUNGLIN DO
58 LIN(CCONTOR.):=' ' ;
59 END;
60 PROCEDURE ENTRY ASTBRK;
61 BEGIN IO(PARAMBRK,PARAMBRK,TERMINAL) END;
62 PROCEDURE ENTRY SCRMESAJ(LIN:LINIE);
63 BEGIN SCRMESAJ(LIN) END;
64 BEGIN
65 SPECIALE:=(EM,CRK,LF,NL,TRM.);
66 WITH PARAMI DO BEGIN
67 OPERATIE:=CITIRE;DV:='TT ' ;DV(.3.):=NNAPARAT;ABORT:=FALSE;
68 LUNGIME:=LUNGLIN;
69 END;
70 WITH PARAME DO BEGIN
71 OPERATIE:=SCRIERE;DV:='TT ' ;DV(.3.1):=NNAPARAT;ABORT:=FALSE;
72 END;
73 WITH PARAMBRK DO BEGIN
74 OPERATIE:=ATT_BRK;DV:='TT ' ;DV(.3.):=NNAPARAT;ABORT:=FALSE;
75 END;
76 END;
77 [#####]
78 TYPE IE_BANDA=CLASS
79 [#####]
80 VAR SFPARAM:PARAMIE;
81 PROCEDURE ENTRY SCRLOCUIVAR ART:DOCUMENT);
82 BEGIN
83 PARAM.STARE:=1; IO(ART,PARAM,BANDA);
84 END;
85 PROCEDURE ENTRY SCREOF;
86 BEGIN IO(SFPARAM,SFPARAM,BANDA) END;
87 BEGIN
88 WITH PARAM DO BEGIN
89 OPERATIE:=SCRIERE; LUNGIME:=LGART;DV:='HTI ' ; ABORT:=TRUE;
90 END;
91 WITH SFPARAM DO BEGIN

```

```

92     OPERATIE:=SCR_ELF;DV:='MT1 ';ABORT:=TRUE;
93     END;
94     END;
95     TYPE MESAJ=ARRAY(.0..LUNGLIN.) OF CHAR;
96     [#####]
97     TYPE IE CONSOLA=CLASS;
98     [#####]
99     VAR PARAMI,PARAME,PARAMC:PARAMIE; SPECIALE:SET OF CHAR;
100    PROCEDURE ENTRY SCRMESSAJ(LIN:LINIE);
101    VAR MES:MESAJ;I:INTEGER; SFR:BOOLEAN;
102    BEGIN
103        SFR:=FALSE; I:=0;
104        WHILE NOT SFR DO BEGIN
105            I:=I+1; MES(I,1):=LIN(I,1);
106            SFR:=(I=LUNGLIN) OR (LIN(I,1)='(::0)');
107        END;
108        MES(0,0):=CHR(I); IO(MES,PARAME,CONSOLA); [LG MESAJ IN MES(0,1)
109    END;
110    PROCEDURE ENTRY CITMESAJ(VAR LIN:LINIE);
111    VAR MES:MESAJ; I:INTEGER; SFR:BOOLEAN;
112    BEGIN
113        MES(0,0):=CHR(LUNGLIN); IO(MES,PARAMI,CONSOLA);
114        SFR:=FALSE; I:=0;
115        WHILE NOT SFR DO BEGIN
116            I:=I+1; LIN(I,1):=MES(I,1);
117            SFR:=(I=LUNGLIN) OR (LIN(I,1) IN SPECIALE);
118        END;
119        IF I<LUNGLIN THEN FOR I:=I TO LUNGLIN DO LIN(I,1):=' ';
120    END;
121    PROCEDURE ENTRY INTERVENTIE;
122    BEGIN IO(PARAMC,PARAMC,CONSOLA) END;
123    BEGIN
124        SPECIALE:=I.E.M,CRR,LF,NL,TRM.);
125        PARAMI.OPERATIE:=CITIKE;PARAMI.ABORT:=TRUE;PARAMI.DV:='TW ';
126        PARAME.OPERATIE:=SCRIERE; PARAME.ABORT:=TRUE;PARAME.DV:='TW ';
127        PARAMC.OPERATIE:=CONTROL; PARAMC.DV:='TW ';
128    END;
129    [#####]
130    TYPE ACCES_BANDA=MONITOR;
131    [#####]
132    VAR B:IE_BANDA; I:INTEGER;
133    LOCATII=ARRAY(1..NRPROCESE) OF QUEUE;
134    ACTIV=ARRAY(1..NRPROCESE) OF BOOLEAN;
135    PROCEDURE ENTRY SCRARTICOL(VAR ART:DOCUMENT;NRP:INTEGER);
136    BEGIN
137        IF NOT ACTIV(NRP) THEN DELAY(LOCATII((NRP)));
138        B.SCRBLOC(ART);
139    END;
140    PROCEDURE ENTRY OPRESTE(NRP:INTEGER);
141    BEGIN ACTIV(NRP):=FALSE END;
142    PROCEDURE ENTRY PORNESTE(NRP:INTEGER);
143    BEGIN ACTIV(NRP):=TRUE; CONTINUE(LOCATII((NRP))); END;
144    PROCEDURE ENTRY INCHIDE;
145    BEGIN
146        FOR I:=1 TO NRPROCESE DO ACTIV(I,1):=FALSE;
147        B.SCREOF;
148    END;
149    BEGIN
150        FOR I:=1 TO NRPROCESE DO ACTIV(I,1):=FALSE;
151        INIT B;
152    END;
153    [#####]
154    TYPE CENTRAL=PROCESS(ACCES:ACCES_BANDA);
155    [#####]
156    VAR C:IE_CONSOLA; LIN:LINIE;
157    PROCEDURE CITNRPROC(VAR NKP:INTEGER);
158    VAR LIN:LINIE;
159    BEGIN
160        REPEAT
161            C.SCRMESAJ('DATI NUMARUL PROCESULUI:(:0)');
162            C.CITMESAJ(LIN); NRP:=ORD(LIN(I,1))-ORD('0');
163        UNTIL (NRP>=1) AND (NKP<=NRPROCESE);
164    END;
165    PROCEDURE LANSARE;
166    VAR NRP:INTEGER;
167    BEGIN CITNRPROC(NRP); ACCES.PORNESTE(NRP); END;
168    PROCEDURE OPRIRE;
169    VAR NRP:INTEGER;
170    BEGIN CITNRPROC(NRP); ACCES.OPRESTE(NRP); END;
171    PROCEDURE INCHIDE;
172    BEGIN ACCES.INCHIDE END;
173    BEGIN
174        INIT C;
175        CYCLE
176            C.INTERVENTIE;
177            C.SCRMESAJ('COMANDA:(:0)');
178            C.CITMESAJ(LIN);
179            IF LIN(I,1)='L' THEN LANSARE ELSE
180            IF LIN(I,1)='O' THEN OPRIRE ELSE
181            IF LIN(I,1)='I' THEN INCHIDE ELSE
182            C.SCRMESAJ('EROARE(:0)');
183        END;
184    END;
185    [#####]

```

ARTICOLUL CONTINUA LA PAGINA 354

„O punte între știință și tehnologie“

(Congresul mondial trienal
I.F.A.C., Budapesta, 1984)

SISTEM DE CONDUCERE CU MICROCALCULATOARE, TOLERANT LA DEFECTE, AL MAȘINII DE ÎNCĂRCARE-DESCĂRCARE A COMBUSTIBILULUI PENTRU CENTRALE NUCLEARE

J. Borka și S. Keresztely

Instit. de Automatizare și
Calculatoare, Academia
Ungară de Științe (R.P.U.)

Rezumat :

Sistemul de conducere (comandă) elaborat pentru mașina de încărcare-descărcare a combustibilului pentru centrale nucleare cuprinde 7 microcalculatoare. Structurile hardware și software redundante asigură o mare fiabilitate și disponibilitate. Senzorii sînt dublați și fiecare set este conectat la propriul său microcalculator, care asigură evaluarea măsurărilor.

Programul de conducere care coordonează 7 acționări electrice este derulat pe două microcalculatoare identice și are acces la ambele seturi de rezultate de măsurare. Există două console operator. Cel de-al șaptelea microcalculator generează un plan simulat al zonei de lucru în jurul poziției reale, iar imaginea este transmisă monitoarelor TV-color de pe pupitrele de comandă.

Fiabilitatea sistemului : la defectarea oricăreia dintre părțile sistemului se produce un mesaj de eroare și nici o acțiune. *Disponibilitatea sistemului* : în caz de avarie, unul din fiecare pereche de subsisteme identice trebuie să fie operațional. În acest regim de avarie se pierde fiabilitatea necondiționată.

Sistemul de antrenare conține acționări de c.c. în patru cadrane, cu comanda curentului de circulație.

Cuvînt cheie : Sisteme redundante (tolerante la defecte); fiabilitate; disponibilitate; conducere comandă prin microcalculatoare; acționari de c.c. în patru cadrane.

Introducere : Mașina de încărcare-descărcare a combustibilului (MID) dintr-o centrală nucleară este supusă unor standarde de securitate ridicată tipice industriei centralelor nucleare. Condițiile principale pentru sistemul de comandă al MID sînt :

- disponibilitate foarte mare,
- fiabilitate ridicată,
- posibilitatea comenzii la distanță,
- blocarea comenzilor eronate.

Disponibilitate foarte mare

MID este o macara mare, cu un braț manipulator de precizie. MID trebuie să fie în stare de funcționare atîta vreme cît părțile mecanice și motoarele electrice rezistă la acțiunea mediului ambiant.

Sistemul de comandă trebuie să se comporte în consecință.

Nivelul de disponibilitate cerut presupune că, în cazul unei singure erori în oricare parte a sistemului de comandă, disponibilitatea se menține, cu o siguranță în funcționare mai scăzută.

Fiabilitate ridicată

Se cere ca o singură eroare în oricare parte a sistemului de comandă, însoțită sau nu de o comandă eronată sau corectă a operatorului, să nu ducă la o acțiune greșită.

Posibilitatea comenzii la distanță

Ținând seama de mediul periculos în care funcționează MID, este necesară și o consolă de comandă la distanță. Pentru informarea completă a operatorului, la acest pupitru trebuie să existe toate afișajele, un ecran TV în circuit închis etc.

Proiectanții consideră că această consolă pentru comandă la distanță trebuie utilizată și pentru operațiunile normale de comandă a mașinii.

Blocarea comenzilor greșite

Brațul MID se mișcă într-un spațiu multidimensional, cu mai multe limitări ale mișcării. Depășirea acestor limitări poate duce la deteriorarea barei de combustibil, a MID sau a ambelor. Sistemul de comandă verifică comenzile operatorului, în funcție de aceste limitări (geometrice și altele, de ex. greutatea sau secvențe operaționale) și blochează execuția în cazul nerespectării lor.

Structura sistemului de comandă

Sistemul de comandă și acționările electrice sînt amplasate în camera de comandă la distanță, unde condițiile de mediu sînt mai puțin severe. Senzorii de poziție, comutatoarele de contact și tensometrele, motoarele și frîncle montate pe MID sînt de tipul „regim greu“.

Sistemul de comandă conține subsisteme de patru tipuri diferite iar cele trei tipuri de subsisteme vitale sînt dublate, după cum se arată în fig. 1. Subsistemele dublate execută aceleași sarcini și trebuie să producă ieșiri identice.

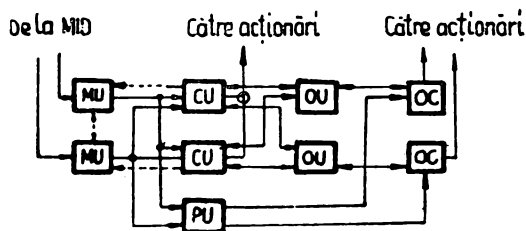


Fig. 1. Structura sistemului de comandă.

Blocul de măsurare (MU)

Ambele blocuri de măsurare își au propriul lor set de senzori pe MID. MU convertește semnalele de fază ale transformatoarelor de comandă a sesizării poziției în coordonate, face verificările logice ale acestor valori și stărilor comutatoarelor de contact și compilează un mesaj de ieșire la fiecare 20 ms.

Acest mesaj conține informațiile de stare referitoare la microcalculatorul blocului de măsurare, la sistemul de măsurare, și cum noile rezultate ale măsurării.

Mesajul este însoțit de o verificare a redundanței ciclice pe 16 biți și este transmis la alte unități fără a aștepta o confirmare. În cazul unei erori de comunicație detectate la capătul de recepție, mesajul trebuie — pur și simplu — anulat.

Informațiile pierdute, se repetă în cursul citorva cicluri de măsurare. Traseele de informații, reprezentate în fig. 1 prin linii întrerupte, nu se utilizează în timpul funcționării MID, ele servesc pentru testare.

Blocul de comandă (CU)

CU primește informații de la blocul de măsurare, comunică cu operatorul (primind comenzi și trimițând răspunsuri) și execută algoritmul de comandă.

Funcțiuni de supraveghere. CU primește comenzi de la operator cu privire la starea de funcționare a sistemului, ține evidența informațiilor de stare a blocului de măsurare și a celuilalt bloc de comandă.

Din acestea se obține starea prezentă a sistemului și se transmite la consola operatorului. Dacă cerințele sistemului nu sînt corespunzătoare se generează un mesaj de eroare.

Funcțiuni de evaluare a comenzii. În regim manual, operatorul inițializează o mișcare, dînd o comandă corespunzătoare mașinii, în timp ce, în regim automat, comenzile sînt generate de către interpretorul de comenzi al blocului de comandă. În ambele cazuri comenzile se testează față de limitele geometrice și de alte limite. Dacă testul reușește, comanda se asociază cu parametrii limită de curent și viteză și cu poziția limită în care trebuie să se oprească acționarea.

După suplimentarea cu aceste informații, comanda este gata de execuție.

Funcțiuni de comandă propriu-zise. În blocul de comandă, acest task compară între ele mesajele de la blocul de măsurare cu informațiile anterioare și actualizează setul de date de măsurare. Dacă setul de date de măsurare este corect, se pot transmite comenzi la acționările electrice. La apropierea de poziția limită, viteza mișcării se reduce pas cu pas și, în ultima fază, funcția de evaluare a comenzii se reinițializează. Progresul realizat în ciclul de lucru poate să aibă ca rezultat noi poziții limită sau alți parametri. Inițializarea repetată a evaluării comenzii are ca rezultat execuția unei secvențe de comenzi cu depășire maximă, dirijate de regulile de limitare.

Pupitrul operatorului (OU)

Există în sistem două pupitre ale operatorului, însă numai unul dintre ele poate fi activ la un moment dat. Cel pasiv — dacă este alimentat — afișează informațiile și nu face nimic altceva.

Funcția principală a OU constă în comprimarea/expandarea informațiilor între blocul de comandă, afișajul operatorului și tastatură. Pentru transmiterea informațiilor și pentru asigurarea eficienței prelucrării, mesajele la blocul de comandă sînt codificate în formă compactă, în timp ce operatorul are la dispoziție textul în clar.

Consola operatorului conține afișajul și tastatura, elementele de comandă manuală conectate direct la acționările electrice, TV-ul în circuit închis și monitorul color conectat la blocul video.

Blocul video (PU)

Blocul video nu face parte din sistemul de comandă, ci este un ajutor vizual pentru operator. Primește rezultatele măsurătorii de la blocul de măsurare și generează un plan simulat al zonei din jurul brațului manipulator al MID.

Locațiile unde pot fi găsite sau așezate barele de combustibil sînt arătate pe monitorul color prin cercuri colorate.

Imaginea constă din 256×384 puncte și poate fi deplasată prin hardware cu o rezoluție de un punct echivalent cu cea 5 mm.

Funcționarea sistemului

În sistem funcționează două blocuri de măsurare, două blocuri de comandă și pupitrul operatorului. Cele două blocuri de măsurare primesc datele de măsurare corecte și rezultatele calculate sînt transmise ambelor blocuri de comandă.

Blocurile de comandă primesc informațiile de la ambele blocuri de măsurare și le compară. În urma comparării, în ambele blocuri de comandă, în cazul echivalenței informațiilor, algoritmul de comandă este activat.

Semnalele de comandă calculate sînt transmise la acționările electrice prin ambele blocuri de comandă prin optoizolatoare conectate în serie. Semnalul de ieșire combinat este reținut de blocurile de comandă și nici unul dintre blocurile de comandă nu detectează limitarea de către celălalt, adică ieșirile sînt aceleași.

Comenzile de la pupitrul operatorului activ sînt primite și confirmate de ambele blocuri de comandă.

Mesajele către pupitrul operatorului sînt transmise numai printr-un singur bloc de comandă. Pupitrul operatorului nu este dublat. Nu există nici un motiv pentru a face acest lucru : sistemul este asigurat împotriva comenzilor

greșite din partea operatorului și cel mai rău lucru pe care-l poate face o unitate operator defectă este să producă un asemenea mesaj de ieșirea.

Se poate vedea cu ușurință : în acest sistem, orice eroare însoțită de o comandă greșită din partea operatorului nu poate niciodată să inițieze o acțiune greșită. Cel puțin unul dintre blocurile de comandă este corect și primește cel puțin un set de rezultate corecte ale măsurării. Înseamnă că acest bloc de comandă nu acționează asupra unui set greșit de rezultate ale măsurării și blochează astfel comenzile eronate ale operatorului (sau OU generează comenzi false) și oprește orice acțiune greșită a celui alt CU de comandă.

Sistemul este asigurat și împotriva dublelor greșeli, cu excepția cazului următor : aceeași eroare este generată în ambele subsisteme (duble) în intervalul timpului de reacție al sistemului.

Sistemul reacționează la erori cu toleranța necesară : încearcă să compenseze erorile de comunicație dintre subsisteme. Durata unui ciclu al programului blocului de măsurare și comunicației este de 20 ms ; se tolerează unul sau două cicluri, în funcție de natura datelor ce lipsesc. Timpul de comunicație al OU și al ciclului de evaluare a comenzii de cca 100 ms și se tolerează două mesaje lipsă sau incorecte.

Semnalele de ieșire către acționările electrice pot să difere timp de cca 200 ms. Limita de curent și semnalele de viteză sînt codificate : o diferență de 1 bit în valoarea codificată nu este tratată ca o nepotrivire. În acest mod, se evită reacția de panică față de erori temporare ne semnificative.

Pentru detectarea prematură a erorilor (în părțile nefolosite ale sistemului) fiecare microcalculator execută un program de autotestare la nivelul cel mai mic de prioritate, iar dezvoltarea acestui program este verificată la nivelul superior. Acest program de autotestare citește memoria de la început la sfîrșit și în caz de eroare la verificarea de paritate oprește microcalculatorul. Blocurile de măsurare trimit informațiile lor de stare la unitățile de comandă.

Cele 2 blocuri de comandă schimbă informații de stare regulat prin intermediul OU activ și verifică ciclic interfața cu acționările electrice pentru evitarea scurtcircuitelor.

Funcționarea cu subsisteme defecte

Imediat ce se detectează un defect se trimite un mesaj de eroare la consola operatorului și mașina de încărcare-descărcare a combustibilului se oprește. Personalul trebuie să repare defectul (să înlocuiască placa cu circuite imprimare cu una de rezervă). Dacă, pentru anumite motive, înlocuirea nu este posibilă, sistemul este totuși capabil să funcționeze la un nivel mai scăzut de siguranță. Pentru inițializarea acestor stări de funcționare este necesară aplicarea unei comenzi.

Erori de măsurare

Dacă unul dintre blocurile de măsurare furnizează o informație greșită blocurilor de comandă, aceasta poate fi exclusă din sistemul de comandă prin alegerea stării de funcționare a unui singur bloc de măsurare. Dezavantajul :

orice eroare nouă de măsurare care nu este detectată în blocul de măsurare poate fi o sursă de eroare nedetectată.

Dacă într-unul din blocurile de măsurare una sau mai multe măsurători sînt greșite este posibil să se aleagă blocuri de măsurare combinate. În această stare de funcționare, numai măsurătoarea detectată (și confirmată de operator) este unică, toate celelalte sînt verificate din punct de vedere al egalității în cele două blocuri de comandă.

Dezavantajul este mai mic decît în cazul unui singur bloc de măsurare : numai perechea măsurătorii care lipsește poate cauza o eroare nedetectată.

Erorile dispozitivului de comandă

Dacă unul dintre blocurile de comandă nu funcționează adecvat, operatorul o poate exclude din sistemul de comandă alegînd starea de funcționare a unui singur bloc de comandă. Dacă microcalculatorul exclus mai este încă în funcționare, el înțelege această comandă și trece în stare de rezervă ; își scurtcircuitează ieșirile la acționările electrice și încetează să se interfereze cu sistemul.

Dacă microcalculatorul este incapabil să execute această comandă de rezervă, trebuie izolat din sistem. Aceasta se poate face prin rearanjarea cablurilor conectînd direct acționările electrice cu blocul de control de funcționare.

Dezavantaj : erorile din blocul de comandă nu sînt detectate de o unitate independentă.

Defectarea consolei operatorului

În acest caz sistemul trebuie să fie deconectat și conectat la cealaltă consolă.

Consola la care a fost conectat sistemul este automat consolă activă și sistemul nu solicită starea de funcționare a celeilalte console. Se asigură o fiabilitate completă.

Testarea sistemului

Testarea de hardware și software pînă la nivel de subsistem individual se face în mod obișnuit. Testarea nivelului sistemului a pus o problemă serioasă, pentru că sistemul de comandă nu a putut fi conectat la mașina de încărcare-descărcare a combustibilului pentru test, dar oricum, fără o platformă de testare la fel de mare ca platforma dintr-o centrală electrică nu ar fi de mare ajutor.

S-a hotărît să se simuleze mașina de încărcare-descărcare a combustibilului. Un program complet de simulare pentru mașina de încărcare-descărcare a combustibilului și a mediului înconjurător ar fi o problemă grea cu multe posibilități de greșală.

Considerînd că nu este necesar să se simuleze situații care au loc numai atunci cînd sistemul de comandă nu funcționează (ex. lovirea de obstacole) problema a devenit foarte simplă.

Funcțiile care trebuie realizate :

— generarea valorilor măsurate în concordanță cu comenzile trimise la acționările electrice :

— oprirea acționărilor într-o anumită poziție și indicarea „limitei de curent atinsă“ ;

— schimbarea rezultatelor măsurării greutății datorită mișcărilor simulate.

Aceste funcții simple sînt realizate în cadrul blocurilor de comandă și a blocurilor de măsurare cu sarcini independente. Pentru a obține o soluție corectă, task-urile de simulare nu împart informațiile cu alte programe din microcalculator și folosesc un canal de date separat pentru comunicare (arătat în fig. 1 prin linia întreruptă).

Înainte de a începe simularea, cablurile către acționările electrice trebuie înlocuite cu un dispozitiv de testare.

Programul de simulare din blocul de comandă poate fi început prin panoul operator al microcalculatorului. El verifică prezența dispozitivului de testare lucrînd în absența acționărilor reale. Programul de simulare citește comenzile care sînt ieșirile combinate ale celor două blocuri de comandă. În concordanță cu aceste comenzi se alege viteza de simulare și se transmite la unul din blocurile de măsură. Acest bloc de măsură — dacă simularea a fost începută la panoul operator al microcalculatorului — generează rezultatele măsurării și le trimite direct la blocurile de comandă prin intermediul celui alt bloc de măsură. În timpul simulării, totul pare normal la consola operatorului, ca și cum mașina de încărcare-descărcare a combustibilului ar funcționa.

Răspunsurile dinamice sînt oarecum diferite, dar toate funcțiile de securitate și limitările de mișcare există, împreună cu mesajele de eroare.

Monitorul color indicînd zona din jurul poziției brațului mașinii de încărcare-descărcare a combustibilului, este de asemenea în funcțiune.

Acest software de simulare a devenit o parte din ansamblul final de software. Poate fi folosit pentru a demonstra funcțiile sistemului și pentru verificare. Un alt domeniu de utilizare este pregătirea operatorilor.

Realizare :

Sistemul de comandă constă din șapte subsisteme. Fiecare subsistem este construit în jurul unui microcalculator de același tip și sînt adăugate circuite de interfață pentru a specializa subsistemul. Microcalculatoarele includ un microprocesor Z80, memorie EPROM de 32 Ko și CMOS RAM de 8 Ko cu control de paritate, pentru porți seriale de I/E și un panou de operare.

Unitățile sînt montate în sertar folosind plăci cu circuite imprimate Eurocard.

Subsistemele comunică între ele prin porțile seriale de I/E ale microcalculatoarelor, folosind protocolul de linie HDLC.

Blocul de măsură este arătat în fig. 2. Cartela de intrare primește un semnal de fază de referință și folosește un multiplicator de frecvență PLL și contoare de 8 biți pentru măsurarea semnalului de fază al transformatoarelor de comandă.

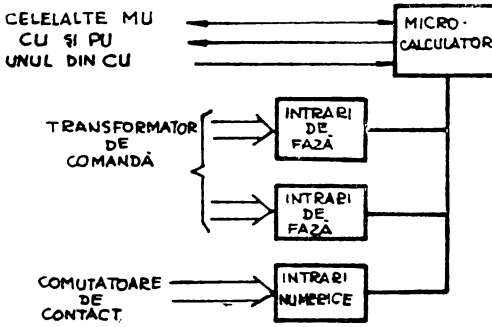


Fig. 2. Blocul de măsură.

Blocul de comandă este arătat în fig. 3. O cartelă de interfață poate comanda două acționări electrice, cele șapte acționări ale mașinii de încărcare-des-cărcare necesită patru cartele.

Pupitrul operatorului este arătat în fig. 4.

Cartela de comandă a display-ului conține memoria de recirculare și generatorul de caractere.

Blocul video este arătată în fig. 5.

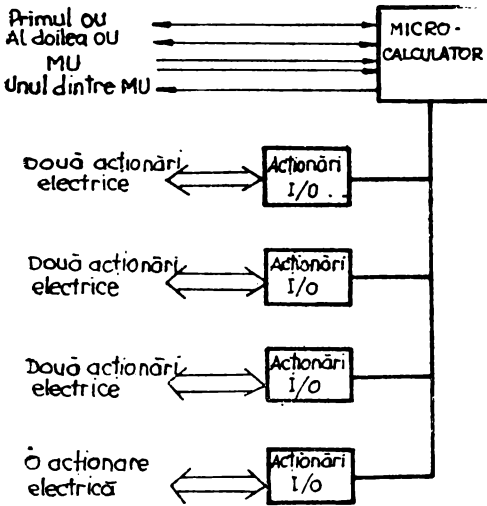


Fig. 3. Blocul de comandă.

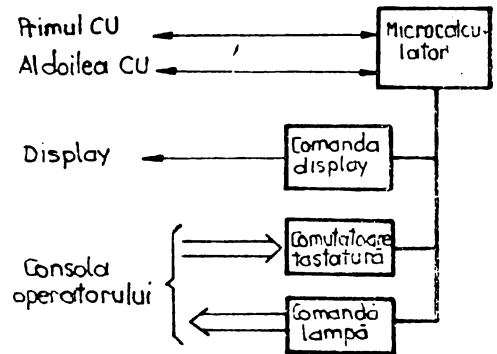


Fig. 4. Pupitrul operatorului.

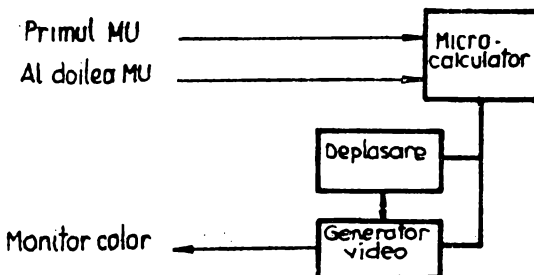


Fig. 5. Blocul video.

Cartele generatorului video conține memoria de recirculare și generatorul setului de caractere semigrafice.

Sistemul de acționare (DS)

Sărcina sistemului de acționare este de a asigura mișcarea comandată în șapte direcții a mașinii de încărcare-descărcare a combustibilului atât în regim manual cât și automat. În regimul manual, operatorul direcționează mișcarea mașinii de încărcare-descărcare a combustibilului (MID) de la o consolă a operatorului OC la alta, dar sistemul de comandă supraveghează permanent întregul ciclu de lucru și comenzile operatorului.

În regim automat consola operatorului este folosită direct de operator pentru a direcționa sistemul de comandă pentru realizarea ciclului de lucru dorit:

Aceasta se face prin darea comenzilor necesare către acționările individuale.

Sistemul de acționare cuprinde șase acționări comandate în c.c. și o acționare de c.a. fără comanda vitezei. Aceste acționări sînt următoarele:

- acționare de cursă lungă;
- acționare de traversare;
- acționare de ridicare, echipată cu șină cu cremalieră;
- acționare de ridicare, echipată cu cablu;
- acționare de rotație pentru mișcarea de rotație a tijei de lucru;
- acționarea de rotație pentru mișcarea de rotație a platformei TV;
- acționarea de ridicare fără comanda vitezei pentru mișcarea verticală a tijei TV.

Schema bloc a unei acționări individuale poate fi văzută în fig. 6.

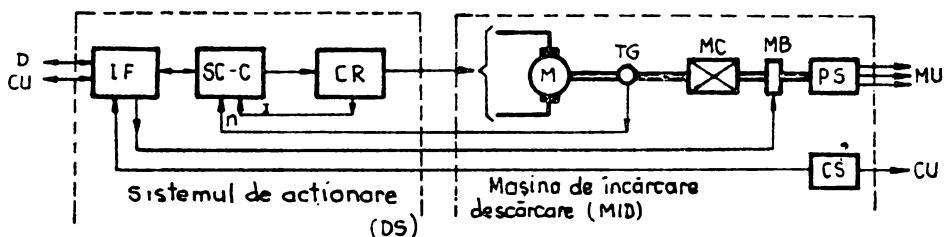


Fig. 6. Schema bloc a sistemului de acționare.

Unitatea de interfață (IF) evaluează semnalele care vin atât de la consola operatorului (OC) și de la blocul de comandă (CU) și în plus furnizează informațiile de stare ale acționărilor individuale către blocul de comandă și transmite valoarea curenților motoarelor consolei operatorului (OC).

Fiecare ansamblu de acționare trebuie să fie capabil să acționeze și să frîneze în ambele sensuri de rotație, respectiv sînt necesare acționări în patru cadrane.

O serie de echipamente tehnologice antrenate ce necesită acționări în patru cadrane; din motive de siguranță impun o trecere prin zero continuă

a curentului. Acesta este motivul pentru care acțiunile simple în curent continuu fără curent de circulație nu au fost selectate deși ele posedă proprietăți preferabile în alte privințe. Mecanismul de ridicat necesită un cuplu nominal chiar și la viteza zero a mașinii de încărcare-descărcare a combustibilului, și pentru a obține o uniformitate, toate acțiunile sînt de același tip.

În acțiunile în curent continuu cu curent de circulație, două seturi de redresoare comandate sînt conectate antiparalele prin intermediul bobinelor de reactanță și un set de redresoare asigură curentul de circulație. În practică există două soluții diferite pentru comanda de amorsare a acestor acțiuni. În cazul uneia dintre acestea, unghiul de amorsare al redresoarelor comandate, variază în funcție de un raport selectat într-un mod adecvat, iar mărimea curentului de circulație într-un regim de lucru discontinuu este limitată doar de inductanța bobinelor de reactanță.

Dezavantajele acestei soluții sînt după cum urmează: curentul de circulație depinde de unghiul de amorsare; în regimul de lucru continuu componenta sa de c.c. este limitată numai de rezistența circuitului de c.c., conținînd în primul rînd bobinele de reactanță. În cazul altor soluții există o dispunere specială de control a curentului de circulație.

Structura regulatorului de curent este relativ complicată, deoarece valoarea instantanee a curentului de circulație mai depinde și de sarcină. În plus, nu este simplu de măsurat curentul de circulație și de selectat redresorul, furnizînd numai curentul de circulație.

În cazul nostru a fost realizat un dispozitiv special de comandă al curentului de circulație și al motorului, pe baza componentelor simetrice ale curentului. Acțiunile de c.c. în patru cadrane constau din două redresoare identice (CR1 și CR2) cu bune proprietăți dinamice și este posibil să se utilizeze traductoare clasice de curent (vezi fig. 7). Semnalele de referință de curent I_{r1} și I_{r2} prescriu valorile medii ale curenților lor de ieșire I_1 și I_2 iar dispozitivele lor de comandă forțează acești curenți, independent de tensiunile lor de ieșire U_1 și U_2 . Um reprezintă tensiunea la borne a motorului; w reprezintă viteza unghiulară; $I_m = I_1 - I_2$ este indusul curentului motorului.

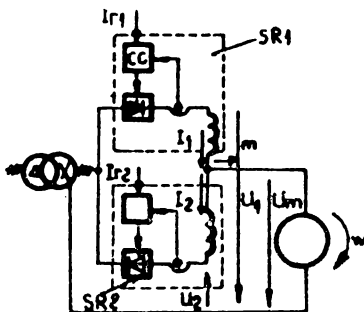


Fig. 7. Redresoare comandate.

Cercetările teoretice au arătat (1) că semnalul de referință I_{rp} al curentului în secvență pozitivă este proporțional cu suma semnalului de referință I_{rc} al curentului de circulație și valoarea absolută a curentului I_m al motorului în timp ce semnalul de referință I_{rn} al curentului în secvență negativă este proporțional cu semnalul de ieșire al regulatorului de viteză. Semnalele de

referință ale curentului celor două redresoare controlate pot fi compuse din semnale de referință ale curentului în secvență pozitivă și negativă prin intermediul unei transformări matriciale corespunzătoare, vezi fig. 8.

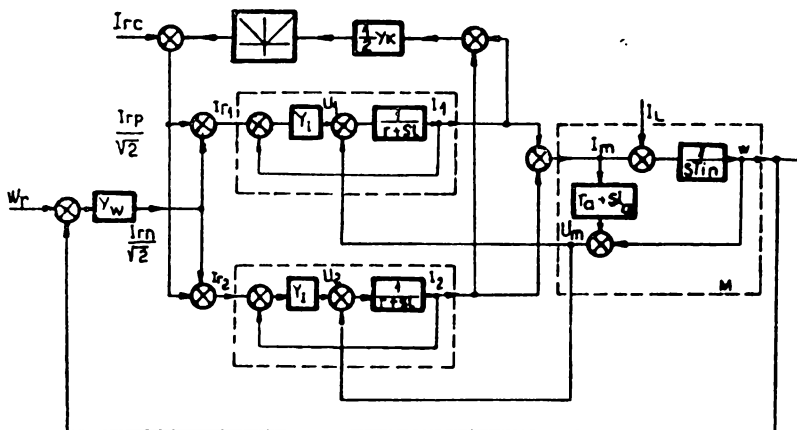


Fig. 8. Modelul dinamic al redresoarelor comandate.

Figura 8 prezintă modelul dinamic potrivit pentru calcularea aproximativă a proceselor tranzitorii din montajul nostru de acționare în curent de circulație.

Figura 8 prezintă influența reciprocă a redresoarelor comandate ce lucrează asupra aceeași sarcini. În realitate sistemul este nelinear datorită convertizorului cu tiristoare acesta avînd de exemplu un timp mort și un conținut ridicat de armonice. Totuși, cînd circuitul de comandă este proiectat, nu este necesar să se efectueze un calcul exact, de vreme ce o metodă simplificată bazată pe modelul dinamic din fig. 8 ce calculează valorile medii ale variabilelor oferă rezultate de o precizie acceptabilă.

Funcția de transfer Y_1 pentru bucla care conține regulatorul de curent, traductorul și blocul cu tiristoare este aproximativă și nu ia în considerare — după cum s-a menționat anterior — nelinearitățile determinate de blocul cu tiristoare. Schema bloc este elaborată eliminînd funcțiile de transfer din reacțiile inverse. Funcțiile de transfer ale bobinelor de reacțianță ce limitează curentul de circulație sînt redată prin raportul: $\frac{1}{r+sL}$. Litera M simbolizează motorul; unde: r_a , L_m sînt rezistența și respectiv inductanța motorului; T_{in} reprezintă timpul său nominal de pornire. Efectul de linearizare este exprimat prin funcțiile de transfer Y_k ; iar I_L simbolizează efectul sarcinii.

După o oarecare modificare a fig. 8 obținem schema bloc transformată din fig. 9, care este utilă pentru a proiecta dispozitivele de control și pentru a analiza limitele de stabilitate și comportamentul dinamic al sistemului de acționare. Porțiunea reprezentată de secvența pozitivă și negativă a circuitului de comandă a fost separată în fig. 9.

Blocul comun D din fig. 9 reprezintă un factor de transfer de ordinul 2 în bucla secvenței negative.

Din cele de mai sus rezultă că secțiunea din schema bloc ce prescrie curentul de circulație, nu este o buclă închisă, deși există o reacție inversă reprezentată de semnalul secvenței pozitive. Parametrii și structura dispozitivelor de control ce asigură comportamentul dinamic adecvat pot fi determinați prin intermediul ecuației indicate pe schema bloc din fig. 9.

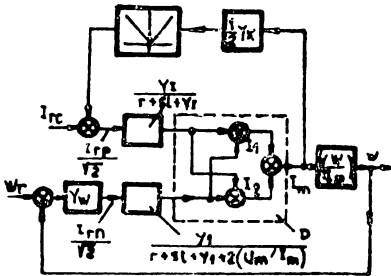


Fig. 9. Schemă bloc transformată pentru calcul.

Oscilogramele indică procesele tranzitorii ale acționării în patru cadrane, realizate pe baza principiului descris. Modificarea curentului motorului I_m și a vitezei motorului n sînt indicate în fig. 10, în timpul unui proces de inversare a vitezei la o sarcină nominală.

Pentru același proces, fig. 11 prezintă curentul celor două redresoare I_p și I_n . Curentul mai ridicat este totdeauna curentul motorului, iar cel mai scăzut este curentul de circulație.

Se poate vedea din oscilogramă că, curentul de circulație este practic constant chiar pe perioada unei stări tranzitorii.

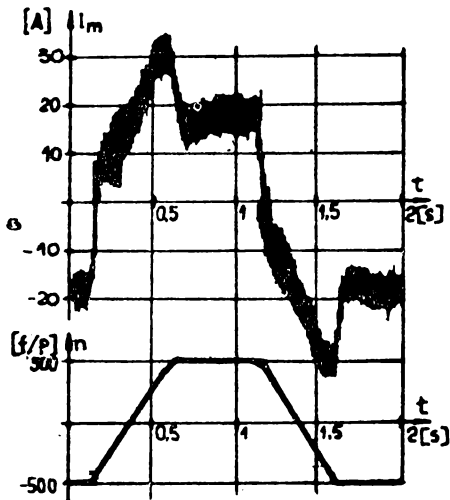


Fig. 10. Proces invers al acționării comandate în curent continuu.

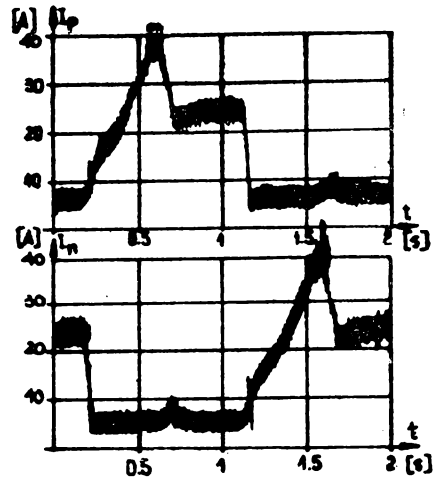


Fig. 11. Formarea curentului motorului și a curentului de circulație.

Sistemul de acționare în curent continuu cu curent de circulație, asigurând funcționarea în patru cadrane cu trecere continuă prin zero a curentului, include două dispozitive de comandă (SC-C, viteză și respectiv curent), vezi fig. 6. Redresorul comandat (CR) furnizează energie servomotorului (M) de c.c. cu magnet permanent, cu tahogenerator (Tg) încastrat. Frâna electromagnetică (MB) și traductorul de poziție (PS) sînt conectați la arborele motorului printr-o cuplă mecanică (MC). Toate dispozitivele de acționare sînt echipate cu întrerupătoare de contact la pozițiile finale geometrice ale mișcărilor. Funcția unității de interfață (IF) a acționării în c.a. este similară acționării în c.c.; normal, nu există nici o unitate de comandă (SC-c) și nici un redresor comandat (CR), în schimb, există o pereche de întrerupătoare trifazice pentru modificarea secvenței fazelor.

După comutarea liniilor de alimentare de la rețea ale unei anumite acționări, unitatea sa de detectare a erorilor verifică;

- parametrii principali ai rețelei trifazice;
- cele mai importante caracteristici ale dispozitivului electronic de acționare;
- limitări asupra mișcării mașinii de încărcare-descărcare a combustibilului.

Dacă nu se detectează defecte, atunci mișcarea dorită este permisă și se va trimite un mesaj către blocul de comandă (CU) cu privire la starea acționării selectate. După aceasta, operarea acționării este posibilă atât în regim automat, cît și manual. În cazul operării de regim automat, comenzile de la blocul de comandă (CU) vor fi verificate de unitatea de interfață (IF).

Combinarea incorectă a comenzilor către dispozitivele de comandă ale acționării (SC-C) va fi blocată de (IF). În cazul unei operări normale (lipsită de greșeli) a dispozitivelor de acționare, unitatea de interfață (IF) furnizează informații de stare de la dispozitivele de acționare la blocul de comandă (CU). Sistemul de comandă al mașinii de încărcare-descărcare a combustibilului, poate permite sau interzice funcționarea dispozitivelor de acționare pe baza comparării comenzilor de la blocul de comandă (CU) cu informațiile de stare de la sistemul de acționare (DS).

De îndată ce se detectează o eroare în regimul de funcționare, mișcarea va fi imediat oprită și se va genera un mesaj de eroare la sistemul de comandă.

Mesajul de eroare va apare pe display-ul consolei operator (OC). În acest caz operatorul sau dezanatorul trebuie să se ducă la camera de comandă, pentru a detecta și înlătura defecțiunea din dulapul de comandă. În cazul apariției unei defecțiuni de orice natură și de orice amploare se va aprinde LED-ul roșu al lămpii de reanclanșare, iar funcționarea acționării selectate poate fi reluată numai dacă defecțiunea a fost înlăturată prin apăsarea butonului de reanclanșare.

Concluzii

Sistemul de comandă are o fiabilitate și disponibilitate ridicată. Costurile moderate se datorează utilizării exclusive a aceluiași tip de microcalculator și faptul că sînt necesare numai două microcalculatoare pentru fiecare subsistem. Acționările în c.c. în patru cadrane, cu comanda curentului de circulație asigură un comportament dinamic de calitate.

BIBLIOGRAFIE

- (1) *Brausz I., J. Borka și A. Varga* (1982). Comanda curentului de circulație prin componente simetrice ale acționărilor în curent continuu în patru cadrane. ICEM 1982, Budapesta, 141—144.

TENDINȚE ÎN EDUCAȚIA ÎN AUTOMATIZĂRI

Z. Benyó*

Universitatea Tehnică din
Budapesta (R. P. Ungară)

Ing. Marcel Sirbu
I.P.A.

Dr. ing. Adrian Davidoviciu
I.T.C.I.

I) EDUCAȚIA ÎN R.P. CHINEZĂ, AUSTRIA, S.U.A., JAPONIA, EUROPA DE VEST

Sesiunea 07.1 de la Congresul IFAC a dezbătut prin referatele prezentate în cele ce urmează chiar tendințele în educația în automatizări.

Lista referatelor prezentate a fost:

1. Stabilizarea unui pendul dublu inverte prin regulator analogic; autori: Z. Feng, Z. Yin, H. Chen (R.P. Chineză);
2. Un experiment de laborator controlat de calculator; autori: C. Wang (R.P. Chineză), R. Henry, R. Cameron, S. Mossaheb (Anglia);
3. Educația în reglarea automată în Austria; autori: P. Kopacek, R. Genser, I. Troch, A. Weinmann (Austria);
4. Tendințe educaționale în automatizări în SUA; autori: A. Desrochers, G. Saridis (SUA);
5. Tendințe în educația în automatizări în Japonia; autori: H. Akashi (Japonia);
6. Orizonturi în educația inginerescă — o prezentare sistemică a situației în Europa de vest; autori: G. Battersby, D. Jenkins (Anglia), W. Schaufelberger (Elveția).

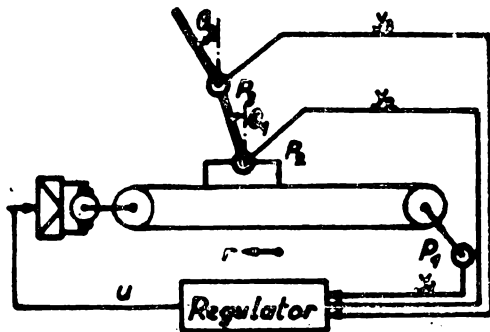
Referatul 07.1—1 prezintă implementarea unui sistem de reglare cu un regulator analogic, atât parțial observabil, cât și linear funcțional observabil. Sistemul prezentat este foarte stabil la perturbații mari și la variații ale parametrilor și este foarte indicat pentru experimentele de educație în teoria modernă a reglării, suscitând un interes major din partea studenților.

Autorii pornesc de la ideea posibilității utilizării unui calculator analogic pentru rezolvarea problemei și susțin că, dacă această întrebare capătă un răspuns afirmativ, atunci costul regulatorului scade simțitor și scopul aplicării teoriei moderne a reglării devine evident. În completare ei enunță o altă idee interesantă și de mare valoare practică și anume că, în aplicarea practică a teoriei moderne a reglării, nu este necesară o mare precizie de calcul sau cerințe stricte de implementare pentru regulator, dar este importantă înțelegerea problemei, eventual cercetarea și analiza atentă a sistemului practic, pentru obținerea unei descrieri corespunzătoare a regulatorului.

Descriptiv, sistemul pendul (Fig. 1) este un sistem nelinier, instabil și complicat. Se presupune linearizarea în jurul punctului de echilibru instabil și atunci el poate fi reprezentat printr-un model linear de ordinul șase, astfel:

$$\dot{x} = Ax + bu, \quad y = Cx \quad (1)$$

Fig. 1. Diagrama schematică a pendulului dublu inverte.



* Lucrarea autorului (secțiunea II) mai amplă în original, este restrinsă la informațiile necesare cititorului seriei AMC.

unde :

$$x = (r, 01, 02 - 01, \dot{r}, \dot{0}1, \dot{0}2 - \dot{0}1)^T$$

$$y = (y1, y2, y3)^T = (r, 01, 02 - 01)^T$$

$$A = \begin{pmatrix} 0 & I \\ A_{21} & A_{22} \end{pmatrix} =$$

$$= \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & -1,982 & 0,09565 & -9,550 & 0,09817 & -0,036 \\ 0 & 26,871 & 13,85000 & 21,76 & -0,1602 & 0,13135 \\ 0 & -31,19 & 57,847 & -25,27 & 0,35027 & -0,4292 \end{pmatrix}$$

$$b = (0, b_2)^T = (0, 0, 0, 6.1836, -14.096, 16.361)^T$$

$$C = (I, 0)$$

și u este tensiunea de intrare a amplificatorului.

Observabilitatea de ordin minim pentru sistemul descris de (1) este dată de :

$$\dot{z} = Fz + Gy + hu \quad \hat{\dot{x}} = Dz + Ey \quad (2)$$

unde :

\hat{x} — este estimatorul lui x ;

$$F = A_{22} - L ; G = -FL + A_{21} ; h = b_2 ;$$

$$D = (0, I)^T ; E = (I, -L)^T.$$

În situația concretă L este astfel determinat încât valorile proprii ale matricei F sînt :

$$(-10.018, -17.463 \pm j11.890)$$

Structura observabilității este prezentată în fig. 2.

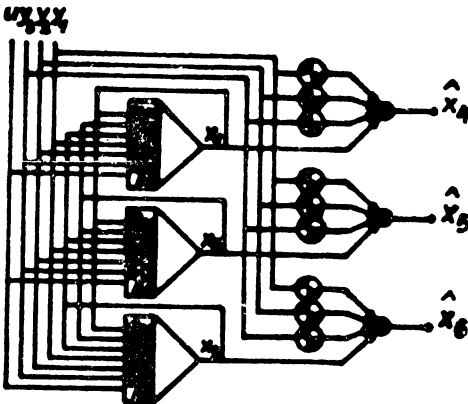


Fig. 2. Structura observabilității.

Pentru descrierea reacției inverse se utilizează teoria reglării liniare optime și deoarece (1) este complet controlabil, pentru un timp inițial de reglare, funcția de comandă u^* care minimizează indicele de performanță

$$J_2 = \int_0^{\infty} (x^T Q x + r u^2) dt \quad (3)$$

este dată de reacția inversă a stării liniare

$$u^* = K \cdot x^*$$

Pentru ușurința practică s-a preferat o reacție inversă liniară invariantă în timp ;

$$u_1 = K_1 x$$

unde :

$$K_1 = -(1.6785, 25.437, 81.513, 1.8432, 13.298, 12.598)$$

Experimentul a fost încorporat în programa de instruire și a dat bune rezultate, studenții executând următoarele sarcini :

1. Formularea modelului liniar care reprezintă sistemul.
 2. Analiza controlabilității, observabilității, stabilității etc., utilizând pachetul de programe CAD pe minicalculatorul HP-1000.
 3. Descrierea regulatorului, care constă din observabilitate și legea de reglare a reacției, prin utilizarea pachetului CAD.
 4. Simularea sistemului de reglare prin integrare numerică.
 5. Implementarea regulatorului prin intermediul circuitelor electrice și experimentarea lui.
- Activitățile 1—4 au constituit, de asemenea, diverse preexperimente și proiecte. Referatul 07.1—2 descrie un experiment de laborator bazat pe măsurători și comenzi executate de un microprocesor specializat.

Suportul procesului studiat a fost o conductă de cupru, având la cele 2 extremități câte o bobină de încălzire (Fig. 3) și implantate de-a lungul conductei 4 traductoare de temperatură. Acest suport permite experimente multiple pentru înțelegerea proceselor cu parametri distribuți.

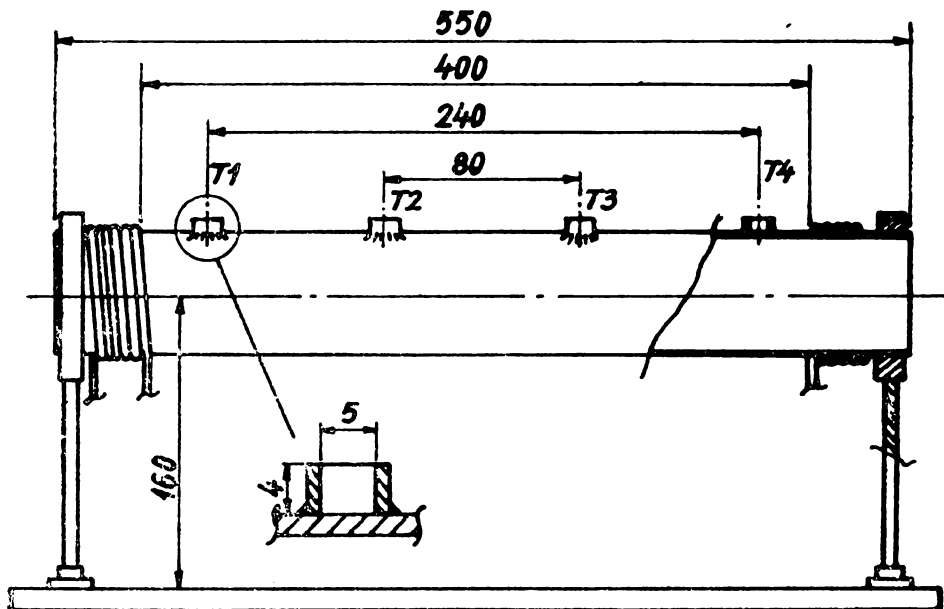


Fig. 3. Structura conductei.

Calculatorul utilizat a fost de tipul NASCOM 2, având la bază un microprocesor Z-80A. Configurația dispune de un monitor TV și o tastatură, precum și de o interfață serială cuplată la o unitate de casetă magnetică. Spațiul de memorie este de 4 K RAM și 4 K EPROM. Sistemul dispune de un microsoft BASIC și de un monitor de 2 K.

Software-ul de interfețe este ars în EPROM și este permanent la dispoziția utilizatorului, care realizează programarea într-un limbaj de nivel superior.

Din punctul de vedere al interfețelor, sistemul dispune de 4 intrări în bucle de curent de la traductoarele de temperatură și de 2 ieșiri care comandă tiristoare. Fiecare canal de intrare dispune de un convertor A/D ZN427E. Toate cele 4 conversii se execută simultan, ieșirile celor 8 biți fiind conectate împreună într-un BUS cuplat la portul A al PIO, ceea ce reprezintă o multiplexare a semnalelor numerice și nu a celor analogice.

Circuitele de ieșire generează întârzieri în gama 0—10 ms, utilizând componente counter-timpe (CTC) din familia Z-80. Fiecare tiristor necesită 2 canale CTC și, deoarece CTC-ul dispune de 4 canale, un CTC poate comanda ambele tiristoare. Timpul de eșantionare este generat de un al doilea CTC. Constantele de timp ale echipamentului sînt în gama 14—28 minute, iar perioada de eșantionare a fost aleasă de 15 secunde (Fig. 4).

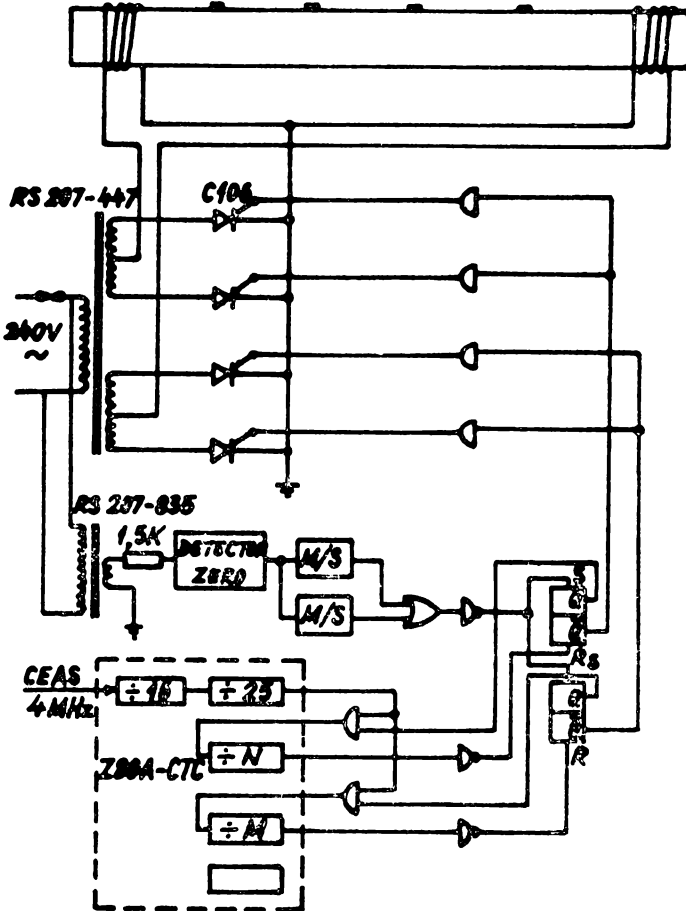


Fig. 4. Interfața de ieșire cu CTC pentru comanda tiristoarelor

Echipamentul se poate utiliza pentru ilustrarea multor aspecte privind descrierea și implementarea sistemelor cu mai multe variabile de comandă. Înaintea exercițiilor de descriere trebuie ținut cont de câteva restricții impuse de echipament.

- (i) Numai câteva profiluri de temperatură pot fi accesibile;

(ii) Variabilele de comandă au caracteristici neliniare, deoarece numai încălzirea poate fi controlată, în timp ce răciră este determinată de construcția sistemului, ceea ce implică un grad de interacțiune în sistem, care nu poate fi eliminat;

(iii) Modelul utilizat este aproximativ, este un model liniar, model cu parametri distribuți ai unui sistem neliniar.

În referat se prezintă descrierea locului caracteristic (MacFarlane și Kouvaritakis, 1977). Regulatorul rezultat a constat dintr-un compensator de înaltă frecvență și un regulator comutativ de aproximare încorporând acțiuni proporționale și integrale. Rezultatele sînt prezentate în Fig. 5 și 6.

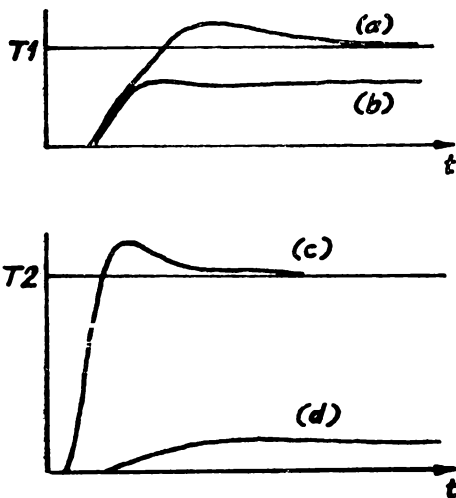


Fig. 5. Răspunsurile la treaptă :
Treaptă la intrarea 1, — (a) și (d),
Treaptă la intrarea 2 — (b) și (c).

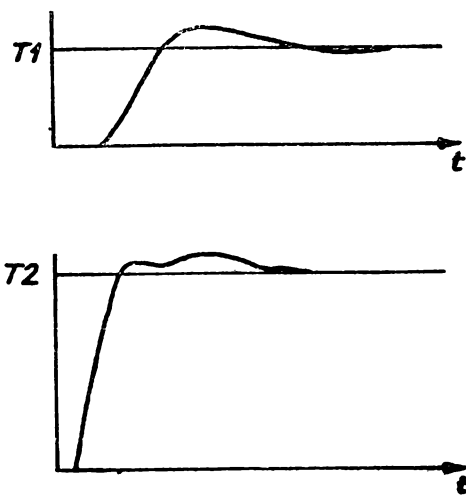


Fig. 6. Răspunsurile la trepte simultane.

Se pot utiliza și alte metode de descriere, cum ar fi metoda domeniului invers Nyquist (Rosenbrock, 1975) sau metoda multivariabilelor Ziegler-Nichols (Kouvaritakis și Klefouri, 1980).

În anul universitar 1983/1984 echipamentul a fost utilizat ca suport de curs.

În referatul 07.1—3, autorii prezintă sistemul educațional în Austria, mai ales în domeniul realizării automate, necesitățile și posibilitățile aplicării practice, precum și sugestii pentru îmbunătățiri la toate nivelele și pentru o legătură mai strînsă între industrie și institutele educaționale.

Introdutiv se prezintă datele generale ale sistemului educațional austriac sumarizate în Fig. 7. Cîteva dintre caracteristicile sistemului sînt :

— Copiii frecventează școala între vîrsta 6—15 ani, urmînd în acest interval școala primară (4 ani) și școala principală sau secundară (4 ani), aceste școli avînd mai mult sau mai puțin aceleași scopuri educaționale, diferențele fiind în raport cu metodele sau nivelul de pregătire obținut, după care viitoarea specializare se face în conformitate cu cîmpul de interes personal.

— Ca rezultat se obțin trei nivele de pregătire. La nivelul cel mai scăzut (I în Fig. 7) se găsesc școlile vocaționale cu pregătire de personal pentru industrie. La nivelul mediu (II în Fig. 7) se găsesc numeroase specializări corespunzătoare diverselor cîmpuri ale ingineriei, cursurile acestor școli, ca și ale școlii generale secundare, încheindu-se cu examen de maturitate. La nivelul superior (III în Fig. 7) se găsește universitatea tehnică, care dă posibilitatea obținerii gradului de „inginer diplomat”, sau după studii adiționale a celui de „doctor în științe tehnice”.

- Nivelul II poate fi atins și la colegiile tehnice (4—5 semestre), care sînt momentan în stadiul de proiect pilot. Următoarea parte a referatului prezintă situația în domeniul reglării automate pe suportul sistemului educațional. Astfel :
- În școlile vocaționale nu se predau nici un fel de noțiuni de reglare automată. Numai în ultimii ani ai școlilor generale secundare se predau noțiuni fundamentale de prelucrare electronică a datelor, dar aceste cursuri sînt opționale.
- La nivelul mediu, intensitatea educației în reglarea automată depinde de timpul de studiu, astfel încît pentru ingineria în electronică și comunicații, timpul afectat însumează 4 ore/săptămîină pe durata unui an, iar pentru ingineria mecanică, chimică și electrotehnică, timpul afectat însumează 2 ore/săptămîină pe durata unui an. Recent, împreună cu Departamentul de inginerie electrotehnică, s-a stabilit o secțiune specifică de reglare automată la una din școlile tehnice industriale, care va pregăti în cîțiva ani primii ingineri în reglarea automată pentru industria austriacă. Totodată, la două din aceleași școli, s-a stabilit un program de instruire de 20 ore/săptămîină pe durata unui an în domeniul microcalculatoarelor și limbajelor de programare, participanții obținînd certificate oficiale.
- La nivelul universitar, facultățile de inginerie electrotehnică și inginerie mecanică oferă posibilitatea studiilor în domeniul reglării automate. Cursurile aferente acestui domeniu se întind pe durata a 3 semestre, au în medie afectate 5—10 ore/săptămîină și cuprind ingineria reglării, construcția și descrierea reguletoarelor, tehnici de măsurare, laboratoare și seminarii, susținerea tezei de diplomă. Totodată, există cursuri suplimentare referitoare la reglare utilizînd microcalculatoare, sistemele multivariabile și identificarea proceselor. La facultățile de științe naturale există cursuri de 2 ore/săptămîină pentru matematică în teoria reglării, teoria stabilității sistemelor, teoria optimizării și teoria sistemelor multivariabile.

În ultima parte a referatului se prezintă cererile industriei în acest domeniu și sugestii pentru activitatea viitoare în domeniul educației referitoare la reglarea automată.

Cererile industriei se referă în mod expres la o educație pentru aplicarea calculatoarelor în automatizări și aceasta deoarece, inginerii electrotehniști trebuie să aibă multe cunoștințe referitoare la construcția echipamentelor și problemele mecanice ale tehnologiilor avansate, absolvenții ingineriei mecanice trebuie să aibă cunoștințe de bază despre electronică și automatizări, iar inginerii chimiști nu au experiență în controlul și prelucrarea datelor, cu toate că ei utilizează curent procedee de estimare și prelucrare statistică a datelor.

Conceptul pentru viitorul educației în acest domeniu este prezentat în Fig. 8 și se referă la 4 nivele de educație :

- Mecanici pentru dispozitive de măsurare și reglare ;
- Tehnicienii pentru măsurători și reglări ;
- Ingineri în domeniul reglării ;
- Ingineri diplomați în domeniul reglării.

Aceste nivele vor putea fi atinse prin :

— Cursuri despre dispozitive de reglare pentru specialiști în fabricația aparatului care trebuie să cuprindă reglarea temperaturii, sisteme de ventilare și condiționare a aerului, vane de reglare, dispozitive de reglare pneumatice, reguletoare programabile, reguletoare cu microcalculatoare, dispozitive de manipulare și roboți industriali.

— Cursuri avansate de ingineria reglării, pentru absolvenții școlilor tehnice certificate, despre comandă nu nerică, sisteme de reglare multivariabile, procese dinamice și modele, microcalculatoare în reglare, dispozitive de manipulare și roboți industriali, montaj automat, reglarea proceselor chimice.

— Lucrul la un proiect concret în domeniul reglării pentru o întreprindere sau un centru de cercetări, pe durata a 6—8 săptămîni, în grupuri de 4—5 persoane.

În referatul 07.1—4, „Tendințe educaționale în automatizări în SUA“, autorii prezintă câteva eforturi recente în elaborarea programelor pentru ingineria sistemelor de producție, în câteva universități mari din SUA și susțin că aceasta este una din căile de cooperare între universitățile publice și industria privată.

În introducerea autorii pornesc de la faptul că din 1960 pînă în 1970, față de producția unui anumit sector, productivitatea muncii a fost în scădere și subliniază că una din cauze este existența unei conduceri tehnice nepregătite pentru producție. În sprijinul acestui fapt ei aduc argumentul că, în prezent, din 1216 programe acreditate de inginerie în SUA, numai 3 sînt programe de inginerie de producție.

O soluție este modificarea pregătirii educaționale, mai ales în domeniile în care tehnologia automatizării poate aduce un efect important privind creșterea productivității muncii (Tabelul 1).

Necesitățile pentru noile programe de ingineria producției trebuie să pornească de la schimbările rapide în tehnologia calculatoarelor, microprocesoarelor, tractoarelor, roboților și inteligenței artificiale. Aceste schimbări rapide vor crea un nou profil de inginer de producție, astfel încât dezvoltarea și individualizarea funcțiilor în cadrul acestor evenimente trebuie să implice o aprofundare, o analiză sofisticată și o prelucrare atentă, pentru înțelegerea implicațiilor intrinseci ale sistemelor de producție.

Următoarea secțiune a referatului se ocupă de concepția existentă referitoare la educația pentru producție și automatizări în SUA. Astfel, este descris fiecare din cele 5 programe MSMSE (Master of Science in Manufacturing Systems Engineering).

Programele universitare relative la robotică, automatizări și producție au fost implementate de câțiva ani la Universitatea din Stanford, M.I.T., Universitatea Mellon-Carnegie, Universitatea Purdue, Universitatea din Michigan, Universitatea din Wisconsin și Institutul Politehnic Rensselaer. Este de notat că fiecare program individual nu satisface complet liniile directoare pentru programele de ingineria producției, stabilite de Societatea Inginerilor de producție din SUA (Un program de nivel ridicat, care să cuprindă practică, teze, cercetări și proiecte speciale, materializat prin 1—3 ani descriptivi și 1—3 ani adiționali de cursuri de nivel ridicat. Domeniile ingineriei de producție vor cuprinde materiale și prelucrări, ingineria și eficiența producției și sisteme de producție și control prin calculatoare).

Direcțiile existente în educația în automatizări în SUA sînt sintetizate prin 3 categorii de programe educaționale: programe care oferă gradații în robotică și automatizări, programe care oferă studii în robotică și automatizări, programe care oferă suport de studii pentru robotică și automatizări.

În continuare referatul descrie noile programe de ingineria sistemelor de producție de la Institutul Politehnic Rensselaer, Universitatea din Stanford, Institutul de Tehnologie Georgia, Universitatea din Wisconsin—Madison, Universitatea Lehigh.

Din descrierea acestor noi programe se pot identifica următoarele tendințe:

— Noile programe sînt interdisciplinare, dar nu necesită crearea de noi departamente de inginerie.

— Aceste programe sînt conduse de industrie. Ele reprezintă eforturile de cooperare între universități și industrie, care este un fapt relativ nou pentru SUA.

— Noile cercetări pun accentul pe partea practică a pregătirii, în ideea trecerii de la un inginer de producție cu statut de artizan, la un inginer de producție orientat larg către producție.

— Un efort continuu și clar se întreprinde pentru stringerea legăturilor între industrie și universități, prin programe de schimb de vizite reciproc, un fapt care este, de asemenea, nou pentru SUA.

— Ingineria sistemelor de producție a fost identificată ca existentă între inginerie și conducere.

— Aceste programe sînt labile și tind să ducă la un răspuns rapid privind realinierea conceptelor de proiectare și producție.

Tabelul 1

Programe de creștere a productivității muncii și eficiența lor

Tipul de program	Eficiența raportată a programelor		
	Ridicată	Medie	Scăzută
Investiții de capital sau automatizări (fără roboți)	48 %	45 %	8 %
Sisteme inovatoare (unități integrate, tehnici de producere de materiale)	37 %	49 %	14 %
Programe de stimulare a salariilor	35 %	41 %	24 %
Introducerea și extinderea utilizării roboticii	32 %	34 %	34 %
Introducerea sau dezvoltarea metodelor de control	30 %	53 %	18 %

În referatul 07.1—5, „Tendințe în educația în automatizări în Japonia“, autorul prezintă rezultatul unui sondaj de opinie, împreună cu un raport referitor la activitatea citorva laboratoare. Concluzia este că, unitar, situația generală pare cam conservatoare, dar activitatea este foarte bine organizată.

În partea introductivă se prezintă sistemul învățămîntului superior, pe baza citorva statistici comparative cu SUA, prezentate în Tabelul 2. Următoarele elemente sînt caracteristice:

— Proporția între numărul universităților și colegiilor pe întreaga populație este de 1 universitate la 116 500 oameni în Japonia, în timp ce în SUA este de 1 universitate la 67 909 oameni.

— Numărul studenților admiși la facultate în Japonia este aproximativ egal cu numărul studenților care o absolvă (aprox. 80 %).

— Numărul studenților în inginerie este aproximativ $1/5$ din total. Dintre aceștia $1/20$ sînt înscriși pentru titlul Master și $1/200$ pentru titlul Doctor.

— Valoarea cheltuielilor în învățămînt a sporit mult în Japonia începînd cu 1975 (Tabelul 3).

O deosebire principală apare la sistemul de admitere în învățămîntul superior, unde, spre deosebire de universitățile americane și europene, se organizează practic două concursuri de admitere pentru același candidat. Totodată, datorită acestui sistem de admitere, este mai ușor să intri în facultate decît să o termini, deoarece mulți din cei care au trecut grelele examene de admitere consideră că munca lor a luat sfîrșit.

În continuare referatul prezintă sistemul educațional în automatizări. Se observă că deși ingineria automatizărilor este relativ recentă în învățămîntul universitar, ea a fost predată în multe universități, sub diverse aspecte, începînd cu sfîrșitul ultimului război mondial.

În cazul Universității din Kyoto, studiile speciale de sisteme de reglare sînt întotdeauna popularizate și adeseori studenții de anul 4 concurează în asociere cu laboratoarele de inginerie pentru automatizări, pentru susținerea tezei de diplomă.

Numărul de ore pentru experimentări este foarte mare (Tabelul 4) și se pare că multe departamente folosesc calculatoare analogice și servomecanisme. Totuși, numărul microcalculatoarelor utilizate este în creștere rapidă.

Subiectele predate și numărul orelor afectate pentru automatizări sînt prezentate în tabelul 5.

În ceea ce privește utilizarea microcalculatoarelor din cele 115 departamente interogate, 101 utilizează microcomputere de diverse tipuri și capacități (Tabelul 6). Majoritatea acestor microcomputere se utilizează pentru măsurători, prelucrări de date, analiză numerică, simulare și ca terminale pentru calculatoarele puternice. Șase departamente utilizează aceste microcalculatoare în comanda roboților.

Din cele prezentate rezultă cîteva trăsături esențiale:

— Predarea se face în domeniile tradiționale (mecanică și electrotehnică) și constă în majoritate în teorie fundamentală.

— Avînd în vedere indecizia între mai multă teorie sau mai multă practică și opțiunile unora pe partea neconvențională se ajunge la concluzia unei necesități a unirii teoriei cu practica,

— Lipsa de fonduri bugetare și deci prea puține ore de predare, precum și lipsa de cadre competente duc la o ineficiență a preștirii, la un gol între teorie și problemele actuale.

În finalul referatului, autorul prezintă tendințele în educația în automatizări pe baza unor cazuri specifice. Concluziile trase s-ar putea rezuma la:

— Extinderea teoriei controlului și reglării prin hardware. Aceasta poate avea implicații directe asupra sistemelor CAD și în special asupra elaborării de software în timp real.

— Necesitatea predării teoriei moderne a reglării chiar de la început, dacă se are în vedere că metodologia de descriere lipsește din teoria clasică a reglării.

— Necesitatea extinderii utilizării microcalculatoarelor pentru studiul reglării automate pentru calcul. De asemenea, teoria modernă a reglării reclamă neapărat utilizarea microcalculatoarelor (rezolvarea matricială a ecuațiilor Ricatti, problemele legate de sistemele discrete, răspunsul treaptă, impuls, diagrama Bode, locul rădăcinilor necesită un software adecvat modelele stochastice sîntelizate ușor, producerea numerelor aleatoare după o distribuție statistică, filtrele Kalman).

— Utilizarea cît mai intensă a microcalculatoarelor în robotică.

În referatul 07.1—6, „Orizonturi în educația inginerescă — o prezentare sistemică a situației în Europa de vest”, autorii utilizează metodele ingineriei sistemelor pentru a investiga sistemul educațional ingineresc și a-i deduce orizonturile.

Introducerea pornește de la aprecierea că, pe baza unor cercetări ale unui grup de lucru SEFI (Société Européenne de la Formation des Ingénieurs), sistemul de educație tehnică în Europa de vest este într-o stare de fapt periculoasă. Autorii utilizează metodele teoriei sistemelor pentru ca punctele subiective de vedere să devină mai obiective.

Sistemul este prezentat în Fig. 9 și se caracterizează prin :

— Sistemul a fost conceput pentru a fi ușor urmat de ingineri și nu de mecanici sau tehnicieni. Sistemul a fost conceput în afara științei și suferă din această cauză.

— Natura ieșirilor sistemului este în principal determinată de sistem însuși, el nerăspunzând presiunilor din afară.

— Presiunile interne sînt concentrate spre introducerea rezultatelor cercetărilor în știință și inginerie.

— Sistemul conține propriile sale metode de studiu.

— Sistemul nu are o preocupare particulară pentru lărgirea orizontului, se autoperpetuează, produce personal pentru educație și în consecință, are un orizont închis.

— Scala timpului pentru învățare este relativ lungă, ciclul de dezvoltare al sistemului este de durată.

— Adeseori se spune că răspunde la feedback, dar timpul de răspuns este așa de mare, încît el tinde să fie un sistem foarte stabil.

— Este foarte puțin receptiv la schimbare, nu este un sistem auto-motivabil.

— Dacă sîntem de acord cu aceste puncte vom vedea clar clasificarea sistemului educațional, ca nu sistem, dar nu ca un sistem de succes.

Pentru îmbunătățirea sistemului, autorii propun două soluții, modelul elementelor organizaționale și procesul de rezolvare al problemei în șase pași.

Modelul elementelor organizaționale constă în stabilirea următoarelor :

— Intrări (resurse materiale)

resurse umane și fizice, decizii guvernamentale, legi, necesități, obiective, fonduri cursuri, programe, cercetări, metode de studiu și tehnologii

— Procese (cum se manifestă acestea)

cursuri și proiecte complete, servicii furnizate către utilizatori licențe, certificate personale, rezultatele cercetărilor, publicații.

— Produse (rezultate implementate)

— Ieșiri (produse agregate ale unei organizații care sînt furnizate către societate)

În partea finală a referatului, autorii încearcă să prezinte tendințele de viitor prin prisma a două categorii de elemente.

Prima categorie de elemente constituie factorii de influență care sînt grupați după :

I. Necesitățile industriei ;

Efectele schimbărilor tehnologice în societate ;

Acceptarea de către societate ;

Reacții împotriva.

II. Evoluția însăși a ingineriei ;

Evoluția comunicațiilor și informaticii ;

Evoluția noilor tehnologii.

III. Efectele sociale :

Schimbările de cerințe asupra profesiei de inginer ;

Schimbările punctelor de vedere tradiționale asupra ingineriei.

A doua categorie de elemente o constituie direcțiile pentru viitor :

I. Cunoștințele profesionale ;

Tehnologia de lucru ;

Natura noilor cunoștințe și aplicarea lor.

II. Natura societății ;

Etica de lucru și modificarea ei ;

Echilibrarea între timpul de muncă și odihnă ;

Efectele cibernetizării și informatizării.

- III. Natura ingineriei;
 Cum va fi ea definită în viitor;
 Care sînt aspectele modificărilor inclusiv în sistemul de învățămînt.
- IV. Categoriile de ingineri în viitor;
 Ce cunoștințe acumulează;
 Ce munci li se vor da;
 Ce responsabilități vor avea;
 Punctul de vedere asupra profesionalismului la sfîrșitul secolului.
- V. Natura cunoștințelor cerute pentru aceste categorii de ingineri;
 Subiectele tradiționale opuse noilor subiecte;
 Modul în care ele vor fi furnizate în școală;
 Efectele conservării energiei, concentrării industriei;
 Atitudinea în școală.
- VI. Natura industriei pentru societate.

Concluzia referatului este că sistemul de educație inginerească trebuie schimbat, dar utilizînd metoda bottom-up, care necesită un timp mai scurt și modificări mai mici, față de metoda top-down.

II) CS.5 SISTEMUL EDUCAȚIONAL DIN R.P. UNGARIA ȘI TRANSFERUL SĂU PESTE HOTARE *

Sistemul educațional din R.P. Ungaria nu este prea bine cunoscut în lume. Prezenta expunere își propune să facă o prezentare generală a condițiilor în care se desfășoară educația în R.P. Ungaria și a sistemului în care sînt pregătiți inginerii. Se prezintă organizarea Universității Tehnice din Budapesta (cea mai mare instituție de învățămînt superior din R.P.U) precum și a Facultății de Electrotehnică. În continuare se fac considerații asupra exportului de sisteme complete de educație.

Ungaria are cca 11 milioane locuitori. Puterea creatoare a poporului este demonstrată de o gamă largă de realizări deosebite. Șase savanți de origine maghiară sînt laureați ai Premiului Nobel: Richard Zsigmond (chimie) în 1925, Albert Szentgyörgyi (medicină) în 1937, György Hevesi (chimie) în 1943, György Békési (medicină) în 1961, Jenő Wigner (fizică) în 1963 și Dénes Gábor (fizică) în 1971.

Dintre oamenii de știință maghiari de faimă mondială fac parte János Bolyai, Ignác Semmelweis, Loránd Ötvös, János Neumann, Todor Kálmán și Leo Szilárd. În lumea muzicii sînt faimoase lucrările lui Liszt, Lehár, Bartók, Kodaly și mulți alți compozitori.

Sistemul educațional din Ungaria cuprinde:

Învățămîntul obligatoriu, care începe la vîrsta de șase ani cu *Școli primare*, în care fiecare copil primește pînă la vîrsta de 14 ani, același tip de educație.

Urmează *școlile pentru pregătire profesională*, în care se pregătesc cadre pentru cca 255 de profesii, cu o durată de 3 sau uneori de 2 ani.

Școlile gimnaziale sînt tipul de școală cu cea mai veche tradiție.

După absolvirea cu succes a celor patru ani de școală gimnazială, elevii se pot prezenta la examenul de maturitate.

Școlile secundare de specialitate experimentale au început să funcționeze începînd cu anul 1959 și ca rezultat s-a născut o școală de tip nou.

Orice cetățean al R.P.U., pînă la vîrsta de 35 de ani poate solicita admiterea în învățămîntul superior, cu condiția absolvirii examenului de maturitate.

Universitatea tehnică din Budapesta își începe istoria în jurul anului 1782; denumirea actuală catcăz din anul 1949. Ea este cea mai mare universitate din țară și este singura care are o facultate de electrotehnică.

Întregul corp profesoral al universității numără cca 2000 de cadre didactice. Biblioteca Universității Tehnice dispune de aproape o jumătate de milion de volume de specialitate.

Universitatea dispune de un calculator electronic E32 de fabricație poloneză. În plus, fiecare facultate are un calculator propriu CDPA pe carez și un calculatcare SM-4 de fabricație

* Z. BENYÓ, Catedra de Automatică, Universitatea Tehnică din Budapesta Facultatea de Electrotehnică — R.P. Ungaria. (*Material adaptat*).

sovietică. Facultatea de Electrotehnică dispune și de un calculator electronic IBM 370/115 și mai multe minicalculatoare de tip PDP-11.

Universitatea Tehnică din Budapesta are Cca 8 000 de studenți.

Facultatea de Electrotehnică de la Universitatea Tehnică din Budapesta a fost înființată în anul 1959. Ca urmare a dezvoltării sale deosebit de dinamice, facultatea are în prezent cea 2 200 studenți și 350 cadre didactice.

Exportul R.P. Ungare în domeniul educației se bazează pe accelerarea diviziunii internaționale a muncii de după cel de al 2-lea război mondial, când schimbul de servicii și de experiență intelectuală, alături de schimburile de bunuri materiale, a devenit din ce în ce mai important.

Specialiștii noștri de înaltă calificare activează în țări în curs de dezvoltare din Africa, Asia și într-o oarecare măsură și în America de Sud. Mai recent, au crescut cerințele pentru experți în agricultură.

O altă formă de cooperare cu țările în curs de dezvoltare este exportul de clase complete pentru învățământ. Este vorba de clase, laboratoare, ateliere de învățământ complet echipate, livrarea echipamentelor necesare, punerea la dispoziție a materialului didactic și a suporturilor de curs, trimiterea de personal didactic și ajutor precum și pregătirea personalului didactic al țării importatoare, fie în Ungaria fie în țara respectivă.

Pentru a realiza o instruire adusă la zi, s-a înființat și un Institut pentru educație și metodologia educației. Ministerul Educației include instituțiile care elaborează programele analitice, cerințele, metodele și mijloacele necesare educației publice. Asemenea instituții sînt Institutul Pedagogic Național, Centrul Național pentru tehnici de instruire, întreprinderile ce produc material didactic, Institutul de artă populară etc.

Instruirea în domeniul agriculturii și industriei alimentare are o situație specială, ca depinzînd direct de Ministerul Agriculturii și Industriei Alimentare. Un Centru de instruire și de cercetare precum și un Institut metodologic și de studii post-universitare contribuie nemijlocit la această instruire și la stabilirea cerințelor.

Întreprinderile mai importante din R.P.U. care realizează export de facilități pentru instruire și expertiză profesională sînt : TESCO, KOMPLEX, TECHNOIMPEX, METRIMPEX, INDUSTRIALEXPOR, KÓZTI, LABOR MIM, AGTI. Aceste întreprinderi sînt toate întreprinderi de stat ; în consecință, produsele și serviciile contractate sînt garantate în întregime de către stat.

Fără a avea pretenția că lista este completă, iată cîteva din realizările noastre :

Țara	Denumire și tip proiect	Caracteristici
1	2	3
Libia	Institut pentru calificarea muncitorilor pentru 500 persoane	Pentru calificarea de specialiști în industria lemnului, livrare echipamente instalare și punere în funcțiune
Peru	Diferite laboratoare de instruire	Pentru fizică, metalografie analitică, mecanică electrochimică, biologie
Algeria	Universitatea de științe și tehnologie din Oran	Livrarea de echipamente pentru laboratoare și ateliere pentru Institutul de Arhitectură și de Inginerie Mecanică. Elaborarea de echipamente auxiliare pentru instruire, suporturi de curs în lb. franceză. Proiectare, conducere tehnică, asistență tehnică și servicii
Iran	Institut pentru calificarea muncitorilor	Livrarea și instalarea de echipamente pentru laboratoare și ateliere necesare predării de cunoștințe în domeniile industriei alimentare, chimice, metalurgie, electrotehnică
Nigeria	Politehnica Federală din Idah	Planificarea pedagogică ; pregătirea suporturilor de curs, îndrumările de practică ; livrarea de echipamente și instalarea lor pentru laboratoare și ateliere pentru facultățile de arhitectură, electrotehnică, mecanică și metalurgie

1	2	3
Nigeria	Politehnica Federală din Yola	Livrarea de echipamente pentru laboratoare și ateliere ; instruirea cadrelor didactice, documentare pedagogică, instruire în R.P.U., asistență tehnică în Nigeria
Nigeria	Institutul de calificare a muncitorilor, Colegiul Tehnic	Livrarea de laboratoare și ateliere necesare pentru instruirea în diferite profesii
Nigeria	Politehnica de Stat din Ogun	Livrarea de laboratoare, instruire în R.P.U. și asistență tehnică
Irak	Institute de calificare a muncitorilor pentru mecanizarea agriculturii	Proiect „la cheie” pentru clădire, livrare și instalare echipament didactic Pregătirea instructorilor în R.P.U.
Cuba	Diferite laboratoare de instruire	Livrare echipamente pentru laboratoare și ateliere universitare.
Bangladesh	Laborator de instruire	Livrare echipamente de laborator
Mexic	Laboratoare de instruire și de testări pentru industria alimentară	Livrarea și instalarea de laboratoare complete
Nigeria	Fabrică de echipamente de cercetare	Proiectare și realizare
Algeria	Școli secundare de specializare în hidrologie la : Banchegout Kaar — Chellala Biskra M' Sila	Proiectarea construcției, livrarea și instalarea de echipamente
Brazilia	Livrarea de echipamente și instrumente pentru universități	
Vietnam	Livrarea de echipamente pentru o școală de specializare a 1 000 de studenți, Hanoi	

BIBLIOGRAFIE

- Benyó, Z.* : Educational system of Hungary professional education, educational export in Hungary, *Komplex-Symposium*, Cairo 28—29 nov., 1982.
- Benyó, Z.* : Programmable teaching equipments. Experiences of professional training in Hungary. *Asean Didacta 83'*, Singapore 7—11 June, 1983.
- Cheng, D.K.* : Electrical Engineering Education in Hungary. *IEEE transaction on education*, Vol. E—17, No. 2, may 1974, pag. 92—99.
- Szabadváry, F.* : Budapest Technical University, 1782—1982 *Periodica polytechnica*, 1982.
- Vágó, I.* : Budapest Technical University, Faculty of Electrical Engineering, 1982.

SISTEME COMPLEXE ȘI MARI ; TEORIE, METODOLOGIE ȘI APLICAȚII

Ing. Laurențiu Orășanu
Dr. ing. Florin Gheorghe Filip
Ing. Melia Muratcea
I.T.C.I.

Lucrările grupate în cele 5 secțiuni ale *colocviului 11.1.* al Congresului 9 IFAC ilustrează -- prin numărul lor (29) și diversitatea problematicii abordate -- interesul constant al specialiștilor pentru găsirea unor noi abordări teoretice și pentru lărgirea gamei de aplicații în conducerea sistemelor complexe.

1. ANALIZA ȘI DESCOMPUNEREA SISTEMELOR COMPLEXE

În sensul teoriei introduse pentru prima dată în [1] și [2], *lucrarea 11.1/A1* prezintă două metode de măsurare exactă a complexității, una bazată pe informația conținută, alta pe aspecte de calcul, cu aplicații în prelucrarea datelor, teoria probabilității și teoria algoritmilor. Pentru a introduce noțiunea de complexitate (Kolmogorov) a informației se consideră datele și programele ca secvențe de 0 și 1 (pe scurt șiruri). Fie X un șir și $|X|$ lungimea sa. Se zice că un șir Y definește pe X dacă Y este un program (BASIC) care la execuție tipărește pe X . Complexitatea Kolmogorov $K(X)$ a șirului X este definită ca lungimea minimă a oricărui șir care definește pe X . Cel mai scurt program care definește pe X este codul Kolmogorov al lui X . Pentru alt program BASIC sau alt limbaj se obține o valoare $K'(X)$ diferită

$$|K(X) - K'(X)| < \text{const.} \quad (1)$$

în care constanta depinde numai de cele 2 limbaje, nu de X .

Complexitatea Kolmogorov măsoară conținutul în informație al șirului. Dacă un șir conține aproximativ $\alpha |X|$ de 1 ($0 < \alpha < 1$), atunci șirul are complexitatea informației cel mult $H(\alpha) |X|$, unde :

$$H(\alpha) = -\alpha \log_2 \alpha - (1-\alpha) \log_2 (1-\alpha) \quad (2)$$

În ceea ce privește complexitatea de calcul, se abordează aspectul timp de calcul. Pentru un program operind cu o intrare dată X se compară de obicei timpul de execuție cu lungimea $|X|$ a intrării. Astfel complexitatea de timp a unui algoritim este definită ca o funcție f , unde $f(n)$ este timpul de calcul maxim al unui algoritim pentru toate intrările de mărime n . Definind o problemă ca o mulțime de șiruri, lucrarea pune în evidență diferite clase de complexitate a problemelor (probleme fără algoritim — axiome matematice, probleme rezolvabile în timp exponențial, în timp polinomial, în timp linear).

În concluzie, lucrarea 11.1/A1 propune o teorie matematică a complexității care să poată fi utilizată în analiza și modelarea sistemelor complexe. Pentru fiecare model ar trebui studiată atent complexitatea datelor care constituie modelul și complexitatea de calcul a algoritmilor cu care va fi rezolvată problema.

Analiza stabilității sistemelor complexe (cazul discret) constituie obiectul *lucrării 11.1/A2*. Există două metode de analiză a stabilității sistemelor complexe bazate pe funcțiile Liapunov :

- 1° metoda funcției scalare Liapunov ;
- 2° metoda funcției vectoriale Liapunov.

Prima metodă prezintă avantajul că cîmtenii de stabilitate pot fi determinate foarte rapid, ceea ce o recomandă ca o metodă eficientă în rezolvarea problemelor tehnice. Metoda a fost frecvent utilizată [3], [4], [5] în diverse cazuri: sisteme invariante sau variante în timp, sisteme cu întârzieri, continue sau discrete. A doua metodă, ca și principiul de comparație (prezentat în lucrare), conduce la condiții mai „slabe” (adică un cîmteniu de stabilitate mai larg) decît în cazul utilizării primei metode. În lucrare se arată că proprietățile de stabilitate ale unui sistem complex de ordinul n pot fi obținute prin analiza stabilității unui sistem de comparație de ordinul r ($1 \leq r \leq n$). Acest principiu de comparație enunțat în [6] este extins în lucrarea menționată pentru cazul sistemelor discrete în timp.

În lucrarea 1.1/A3 este extins principiul de incluziune pentru conducerea descentralizată cu suprapunerea stărilor pentru sistemele interconectate (subsistemele au anumite intrări și ieșiri în comun). Cînd un sistem este compus din subsisteme interconectate, spațiul stărilor este extins într-un spațiu mai mare printr-o transformare lineară (singulară), astfel că sistemele apar ca disjuncte. Dacă extensia este realizată astfel încît sistemul extins include sistemul inițial, se pot analiza stabilitatea și optimalitatea sistemului extins utilizînd tehnicile standard de descompunere disjunctă. Legea de comandă proiectată în spațiul extins poate fi „comprimată” pentru implementarea în sistemul inițial.

Se consideră sistemele S și \tilde{S} :

$$S: \dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \quad \mathbf{y} = \mathbf{C}\mathbf{x} \quad (3)$$

$$\tilde{S}: \dot{\tilde{\mathbf{x}}} = \tilde{\mathbf{A}}\tilde{\mathbf{x}} + \tilde{\mathbf{B}}\tilde{\mathbf{u}} \quad \tilde{\mathbf{y}} = \tilde{\mathbf{C}}\tilde{\mathbf{x}} \quad (4)$$

și perechile de transformări lineare (singulare) între stările, intrările și ieșirile lor:

$$\begin{aligned} \tilde{\mathbf{x}} &= \mathbf{V}\mathbf{x} & \mathbf{x} &= \mathbf{U}\tilde{\mathbf{x}} & \mathbf{U}\mathbf{V} &= \mathbf{I}_n \\ \tilde{\mathbf{u}} &= \mathbf{R}\mathbf{u} & \mathbf{u} &= \mathbf{Q}\tilde{\mathbf{u}} & \mathbf{Q}\mathbf{R} &= \mathbf{I}_m \\ \tilde{\mathbf{y}} &= \mathbf{T}\mathbf{y} & \mathbf{y} &= \mathbf{S}\tilde{\mathbf{y}} & \mathbf{S}\mathbf{T} &= \mathbf{I}_p \end{aligned} \quad (5)$$

Principiul de incluziune extins: Se spune că sistemul \tilde{S} include sistemul S (sau S este inclus în \tilde{S}) dacă există perechile de matrici (\mathbf{U}, \mathbf{V}) și (\mathbf{R}, \mathbf{S}) astfel încît, pentru orice stare inițială \mathbf{x}_0 și orice intrare fixată $\mathbf{u}(t)$ a lui S , alegerea stării inițiale $\tilde{\mathbf{x}}_0$ și a intrării $\tilde{\mathbf{u}}(t)$ a lui \tilde{S} :

$$\begin{aligned} \tilde{\mathbf{x}}_0 &= \mathbf{V}\mathbf{x}_0 \\ \tilde{\mathbf{u}}(t) &= \mathbf{R}\mathbf{u}(t) \quad \text{pentru } t \geq 0 \end{aligned} \quad (6)$$

implică:

$$\begin{aligned} \mathbf{x}(t; \mathbf{x}_0, \mathbf{u}) &= \mathbf{U}\tilde{\mathbf{x}}(t; \tilde{\mathbf{x}}_0, \tilde{\mathbf{u}}) \\ \mathbf{y}[\mathbf{x}(t)] &= \mathbf{S}\tilde{\mathbf{y}}[\tilde{\mathbf{x}}(t)] \quad \text{pentru } t \geq 0 \end{aligned} \quad (7)$$

Condiția necesară și suficientă pentru incluziunea extinsă este:

$$\begin{aligned} \mathbf{A}^i &= \mathbf{U}\tilde{\mathbf{A}}^i\mathbf{V} & \mathbf{A}^i\mathbf{B} &= \mathbf{U}\tilde{\mathbf{A}}^i\tilde{\mathbf{B}}\mathbf{R} \\ \mathbf{C}\mathbf{A}^i &= \mathbf{S}\tilde{\mathbf{C}}\tilde{\mathbf{A}}^i\mathbf{V} & \mathbf{C}\mathbf{A}^i\mathbf{B} &= \mathbf{S}\tilde{\mathbf{C}}\tilde{\mathbf{A}}^i\tilde{\mathbf{B}}\mathbf{R} \quad i=0, 1, 2, \dots \end{aligned} \quad (8)$$

Cînd considerăm descompunerea cu suprapunere a lui S , dacă perechile de matrici de transformare (\mathbf{U}, \mathbf{V}) , (\mathbf{Q}, \mathbf{R}) , (\mathbf{S}, \mathbf{T}) sînt specificate, atunci matricile \mathbf{A} , \mathbf{B} , \mathbf{C} ale sistemului extins S sînt date de:

$$\tilde{\mathbf{A}} = \mathbf{V}\mathbf{A}\mathbf{U} + \mathbf{M} \quad \tilde{\mathbf{B}} = \mathbf{V}\mathbf{B}\mathbf{Q} + \mathbf{N} \quad \tilde{\mathbf{C}} = \mathbf{T}\mathbf{C}\mathbf{U} + \mathbf{L} \quad (9)$$

unde \mathbf{M} , \mathbf{N} , \mathbf{L} sînt matrici complementare de dimensiuni corespunzătoare. Alegerea matricilor \mathbf{M} , \mathbf{N} , \mathbf{L} astfel ca \tilde{S} să fie o extensie a lui S se poate realiza astfel:

$$\begin{aligned} \mathbf{U}\mathbf{M}^i\mathbf{V} &= \mathbf{0} & \mathbf{U}\mathbf{M}^{i-1}\mathbf{N}\mathbf{R} &= \mathbf{0} \\ \mathbf{S}\mathbf{L}\mathbf{M}^{i-1}\mathbf{V} &= \mathbf{0} & \mathbf{S}\mathbf{L}\mathbf{M}^{i-1}\mathbf{N}\mathbf{R} &= \mathbf{0} \quad i=1, \tilde{n} \end{aligned} \quad (10)$$

Lucrarea formulează legea de comprimare sub forma: Legea de comandă $\tilde{u} = \tilde{K}\tilde{x} + \tilde{V}$ pentru extinderea \tilde{S} este comprimabilă în legea de comandă $u = Kx + V$ pentru sistemul (inițial) S dacă:

$$\begin{aligned}\tilde{x}_0 &= Vx_0 \\ \tilde{u}(t) &= Ru(t) \quad (t \geq 0)\end{aligned}\quad (11)$$

implică:

$$\tilde{K}\tilde{x}(t; \tilde{x}_0, \tilde{u}) = RKx(t; x_0, u) \quad (t \geq 0) \quad (12)$$

pentru orice stare inițială x_0 și orice intrare fixată $u(t)$ a lui S .

Definind matricea F prin $\tilde{K} = RKU + F$, condițiile necesare și suficiente de comprimabilitate sînt date de:

$$RKA^i = \tilde{K}\tilde{A}^iV, \quad \tilde{R}KA^iB = \tilde{K}\tilde{A}^i\tilde{B}R \quad i=0, 1, 2, \dots \quad (13)$$

sau echivalent:

$$FM^{i-1}V = 0 \quad FM^{i-1}NR = 0 \quad i=1, \bar{n} \quad (14)$$

În consecință, proiectînd legea de comandă pentru sistemul extins și aceasta fiind comprimabilă se poate obține legea de comandă pentru sistemul inițial pe baza:

$$K = Q\tilde{K}V \quad (15)$$

Pentru exemplificare se consideră sistemul $S: \dot{x} = Ax + Bu$ de forma:

$$S: \begin{vmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{vmatrix} = \begin{vmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{vmatrix} \begin{vmatrix} x_1 \\ x_2 \\ x_3 \end{vmatrix} + \begin{vmatrix} B_{11} & 0 & 0 \\ 0 & B_{22} & 0 \\ 0 & 0 & B_{33} \end{vmatrix} \begin{vmatrix} u_1 \\ u_2 \\ u_3 \end{vmatrix} \quad (16)$$

Sistemul S poate fi privit ca un sistem interconectat, compus din 2 subsisteme cu starea x_2 suprapusă:

$$S_1: \begin{vmatrix} \dot{x}_1 \\ \dot{x}_2 \end{vmatrix} = \begin{vmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{vmatrix} \begin{vmatrix} x_1 \\ x_2 \end{vmatrix} + \begin{vmatrix} B_{11} & 0 \\ 0 & B_{22} \end{vmatrix} \begin{vmatrix} u_1 \\ u_2 \end{vmatrix} \quad (17)$$

$$S_2: \begin{vmatrix} \dot{x}_2 \\ \dot{x}_3 \end{vmatrix} = \begin{vmatrix} A_{22} & A_{23} \\ A_{32} & A_{33} \end{vmatrix} \begin{vmatrix} x_2 \\ x_3 \end{vmatrix} + \begin{vmatrix} B_{22} & 0 \\ 0 & B_{33} \end{vmatrix} \begin{vmatrix} u_2 \\ u_3 \end{vmatrix}$$

În extinderea \tilde{S} , S_1 și S_2 apar ca disjuncte:

$$\tilde{S}: \begin{vmatrix} \dot{\tilde{x}}_1 \\ \dot{\tilde{x}}_2 \end{vmatrix} = \begin{vmatrix} A_{11} & A_{12} & 0 & A_{13} \\ A_{21} & A_{22} & 0 & A_{23} \\ A_{21} & 0 & A_{22} & A_{23} \\ A_{31} & 0 & A_{32} & A_{33} \end{vmatrix} \begin{vmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{vmatrix} + \begin{vmatrix} B_{11} & 0 & 0 & 0 \\ 0 & B_{22} & 0 & 0 \\ 0 & 0 & B_{22} & 0 \\ 0 & 0 & 0 & B_{33} \end{vmatrix} \begin{vmatrix} u_1 \\ u_2 \end{vmatrix} \quad (18)$$

comenzile descentralizate fiind date de:

$$\tilde{u}_1 = \begin{vmatrix} \tilde{K}_{11} & \tilde{K}_{12} \\ \tilde{K}_{21} & \tilde{K}_{22} \end{vmatrix} \tilde{x}_1 \quad \tilde{u}_2 = \begin{vmatrix} \tilde{K}_{33} & \tilde{K}_{34} \\ \tilde{K}_{43} & \tilde{K}_{44} \end{vmatrix} \tilde{x}_2 \quad (19)$$

Matricea de transfer pentru sistemul extins \tilde{S} :

$$\tilde{K}_1 = \begin{vmatrix} \tilde{K}_{11} & \tilde{K}_{12} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \tilde{K}_{33} & \tilde{K}_{34} \\ 0 & 0 & \tilde{K}_{13} & \tilde{K}_{14} \end{vmatrix} \quad (20)$$

nu este comprimabilă la:

$$K = \begin{vmatrix} K_{11} & K_{12} & 0 \\ K_{21} & K_{22} & K_{23} \\ 0 & K_{32} & K_{33} \end{vmatrix} \text{ pentru sistemul } S \quad (21)$$

De aceea este necesară introducerea unei componente locale a comenzii $\tilde{u} = \tilde{K}_g \tilde{x}$ pe lângă componenta locală $\tilde{v} = \tilde{K}_1 x$ astfel ca legea de comandă: $\tilde{u} = (\tilde{K}_1 + \tilde{K}_g) \tilde{x}$ să fie comprimabilă. Deși această reacție suplimentară nu este descentralizată în extensia \tilde{S} , asigură conducerea descentralizată în sistemul S .

Lucrarea prezintă în final o aplicație a principiilor teoretice enunțate pe un model cunoscut în literatură [7].

Cazul interacțiunilor dinamice (cu sau fără întârzieri) între subsistemele interconectate este abordat în lucrarea 11.1/A4. Pe baza vectorilor proprii ai subsistemelor care descriu interacțiunile se găsește spectrul derivat de valori proprii pentru sistemul compus.

Este cunoscut faptul că utilizarea unor legi de comandă care începereză și valorile anterioare ale vectorului de stare (istoricul) conduce la performanțe superioare privind robustețea și sensibilitatea la modificarea valorilor nominale ale parametrilor sistemului și la perturbațiile exterioare [8]. Acest rezultat poate fi utilizat și în cazul sistemelor interconectate.

Lucrarea 11.1/A5 tratează rolul strategiilor de memorie în jocurile dinamice, în particular în legătură cu existența și unicitatea unor tipuri variate de echilibru. Unitățile de conducere descentralizată sînt privite ca „jucători” din teoria jocurilor.

2. ESTIMATOARE ȘI REGULATOARE DESCENTRALIZATE

Lucrările grupate în cadrul secțiunii 11.1/B prouun diferite abordări pentru proiectarea estimatoarelor și reguletoarelor descentralizate, explicatînd structura fizică a sistemelor complexe (compuse din subsisteme interconectate) sau operînd asupra subsistemelor cîștîtute prin metodele clasice de descompunere. Proiectarea unui regulator descentralizat, pentru fiecare subsistem, atrage după sine necesitatea estimării (observării) de o manieră descentralizată a stărilor, ieșirilor, intrărilor de interacțiune ale subsistemelor.

O primă abordare este prezentată în lucrarea 11.1/B1, pentru cazul sistemelor cu două scări de timp (două dinamici: una lentă, cealaltă rapidă): Fără utilizarea metodelor de agregare (înlocuirea modelului original de ordinul n printr-un model redus, de ordinul r_1), informația corespunzătoare celor $r - n_1$ stări suprimate este pierdută. Modelul redus (agregat) poate conduce la un sistem de comandă departe de optim, chiar instabil, metodele de agregare nepermițînd reintroducerea modurilor suprimate. Acest lucru este posibil, așa cum s-a arătat în [9], în cazul sistemelor care pot fi reprezentate prin modele singular perturbate, și anume cu un parametru mic ϵ acționînd multiplicativ asupra derivatelor în timp ale variabilelor de stare:

$$\begin{aligned} \dot{\tilde{x}} &= f(x, y, \epsilon t) \\ \epsilon \dot{y} &= g(x, y, \epsilon t) \end{aligned} \quad (1)$$

Astfel de sisteme se caracterizează prin existența a două dinamici, una lentă, luată în considerație la primul pas, cealaltă rapidă, care poate fi tratată separat. Când aceste două dinamici pot fi puse în evidență se poate aplica cea mai mare parte a rezultatelor obținute în cazul sistemelor singular perturbate.

Pentru separarea dinamicii globale a sistemului în două părți se folosesc diverse tehnici, dintre care în lucrare este prezentată tehnica de triunghiularizare. Dat fiind sistemul dinamic:

$$\dot{X} = AX \quad (2)$$

partiționat arbitrar sub forma:

$$\begin{bmatrix} \dot{X}_1 \\ \dot{X}_2 \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \cdot \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \quad (3)$$

prin utilizarea transformării:

$$\begin{bmatrix} X_1 \\ Z_2 \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} \cdot \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \mathfrak{B} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \quad (4)$$

sistemul poate fi pus sub forma bloc triunghiularizată:

$$\begin{bmatrix} \dot{X}_1 \\ \dot{Z}_2 \end{bmatrix} = \begin{bmatrix} \tilde{A}_1 & \tilde{A}_{12} \\ R(L) & \tilde{A}_2 \end{bmatrix} \begin{bmatrix} X_1 \\ Z_2 \end{bmatrix} \quad (5)$$

unde:

$$\tilde{A}_1 = A_{11} - A_{12}L \quad \text{și} \quad \tilde{A}_2 = A_{22} + LA_{12} \quad (6)$$

L — soluția ecuației Riccati:

$$R(L) = 0 = A_{21} + LA_{11} - LA_{12} - A_{22}L \quad (7)$$

Calculul matricii de transformare L se poate face fie prin utilizarea vectorilor proprii, fie prin utilizarea polinoamelor caracteristice.

O contribuție importantă o aduce lucrarea în domeniul conducerii sistemelor cu două scări de timp. Utilizând transformarea (4), considerind și matricile de intrare (B) și ieșire (C), sistemul (2) este pus sub forma:

$$\begin{bmatrix} \dot{X}_1 \\ \dot{X}_2 \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \cdot \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} + \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} u \quad (8)$$

$$\begin{bmatrix} \dot{X}_1 \\ \dot{Z}_2 \end{bmatrix} = \begin{bmatrix} A_s & A_{12} \\ 0 & A_f \end{bmatrix} \cdot \begin{bmatrix} X_1 \\ Z_2 \end{bmatrix} + \begin{bmatrix} B_1 \\ B_f \end{bmatrix} u$$

$$y = [C_1 \quad C_2] \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \quad y = [C_s \quad C_2] \begin{bmatrix} X_1 \\ Z_2 \end{bmatrix} \quad (9)$$

cu:

$$A_s = A_{11} - A_{12}L \quad A_f = A_{22} + LA_{12}$$

$$B_f = LB_1 + B_2 \quad C_s = C_1 - C_2L$$

Considerind dinamica A_s și A_f distincte, intrarea u compusă din componente lente u_s și rapide u_f (u_s transmisă direct de $-A_f^{-1}B_f$, iar u_f puternic atenuată de A_s), sistemul este reprezentat ca în fig. 1. Această descriere permite sîrteza utilizînd reacția de la stările și ieșirile decuplate, mai ușor decît utilizînd informația de stare globală. Dacă K_0 și K_f sînt corectorii de stare proiectați pentru cele 2 subsisteme — lent și rapid — atunci este suficientă identificarea lui u :

$$u = K_0 X_s + K_f X_f = K_1 X_1 + K_2 X_2 \quad (10)$$

Conform fig. 1 :

$$X_2 = X_1 + X_{s,r} - LX_0 = X_1 - A_f^{-1} B_r K_0 X_s - LX_0 \quad (11)$$

rezultând :

$$K_1 K_2 = [(1 + K_r A_f^{-1} B_r) K_0, K_r] \begin{bmatrix} I & 0 \\ L & I \end{bmatrix} \quad (12)$$

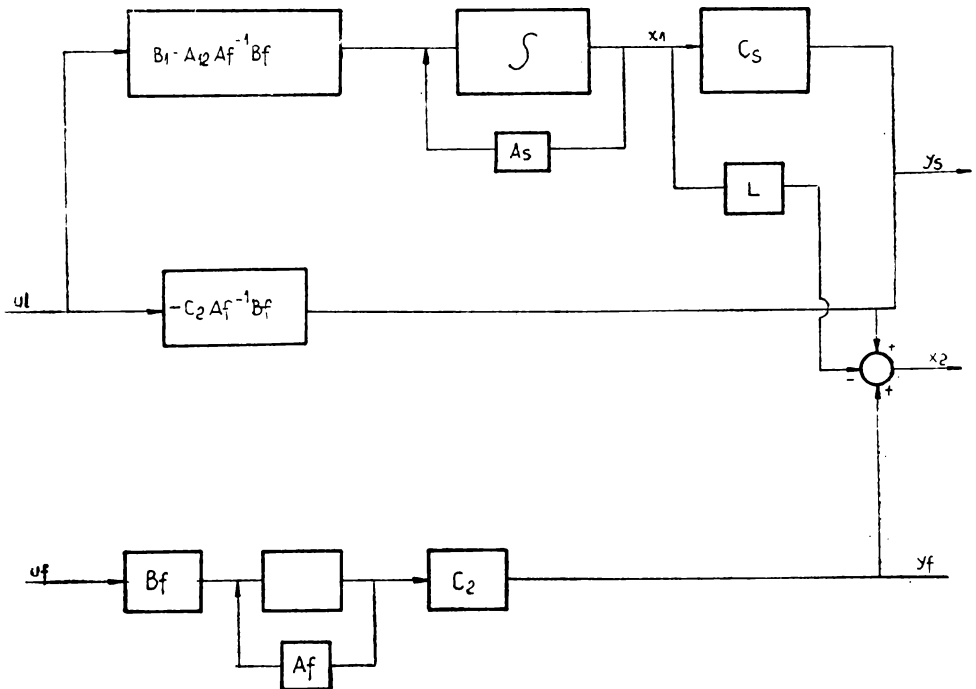


Fig. 1. Reprezentarea unui sistem cu două dinamici.

Subsistemul rapid fiind în general stabil ($K_r=0$), reacția de la stare poate fi limitată la variabilele lente.

Principala avantaj al acestui mod de abordare constă în economisirea unui volum important de calcule matriciale. Totuși când dinamica subsistemelor nu pot fi diferențiate, trebuie apelat din nou la abordarea algebrică.

În lucrarea 11.1/B3 la sinteza estimatoarelor descentralizate este utilizată informația privitoare la vectorul local de stare $x_1(t)$ și intrarea de cuplare (de interacțiune) $h_1(t)$. În sistemul dinamic considerat :

$$\dot{z}_1(t) = F_1 z_1(t) + G_1 y_1(t) + J_1 w_1(t) \quad z_1(t_0) = z_{10} \quad (13)$$

$$w_1(t) = L_1^1 z_1(t) + L_1^2 y_1(t)$$

Ieșirea $w_1(t)$ trebuie să convergă către o funcție necunoscută $p_1(t) = M_1 x(t)$ de stările sistemului observat. Așa cum s-a menționat, se consideră :

$$p_1(t) = [X_1^T \quad n_1^T]^T \quad (14)$$

$$M_1 = \begin{bmatrix} 0 & \dots & 0 & I & 0 & 0 \\ A_{11} & \dots & A_{11-1} & 0 & A_{11+1} & \dots & A_{1N} \end{bmatrix}$$

unde: N este numărul de subsisteme.

În acest caz estimatorul realizează o funcție lineară de stările locale $x_i(t)$ și intrările de interacțiune $h_i(t)$ (fig. 2). S-ar putea folosi $p_1(t) = x(t)$ ($M_1 = I$) (maximum de informație despre sistemul global), dar efortul de proiectare ar crește considerabil și în plus ar fi estimate stări care nu ar intra în componența comenzii descentralizate.

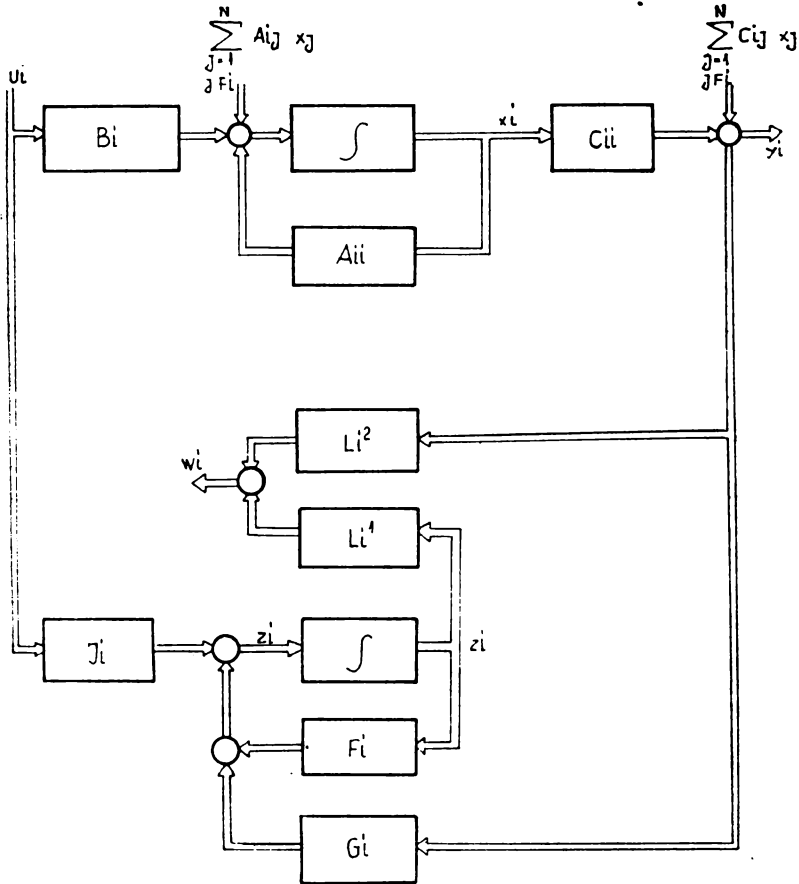


Fig. 2. Estimator descentralizat pentru subsistemul i .

Estimatorul descentralizat din fig. 2 îndeplinește teorema de separare; de asemenea ecuațiile diferențiale ale erorii de estimare ale subsistemelor sînt decuplate. Prin considerarea ieșirilor estimatorului ca intrări ale subsistemului i : $u_1(t) = w_1(t)$ se aplică estimatorul la conducerea descentralizată, în lucrare prezentîndu-se condițiile de existență ale acestui estimator de comandă și aplicația la un sistem de conducere a unei turbine (8 stări, 2 subsisteme). Alte

aplicații posibile sînt sistemele de conducere pentru colcane de distilare și pentru cupatoare cu încălzire continuă.

În lucrarea 11.1./B4 sînt prezentate trei structuri de bază pentru estimatoarele de stare descentralizate :

1° estimator complet descentralizat, utilizînd exclusiv date și semnale ale subsistemului :

2° estimator parțial descentralizat, utilizînd și semnale de intrare și/sau ieșire globale ;

3° estimator descentralizat incluzînd un model de ordin redus al interacțiunilor.

În cazul 1°, considerînd subsistemul S_1 :

$$\begin{aligned}\dot{\hat{x}}_1(t) &= A_1 x_1(t) + B_1 u_1(t) + V_1 \zeta_1(t) \\ y_1(t) &= C_1 x_1(t)\end{aligned}\quad (15)$$

(unde $\zeta_1(t)$ nu e cunoscut de subsistemul i), starea estimată \hat{x}_1 este presupusă ca fiind ieșirea unui estimator :

$$\begin{aligned}\hat{x}_1(t) &= \xi_1(t) + M_1 y_1(t) \\ \dot{\xi}_1(t) &= (A_1 - L_1 C_1) \xi_1(t) + T_1 B_1 u_1(t) + [(A_1 - L_1 C_1) M_1 + L_1 + N_1] y_1(t)\end{aligned}\quad (16)$$

Proiectarea acestuia constă în găsirea matricilor L_1 , M_1 , N_1 , T_1 astfel încît estimatorul să fie stabil și eroarea :

$$e_1(t) = \hat{x}_1(t) - x_1(t)\quad (17)$$

să tindă la zero pentru $t \rightarrow \infty$.

Procedura de proiectare propusă acționează în două etape :

a) Verificarea stabilității conform condiției :

$$\text{rang} \begin{bmatrix} \lambda I - A_1 \\ C_1 \end{bmatrix} = n_1 \text{ pentru toți } \lambda \text{ cu } \text{Re } \lambda \geq 0\quad (18)$$

și proiectarea clasică a estimatorului $-L_1$ ales astfel încît să se obțină o matrice stabilă $A_1 - L_1 C_1$ pentru ecuația :

$$\dot{e}_1(t) = (A_1 - L_1 C_1) e_1(t)\quad (19)$$

b) Verificarea condiției :

$$\text{rang} \begin{bmatrix} C_1 & 0 \\ C_1 A_1 & C_1 V_1 \end{bmatrix} = \text{rang } V_1 + \text{rang} \begin{bmatrix} C_1 \\ C_1 A_1 \end{bmatrix}\quad (20)$$

(aceasta asigură decuplarea interacțiunilor necunoscute) și rezolvarea ecuației liniare :

$$[T_1 : N_1 : M_1] \begin{bmatrix} I & 0 & 0 \\ 0 & C_1 & 0 \\ C_1 & C_1 A_1 & C_1 V_1 \end{bmatrix} = [I : 0 : V_1]\quad (21)$$

Procedura descrisă neputînd fi aplicată întotdeauna, se pot introduce informații adiționale bazate pe măsurarea intrărilor și ieșirilor globale (2°) ; această abordare prezintă inconveniente legate de transmiterea acestor semnale. Se ajunge astfel la utilizarea unui model redus al interacțiunilor (3°), așa cum s-a propus în [10]. Metoda poate fi privită ca o formă de agregare de aproximare a sistemului complement sau ca o modelare a interacțiunilor ca perturbații acționînd din exterior asupra subsistemului S_1 .

Stabilind teoreme de existență pentru tipurile 1°, 2° și 3° de estimatoare descentralizate, lucrarea lasă deschise unele aspecte privind analiza sensibilității și condițiile de stabilitate pentru un sistem de comandă în buclă închisă realizat printr-un estimator.

Alte două lucrări — [11.1/B5] și [11.1/B6] — se referă la problema proiectării reguletoarelor descentralizate pentru cazul sistemelor liniare.

În lucrarea 11.1/B5, pentru cazul sistemelor dinamice discrete, lineare, invariante în timp se propune o metodă de proiectare a reguletoarelor descentralizate bazată pe optimizare parametrică. Modul de abordare este similar cu cel propus în [11] pentru cazul sistemelor dinamice continue. Metoda de proiectare este foarte flexibilă și poate fi aplicată în cazurile în care asupra structurii reguletoarelor se impun restricții de tipul :

- reguletoare descentralizate ;
- specificare parțială a relațiilor funcționale pe care reguletoarele le stabilesc între intrări și ieșiri.

Procedura de proiectare propusă — bazată pe optimizare parametrică — cuprinde două etape :

1° alegerea structurii reguletorului care satisface restricțiile de informație necesară și complexitate ;

2° acordarea parametrilor reguletorului, prin optimizarea unui indice de ponderare a dinamicii sistemului de comandă, luând în considerație cerința de anulare a erorii de reglare.

Structura unor astfel de reguletoare este prezentată în fig. 3, în care: R_0 constă din p integratoare discrete (în paralel), $R_1(q)$ și $R_2(q)$ sînt sisteme lineare (q — vectorul parametrilor de proiectare), invariante în timp, de ordin dat (R_1 și R_2 pot fi dinamice sau statice, centralizate sau descentralizate, parametrizate).

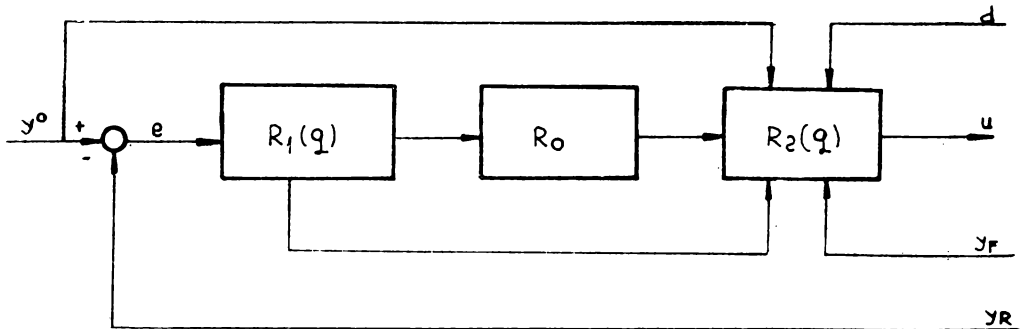


Fig. 3. Structura reguletorului descentralizat.

Se apreciază ca interesante formularea problemei de proiectare a reguletoarelor ca o problemă de programare matematică (rezolvată cu algoritmul de gradient conjugat Fletcher-Powell) și aplicația privind conducerea unei mașini de hirtie. Stabilitatea asimptotică a sistemului de reglare este asigurată de o manică robustă (pentru orice variații ale matricilor sistemului cendus, orice perturbații ale matricilor de filtrare etc.).

În lucrarea 11.1/B6 se extinde metodologia propusă în [12] pentru problema generală a servomecanismelor la cazul reguletoarelor descentralizate (în funcție de ieșire) pentru sistemele lineare complexe supuse unor perturbații externe constante. Procedura descrisă în [13] pentru proiectarea reguletoarelor PI, PID și PID generalizat în cazul sistemelor cu informație centralizată este aplicată la cazul descentralizat. Abordarea propusă în lucrare se bazează pe extinderea vectorilor de stare și de ieșire și a matricilor corespunzătoare și pe utilizarea conceptului de comandă predictivă [13]. Structurile proiectate pot fi implementate pe microprocesoare.

În cazul utilizării unor structuri de prelucrare distribuită pentru rezolvarea problemelor de optimizare și conducere multinivel, algoritmi utilizați sînt în general de tip sincron. (Sarcina de prelucrare la o iterație este împărțită între procesoarele disponibile, acestea comunicînd informațiile privind ieșirile iterației curente înainte de începerea iterației următoare). Acest mod de prelucrare urmărește ordinea operațiilor din algoritm și are avantajul de a simplifica analiza convergenței. Algoritmi distribuiți de tip sincron prezintă totuși dezavantajul

de a necesita un protocol de inițializare a iterației și de a limita viteza de calcul la viteza celui mai lent procesor.

În lucrarea 11.1/B2 se luă în considerație algoritmi care tolerează o ordine mai flexibilă a calculului și a comunicațiilor între procesoare (algoritmi asincroni). Astfel de algoritmi au găsit aplicații în rețelele de calculatoare (este citat exemplul rețelei ARPA), unde — în cazul defectării procesoarelor — este complicat să se mențină sincronizările între nodurile întregii rețele când acestea execută funcțiile de control în timp real (dirijarea pachetelor de mesaje). Pentru astfel de algoritmi este interesant de stabilit gradul minim de coordonare necesar pentru ca un algoritmi dat să conducă la soluția corectă. Lucrarea se rezumă la enunțarea acestor probleme și a aplicațiilor posibile (optimizare distribuită, estimarea parametrilor, rețelele de calculatoare).

3. CONDUCERE DESCENTRALIZATĂ

Lucrările secțiunii 11.1/C completează tematica secțiunii 11.1/B cu aspecte privind proiectarea structurilor de conducere descentralizată în cazul sistemelor stohastice și în cazul prezenței modurilor fixe.

În lucrarea 11.1/C1 abordarea problemei de proiectare a reguletoarelor descentralizate (cazul continuu) robuste este similară cu cea prezentată în lucrarea 11.1/B5.

Comanda optimă în sistemele stohastice, în cazul unei structuri nestandard a informației utilizate în sinteza legii de conducere, face obiectul lucrării 11.1/C2. Este interesantă și originală metoda propusă pentru reducerea informației necesare, pentru care se formulează o condiție suficientă de existență. Această condiție este aplicată în cazul sistemelor compuse cu structură specială (de exemplu, considerând zgomotele multiplicative drept cuplaje impredictibile). Deși aplicabilitatea condiției de suficiență nu se reduce la cazurile prezentate în lucrare, totuși este dificil să se extindă condiția la cazuri mai generale. O tratare mai completă a situației cuplajelor mutuale (reciproce) (enunțată în lucrarea 11.1/C2) este făcută în 11.1/C6.

Lucrările 11.1/C3 și 11.1/C4 tratează problema conducerii descentralizate în prezența modurilor fixe, concept introdus pentru prima dată în [14.] Considerând sistemul multivariabil linear invariant în timp :

$$\dot{x}(t) = Ax(t) + \sum_{i=1}^n B_i u_i(t) \quad i = \overline{1, n} \quad (1)$$

$$y_i = C_i x(t)$$

cu reguletoarele descentralizate :

$$u_i(t) = K_i y_i(t) + Q_i z_i(t) \quad (2)$$

$$\dot{z}_i(t) = S_i z_i(t) + R_i y_i(t)$$

și mulțimea :

$$K^* = \{K/K = \text{blockdiag} [K_1, \dots, K_n]\} \quad (3)$$

atunci mulțimea modurilor fixe descentralizate (MFD) în raport cu K este :

$$\Lambda(C, A, B, K) = \bigcap_{K \in K^*} \sigma(A + BK) \quad (4)$$

în care : (\cdot) reprezintă mulțimea valorilor proprii pentru (\cdot) ;

$$B = B_1, B_2, \dots, B_n \quad (5)$$

$$C^T = [C_1^T, C_2^T, \dots, C_n^T]^T$$

Condiția necesară și suficientă de existență a regulatcărclor descentralizate (2) pentru ca sistemul (1) să fie asimptotic stabil este ca MFD ale lui (1) să fie toate în partea stângă deschisă a planului complex. MFD joacă un rol important în multe probleme de conducere (problema stabilizării descentralizate, problema servomecanismului descentralizat robust) și în determinarea controlabilității și observabilității în sistemele puternic interconectate. O descriere algebrică a MFD a fost realizată în [15]. În [16] s-a introdus conceptul de moduri fixe structurate. În cazul când MFD sînt o reflectare a parametrilor sistemului, acestea pot fi eliminate prin modificarea parametrilor; cînd MFD reflectă structura sistemului, eliminarea lor se poate face fie modificînd structura sistemului fie relaxînd restricțiile impuse asupra fluxului de informații între regulatcările locale (schimbarea structurii matricii de reacție K). MFD pot fi eliminate printr-o descompunere adecvată a sistemului (cînd aceasta nu este impusă). Cînd descompunerea sistemului este impusă și are MFD, pentru eliminarea acestora se poate recurge la procedura propusă în [17]; MFD sînt caracterizate prin matrici dominante blocdiagonale și se determină mulțimea de legături între subsisteme (blocuri K_{ij}) care elimină MFD (pentru matricea K rezultă o structură redundantă).

În lucrarea 11.1/C3 se propune o nouă modalitate de caracterizare a MFD. Sînt deduse condițiile necesare și suficiente pentru ca un pol al sistemului să fie un MFD și unele condiții suficiente pentru ca un sistem să nu aibă MFD. Cînd se obțin MFD instabile este sugerată modalitatea de alegere a unei noi structuri a informației de control, care elimină MFD (cu condiția ca sistemul să fie controlabil și observabil). Pentru exemplificare se folosesc exemplele cunoscute în literatura de specialitate [17].

În lucrarea 11.1/C4 se face o trecere în revistă a diverselor abordări legate de MFD. Dintre acestea menționăm pe cele cu un grad mai ridicat de aplicabilitate [18], [19], [20]. În [18] se asociază sistemului un graf $\Gamma_s = (V_s, L_s)$ în care fiecare nod reprezintă o variabilă de intrare sau ieșire, iar arcele reprezintă o funcție de transfer fără zecuri sau o legătură de reacție. O valoare proprie a sistemului este fixă dacă și numai dacă ea nu este un pol al vreunui ciclu elementar al lui Γ_s . Abordarea propusă în [19] o include pe cea din [18], fiind singura care permite să se obțină o formă adecvată a matricii K de reacție de la ieșire — cu structură și coeficienți optimați — fără ca să fie necesară testarea prealabilă a existenței modurilor fixe. Dacă sistemul nu are moduri fixe, comanda rezultată va fi complet descentralizată. În [20], utilizînd principiul conducerii ierarhizate a sistemelor complexe se stabilizează un număr de moduri fixe cu ajutorul unui coordonator global. Acest coordonator utilizînd totalitatea stărilor, tratarea este mai degrabă descentralizată.

Dezvoltînd concluziile din [14], lucrarea 11.1/C5 tratează problema conducerii descentralizate robuste, cu perturbații parțiale asupra parametrilor sistemului, cu urmărirea comenzilor de referință fără ercarea staționară. În cazul a numai 2 unități de conducere locală și în ipotezele menționate, condițiile necesare și suficiente din [14] devin condiții suficiente pentru problemele de tipul menționat.

În lucrarea 11.1/C6, pe baza lucrărilor [21], [22], se propune o trunchiere a interconexiunilor considerîndu-se pentru ele o formă simetrică (mutuală), ceea ce are ca o presupunere mai realistă decît ignorarea lor totală. Pentru sistemul stohastic :

$$dx = [A_D dt + A_C(dw)] x + B u dt \quad (6)$$

$$A_C = (a_{ij}) \quad i, j = \overline{1, n}$$

cu indicele de performanță :

$$J = \int_0^\infty [x^T(t) Q x(t) + u^T(t) R u(t)] dt \quad (7)$$

descompus în s subsisteme :

$$dx_i = A_i x_i dt + \sum_{j=1}^s A_{ij}(dw) x_j + B_i u_i dt \quad i = \overline{1, s} \quad (8)$$

$$J_i = \int_0^\infty [x_i^T(t) a_i x_i(t) + u_i^T(t) R_i u_i(t)] dt, \quad (9)$$

problema conducerii descentralizate optimale constă în calculul legii de comandă cu restricții :

$$u^T(x) = [u_1(x_1), u_2(x_2), \dots, u_s(x_s)] \quad (10)$$

care minimizează valoarea medie a indicelui J.

Se impun condițiile :

$$1^\circ E(a_i a_j^T) = -E(a_i a_j^T)^T \quad \gamma, \quad v=1, s \quad \gamma \neq v \quad (11)$$

(Acțiunile componentelor x^1 și x^j asupra componentelor x^k și x^l sînt opuse și linear dependente — în valoare medie).

Introducînd constanta :

$$\beta = \inf_K \left\| \int_0^\infty \exp[A_D - BK]^T t \exp[(A_D - BK)t] dt \right\| \quad (12)$$

în care $\|D\|$ — norma vectorială euclidiană, condiția :

$$2^\circ \sum_{k=1}^P \|G_k\|^2 \leq (\beta P)^{-1} \quad (13)$$

semnifică exercitarea unor interacțiuni slabe între subsisteme.

Considerînd sistemul stohastic interconectat (6), condițiile 1° și 2° îndeplinite și (A_D, B) controlabile, legea de comandă :

$$u^\circ(x) = -Kx \quad (14)$$

satisfacă (10) și minimizează indicele de performanță :

$$EJ = E \int_0^\infty (x^T Q x + u^T R u) dt \quad (15)$$

Matricea de comandă K este calculată cu :

$$K = \text{diag}[K_1, K_2, \dots, K_s] \quad (16)$$

în care : $K_k = P_k^{-1} B_k^T P_k$

iar P_k ($k=1, s$) sînt blocuri ale diagonalei matricii pozitiv definite : $P = \text{diag}[P_1, P_2, \dots, P_s]$ care satisfac sistemul de ecuații algebrice :

$$A_k^T P_k + P_k A_k - P_k B_k R_k^{-1} B_k^T P_k + Q_k + \Gamma_h(P) = 0$$

în care :

$$\Gamma_k(P) dt = E(a_i^T P a_j) \quad k=1, s \quad (17)$$

Matricea $A_D - BK$ este stabilă. Mai mult, sistemul stohastic în buclă închisă (6) cu legea de comandă (14) este (cu probabilitatea 1) exponențial stabil și ordinul de stabilitate cel mai mic este :

$$\alpha = \lambda_m(Q + K^T R K) / \lambda_M(P) \quad (18)$$

în care λ_m și λ_M sînt valorile proprii minime și maxime ale matricilor respective.

Creșterea volumului de calcule datorită cuplajelor ecuațiilor Riccati este compensată de faptul că procedura furnizează imediat o structură de conducere descentralizată stabilă.

În concluzie lucrările acestei secțiuni dezvoltă principiile conducerii descentralizate a sistemelor interconectate stabilite în [14], [21] și [22].

BIBLIOGRAFIE

1. *Kolmogorov, A.N.* (1965). **Three approaches for defining the concept of information quantity.** Prob. Inform. Trans. 1, p. 1–7.
2. *Chaitin, G.J.* (1966). **On the length of programs for computing finite binary sequences.** Journal of ACM, 13, p. 547–569.
3. *Liu Yong-Qing* (1965). * **Decomposition of Liapunov's function.** ACTA Automatica Sinica vol. 3, No. 3.
4. *Michel, A.N., Miller, R.K.* (1977). **Qualitative analysis of large-scale dynamical system.** Academic Press.
5. *Siljak, D.D.* (1978). **Large-scale dynamic systems, stability and structure.** North Holland.
6. *Bailey, F.N.* (1966). **The application of Lyapunov's second method to interconnected systems.** J. SIAM Control Ser. A.3, p. 443–462.
7. *Levine, W.S., Athans, M.* (1966). **On the optimal error regulation of a string of moving vehicles.** IEEE Transactions on Automatic Control, AC-11, p. 355–361.
8. *Sundarashan, N., Cruz, J.B.* (1972). **Sensitivity reduction in timevarying linear and nonlinear systems.** Int. Journal of Control, 15, 5, p. 937–943.
9. *Kokotovic P.V., Yackel R.A.* (1977) **Singular perturbation of linear regulators: basic theorems.** IEEE Trans. on Autom. Control – Vol. AC 17, no. 1.
10. *Tacker, E.C., Sanders, C.W.* (1982). **Decentralized structures for state estimation in large scale systems.** Large Scale Systems 3, p. 255–266.
11. *Guardabassi, G., Locatelli, A., Maffezzoni, C., Schiavoni, N.* (1979). **Parameter optimization in decentralized process control: a unified setting for multivariable industrial regulator design.** M.A. Cuénod (Ed.), Proc. IFAC Symposium on Computer Aided Design of Control Systems, Pergamon-Press, Oxford, p. 87–92.
12. *Davison, E.J.* (1976). **The robust Decentralized Control of General Servomechanism Problem.** IEEE Trans. on Automatic Control, 21, p. 14–24.
13. *Medanic, J.* (1979). **Synthesis of Decentralized Output Regulators.** Systems Engineering for Power Vol. II, DOE, Davos Switzerland, p. 3 121–3 134.
14. *Davison, E.H., Wang, S.H.* (1973). **On the stabilization of decentralized control systems.** IEEE Trans. on Automatic Control, vol. AC-18, p. 473–478.
15. *Anderson, B.D.O., Clements, D.J.* (1981). **Algebraic characterization of fixed modes in decentralized control.** Automatica, 17, p. 703–712.
16. *Sezer, M.E., Siljak, D.D.* (1981). **Structurally fixed modes.** Systems and control letters, vol. 1, no. 1, p. 60–64.
17. *Armentano, V.A., Singh, M.G.* (1981). **A new approach to the decentralized controller initialization problem.** Preprints IFAC Congress, Kyoto.
18. *Locatelli, A., Schiavoni, N., Tarantini, A.* (1977). **Pole placement: role and choice of the underlying information pattern.** Recherche di Automatica, vol. 18, no. 1., p. 107–126.
19. *Senning, M.F.* (1979). **Feasibly decentralized control.** Ph. D. Thesis ETH Zürich.
20. *Groumpos, P.P., Loparo* (1980). **Structural control of large-scale systems.** 19-th IEEE Conference on Decision and Control.
21. *Sandell, N.R., Varaiya, P., Athans, M., Safonov, M.G.* (1978). **Survey of Decentralized Control Methods for Large Scale Systems.** IEEE Transactions, AC-23, p. 108–128.
22. *Wonham, N.M.* (1967). **Optimal Stationary Control of a Linear System with State – Dependent Noise.** SIAM Journal of Control, 5, p. 486–500.

4. APLICAȚII ALE METODOLOGIILOR SISTEMELOR COMPLEXE

După cum s-a arătat într-o lucrare anterioară [1] dedicată sintezei unor comunicări științifice legate de sistemele mari și complexe prezentate la al 8-lea Congres IFAC de la Kyoto, eforturi importante de cercetare sînt depuse în vederea rezolvării problemelor integrate de optimizare și estimare a parametrilor în cazul dinamic. Și în cadrul grupajului de analizat în prezenta

sinteză, două lucrări au abordat problema amintită. Astfel Singh (11.1/D6), în contextul unei aplicații legate de procesele de fermentație, arată că procesul poate fi reprezentat de modelul matematic:

$$\dot{\underline{x}} = \underline{f}(\underline{x}, \underline{m}, \underline{\alpha}) \quad (1)$$

unde: vectorul de stare, \underline{x} , indică creșterea celulelor și sinteza penicilinei, vectorul de comandă \underline{m} reprezintă temperaturile, iar vectorul $\underline{\alpha}$ conține coeficienții de reacție catalitică enzimatică.

Funcția obiectiv de extremizat este de tipul Mayer:

$$J = \underline{c}^T \underline{x}(t_f) \quad (2)$$

unde \underline{c} indică un vector constant, t_f este timpul final, iar T indică transpunerea.

În condițiile în care $\underline{\alpha}$ nu este cunoscut cu precizie, problemele de optimizare și estimare a parametrilor se pot combina într-o problemă de extremizare (minimizare) a următorului criteriu combinat:

$$J_1 = \min_n \left\{ \frac{1}{2} (1 - \beta) \int_0^T (\underline{x}^T \underline{Q} \underline{x} + \underline{m}^T \underline{R} \underline{m}) dt + \frac{1}{2} \beta \int_0^T (\underline{y}^o - \underline{y})^T \underline{W} (\underline{y}^o - \underline{y}) dt \right\} \quad (3)$$

cu restricțiile (1) și

$$\underline{y} = \underline{g}(\underline{x}, \underline{m}, \underline{\alpha}) \quad (4)$$

unde \underline{y}^o este vectorul mărimilor măsurate, iar β este un coeficient subunitar pozitiv.

Procedura constă în rezolvarea iterativă a problemei definite de criteriul (3) și restricțiile (1) și (4) și aplicarea comenzilor \underline{m} procesului real în condițiile scăderii continue a valorii parametrului β . În lucrare se arată că procedura a fost implementată pe un microcalculator.

Dacă elementele prezentate mai sus se refereau la o problemă dinamică, pentru cazul static, Ellis, Mihalska și Roberts (11.1/D3), prezintă extinderi ale unor rezultate proprii anterioare [2] în cazul sistemelor interconectate. Dată fiind importanța metodei, care permite, în principiu, economii substanțiale atât în efortul de calcul (decarece admite folosirea unor modele simple), cât și, uneori, în costul instrumentației de măsurare, în condițiile obținerii de valori concrete optime ale referințelor reguletoarelor pentru procese complicate, în continuare va fi descrisă pe larg.

Fie un sistem format din N subsisteme interconectate, prevăzute cu bucle de reglare, a căror funcționare în regim static este dată de:

$$\underline{y}_i = \underline{f}_i^*(\underline{c}_i, \underline{u}_i) \quad i = \overline{1, N} \quad (5)$$

unde \underline{y}_i , \underline{c}_i și \underline{u}_i sînt: ieșirile, consemnele reguletoarelor și respectiv intrările de interconexiune ale subsistemelor aparținînd mulțimilor \mathcal{Y}_i , \mathcal{C}_i și \mathcal{U}_i .

Cuplajul dintre subsisteme este dat de:

$$\underline{u}_i = \sum_{j=1}^N H_{ij} \underline{y}_j, \quad i = \overline{1, N} \quad (6)$$

unde elementele matriei de interconexiune, H_{ij} , au valoarea zero sau unu.

Descrierea globală a sistemului este:

$$\underline{y} = \underline{f}^*(\underline{c}, \underline{u}) \quad (5')$$

unde vectorii globali (la nivelul sistemului) sînt :

$$\begin{aligned}\underline{y} &\triangleq [\underline{y}_1^T \dots \underline{y}_N^T]^T \in \mathfrak{Y}_1 \times \dots \times \mathfrak{Y}_N \triangleq \mathfrak{Y} \\ \underline{u} &\triangleq [\underline{u}_1^T \dots \underline{u}_N^T]^T \in \mathcal{U}_1 \times \dots \times \mathcal{U}_N \triangleq \mathcal{U} \\ \underline{c} &\triangleq [\underline{c}_1^T \dots \underline{c}_N^T]^T \in \mathcal{C}_1 \times \dots \times \mathcal{C}_N \triangleq \mathcal{C}\end{aligned}$$

Deoarece în cazul unui sistem complex funcțiile corecte ale subsistemelor (1) pot să nu fie cunoscute cu precizie, sau să fie prea complicate pentru o folosire on-line, se pot folosi modele simplificate :

$$\underline{y}_i = f_i(\underline{c}_i, \underline{u}_i, \underline{\alpha}_i), \text{ sau} \quad (7)$$

$$\underline{y} = f(\underline{c}, \underline{u}, \underline{\alpha})$$

unde : $\underline{f} = [\underline{f}_1^T \dots \underline{f}_N^T]^T$, $\underline{\alpha}_i$ reprezintă parametri modelelor.

Se presupune (cu suficientă acoperire practică) că funcția de cuplaj adevărată (6) poate fi cunoscută cu precizie. De asemenea, funcțiile care definesc restricția $\underline{c} \in \mathcal{C}$ pot fi definite cu precizie (în continuare se vor considera numai restricțiile privind variația valorilor consemnelor).

Ținînd seama de structura interconectată a sistemului, într-o problemă de optimizare apare ca naturală considerarea unui criteriu de performanță aditiv de tipul

$$\min_{\underline{c} \in \mathcal{C}} \left\{ J(\underline{c}, \underline{u}, \underline{y}) = \sum_{i=1}^N J_i(\underline{c}_i, \underline{u}_i, \underline{y}_i) \right\} \quad (8)$$

În definirea problemei de optimizare, pe lângă criteriul de performanță (8), se vor considera restricțiile de model (7).

Să considerăm cazul particular cînd în definirea criteriului de performanță (8) nu intervin și intrările de interacțiune \underline{u}_i :

$$J(\underline{c}, \underline{y}) = \sum_{i=1}^N J_i(\underline{c}_i, \underline{y}_i)$$

Modelul subprocesului (7) poate fi rescris în forma :

$$\underline{y}_i = \underline{f}_i(\underline{c}_i, \underline{\alpha}_i) = \bar{\underline{f}}_i(\underline{c}_i) + \underline{\alpha}_i; \quad i = \overline{1, N} \quad (9)$$

Rezultă

$$\min_{\underline{c} \in \mathcal{C}} \{ J(\underline{c}, \underline{y}) = J(\underline{c}, \underline{y}(\underline{c}, \underline{\alpha})) \} \quad (10)$$

Presupunînd că toate variabilele de ieșire \underline{y} care intervin în formularea criteriului de performanță (10) sînt măsurabile, problema de estimare a vectorului parametrilor $\underline{\alpha}$ este :

$$\underline{f}_i(\underline{c}_i, \underline{\alpha}_i) = \underline{y}_i^* \rightarrow \underline{\alpha}_i; \quad i = \overline{1, N} \quad (11)$$

Soluția acestei probleme este imediată :

$$\underline{\alpha}_i = \underline{y}_i^* - \bar{\underline{f}}_i(\underline{c}_i); \quad i = \overline{1, N}$$

unde \underline{y}_i^* este valoarea măsurată a vectorului de ieșire al subsistemului i .

Datorită diferențelor posibile între model și realitate, în mod inevitabil, apar interacțiuni între problema de optimizare definită de (10) și cea de estimare a parametrilor definită de (11).

Pentru a ocoli aceste interacțiuni și a decupla cele două probleme, vectorul $\underline{\alpha} = [\underline{\alpha}_1^T \dots \underline{\alpha}_N^T]^T$ este înlocuit prin vectorul auxiliar $\underline{\sigma} = [\underline{\sigma}_1^T \dots \underline{\sigma}_N^T]^T$ în probleme de optimizare, iar vectorul \underline{c} este înlocuit cu vectorul auxiliar $\underline{v} = [\underline{v}_1^T \dots \underline{v}_N^T]^T$ în problema de estimare a parametrilor. Ca urmare, apar următoarele restricții egalitate suplimentare :

$$\underline{v} = \underline{c} \quad (12)$$

$$\underline{\sigma} = \underline{\alpha} \quad (13)$$

Din (10) și (12) se redefinește problema de optimizare :

$$\min_{\underline{c} \in \mathcal{C}} J(\underline{c}, \underline{y}(\underline{c}, \underline{\sigma})) \quad (14)$$

Ținând seama de (11) și (13) problema de estimare a parametrilor devine :

$$\underline{y}_1 = \underline{f}_1(\underline{v}_1, \underline{\alpha}_1) \quad (15)$$

Pentru rezolvarea problemei integrate de optimizare și estimare a parametrilor, definită de (14), (15) și restricțiile (12) și (13), se poate defini Lagrangeanul :

$$\mathcal{L} = J(\underline{c}, \underline{f}(\underline{c}, \underline{\sigma})) + \underline{\lambda}^T (\underline{v} - \underline{c}) + \underline{\mu}^T (\underline{\sigma} - \underline{\alpha}) + \underline{\eta}^T (\underline{f}(\underline{v}, \underline{\alpha}) - \underline{y}^*) \quad (16)$$

unde $\underline{\lambda}, \underline{\mu}, \underline{\eta}$ reprezintă multiplicatorii Lagrange.

Presupunând că derivatele diferitelor funcții există și sint continue, din aplicarea condițiilor de staționaritate, rezultă problema de optimizare modificată :

$$\min_{\underline{c} \in \mathcal{C}} \{J(\underline{c}, \underline{f}(\underline{c}, \underline{\sigma})) - \underline{\lambda}^T \underline{c}\} \quad (17)$$

unde, tot din condițiile de staționaritate, vectorul $\underline{\lambda}$ se calculează cu :

$$\underline{\lambda} = \left(\frac{\partial f^T}{\partial v} - \frac{\partial y^{*T}}{\partial v} \right) \left(\frac{\partial f^T}{\partial \alpha} \right)^{-1} \left(\frac{\partial J}{\partial \sigma} \right) \quad (18)$$

Ținând cont de forma particulară a problemei modificate de optimizare, este o posibilă rezolvare descentralizată :

$$\min_{\underline{c}_1 \in \mathcal{C}_1} \{J_1(\underline{c}_1, \underline{\sigma}_1) - \underline{\lambda}_1^T \underline{c}_1\} \quad (19)$$

La rîndul său problema de estimare a parametrilor se rezolvă ca și mai sus, direct, în mod descentralizat :

$$\underline{\alpha}_1 = \underline{y}_1 - \underline{f}_1(\underline{v}_1) \quad (20)$$

Rolul coordonatorului constă în calculul vectorului folosind (18) și repartizarea valorilor $\underline{\lambda}_1$ către subsistemele locale.

Procedura iterativă de rezolvare on-line a problemei integrate de optimizare și estimare a parametrilor este ilustrată în fig. 1. La fiecare pas, k , se estimează local vectorii $\underline{\alpha}_i$; $i = \overline{1, N}$, în urma măsurării ieșirilor rezultate obținute la stabilizarea sistemului după aplicarea referințelor \underline{v}_1 . La nivelul coordonatorului se calculează vectorul $\underline{\lambda}$.

Pe baza transferului semnalelor de coordonare $\underline{\lambda}_i$, la nivelul subsistemelor, se rezolvă problemele modificate de optimizare definite de (20).

Pentru a nu perturba prea mult procesul prin modificări substanțiale ale referințelor, nu se aplică direct soluțiile problemei de optimizare modificate, \underline{c} , ci valorile filtrate :

$$\underline{v}_1^{[k]} = \underline{v}_1^{[k-1]} + K_1^{[k]} (\underline{c}_1^{[k]} - \underline{v}_1^{[k-1]}) \quad (21)$$

COORDONATOR

$$\lambda = \left(\left(\frac{\partial f}{\partial v} \right)^T - \left(\frac{\partial \ddot{y}}{\partial v} \right)^T \left(\frac{\partial \ddot{y}}{\partial \lambda} \right)^{-1} \left(\frac{\partial J}{\partial \sigma} \right) \right)$$

$$\theta_N = \left\{ \frac{\partial f_N}{\partial v}, \frac{\partial y_N^*}{\partial v}, \frac{\partial f_N}{\partial \sigma}, \frac{\partial J_N}{\partial \sigma} \right\}$$

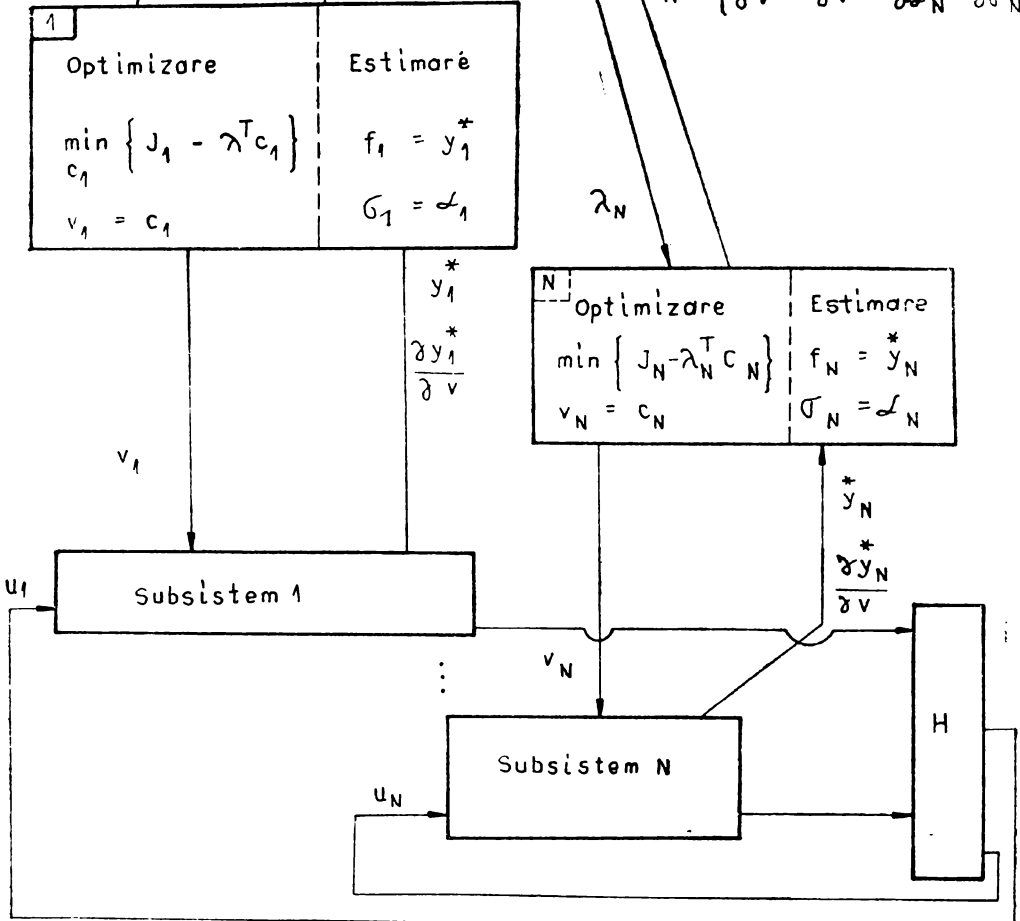


Fig. 4. Rezolvarea on-line a problemei integrate de optimizare și estimare a parametrilor.

unde k este numărul iterației, iar K este o matrice diagonală cu coeficienți pozitivi subunitari. Vom observa în treacăt că folosirea unei matrici K variabile, cu coeficienți adaptivi poate asigura proprietăți sporite de convergență ale procedurii chiar și atunci când funcția procesului $f(\underline{c}, \underline{u})$ este variabilă în timp [3].

Rezultatele prezentate mai sus pot fi extinse și în cazul în care variabilele de interconexiune intervin aditiv în criteriile de performanță locale:

$$J_i(\underline{c}_i, \underline{u}_i, \underline{y}_i) = J_{1i}(\underline{c}_i, \underline{y}_i) + J_{2i}(\underline{u}_i), \quad i=1, N \quad (22)$$

Dacă matricea de cuplaj H_{ij} din (6) conține pe fiecare linie cel mult un element nenul, obținem:

$$J_i(\underline{c}_i, \underline{u}_i, \underline{y}_i) = J_{1i}(\underline{c}_i, \underline{y}_i) + J_{2i} \left(\sum_{j=1}^N H_{ij} \underline{y}_j \right)$$

$$J_i(\underline{c}_i, \underline{y}_i) = J_{1i}(\underline{c}_i, \underline{y}_i) + \sum J_{2ij}(\underline{y}_j) \quad (23)$$

Criteriul de performanță global devine

$$J(\underline{c}, \underline{y}) = \sum_{i=1}^N \tilde{J}_i(\underline{c}_i, \underline{y}_i) = \sum_{i=1}^N (J_{1i}(\underline{c}_i, \underline{y}_i) + \sum_{j=1}^N J_{2ij}(\underline{y}_j)) \quad (24)$$

În consecință se pot defini subprobleme locale de optimizare rezolvabile în mod descentralizat:

$$\min_{\underline{c}_i \in \mathcal{C}_i} \{ \tilde{J}_i(\underline{c}_i, \underline{y}_i) - \lambda_i^T \underline{c}_i \} \quad (25)$$

Exemplul numeric prezentat în (11.1/D3) se referă la un sistem compus din 5 subsisteme.

Dacă folosirea modelelor deliberat simplificate permite conducerea proceselor dinamice sau statice cu caracteristici prea complicate (sau variabile în timp) pentru a „merita” o modelare precisă, pe aceeași direcție de cercire a dificultăților legate de construirea unor modele prea complicate, se situează conducerea situațională a proceselor complexe (11.1/D1).

Pentru un sistem complex o situație cuprinde efectul global al factorilor interni și externi și exprimă relația dintre sistemul complex și mediul său. Dificultatea conducerii sistemelor complexe este legată de discrepanța dintre numărul în mers de situații și numărul limitat de strategii de conducere. De multe ori, conducerea situațională este redusă la o problemă de clasificare și se atașează strategii de conducere unei clase de situații. Dezvoltând unele rezultate proprii, anterioare, Șpal și Madrasz (11.1/D1) propun o formă generalizată de clasificare și anume clasificarea „fuzzy” (vagă) prin care se abandonează asigurarea unei situații date unei singure clase. Deosebit de interesante sînt cele două aplicații ale abstracției analizate: conducerea situațională a unui sistem de producție robotizat și cea a unui sistem energetic.

Alte aspecte legate de conducerea sistemelor energetice sînt redate de Singh [11.1/D6] de astă dată folosind modele matematice ale procesului și metode de descompunere și coordonare.

O formulare deveritată clasică a problemelor de optimizare legate de sistemele mari a fost propusă de Tamura [4] și este cea descrisă de modelul, restricțiile și criteriul de performanță de mai jos:

$$\underline{x}(k+1) = A \underline{x}(k) + \sum_{j=0}^m B_j \underline{m}(k-j) + \underline{w}(k) \quad (26)$$

$$\underline{v}_m \leq \underline{v} \leq \underline{v}_M \quad \underline{v} = [\underline{x}^T, \underline{m}^T]^T \quad (27)$$

$$J = \sum_{k=0}^{K-1} (\underline{v}(k) - \underline{v}_d(k))^T R (\underline{v}(k) - \underline{v}_d(k)) \quad (28)$$

unde \underline{w} este vectorul perturbațiilor, \underline{v}_m și \underline{v}_M arată limitele inferioare și respectiv superioare pentru variația lui \underline{v} , iar \underline{v}_d indică o valoare dorită.

Metoda de rezolvare constă dintr-o procedură ierarhizată iterativă; se bazează pe construirea unui Lagrangean în care restricția egalitate (26) apare înmulțită cu vectorul de costare \underline{p} . La nivel superior se ajustează traiectoria lui \underline{p} pînă la satisfacerea (aproximativă a ecuației 28), iar la nivel inferior se găsesc soluțiile analitice pentru $\underline{x}(k)$ și $\underline{m}(k)$ [4].

În (11.1/D6) se trec în revistă unele aplicații mai noi sau mai vechi din domeniul conducerii traficului urban, conducerii traficului pe autostrăzi, distribuția optimală a resurselor de apă, asigurarea calității apei râurilor, arătîndu-se cum astfel de probleme pot fi formulate (sau reduse la) forma de mai sus. De menționat în treacăt realizarea la I.C.I. a unui pachet de programe denumit DISPATCHER (sau DICOTR-C), care rezolvă probleme legate de conducerea sistemelor cu producție continuă interconectate prin intermediul rezervoarelor [5]. Produsul, care implementează o variantă îmbunătățită a metodei lui Tamura pe minicalculatoarele românești, este construit sub forma unui sistem de sprijinire a deciziilor la nivel operativ și a fost deja implementat sau testat în rafinării, combinate siderurgice și sisteme hidrotehnice.

Alte aplicații tratate în cadrul secțiunii se referă la conducerea sistemelor telefonice bazat pe sisteme de așteptare cu reacție (11.1/D4) sau pe organizare pe niveluri ierarhice (11.1/D2).

În concluzie, în cadrul secțiunii s-au prezentat o serie de comunicări foarte interesante, unele avînd caracter de sinteză (11.1/D2) și (11.1/D6) accentul fiind pus pe încercarea de a formaliza și rezolva probleme practice din diferite domenii. Totuși, după cum arăta Singh (11.1/D6), rezultatele prezentate se referă încă doar la simulări, nu și la implementări.

BIBLIOGRAFIE

1. F.G. Filip (1978). Sisteme complexe : teorie și aplicații, AMC, 39, p. 189—193.
2. P. Roberts (1977). Algorithms for integrated system optimization and parameter estimation. Electronic Letters, 14 (5), p. 196—197.
3. F.G. Filip, D.A. Donciulescu, M. Muratcea (1984). O comparație a metodelor de coordonare în problema integrată de estimare a parametrilor și optimizare în regim staționar a sistemelor. Buletinul României de Informatică V(3), p. 31—45.
4. H. Tamura (1975). Decentralized optimization for distributed lag models of discrete systems. Automatica, 11, p. 593—602.
5. D.A. Donciulescu, F.G. Filip (1983). DISPATCHER : a decision support system based on hierarchical approach. 5th Intern. Conf. on Contr. Syst. and Comp. Sci, București vol. I, p. 189—194.

5. APLICAȚII NESTANDARD ALE SISTEMELOR MARI

În cadrul cercetărilor în domeniul conducerii sistemelor mari, o preocupare importantă se referă la stabilirea modelului matematic folosit. În majoritatea cazurilor, pentru sistemele mari nu se dispune de model matematic, sau modelul matematic disponibil este de dimensiuni mari, ceea ce ridică probleme în alegerea metodei de conducere și a configurației de echipamente folosite (în sensul unui cost ridicat al instrumentației de măsurare și control). O serie de lucrări ale secțiunii 11.1/E tratează aspecte legate de această problemă.

Astfel, Davison și Solomon (11.1/E4) tratează problema alegerii structurii de comandă pentru reglarea robustă a sistemelor multivariabile mari al căror model matematic nu este cunoscut. Selectarea intrărilor care urmează a fi folosite pentru comanda sistemului și a tipului structurii de comandă este una dintre cele mai critice probleme ale conducerii sistemelor

mari. Lucrarea tratează această problemă pentru procese care pot fi descrise de modele lineare invariante, de tipul ecuației de stare și ieșire următoare :

$$\dot{x} = Ax + Bu + E\omega \quad (1)$$

$$y = Cx + Du + F\omega, \quad e \stackrel{\Delta}{=} y - y_{ref}$$

unde $x \in \mathbb{R}^n$ este vectorul de stare, $u \in \mathbb{R}^m$ este vectorul intrărilor, $y \in \mathbb{R}^r$ este vectorul ieșirilor, $e \in \mathbb{R}^r$ este vectorul erorii în sistem, y_{ref} este vectorul referințelor, iar $\omega \in \mathbb{R}^Q$ este vectorul perturbațiilor în sistem.

Se consideră cazul proceselor stabile în buclă deschisă. În aceste condiții, sînt prezente o serie de structuri de comandă (distingute pentru cazurile perturbațiilor și referințelor constante, sau de tip rampă), cuprînd matrici care pot fi totdeauna determinate pe baza unor experimente realizate în regim staționar. Rezultă că nu este necesară cunoașterea modelului matematic al procesului nici pentru verificarea condițiilor de existență a soluției problemei de reglare robustă a procesului respectiv, nici pentru implementarea structurilor de comandă.

Structurile de comandă sînt parametrizate prin intermediul unor scalari, care sînt determinate prin acordarea on-line a reguletoarelor. Configurațiile de comandă obținute prin aplicarea structurilor de comandă prezentate în lucrare pot fi complet centralizate, parțial descentralizate sau complet descentralizate, în funcție de structura matricilor obținute experimental.

Se arată că dacă matricea funcție de transfer evaluată la frecvență zero, este dominantă diagonală, atunci, pentru rezolvarea problemei servomecanismului robust poate fi folosită totdeauna o structură de comandă descentralizată. De asemenea sînt prezentate o serie de exemple (comanda unui furnal cu 4 intrări și 4 ieșiri, a unei colcană de distilare cu 3 intrări și 3 ieșiri și a unei instalații chimice cu 5 intrări și 4 ieșiri), pentru ilustrarea rezultatelor obținute prin aplicarea structurilor de comandă prezentate în lucrare, unor procese al căror model matematic nu este cunoscut.

Hodžic și Šiljak (11.1/E2) prezintă o metodă de proiectare linear pătratică gaussiană (LPG) bazată pe teoria sistemelor ierarhizate, care permite reducerea considerabilă atât a calculului off-line, cît și a celui on-line.

O metodă standard de reducere a dimensiunii problemei, în cazul sistemelor algebrice rare mari, este reordonarea variabilelor și ecuațiilor pentru a aduce sistemul în forma bloc-triunghiulară. Sistemul global poate fi apoi rezolvat ca o secvență de subsisteme de dimensiuni mai mici, rezultînd o scădere considerabilă a cerințelor de calcul. Folosind această metodă, a fost dezvoltat un algoritm bazat pe teoria grafurilor, care poate fi folosit pentru transformarea unui sistem dinamic într-un sistem ierarhizat format din subsisteme structural observabile și controlabile. Pentru această structură, lucrarea prezintă o metodă de proiectare a unor estimatoare și reguletoare de ordin mic, pentru fiecare subsistem separat, pornind de sus în jos în cadrul structurii ierarhizate.

Se consideră sistemul discret \mathcal{O} compus din s subsisteme :

$$\begin{aligned} \mathcal{O} : x_i(t+1) &= \sum_{j=1}^i A_{ij}x_j(t) + \sum_{j=1}^i \Gamma_{ij}w_j(t) \\ y_i(t) &= \sum_{j=1}^i C_{ij}x_j(t) + v_i(t) \end{aligned} \quad (2)$$

$i = 1, 2, \dots, s$

care este reprezentată de o structură ierarhizată (bloc triunghiulară), compusă din subsistemele :

$$\begin{aligned} \mathcal{O}_i : x_i(t+1) &= A_{ii}x_i(t) + \Gamma_{ii}w_i(t) \\ y_{ii}(t) &= C_{ii}x_i(t) + v_i(t) \end{aligned} \quad (3)$$

$i = 1, 2, \dots, s$

Eroarea de estimare $\tilde{x}(t+1 | t) = x(t) - \hat{x}(t+1 | t)$ poate fi scrisă sub forma :

$$\tilde{x}(t+1 | t) = A\tilde{x}(t | t) + \Gamma w(t) \quad (13)$$

Din ecuația (10) rezultă că matricea de covarianță a erorii de actualizare staționare este de forma :

$$P^* = (I - K^*C)P^-(I - K^*C)^T + K^*R_vK^{*T} \quad (14)$$

unde P^- reprezintă matricea de covarianță a erorii de predicție staționare :

$$P^- = AP^+A^T + \Gamma R_w \Gamma^T \quad (15)$$

Fiecărui subsistem S_i i se asociază un indice de performanță de forma :

$$J_i = \text{tr} P_{ii}, \quad i=1, 2, \dots, s \quad (16)$$

Acești indici de performanță sînt optimizați separat, pentru fiecare subsistem, pornind de sus în jos în structura ierarhizată a sistemului.

Această procedură nu conduce, desigur, la un estimator global optimal, dar acest estimator este suboptimal și stabil, în condiții destul de lejere.

Se obține :

$$K_{ii}^* = \left(\sum_{j=1}^i P_{ij} C_{ij} \right) \left[\sum_{j=1}^i C_{ij}^* \sum_{k=1}^i P_{jk} C_{ik}^{*T} + \left(\sum_{j=1}^i D_{ij}^* \right) R_v \left(\sum_{j=1}^i D_{ij} \right)^T \right]^{-1} \quad (17)$$

unde :

$$C_{ij} = \begin{cases} C_{ii}, & i=j \\ - \sum_{k=j}^{i-1} \sum_{l=k}^{i-1} C_{il} K_{lk} C_{kj} + C_{ij}, & i > j \end{cases} \quad (18)$$

$$D_{ij}^* = \begin{cases} D_{ii}, & i=j \\ - \sum_{k=j}^{i-1} C_{ik} K_{kj} D_{jj}, & i > j \end{cases}$$

Metoda de proiectare ierarhizată a estimatorilor prezentată în lucrare este o generalizare a unei metode propuse de Pichai, Sezer și Šiljak.

Datorită structurii bloc triunghiulare a estimatorului $\hat{\mathcal{O}}^\oplus$, stabilitatea fiecărui estimator de subsistem, $\hat{\mathcal{O}}_i^\oplus$, implică stabilitatea estimatorului global, descris de ecuația (5), care poate fi rescrisă sub forma :

$$\hat{\mathcal{O}}^\oplus : \hat{x}(t+1 | t) = (A - K^\oplus C) \hat{x}(t | t-1) + K^\oplus y(t) \quad (19)$$

Algoritmul bazat pe teoria grafurilor, folosit pentru a reprezenta sistemul original \mathcal{O} sub forma ierarhizată (3), poate demonstra că fiecare pereche (A_{ii}, C_{ii}) este structural observabilă și $[A_{ii}, \Gamma_{ii} (R_w^{ii})^{-1}]$ este structural controlabilă. Acest fapt, ca și metoda de optimizare ierarhizată, garantează că toate estimatorile subsistemelor $\hat{\mathcal{O}}_i^\oplus$ sînt stabile, ceea ce asigură stabilitatea lui $\hat{\mathcal{O}}^\oplus$.

Pentru problema de comandă stohastică, se consideră sistemul \mathcal{O} descris de :

$$\mathcal{O} : x_i(t+1) = \sum_{j=1}^i A_{ij} x_j(t) + \sum_{j=1}^i B_{ij} u_j(t) + w_j(t) \quad (20)$$

$$y_i(t) = \sum_{j=1}^i C_{ij} x_j(t) + v_i(t) \\ i=1, 2, \dots, s$$

unde $u(t) \in \mathbb{R}^{m_1}$ sînt intrările deterministe, sau :

$$\begin{aligned} \mathcal{S}: x(t+1) &= Ax(t) + Bu(t) + w(t) \\ y(t) &= Cx(t) + v(t) \end{aligned} \quad (21)$$

unde B are aceeași structură bloc triunghiulară ca și Γ . Indicele de performanță asociat sistemului \mathcal{S} este descris de :

$$J_1 = \lim_{T \rightarrow \infty} \frac{1}{T} \left\{ \sum_{t=0}^{T-1} x_i^T(t) Q_x^{ii} x_i(t) + u_i^T(t) Q_u^{ii} u_i(t) \right\} \quad (22)$$

$$i=1, 2, \dots, s$$

unde matricile constante Q_x^{ii} sînt ne-negativ definite, iar Q_u^{ii} sînt pozitiv definite.

Metoda de descompunere folosită, bazată pe teoria grafurilor, asigură satisfacerea automată a condițiilor de controlabilitate structurală și observabilitate structurală a perechilor (A_{ii}, B_{ii}) , respectiv $[A_{ii}, (Q_x)^{ii}]$. Urmează un proces de optimizare secvențială a subsistemelor $\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_s$, obținându-se vectorul de comandă global $u^*(t) = [u_1^{*T}(t), u_2^{*T}(t), \dots, u_s^{*T}(t)]^T$ sub forma :

$$u^*(t) = -L^* x(t) \quad (23)$$

unde $\hat{x}(t|t)$ este definit în (9), iar matricea de amplificare L^* este bloc triunghiulară. În lucrare este demonstrată stabilitatea sistemului global obținut prin metoda de proiectare LPG ierarhizată prezentată.

În (11.1/E6), Jamshidi și Wang abordează problema conducerii sistemelor mari cu întârzieri, pe baza teoriei sistemelor ierarhizate, cu aplicații în domeniul utilizării resurselor de apă. Această problemă a mai fost abordată pe baza principiului maximului. Sistemul mare nelinier cu multiple întârzieri este descris sub forma :

$$\begin{aligned} \dot{X} &= F(x(t), x(t-\tau_1), \dots, x(t-\tau_p), U(t-h_1), \dots, U(t-h_q)) \quad (24) \\ t &\geq t_0, \quad \tau_p \geq \dots \geq \tau_1 \geq \dots \geq \tau_1 \\ h_q &\geq \dots \geq h_1 \geq \dots \geq h_1 \\ X(t) &= X_0(t) \quad t_0 - \tau_p \leq t \leq t_0 \\ U(t) &= U_0(t) \quad t_0 - h_q \leq t \leq t_0 \end{aligned}$$

unde $X(t) \in \mathbb{R}^n$ și $U(t) \in \mathbb{R}^m$ sînt vectorii de stare, respectiv de comandă, $\tau_i, i=1, 2, \dots, p$ și $h_j, j=1, 2, \dots, q$ sînt constante pozitive reprezentînd întârzierile. $F(\cdot)$ este o funcție continuu diferențială în raport cu argumentele, t_0 este momentul inițial iar $X_0(t)$ și $U_0(t)$ sînt funcții inițiale specificate. Indicele de performanță de minimizat este :

$$J = 1/2 X^T(t_f) S X(t_f) + 1/2 \int_{t_0}^{t_f} [X^T(t) Q(t) X(t) + U^T(t) R(t) U(t)] dt \quad (25)$$

unde S și Q sînt matrici diagonale pozitiv semidefinite, iar matrice $R(t)$ este diagonală pozitiv definită și continuă pe porțiuni. Se pune problema determinării vectorului de comandă $U(t)$, $t_0 \leq t \leq t_f$, care la momentul final t_f , avînd starea finală liberă $X(t_f)$, să satisfacă (24) și să minimizeze indicele de performanță (25).

Modelul sistemului (24) este linearizat folosind metoda dezvoltării în serii Taylor în jurul punctului nominal (X_n, U_n) .

Se obține următoarea aproximație lineară a sistemului :

$$\dot{x}(t) = A_0(t) x(t) + \sum_{k=1}^p A_k(t) x(t-\tau_k) + B_0(t) u(t) + \sum_{l=1}^q B_l(t) u(t-h_l) \quad (26)$$

cu traiectoriile inițiale :

$$\begin{aligned} \mathbf{x}(t) &= \mathbf{x}_0(t) - \mathbf{x}_n(t) = \mathbf{x}_0(t) \dots t_0 - \tau_p \leq t \leq t_0 \\ \mathbf{u}(t) &= \mathbf{U}_0(t) - \mathbf{U}_n(t) = \mathbf{u}_0(t) \dots t_0 - \tau_q \leq t \leq t_0 \end{aligned}$$

iar indicele de performanță (27) devine :

$$\begin{aligned} \hat{J} &= \frac{1}{2} \mathbf{x}^T(t_f) (\mathbf{S}\mathbf{x}(t_f) + 2\mathbf{S}\mathbf{X}_n(t_f)) + \frac{1}{2} \int_{t_0}^{t_f} [\mathbf{x}^T(\mathbf{Q}(t)\mathbf{x}(t) + 2\mathbf{Q}(t)\mathbf{X}_n(t)) + \\ &+ \mathbf{u}^T(t)(\mathbf{R}(t)\mathbf{u}(t) + 2\mathbf{R}(t)\mathbf{U}_n(t))] dt \end{aligned} \quad (27)$$

Sistemul este descompus în N subsisteme, prin partiționarea vectorului de stare \mathbf{x} și al celui de comandă, \mathbf{u} , (de cele mai multe ori, în cazul sistemelor mari, descompunerea în subsisteme a sistemului global, e naturală) :

$$\dot{\mathbf{x}}_1(t) = \mathbf{A}_{01}(t)\mathbf{x}_1(t) + \mathbf{B}_{01}(t)\mathbf{u}_1(t) + \mathbf{Z}_1(t) \quad t \geq t_0 \quad (28)$$

$$\mathbf{x}_1(t) = \mathbf{x}_{10}(t) \quad \tau_p \leq t \leq t_0$$

$$\mathbf{u}_1(t) = \mathbf{u}_{10}(t) \quad t_q \leq t \leq t_0$$

$$\mathbf{z}_1(t) = \hat{\mathbf{A}}_{01}(t)\mathbf{x}(t) + \hat{\mathbf{B}}_{01}(t)\mathbf{u}(t) + \sum_{k=1}^p \mathbf{A}_{k1}(t)\mathbf{x}(t - \tau_k) + \sum_{l=1}^q \mathbf{B}_{l1}(t)\mathbf{u}(t - h_l) = \mathbf{z}_{1e}(t) \quad (29)$$

unde $\hat{\mathbf{A}}_{01} = \text{Bloc-diag}[\mathbf{A}_{11}, \mathbf{A}_{12}, \dots, \mathbf{A}_{1,1-1}, 0, \mathbf{A}_{1,1+1}, \dots, \mathbf{A}_{1N}]$. Se consideră de asemenea că :

$$\min_{\mathbf{u}} \hat{J} = \sum_{i=1}^N \min \hat{J}_i \quad (30)$$

unde :

$$\begin{aligned} \hat{J}_i &= \frac{1}{2} \mathbf{x}_i^T(t_f) (\mathbf{S}_i \mathbf{x}_i(t_f)) + \frac{1}{2} \int_{t_0}^{t_f} [\mathbf{x}_i^T(t) (\mathbf{Q}_i(t)\mathbf{x}_i(t) + 2\mathbf{Q}_i(t)\mathbf{x}_{n1}(t)) + \\ &+ \mathbf{u}_i^T(t) (\mathbf{R}_i(t)\mathbf{u}_i(t) + 2\mathbf{R}_i(t)\mathbf{U}_{n1}(t))] dt \end{aligned} \quad (31)$$

Lucrarea propune un algoritm de determinare a comenzii suboptimale a problemei definite de (26) și (27), pe baza metodei predicției interacțiunilor. Termenii de interacțiune și cei de întârziere sînt considerați ca intrări de perturbație, obținându-se subsisteme fără întârzieri. În cadrul structurii de calcul ierarhizate obținute, la nivelul coordonatorului se realizează simple înlocuiri, ceea ce facilitează folosirea metodei. În lucrare este prezentat un exemplu de aplicare a algoritmului în cazul unui sistem de utilizare a resurselor de apă.

În continuare, sînt prezentate pe scurt celelalte lucrări ale acestei secțiuni.

Mohler, Kolodziej și Bugnon analizează în [11.11.1] modele pentru sisteme bilineare cuplate, care, în unele cazuri, apar în mod natural, sau sînt folosite ca aproximații converabile ale unor sisteme de nelinearități mai mari.

Santiago, Lange și Jamshidi prezintă în [11.1E5] preocupări legate de modelarea și comanda structurilor spațiale flexibile mari. Lucrarea prezintă problemele de bază, caracteristicile și dificultățile inerente în modelarea și comanda unor astfel de structuri.

În [11.1E5] este prezentat un sistem întrebare-răspuns pentru interfețe pentru baze de date inteligente. Relațiile de similaritate a datelor dintr-o bază de date avînd un model bazat pe similaritate psihologică, sînt reprezentate sub formă de rețea. O funcție de generalizare permite propagarea în rețea a informației obținute din răspunsurile de similaritate ale utilizatorului. Folosind funcția de generalizare, calculatorul determină tipul generic de întrebare pe care trebuie să o pună în continuare utilizatorului, în scopul folosirii eficiente a informației. Utilizatorului i se permite să dea răspunsuri nedefinite sau imprecise, care reflectă lipsa de cunoștințe. Prin acest proces de întrebare-răspuns, sistemul bazat pe cunoștințe facilitează specificarea de către utilizator a unor cereri de date relevante, ca și regăsirea de date din baza de date.

ANALIZA ȘI PROIECTAREA ASISTATE DE CALCULATOR A SISTEMELOR

Dr. ing. A. Varga

Ing. M. Dumitriu

Ing. A. Moangă

I.T.C.I.

PACHETE DE PROGRAME DE PROIECTARE ASISTATĂ DE CALCULATOR

În secțiune a 11.2/A se prezintă câteva pachete de programe destinate problematicilor asociate proiectării asistate de calculator a sistemelor automate. Lucrările [11.2/A4] și [11.2/A5] prezintă pachete interactive pentru identificarea experimentală a proceselor tehnologice. Pachetul descris în lucrarea [11.2/A3] este destinat determinării modelelor de ordin redus, ca și pentru validarea modelelor rezultate. Un pachet interactiv pentru proiectarea sistemelor în domeniul frecvență utilizând metoda sintezei totale este descris în [11.2/A2].

Lucrarea [11.2/A1] descrie un sistem integrat de pachete de programe interactive care acoperă aspectele de identificare, analiză, proiectare și simulare și oferă facilități pentru implementarea on-line a rezultatelor proiectării. Lucrarea [11.2/A6] prezintă o bibliotecă de subprograme FORTRAN portabile, care implementează o serie de algoritmi numerici performanți pentru rezolvarea unor probleme matematice de bază ce intervin în analiza, modelarea și proiectarea sistemelor automate multivariabile. Vom prezenta în cele ce urmează pe scurt caracteristicile principale ale acestor pachete.

Realizarea unei aplicații de conducere automată cu calculator a unui proces tehnologic necesită în general parcurgerea a patru etape succesive: (1) modelarea procesului; (2) proiectarea sistemului de conducere automată; (3) implementarea algoritmilor de conducere numerică directă pe calculatorul de proces; și (4) analiza performanțelor sistemului de conducere proiectat. Pentru realizarea primelor două etape se pot utiliza pachete de programe interactive de modelare și proiectare asistată de calculator. Parcurgerea etapei a treia presupune elaborarea programelor (de regulă în limbajul de asamblare a calculatoarelor de proces) care efectuează calculele specifice algoritmilor de conducere. Etapa a patra realizează atât o testare a corectitudinii programelor de conducere, cât și o validare a rezultatelor proiectării în condițiile de funcționare în timp real a sistemului de conducere cu calculatorul. De regulă etapele (2)÷(4) se parcurg repetat, pînă la găsirea valorilor optime ale parametrilor care apar în algoritmi de conducere.

Lucrarea 11.2/A1 prezintă un sistem integrat de pachete de programe interactive, denumit CACOS, care realizează pe lângă funcțiile specifice etapelor de modelare și proiectare asistate de calculator și pe aceea de generare automată a programelor de conducere în limbajul de asamblare al minicalculatorului NOVA3. Sistemul CACOS are la bază pachetele MAT (Calculule matriciale), TS (Serii de timp), PIPACK (Identificarea parametrilor) și DPAGS (Analiza și proiectarea sistemelor) cu ajutorul cărora se pot efectua funcțiile de modelare și proiectare.

Generarea automată a codului sursă pentru programul de conducere se face cu programul PCODE. Pentru obținerea unui timp de calcul cât mai redus, PCODE generează instrucțiunile de adunare și înmulțire în virgulă mobilă numai pentru elemente nenule ale matricilor care intervin în algoritmi de conducere. Codul generat de FCCLL este compilat și înglobat în programul RCOR care efectuează funcția de conducere în timp real. Alte funcțiuni pe care

RCOR le poate realiza sint : culegere de date, afișare parametri, listare serii de timp, introducere valori pentru referințe etc.

Sistemul CACOS a fost utilizat în proiectarea unui sistem automat de levitație magnetică. Timpul necesar proiectării și experimentării (aproximativ 1 oră) a fost mult mai mic decât timpul necesar codificării manuale a algoritmilor de conducere. Soluția adoptată în sistemul CACOS reprezintă una dintre căile de promovare a rezultatelor noi ale teoriei sistemelor în aplicații practice.

În lucrarea 11.2/A2 este prezentat un pachet interactiv, denumit TSPSP, care implementează o metodologie de proiectare în domeniul frecvență. Problema de proiectare, denumită *problema sintezei totale* (PST), se formulează în termenii a trei aplicații lineare fundamentale : P, M și T definite după cum urmează

$$P : U(s) \rightarrow Y(s)$$

$$M : R(s) \rightarrow U(s)$$

$$T : R(s) \rightarrow Y(s)$$

În care U(s), Y(s) și R(s) sînt spații vectoriale raționale ale transformatorilor (Laplace sau Z) ale intrărilor, ieșirilor și referințelor, respectiv. Pentru baze fixate, P, M și T se reprezintă prin matrici de transfer avînd următoarele semnificații : P(s) este matricea de transfer a instalației, M(s) este matricea de transfer dintre intrări și referințe, iar T(s) este matricea de transfer dintre ieșiri și referințe. Între aceste matrici există relația evidentă

$$T(s) = P(s)M(s)$$

Ecuția de mai sus se poate reformula ca o problemă de calcul al nucleului unei aplicații, în forma

$$[P \quad -I] \begin{bmatrix} M \\ T \end{bmatrix} = 0$$

PST constă în alegerea perechilor (M, T) din nucleul aplicației $[P \quad -I]$, astfel încît să satisfacă anumite cerințe de proiectare și realizabilitate fizică. Stabilitatea internă impune ca matricile M(s) și T(s) să fie stabile și proprii (gradele numitorilor mai mari decît sau egale cu gradele numărătorilor). Prin manipularea bazei calculate pentru nucleu, se determină (M, T) astfel încît să conducă la răspunsuri ale ieșirilor și la comenzi asociate, dorite de proiectant.

După ce M(s) și T(s) au fost determinate, proiectarea continuă prin găsirea structurii de comandă adecvate. Configurația generală este prezentată în fig. 1, unde G și H sînt matrici de transfer care se aleg astfel încît să-l realizeze pe M.

Din relația intrare-ieșire rezultată din fig. 1, obținem ecuația de bază care trebuie satisfăcută

$$(I + PGH)^{-1}PG = PM$$

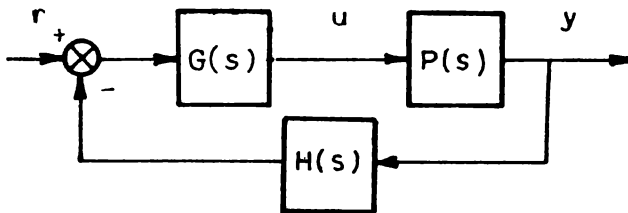


Fig. 1. Configurația de comandă pentru PST.

Obiectivul proiectantului este de a distribui dinamica din M, în G și H astfel încît să rezulte o sinteză care este insensibilă la variații parametrice în instalație. Dacă alegem $H=0$ și $G=M$, structura de comandă este în buclă deschisă. O altă posibilitate este alegerea lui $H=I$, deci utilizarea unei reacții unitare. Se obține imediat

$$G = M(I - PM)^{-1}$$

O altă posibilitate de alegere, bazată pe minimizarea sensibilității sistemului în buclă închisă, este descrisă în lucrare.

Pachetul TSPSP constă dintr-un număr de programe care realizează calculele implicate de PST. Astfel, sînt prevăzute programe de calcul a bazei nucleului unui operator rațional sau polinomial, de manipulare a bazei, de calcul cu matrici de transfer, de realizare minimală. Verificarea proiectării se face prin simulare. Comunicarea între programe se face prin fișiere care utilizează un format comun de memorare a datelor. Programele sînt interactive și permit efectuarea proiectării plecînd de la descrierea originală a instalației și terminînd cu obținerea structurii finale a compensatorului.

În lucrarea 11.2/A3 este descris un pachet interactiv de programe pentru determinarea modelelor de ordin redus ale sistemelor complexe. Pachetul de programe are la bază o bibliotecă de subprograme FORTRAN ce implementează principalele metode cunoscute de reducere a ordinului în domeniul frecvență sau în domeniul timp. O altă bibliotecă de subprograme este destinată testării calității modelului redus, în comparație cu modelul original. O rutină specială este prevăzută pentru predicția ordinului modelului pornind de la criteriile de reducere deterministe sau utilizînd proceduri iterative.

Funcțiile principale ale pachetului de programe sînt :

- 1) transformări între reprezentările de stare și reprezentările în frecvență ;
- 2) determinarea ordinului modelului redus ;
- 3) determinarea modelului de ordin redus printr-una dintre metodele de reducere disponibile ;
- 4) simularea sistemului redus pentru semnale de intrare date și analiza în domeniul frecvență a modelului redus ;
- 5) evaluarea valabilității procedurii de reducere utilizate.

În lucrare se prezintă o aplicație de reducere a ordinului unui model al unui sistem energetic folosind atât metode în domeniul timp, cît și în frecvență. Modelul redus a fost determinat în vederea proiectării unui regulator suboptimal de ordin cît mai mic.

Pachetul este implementat pe un calculator WAX 11/780.

Lucrarea 11.2/A4 prezintă pachetul de programe interactive SATER, destinat identificării sistemelor cu o intrare și o ieșire. Acest pachet a fost dezvoltat în primul rînd în scop didactic, pentru a se pune la dispoziția studenților o gamă largă de metode de identificare în vederea comparării acestora. Datele necesare pentru determinarea modelelor se pot obține fie prin simulări, fie prin culegere directă din procesul real. Pachetul are o structură modulară și se poate extinde relativ ușor cu alte programe.

Pachetul SATER are trei componente principale : supervisorul, programele aplicative și rutinele de serviciu. *Supervisorul* constă dintr-un executiv care are rolul de a lansa programul aplicativ selectat de utilizator și dintr-un modul de întrebare-interpretare, care solicită datele și parametrii necesari rulării programului aplicativ dorit. *Programele aplicative* formează partea cea mai mare a pachetului SATER și implementează diferite metode de estimare a parametrilor, de determinare a ordinului modelului, de analiză, de simulare etc. *Rutinele de serviciu* efectuează operațiile de intrare-ieșire și includ subsistemul pentru întrebări-răspunsuri și subsistemul pentru grafică pe terminal sau ploter.

Vom prezenta în continuare principalele operații implementate în pachetul SATER.

- a) Culegerea și prelucrarea primară a datelor, analiza semnalelor
 - eșantionare pe maximum 8 canale ale sistemului de culegere DEC LPS
 - filtrarea datelor cu filtru Butterworth de ordinul 3
 - prelucrarea statistică a semnalelor
 - analiza armonică a semnalelor
 - analiza spectrală
- b) Generarea datelor simulate

Se pot utiliza șase tipuri de semnale pentru simulări

- semnale sinusoidale
- semnale polinomiale (impuls, treaptă, rampă etc.)
- zgomot alb uniform
- secvență maximală
- zgomot alb gaussian
- serii de timp arbitrare memorate

c) Teste pentru ordinul modelului, validare

Testele bazate pe rezultatele estimării parametrilor includ :

— simplificare poli-zeroouri

— netezimea ajustărilor recursive ale parametrilor pentru ordine crescătoare ale modelului

— reziduali, erori de predicție, funcții de eroare

— testul abaterii semnalelor

— testul produsului matricilor de moment.

Testele care nu utilizează estimarea parametrilor includ :

— funcția de pierdere (Woodside)

— testul de singularitate a produsului matricilor de momente ale datelor.

d) Metode de estimare a parametrilor, estimarea marginii Cramér-Rao

— metoda matricii extinse

— metoda celor mai mici pătrate generalizate

— metoda variabilelor instrumentale

— metoda verosimilității maxime

e) Conversii ale modelelor discrete în modele continue

— cu extrapolator de ordin zero

— cu extrapolator de ordin unu

— cu extrapolator triunghiular

— transformare — Z bilineară

f) Analiza proceselor discrete și continue

— calculul poli-zerourilor

— trasarea diagramelor Nyquist sau Bode pentru sisteme continue

— trasarea locului rădăcinilor

Pachetul SATFER este implementat pe un minicalculator DEC PDP 11/60 sub sistemul de operare RSX 11M.

În lucrarea 11.2/A 5 se prezintă pachetul IP-DEM destinat identificării sistemelor cu mai multe intrări și o ieșire, utilizându-se descrierea prin ecuații cu diferențe. Pachetul este format din trei programe independente. P-1 efectuează culegerea și prelucrarea primară (statistică) a datelor. Semnalele de perturbație ale procesului pot fi generate manual sau automat, sub formă de semnale pseudoaleatoare binare. P-2 efectuează identificarea modelului utilizând datele rezultate de la P-1. Sunt disponibile patru metode de identificare : (1) metoda variabilelor instrumentale cu ieșiri întârziate ; (2) metoda celor mai mici pătrate ; (3) metoda filtrării neliniare aproximative ; și (4) metoda verosimilității maxime. Modelele obținute cu P-2 se pot compara cu programul P-3 în vederea selectării modelului celui mai adecvat.

Pachetul IP-DEM este implementat pe un microcalculator DEC LSI 11/23 și este scris în limbajul FORTRAN. În lucrare este demonstrată utilizarea pachetului atât în cazul unui model simulat cât și în cazul determinării modelului unei coloane de distilare binare.

Lucrarea 11.2/A 6 prezintă pachetul de programe BIMAS elaborat în cadrul Institutului pentru tehnică de calcul și informatică din București. BIMAS este un pachet de subprograme FORTRAN, destinat rezolvării numerice a problemelor matematice de bază care intervin în analiza și proiectarea asistată de calculator a sistemelor automate lineare. Algoritmii implementați au fost riguros selectați în vederea satisfacerii atributelor de generalitate, fiabilitate, stabilitate numerică, precizie și eficiență. BIMAS are la bază pachetele bine cunoscute de algebră liniară BLAS, LINPACK și EISPACK, și poate fi considerat ca o extindere a acestor pachete la probleme de calcul matricial. BIMAS constă din aproximativ 50 de subprograme originale sau adaptate din literatura de specialitate și circa 25 de subprograme din pachetele amintite mai sus.

Pachetul BIMAS conține subprograme pentru următoarele probleme :

- calculul și ordonarea formei Schur reale (FSR) a unei matrici pătrate ;
- calculul și ordonarea formei Schur reale generalizate a unei perechi de matrici pătrate ;
- calculul formei bloc-diagonale a unei matrici ;
- rezolvarea ecuațiilor matriciale lineare de tip Sylvester și Liapunov ;
- stabilizarea sistemelor lineare multivariabile ;
- alocarea polilor sistemelor lineare multivariabile ;
- rezolvarea ecuațiilor matriciale algebrice de tip Riccati ;
- calculul exponențialelor matriciale.

Vom prezenta pe scurt câteva caracteristici ale pachetului BIMAS. Majoritatea algoritmilor implementați utilizează FSR a unei matrici, calculată prin transformări ortogonale. Utilizarea largă a acestor transformări mărește fiabilitatea calculului numeric. Pentru multe probleme, programele evaluează numere de condiționare adecvate sau informații echivalente.

În elaborarea pachetului s-a utilizat o abordare modulară pentru rezolvarea problemelor complexe. Majoritatea calculului implică efectuarea unei secvențe de apeluri la subrutinele pachetului. Modularitatea sporește capabilitățile de segmentare, astfel încât pachetul se poate utiliza eficient chiar pe minicalculatoare.

Portabilitatea pachetului a fost asigurată prin utilizarea limbajului FORTRAN standard, fără să se exploateze facilitățile oferite de implementări particulare ale compilatoarelor. În acest scop, toți parametrii dependenți de mașină, cum sînt de exemplu precizia relativă sau radix-ul mașinii, sînt furnizați prin intermediul a două rutine speciale. La transportarea pachetului de pe un echipament pe altul, doar aceste rutine sînt modificate.

Pachetul BIMAS a fost dezvoltat pe minicalculatorul CORAL-4011 în dublă precizie. Subprogramele sînt documentate extensiv, într-un mod unitar. Pentru toate rutinele de bază s-au elaborat programe de test. Pachetul BIMAS a fost dezvoltat în continuare prin realizarea pachetului de subprograme BIMASC, destinat rezolvării problemelor specifice de calcul ce intervin în modelarea, analiza, proiectarea și simularea sistemelor lineare multivariabile. Utilizîndu-se subprogramele pachetelor BIMAS și BIMASC, s-a elaborat (tot în cadrul ITCI) un puternic pachet interactiv, CASAD, destinat proiectării asistate de calculator a sistemelor automate. CASAD este implementat pe minicalculatoarele românești și este unul dintre pachetele cele mai performante de proiectare în domeniul timp existente pe plan mondial.

Concluzii. Din prezentarea pachetelor, se evidențiază unele tendințe existente pe plan mondial în elaborarea de software pentru proiectarea asistată de calculator a sistemelor automate :

1) elaborarea de pachete interactive care implementează metodologii complete de identificare, analiză sau proiectare, utilizînd tehnici numerice noi ;

2) utilizarea unor algoritmi robusți și a unui software numeric de calitate în rezolvarea problemelor specifice de automată ;

3) constituirea unor biblioteci de subprograme FORTRAN portabile pentru rezolvarea problemelor de automată, elaborate pe baza unor standarde de programare, utilizare, testare și documentare ;

4) implementarea pachetelor pe echipamente de calcul de tip mini- și micro-calculatoare, cu folosirea intensivă a unor facilități grafice.

Menționăm că țara noastră se află printre țările avansate în domeniul elaborării de pachete de programe destinate proiectării asistate de calculator a sistemelor automate.

METODE DE PROIECTARE ASISTATĂ DE CALCULATOR A SISTEMELOR AUTOMATE

În secțiunea 11.2/C se prezintă câteva metode utilizate în analiza și proiectarea asistată de calculator a sistemelor automate. Lucrarea [11.2/C1] prezintă unele tehnici de bază pentru analiza și evaluarea performanțelor sistemelor proiectate prin metode în domeniul frecvență. O metodă curistică bazată pe grafică interactivă pentru aproximarea caracteristicilor de frecvență a sistemelor cu dinamică complexă este prezentată în [11.2/C2]. Lucrarea [11.2/C3] propune un algoritim numeric stabil pentru calcularea numitorilor de ordin superior care intervin în studiul comportării buclilor de reacție, individuale, ale sistemelor multivariabile. O metodologie de proiectare în domeniul timp a sistemelor cu neliniarități în bucla de reacție este propusă în [11.2/C4], iar în [11.2/C5] se prezintă o abordare prin optimizare multicriterială a proiectării reguletoarelor lineare. Tehnici de analiză structurală a unor proprietăți de bază ale sistemelor lineare sînt propuse în lucrarea [11.2/C6]. Voi trece în revistă pe scurt toate aceste metode și tehnici în cele ce urmează.

Cînd se utilizează un pachet de proiectare asistată de calculator, proiectantul comunică cu calculatorul prin intermediul unei interfețe, ce-i permite interpretarea rezultatelor furnizate de calculator și specificarea operațiilor pe care el dorește să le efectueze. Orice informație pre-

zentată de calculator proiectantului, care-i permite evaluarea cantitativă a unui aspect relevant al procesului de proiectare, poartă numele de *indicator*. Astfel de indicatori sînt intim legați de modul de lucru conceptual al proiectantului, mod de lucru care, la rîndul său, trebuie să fie strîns legat de posibilitățile de calcul prevăzute pentru efectuarea procedurilor de sinteză cu calculatorul. Pe baza indicatorilor, proiectantul va trebui să fie în măsură să identifice sursele dificultăților care apar în rezolvarea problemei de proiectare și va trebui să facă compromisurile cele mai favorabile între performanțele dorite ale sistemului automat și costul necesar pentru realizarea acestor performanțe.

În lucrarea 11.2/C1 se prezintă o serie de indicatori pentru unele caracteristici de bază ale unui sistem multivariabil, cum sînt: stabilitatea, performanța, robustețea sau sensibilitatea sistemului. Indicatorii propuși prezintă generalizări netriviiale ale conceptelor de *amplificare și fază*, larg utilizate în teoria clasică a reglării automate.

Considerăm un sistem cu m intrări și m ieșiri descris de o matrice de transfer $G(s)$ ale cărei elemente sînt funcții raționale în variabila complexă s . Indicatorul de stabilitate a sistemului în buclă închisă se definește în termenii valorilor proprii ale matricei $G(s)$, numite și *amplificări caracteristice*, pentru valori ale lui s pe un contur Nyquist. Acest indicator apare fie sub forma *diagramelor Nyquist generalizate* obținute prin reprezentarea locurilor valorilor proprii, fie sub forma *diagramelor Bode generalizate* obținute prin reprezentarea modulelor și fazelor amplificărilor caracteristice.

Pentru studierea performanțelor sistemului în buclă închisă, se utilizează ca indicatori valorile singulare sau *amplificările principiale* ale operatorilor $F(s) = 1 + Q(s)$ și $L(s) = 1 + Q^{-1}(s)$, unde $Q(s)$ este matricea de transfer a sistemului în buclă deschisă. Robustețea stabilității sistemului, ca și sensibilitatea sistemului la perturbații parametrice pot fi caracterizate convenabil în termenii amplificărilor principale.

Indicatorii propuși în lucrare servesc la fundamentarea unei metodologii de proiectare *quasi-clasice* a sistemelor automate multivariabile. Această metodologie este adecvată transpunerii pe calculator sub forma unei proceduri interactive de proiectare asistată de calculator.

În lucrarea 11.2/C2 se prezintă o metodă euristică bazată pe grafică interactivă pentru determinarea unor funcții de transfer simple pentru sisteme continue. Modelul inițial al procesului rezultă de multe ori prea complicat pentru a putea fi manipulat comod. Astfel de modele, cu factori iraționali sau de ordin mare, rezultă frecvent în modelarea teoretică a proceselor. Scopul problemei de reducere a modelului constă în determinarea unei aproximări suficient de exacte care să surprindă proprietățile esențiale ale sistemului. Modelele inițiale acceptate de metoda propusă pot avea drept componente funcții de transfer iraționale, funcții de transfer raționale cu ordin ridicat al numitorului. De asemenea, o descriere neparametrică prin valori discrete ale răspunsului în frecvență este acceptată.

Reducerea modelului se bazează pe reprezentarea grafică a diagramelor Bode: *amplificare-frecvență și fază-frecvență*, într-un domeniu de frecvențe fixat, pentru modelul inițial al sistemului. Modelul redus se obține sub forma

$$G(s) = \frac{v}{s^r} \frac{\prod_i \left(1 + \frac{s}{z_i}\right) \prod_j \left(1 + \frac{2\zeta_j}{\omega_j} s + \frac{s^2}{\omega_j^2}\right)}{\prod_m \left(1 + \frac{s}{p_m}\right) \prod_n \left(1 + \frac{2\tilde{\zeta}_n}{\tilde{\omega}_n} s + \frac{s^2}{\tilde{\omega}_n^2}\right)} e^{-T_n s} \quad (1)$$

Procedura de reducere pornește cu aproximarea asimptotică a diagramei amplificare-frecvență cu drepte de pantă $k \cdot 20$ dB/decadă (k număr întreg) utilizînd facilități de grafică interactivă. Proiectantul efectuează această etapă prin fixarea frecvențelor de tăiere sau a coeficienților de amortizare ζ și a frecvențelor caracteristice ce intervin în termenii pătraticei din (1). Decizia proiectantului se validează la fiecare pas de către calculator prin reprezentarea suprapusă a graficului aproximativ rezultat. Printr-o procedură de încercare-erocare se potrivesc valorile caracteristice astfel încît în final să rezulte o aproximare cît mai bună a caracteristicii inițiale. Etapa a doua constă în adăugarea unor factori suplimentari, de regulă întârzieri, pentru găsirea unei aproximări cît mai bune și pentru caracteristica fază-frecvență. Metoda se dovedește utilă în multe cazuri practice și se poate utiliza ușor de proiectanții familiarizați cu utilizarea diagramelor Bode.

În lucrarea 11.2/C3 se prezintă un algoritm numeric stabil pentru calcularea numitorilor de ordin superior (NOS). Acești numitori intervin în studierea comportării buclă cu buclă a sistemelor multivariabile cu reacție proporțională de la ieșire. NOS se definesc ca determinanți obținuți prin expandarea polinomului caracteristic al unui sistem în buclă închisă în termenii coeficienților matriciei de reacție. Ei constituie generalizări ale polinomului de la numitor și se demonstrează că reprezintă polinomul caracteristic al sistemului pentru amplificări infinite. Prin urmare, ei se pot utiliza în examinarea buclelor de reacție individuale când există amplificări infinite în celelalte bucle.

Fie $G(s)$ matricea de transfer a unui sistem cu m intrări și n ieșiri și fie factorizarea sa sub forma de produse de matrici polinomiale

$$G(s) = L^{-1}(s) H(s)$$

sau

$$L(s)y = H(s)u$$

unde u și y sînt intrările și ieșirile sistemului respectiv. Dacă legea de comandă are forma

$$u = -Ky$$

atunci polinomul caracteristic al sistemului în buclă închisă este

$$\chi(s) = \det [L(s) + H(s)K]$$

În cazul general există m^2 bucle de reacție. Însă pentru introducerea noțiunii de NOS este suficient să considerăm K de formă diagonală

$$K = \text{diag} [k_{ii}]$$

unde $k_{ii} \neq 0, i = 1, \dots, p$ și $k_{ii} = 0, i = p+1, \dots, m$. Atunci polinomul caracteristic al sistemului se poate scrie într-o formă expandată în termenii coeficienților k_{ii} în modul următor

$$\chi(s) = \Delta(s) + \sum_{i=1}^p k_{ii} N_{u_i}^{y_i} + \sum_{j=2}^p \sum_{i=1}^{j-1} k_{ii} k_{jj} N_{u_i u_j}^{y_i y_j} + \dots + \left(\prod_{i=1}^p k_{ii} \right) N_{u_1 \dots u_p}^{y_1 \dots y_p}$$

Coefficienții $N_{u_1 \dots u_j}^{y_1 \dots y_j}$ se definesc ca numitori de ordin superior și se pot exprima ca determinanți ai unor matrici formate din elementele matricilor $L(s)$ și $H(s)$. În expresia de mai sus, $\Delta(s)$ este polinomul caracteristic al sistemului în buclă deschisă.

Metoda de calcul propusă în lucrare, spre deosebire de metodele existente, evită evaluarea determinanților. NOS rezultă ca produsul polilor și zerourilor unor sisteme de ordin redus, sisteme obținute utilizând algoritmul numeric stabil de determinare a structurii spectrale a unui sistem multivariabil propus de Van Dooren. Metoda prezentată pune în evidență conexiunea care există între NOS și zerourile de transmisie ale sistemului.

Lucrarea 11.2/C4 descrie aplicarea în proiectarea sistemelor automate multivariabile neliniare a unor margini funcționale ale răspunsurilor în timp ale erorilor rezultate prin lineări-zări. Structura sistemului este prezentată în fig. 2.

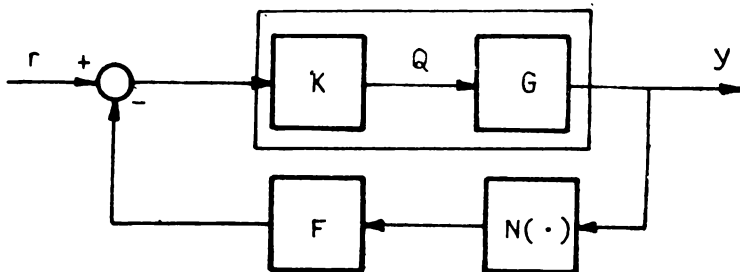


Fig. 2. Configurația sistemului nelinear.

Componentele sistemului includ elementul de pe legătura directă cu m-intrări și m-ieșiri care poate combina modelul G al procesului și compensatorul K. Pe bucla de reacție găsim o reacție lineară proporțională F și o nelinearitate statică de măsurare N(·). Se presupune că funcția componentă $N_{ij}(\cdot)$ se poate descompune în forma

$$N_{ij}(x) = L_{ij}x + E_{ij}(x) + V_{ij}(x)$$

unde L_{ij} este o amplificarea lineară pură, $E_{ij}(\cdot)$ este o funcție mărginită de un sector satisfăcând

$$|E_{ij}(x)| \leq \bar{E}_{ij}|x|, \quad \forall x \in \mathbb{R}$$

iar $V_{ij}(\cdot)$ este o funcție nelineară absolut mărginită satisfăcând

$$|V_{ij}(x)| \leq \bar{V}_{ij}, \quad \forall x \in \mathbb{R}$$

Fie $y_i(t)$ și $y_{Li}(t)$, $i=1, \dots, m$ componentele ieșirilor sistemului inițial (nelinear) și ale sistemului linearizat obținut pentru $E_{ij}=0$ și $V_{ij}=0$. În lucrare se deduc, utilizând concepte ale analizei funcționale, un set de funcții crescătoare $\Phi_i(t)$, $i=1, \dots, m$ care satisfac relațiile

$$y_{Li}(t) - \Phi_i(t) \leq y_i(t) \leq y_{Li}(t) + \Phi_i(t)$$

Această relație are o interpretare grafică simplă, reprezentând o pereche de curbe echidistante față de $y_{Li}(t)$ între care este situat răspunsul sistemului nelinear $y_i(t)$.

O proprietate importantă a marginilor $\Phi_i(t)$ este că depind numai de parametrii sistemului linearizat. În acest fel, proiectarea sistemului se poate face utilizând procedurile bine puse la punct ale teoriei lineare ale sistemelor, urmând ca prin evaluarea marginilor $\Phi_i(t)$ să se verifice performanța sistemului nelinear. În lucrare se prezintă o procedură interactivă de proiectare asistată de calculator bazată pe această idee. Proiectarea se face în domeniul timp și se poate efectua atât în cazul sistemelor continue, cât și discrete.

În lucrarea 11.2/C5 se prezintă o abordare a proiectării reguletoarelor optime pentru sisteme lineare constante prin tehnici de optimizare multicriteriale. Considerăm sistemul

$$\dot{x} = Ax + Bu, \quad x(t_0) = x_0$$

unde $x(t) \in \mathbb{R}^n$ este vectorul de stare, $u(t) \in \mathbb{R}^m$ este vectorul de comandă și $t \in [t_0, t_f]$. Criteriul de performanță care trebuie minimizat se alege de forma

$$I = x^T(t_f) S x(t_f) + \int_{t_0}^{t_f} [x^T(t) Q x(t) + u^T(t) R u(t)] dt$$

în care matricile S, Q și R se presupun diagonale, cu R pozitiv definită, iar S și Q pozitiv semi-definite. Alternativ, putem considera următoarea formulare a problemei de optimizare cu $N=2n+m$ criterii

$$J_1 = \min x_1^2(t_f), \quad J_{n+j} = \min \int_{t_0}^{t_f} x_j^2 dt, \quad J_{2n+k} = \min \int_{t_0}^{t_f} u_k^2 dt$$

pentru $i=1, \dots, n$; $j=1, \dots, n$; $k=1, \dots, m$.

Fie vectorul ponderilor

$$w^T = (s_1, \dots, s_n, q_1, \dots, q_n, r_1, \dots, r_m)$$

format din elementele diagonale ale matricilor S, Q și R, respectiv. Din teoria optimizării multicriteriale avem că toate comenzile optime $u(t)$ rezultate prin minimizarea criteriului $I = w^T J$, ($J=(J_1, \dots, J_N)$) cu $w_i \geq 0$ sînt „soluții neinferioare“ ale problemei multicriteriale. Cu alte cuvinte, dacă u^* este soluția optimă care rezolvă problema de minimizare inițială cu criteriul pătratic pentru $s_i, q_i, r_i > 0$ date, atunci nu există nici o comandă \hat{u} astfel încît

$$J_i(\hat{u}) \leq J_i(u^*) \quad i=1, \dots, N$$

cu cel puțin una dintre inegalități fiind strictă.

Datorită convexității problemei multicriteriale, toate soluțiile neinferioare se pot obține prin varierea componentelor lui w în mulțimea $\Omega = \{w_i \mid w_i > 0\}$. Problema care se ridică este alegerea soluției convenabile din mulțimea legilor de reglare neinferioare utilizând, de exemplu, o procedură de evaluare interactivă prin simulări repetate. Aceste legi de reglare se determină prin calcularea soluției unei probleme de optimizare linear-pătratică de dimensiune n .

Problema determinării comenzii care simultan optimizează criteriul de performanță pătratic și minimizează sensibilitatea sistemului în buclă închisă în raport cu perturbații parametrică se formulează ca o problemă de minimizare bicriterială. Rezolvarea acestei probleme presupune extinderea sistemului de ecuații diferențiale și rezolvarea unei probleme de optimizare linear-pătratică de ordin $2n$.

Procedurile de rezolvare s-au implementat sub forma unui pachet de programe interactiv. Pe lângă modulele de calcul, pachetul dispune de un modul de interacțiune, care implementează o procedură de interferență lingvistică bazată pe utilizarea algoritmilor relaționali „fuzzy”. Această procedură evaluează, pe baza informațiilor lingvistice primite de la utilizator, modificările care se impun în componentele vectorului de ponderi w în vederea reluării calculului de optimizare. Programele sînt scrise în limbajul PASCAL MT+ pe un microcalculator cu procesor cu 8 biți și lucrează sub sistemul de operare CP/M.

Lucrarea 11.2/C6 prezintă utilizarea tehnicilor de manipulare algebrică în testarea unor proprietăți structurale ale sistemelor lineare descrise de ecuații de stare.

Procedurile propuse pentru testarea controlabilității, observabilității, identificabilității etc. utilizează informațiile referitoare la structura de zerouri fixate în matricile sistemului. O parte dintre algoritmii propuși implică rezolvarea unui set de ecuații polinomiale cu mai multe necunoscute. Metoda de bază pentru aceasta se bazează pe teoria eliminării. Utilizarea acestor tehnici necesită, de regulă, un mare volum de calcul și se recomandă numai pentru sisteme de dimensiuni relativ mici.

PROIECTAREA ASISTATĂ DE CALCULATOR A SISTEMELOR CU EȘANTIONARE

Proiectarea asistată de calculator a sistemelor cu eșantionare prezintă un interes deosebit în contextul folosirii din ce în ce mai intense a structurilor de conducere cu microprocesoare a proceselor tehnologice.

În *lucrarea 11.2/B1* este prezentat un sistem de simulare interactiv ISAC-D (Interactive Simulation System for Automatic Control-Digital) destinat analizei și sintezei sistemelor de conducere numerică. Sistemul de programe este realizat folosind conceptul de diagrame-bloc introduse prin dialogul utilizatorului cu un terminal grafic inteligent (HITAC-G760) legat la un calculator mai puternic (HITAC-M200H). Este introdus un concept de organizare ierarhică a descrierii diagramei-bloc pe 3 nivele, ceea ce permite descrierea celor mai complexe scheme de conducere. Sînt furnizate blocuri predefinite atât pentru părțile analogice, cît și pentru cele numerice ale schemelor de conducere. De asemenea, sistemul de programe conține și funcții pentru calculul răspunsului în timp al sistemului dinamic hibrid, răspunsul la frecvență, locul rădăcinilor.

În cadrul configurației descrise, terminalului grafic îi revin sarcinile de pre- și postprocesare date și de dialog om-mașină, în vederea minimizării încărcării calculatorului mare. Un meniu de funcții disponibile este oferit utilizatorului în partea de preprocesare în care se introduc datele și se descrie modelul sistemului hibrid (continuu-numeric). Partea de postprocesare oferă funcții grafice, cum ar fi rotații, extinderi și retrănscrieri de imagini.

Sistemul de programe are următoarele componente :

- a) software de bază ;
- b) subsistemul de introducere a diagramei-bloc ;
- c) subsistemul de analiză a răspunsului în timp ;
- d) subsistemul de analiză a răspunsului la frecvență ;
- e) subsistemul de gestiune a bazei de date.

Software-ul de bază controlează fluxul de date între terminalul grafic și calculatorul gazdă și gestionează programele utile, inclusiv de grafică.

Subsistemul de gestiune a bazei de date a fost special dezvoltat și tratează o bază de date simplă, cu structură arborescentă.

Subsistemul de intrare a diagramelor-bloc permite construirea acestora pe terminajul video funcție de comenzile și structura descrisă de utilizator.

Se pot folosi elemente dinamice, algebrice lineare, algebrice nelinare, logice, elemente de intruziune, convertoare A/N și N/A. Pentru a simula aplicații mari se permite introducerea a cel mult 30 diagrame-bloc și cel mult 500 de elemente pentru descrierea unui sistem de conducere numerică.

Intrucât pentru sistemele de conducere de mari dimensiuni afișarea lor este greoaie, necesitând multe pagini de display, este dificil pentru utilizator să coreleze legăturile între pagini. Pot apărea erori de construcție și descriere a diagramelor de dimensiuni mari. Pentru evitarea lor s-a introdus conceptul de descriere structurată a diagramelor-bloc. Diagramele-bloc structurate sînt clasificate după cum urmează :

- (i) *nivelul overview* (general), care descrie întreg sistemul pe subsisteme componente ;
- (ii) *nivelul detaliu*, care poate conține pînă la 28 de pagini și conține componentele fiecărui subsistem ; fiecare pagină se corectază cu cele adiacente, prin intrări/ieșiri ;
- (iii) *nivelul rutină* (pagină), pe o singură pagină de ecran se grupează elemente care se consideră o singură funcție de transfer la nivelul dci.

Separarea părții continue de cea discretă în descriere se face cu ajutorul blocurilor A/N, N/A cu cîte 6 intrări/ieșiri fiecare. Pe lângă frecvența de eșantionare, blocurile sînt prevăzute și cu timp de calcul necesar algoritmului de reglare, ercarea de trunchiere etc., făcînd astfel ca aceste canale de trecere a informației să se apropie foarte mult de implementarea lor reală, cu microprocesoare.

Pentru a obține rezultate numerice stabile în procesul de simulare a funcționării sistemului de conducere, autorii au ales o metodă de discretizare a părții continue (nelineară în general), cu ajutorul metodei tabloului, după cum urmează :

$$\dot{x}_1(t) = a_{11}x_1(t) + a_{21}x_{\alpha_1}(t) + a_{31}x_{\beta_1}(t) \quad (1)$$

unde α_1, β_1 sînt indici pentru mărimile de intrare ;

$$x_j(t) = a_{2j}x_{\alpha_2}(t) + a_{3j}x_{\beta_2}(t) + a_{4j}x_{\gamma_2}(t) + \delta_j \quad (2)$$

unde δ_j este intrarea în proces

$$x_1(t) = f_1(x_{\alpha_3}(t), x_{\beta_3}(t)) \quad (3)$$

Aplicînd o formulă de discretizare într-un pas, ecuația (1) devine :

$$\begin{aligned} [1 - h(1 - \theta)] x_1(k+1) - h(1 - \theta) a_{21}x_{\alpha_1}(k+1) - h(1 - \theta) a_{31}x_{\beta_1}(k+1) = \\ = (1 + h\theta a_{11}) x_1(k) + h\theta a_{21}x_{\alpha_1}(k) + h\theta a_{31}x_{\beta_1}(k) \end{aligned} \quad (4)$$

unde θ este o pondere între 0 și 1.

Intervalul $[0, T]$ se divide în $0, t_1, \dots, t_n$ puncte.

Pentru ecuația (3) se aplică iterații de tip Newton-Raphson pornind de la valorile $x_1(k), x_{\alpha_3}(k)$ și $x_{\beta_3}(k)$

Obținem :

$$\begin{aligned} x_1(k+1) - \left(\frac{\partial f_1}{\partial x_{\alpha_3}} \right) x_{\alpha_3}(k+1) - \left(\frac{\partial f_1}{\partial x_{\beta_3}} \right) x_{\beta_3}(k+1) = \\ = x_1(k) - \left(\frac{\partial f_1}{\partial x_{\alpha_3}} \right) x_{\alpha_3}(k) - \left(\frac{\partial f_1}{\partial x_{\beta_3}} \right) x_{\beta_3}(k). \end{aligned} \quad (5)$$

Rezultă astfel tabloul valorilor discrete

$$A(k) \cdot X(k+1) = B(k) \quad (6)$$

unde X este un vector N -dimensional (x_1, \dots, x_N) cu N — numărul de blocuri, $A(k) \in \mathbb{R}^{N \times N}$; $B(k) \in \mathbb{R}^N$.

Pentru partea discretă este furnizată o tehnică de calcul secvențial bazată pe observația că un algoritmic numeric de reglare este un model discret în timp. Deci o diagramă-bloc a unui regulator numeric poate fi transformată în ecuații cu întârzieri (cu unitatea de timp dată de pasul de calcul), în ecuații algebrice lineare și nelineare și în ecuații logice. Secvența pornește de la valorile inițiale pe ecuațiile cu întârzieri, valorile intrărilor și ieșirilor din convertoarele A/N și N/A. Comunicația între cele 2 părți analogice și numerice se face prin intermediul intrărilor și ieșirilor acestor convertoare.

O altă componentă a sistemului descris se referă la calculul răspunsului la frecvență a sistemului de reglare, care-și găsește utilitatea pentru sisteme complexe de reglare, pentru care nu se poate calcula atât de ușor funcția de transfer a acestuia. Metoda propusă de la sistemul discret :

$$(zI - A_D) X_D(z) = D_D U(z) \tag{7}$$

$$Y(z) = C_D X_D(z) \tag{8}$$

și rezolvă o ecuație algebrică de tipul :

$$[\exp(j\omega T_s) I - A_D] X_D(\omega) = (D_D)_i \tag{9}$$

unde : T_s este perioada de eșantionare ; matricile A_D , $(D_D)_i$ sînt matricile sistemului dinamic discretizat, X_D — starea sistemului dinamic discret, $(D_D)_i$ — e coloana (i) a matricii (D_D) .

Din (9) se obține :

$$Y_i = (C_D)_i (X_D)_R + (C_D)_i (X_D)_I = Y_R + Y_I \tag{10}$$

unde : $(C_D)_i$ este coloana i a matricii C_D ;

R, I — atributul real sau imaginar al părții lui X_D .

Se obțin astfel ușor diagramele Bode pentru întreg sistemul de conducere. În plus, folosind tehnici QR se trasează și poziția rădăcinilor sistemului, obținându-se o analiză a stabilității sistemului cînd acesta este liniarizat și discretizat (ec. 7).

În lucrarea 11.2/B2, autorii prezintă o metodă exactă pentru evaluarea ciclurilor limită a frecvențe subarmonice în sisteme numerice nelineare, obținută prin modificarea metodei pentru sistemele continue cu nelinearitate de tip releeu. Metoda se poate aplica la orice tip de nelinearitate, spre deosebire de cazul continuu.

Metoda propusă implică găsirea soluțiilor unor ecuații algebrice nelineare și presupune existența unei estimări inițiale a soluțiilor. Pulsurile funcției de nelinearitate au lățimea $\Delta t_1 = T_s$. Într-o extrapolară de ordin zero precede elementul nelinear, ieșirea $y(t)$ a acestuia va fi un tren de impulsuri a cărui expresie Fourier este :

$$y_i(t) = \frac{h\Delta t_1}{T} + \frac{h}{\pi} \sum_{n=1}^{\infty} \frac{1}{n} [\sin n\omega\Delta t_1 \cos n\omega(t-t_1) + (1 - \cos n\omega\Delta t_1) \sin n\omega(t-t_1)] \tag{11}$$

Deci ieșirea sistemului va deveni :

$$c_i(t) = \frac{h\Delta t_1 G(0)}{T} + \frac{h}{\pi} \sum_{n=1}^{\infty} \frac{g_n}{n} [\sin n\omega\Delta t_1 \cos(n\omega t - n\omega t_1 + \Phi) + (1 - \cos n\omega\Delta t_1) \sin(n\omega t - n\omega t_1 + \Phi)] \tag{12}$$

În continuare autorii arată că pentru a obține condițiile de oscilație, trebuie aflată amplitudinea ciclului limită la intrarea în elementul nelinear. Deci :

$$-x [(K-1) T_s] = c [(K-1) T_s] \tag{13}$$

unde : $K=1, 2, \dots, n$;

T_s este perioada de eșantionare.

Din expresiile celor 2 membri ai ecuației (13) rezultă ecuațiile algebrice nelineare care trebuie rezolvate. De cele mai multe ori procedura iterativă de căutare a soluțiilor este convergentă chiar dacă punctul inițial nu este bine precizat.

Pentru a estima valorile perioadelor ciclurilor limită din sistem se folosește o funcție de descriere a ansamblului eșantionator, extrapolator de ordin zero și element nelinear. Această funcție de descriere este valabilă pentru cicluri limită care au perioada un multiplu întreg al perioadei de eșantionare.

Funcția de descriere evaluată prin această metodă poate fi afișată grafic ca o dreaptă ce trece prin origine și face un unghi α cu axa pe diagrama Nyquist. Locul de intersecție al hodografului $G(j\omega)$ cu $-\frac{1}{N(a, \omega)}$ va da ciclul limită.

Programele dezvoltate pentru rezolvarea acestor probleme sînt în număr de 3 și permit estimarea ciclurilor limită în astfel de sisteme nelineare cu eșantionare și calculul soluțiilor exacte.

Astfel, programul SAMDF.F77 implementează metoda funcției de descriere pentru a putea estima valorile inițiale perioadelor ciclurilor limită, care pot fi folosite în celelalte 2 programe pentru aflarea soluțiilor exacte.

Utilizatorul descrie funcția Laplace a instalației și selectează nelinearitatea dintr-o tabelă dată de program. Pe lângă metoda funcției de descriere, programul mai trasează și diagramele Nyquist și Bode pentru partea lineară a sistemului. Se efectuează o căutare automată a intersecțiilor într-un domeniu dat al frecvenței. Funcția de grafică interactivă permite dialogul și trasarea succesivă a diagramelor, permițîndu-se astfel obținerea ușoară a estimărilor. Celelalte 2 programe SSAM.FTN și ASAM.FTN implementează metoda exactă de găsire a ciclurilor limită cu simetrie impară (SSAM.TFN) sau asimetrice (ASAM.FTN).

APLICAȚII ALE PROIECTĂRII ASISTATE DE CALCULATOR

Secțiunea 11.2/1) prezintă aplicații ale proiectării asistate de calculator în chimie [11.2/D2], robotică [11.2/D3, 11.2/D4] și acționări hidraulice [11.2/D5]. Lucrarea [11.2/D2] prezintă un sistem de programe care simulează reglarea pH-ului pentru o clasă generală de procese chimice. Sistemul asigură posibilitatea folosirii algoritmilor clasici sau adaptivi de comandă, precum și identificarea în timp real a procesului condus. Programele pot fi folosite în două moduri: a) pentru proiectarea structurilor de comandă ale unei instalații existente; b) pentru analiza unei instalații noi din punct de vedere al comenzii.

Lucrarea [11.2/D3] prezintă o metodă de reprezentare a dinamicii structurilor mecanice de manipulare. Noțiunea de elipsoid de inerție, folosită la caracterizarea unui singur corp rigid, este extinsă la întreaga structură de manipulare. Prin desenarea elipsoidului general de inerție se pot pune în evidență proprietățile distribuției de masă și caracteristicile dinamice ale întregului manipulator. Această metodă este folosită la optimizarea distribuției de masă și la alegerea dimensiunilor structurii mecanice de manipulare.

Lucrarea [11.2/D4] prezintă un sistem de programe pentru modelarea automată a structurilor mecanice de manipulare antropomorfe, folosit în proiectarea asistată de calculator a manipuletoarelor și roboților industriali. Sistemul generează automat, sub formă simbolică, modelele geometrice, cinematice și dinamice ale structurilor mecanice de manipulare. Pot fi analizate structuri mecanice complexe, cu lanțuri cinematice deschise și (sau) închise.

Lucrarea [11.2/D5] prezintă un pachet de programe pentru analiza și sinteza sistemelor de acționare hidraulică și electrohidraulică. Este prezentată o metodă teoretică nouă de modelare matematică a componentelor și a întregului sistem de acționare.

Lucrarea 11.2/1)2 descrie un sistem de programe realizat în limbaj FORTRAN și implementat pe calculatorul VAX 11/780, pentru simularea reglării pH-ului în instalații chimice. Acest sistem poate fi folosit pentru instalații existente la determinarea performanțelor de reglare optimă, dintr-un set de algoritmi de comandă cunoscuți de sistem, la determinarea eficienței implementării unui anumit algoritm de reglare, la analiza modificării performanțelor când se schimbă condițiile de operare.

În cazul proiectării unei noi instalații tehnologice, se pot determina cu ajutorul sistemului de programe condițiile de operare, valorile parametrilor, structura instalației etc. Algoritmii de comandă pe care îi poate folosi sistemul de programe variază de la clasicul PI, la algoritmi adaptivi autoacordabili sau la algoritmi stochastici de comandă suboptimală. În lucrare se pre-

zintă pe larg metoda de modelare matematică a instalațiilor tehnologice folosite de sistemul de simulare. Caracteristicile interactive ale sistemului permit proiectantului să definească, să modifice și să memoreze modelele matematice ale instalațiilor conduse, să efectueze simulări și să memoreze rezultatele de simulare. În lucrare se prezintă și aplicații tipice în care a fost folosit sistemul de programe.

Lucrarea 11.2./D3 prezintă caracteristicile structurale și dinamice ale manipuloarelor într-o formă ușor de analizat și modificat de către proiectantul structurii mecanice. Tehnicile actuale de simulare a dinamicii roboților sînt greu de utilizat, necesitînd un volum mare de calcule pentru diferitele traiectorii și configurații ale robotului. Noua metodă prezentată este eficientă din punctul de vedere al proiectării și evaluării caracteristicilor dinamice ale structurilor mecanice de manipulare.

Extinderea noțiunii de elipsoid de inerție la întreaga structură și folosirea unui sistem grafic puternic reprezintă caracteristicile principale ale sistemului prezentat în lucrare. Elipsoidul de inerție este asociat cu energia cinetică immagazinată de corpul rigid. Pentru definirea elipsoidului de inerție generalizat asociat întregii structuri, se calculează energia cinetică totală a manipulatorului. Aplicînd una din metodele matriciale de calcul a energiei cinetice totale, pentru structura mecanică de manipulare, rezultă :

$$T = 1/2 \dot{q}^t G(\dot{q}) \dot{q}$$

unde T este energia cinetică, \dot{q} , \dot{q}^t — vectorul și vectorul transpus al vitezelor generalizate, G(q) — matricea energiei cinetice, care se definește ca tensorul generalizat de inerție pentru o structură de corpuri rigide. Tensorul generalizat de inerție are o interpretare geometrică, care pune sugestiv în evidență caracteristicile dinamice ale manipulatorului. Se asociază tensorului generalizat de inerție suprafața :

$$u^t G(q) u = 1$$

care reprezintă ecuația unui elipsoid, denumit elipsoidul generalizat de inerție (EGI). Axele principale de inerție ale elipsoidului (EGI) sînt colinare cu vectorii proprii ai matricii G, iar lungimea axelor este funcție de rădăcina pătrată a valorilor proprii.

Deoarece este interesantă dinamica mișcării minii robotului în coordonate carteziene, se calculează matricea G în coordonate carteziene și se desenează mai mulți elipsoizi (EGI) în spațiul de lucru al robotului. Formele grafice ale acestor elipsoizi trasate în diferite poziții ale minii robotului, în spațiul de lucru, pun în evidență caracteristicile dinamice ale structurii mecanice, funcție de poziția și direcția de mișcare a minii. În lucrare se extinde și noțiunea de moment de inerție a unui corp rigid, la momentul de inerție generalizat a unei structuri mecanice de manipulare :

$$h = \dot{q}^t G \dot{q} / \dot{q}^t \dot{q}$$

Momentul de inerție generalizat depinde de poziția în spațiul de lucru și de direcția de mișcare a minii robotului. Valoarea maximă (minimă) a lui h corespunde la valorile proprii maxime (minime) ale matricii G, care corespund la axele mici (mari) ale elipsoidului generalizat de inerție. Viteza de mișcare a structurii (raportată la mina robotului) este minimă în direcția axelor mici ale elipsoidului (EGI). Rezultă deci că din interpretarea formelor grafice (EGI) desenate pentru spațiul de lucru, se pot trage concluzii asupra posibilităților de mișcare ale robotului, asupra punctelor singulare, asupra gradului de singularitate etc.

Forma elipsoidului (EGI), deformarea și rotația lui sînt legate de valerile forțelor Coriolis și centrifugale ale structurii mecanice. În lucrare se demonstrează că pentru zonele din spațiul de lucru, în care elipsoidul (EGI) se transformă într-o sferă, forțele nelineare Coriolis și centrifugale dispar. Efectele inerțiale, variația forțelor Coriolis și centrifugale, în spațiul de lucru, interpretate prin reprezentările grafice ale elipsoidului generalizat de inerție, sînt tot atîtea date utile pentru proiectantul structurii mecanice. În faza de proiectare se modifică dimensiunile geometrice și distribuția de masă, astfel ca elipsoidele de inerție să tindă, în spațiul de lucru, cît mai mult spre sferă. În acest fel se asigură bune proprietăți dinamice pentru structura mecanică de manipulare.

Lucrarea 11.2.D4 prezintă modelarea automată a structurilor mecanice complexe cu lanțuri cinematice deschise și (sau) închise. Pentru automatizarea modelării structurilor mecanice cu lanțuri cinematice închise se asociază manipulatorului un graf orientat (corp rigid—virf de graf; legătură cinematică—arc de graf). În etapa de analiză topologică se determina drumul minim în graf, corespunzător lanțului fundamental de poziționare (succesiunea de corpuri rigide care unesc mina robotului de batiul său) și o bază de cicluri minimale independente, corespunzătoare grupelor minimale (lanțuri cinematice închise caracterizate de o succesiune predeterminată de corpuri, un corp de bază, un operator global de mișcare, o ieșire și una sau mai multe intrări).

Sistemul de programe folosește un fișier cu grupe minimale cunoscute, descrise prin relațiile analitice ale operatorilor globali de mișcare. În prima fază a analizei topologice se pune în evidență lanțul fundamental de poziționare și grupele minimale ale structurii particulare, identificându-se automat formele analitice ale operatorilor globali de mișcare. Dacă nu toate legăturile motoare se află în lanțul fundamental de poziționare, atunci se construiesc automat lanțurile de acționare (succesiunea de corpuri care unesc legăturile motoare cu o legătură din lanțul fundamental de poziționare).

Folosind operatorii globali de mișcare ai grupelor minimale, se generează automat, în formă simbolică, pentru fiecare corp rigid al robotului, vectorul de poziție al originii reperului de coordonate legat de corp, vectorul de poziție al centrului de masă al corpului și matricea de rotație față de sistemul de referință inerțial. Elementele acestor vectori și matrici au ca variabile coordonatele generalizate ale robotului ($q \in R^n$, n — numărul gradelor de libertate). Vectorul și matricea simbolică corespunzătoare minii robotului reprezintă modelul geometric al întregii structuri mecanice de manipulare.

În faza de analiză cinematică se generează automat, folosind derivarea simbolică, matricile de transmitere de ordinul întâi, corespunzătoare vitezei lineare și unghiulare a fiecărui corp, față de sistemul de referință inerțial. Matricile de transmitere corespunzătoare minii robotului formează matricea Jacobian a modelului cinematic.

Cu ajutorul vitezelor unghiulare, a vitezelor lineare ale centrelor de greutate, a maselor și momentelor de inerție ale corpurilor rigide, se calculează energia cinetică a întregii structuri. Folosind rezultatele analizei cinematice și poziționale, aplicate tuturor corpurilor, se calculează și energia potențială a manipulatorului. Cu ajutorul ecuațiilor lui Lagrange, se determină ecuațiile diferențiale de mișcare, respectiv modelul dinamic al structurii mecanice de manipulare. Forma analitică a modelului dinamic rezultat este :

$$A(q)\ddot{q} + B(q)\dot{q}\dot{q} + C(q)\dot{q}^2 = Q_r(q) + Q_m + Q_f(q)$$

unde $A(q) \in R^{n \times n}$ este matricea energiei cinetice sau matricea forțelor de inerție, $B(q) \in R^{n \times n(n-1)/2}$ — matricea forțelor Coriolis, $C(q) \in R^{n \times n}$ — matricea forțelor centrifugale, $Q_r(q) \in R^n$ — vectorul forțelor de rezistență, $Q_m \in R^n$ — vectorul forțelor motoare, $Q_f \in R^n$ — vectorul greutatei proprii, q, \dot{q}, \ddot{q} — coordonatele, vitezele și accelerațiile generalizate, n — numărul gradelor de libertate.

Sistemul de programe este scris în limbajele PASCAL, FORTRAN, MACRO-11 și funcționează pe minicalculatoare de tip DEC PDP-11. Acest sistem cuprinde următoarele funcții :

- descrierea și modificarea grupelor minimale standard ;
- descrierea topologică a manipulatorului sau a robotului industrial ;
- analiza topologică și pozițională ;
- analiza cinematică și dinamică ;
- calculul simbolic general la nivel de matrici (+, —, ×, inversare, derivare etc.) ;
- generarea automată a programelor în limbaj FORTRAN, care descriu modelul dinamic în vederea simulării ;
- simularea numerică ;
- reprezentarea grafică a rezultatelor.

Evident, structurile mecanice formate din lanțuri cinematice deschise sînt un caz particular a structurilor mecanice cu lanțuri cinematice închise.

În încheierea lucrării se prezintă două exemple de modelare, o structură cu două grade de libertate și lanțuri cinematice închise și o structură cu trei grade de libertate și lanțuri cinematice deschise.

Lucrarea 11.2/D5 prezintă un sistem de programe pentru proiectarea instalațiilor de acționare hidraulice și electrohidraulice, folosind modelarea structurată. Modelarea structurată

permite organizarea sistematică a elementelor componente, care produc, controlează și transformă energia hidraulică și informația în instalații. Metoda prezentată permite o bună structurare, o optimizare locală, eliminarea erorilor de modelare și o analiză mai ușoară a instalațiilor complexe. Modelarea efectuată ajută sinteza asistată de calculator a instalațiilor hidraulice și electrohidraulice. Lucrarea prezintă pe larg unitățile elementare folosite, structurile tipice de instalații, descrierea lor dinamică și determinarea modelului matematic pentru analiza dinamică și de regim staționar. Sistemul de programe cuprinde trei părți: preprocesor, procesor, postprocesor. Rolul preprocesorului este de a genera un model de calcul, selectând și asamblând modele matematice ale modulelor elementare. Procesorul analizează în mod opțional regimul staționar, regimul tranzitoriu, răspunsul la frecvență etc.

Sistemele grafice sînt folosite pentru prezentarea rezultatelor într-o formă ușor de analizat. Postprocesorul permite selectarea acelor module elementare, care fac instalația să funcționeze conform parametrilor de proiectare doriți. Sistemul folosește indici globali de calitate, permițînd utilizarea și de criterii economice. Postprocesorul furnizează operatorului o listă de elemente modulare alese, asociată cu o listă a performanțelor obținute de instalația construită din elementele alese. Comunicarea cu sistemul de programe, care funcționează pe un minicalculator CORAL 4011, se poate realiza în mod interactiv. Lucrarea pune în evidență noi concepte și metode de modelare și construcție automată a instalațiilor de acționare hidraulică și electrohidraulică.

INGINERIA SISTEMELOR INDUSTRIALE

Ing. Alexandru Dan Donciulescu

I.T.C.I.

Sub titlul „Ingineria sistemelor industriale“ au fost grupate lucrările, susținute la al IX-lea Congres IFAC, care prezintă tehnici, algoritmi, structuri și metode de proiectare specifice sistemelor de conducere cu calculator a proceselor industriale complexe.

Prima dintre cele două secțiuni (11.3/A) ale colocviului cu titlul de mai sus a conținut 6 lucrări (dintre care lucrarea 11.3/A4 nu este cuprinsă în volumul lucrărilor congresului) axate pe conducerea unităților industriale mari din domeniul metalurgiei, industriei sticlei și energiei, precum și pe prezentarea unor metode de analiză, sinteză și evaluare a costului pentru sistemele complexe de conducere din domeniul industrial. A doua secțiune (11.3/B) a cuprins 3 lucrări referitoare la conducerea cu calculator a unor activități din industria extractivă și de prelucrare a minereurilor.

Nefiind profund diferențiate, prezentarea comunicărilor cuprinse în volumul lucrărilor Congresului în secțiunile 11.3/A și 11.3/B se va face împreună. În cele ce urmează vor fi succint prezentate întâi lucrările 11.3/A1, 11.3/A3 și 11.3/B1, care abordează conducerea cu calculator a unor unități de producție din industria siderurgică, inclusiv a depozitelor de materie primă care le deservesc. În continuare se va prezenta abordarea problemei exploatării câmpurilor petrolifere [11.3/B2] și îmbunătățirii factorului de recuperare în extracția petrolului [11.3/B3]. Apoi se vor descrie sumar conducerea cu calculator de proces a două sisteme din industria sticlei [11.3/A2] și dintr-un accelerator complex [11.3/A5], care vor permite abordarea problemei a analizei și sintezei sistemelor de conducere a proceselor complexe [11.3/A6]. În final se vor face câteva referiri la o metodă de evaluare a efortului de realizare a software-ului de aplicație pentru sisteme complexe [11.3/A5].

Pătrunderea tot mai accentuată a dezvoltărilor tehnologice în sistemele industriale creează premisele perfecționării corespunzătoare a organizării și metodelor de conducere. Un exemplu de sistem integrat de conducere a activităților de producție dintr-un laminor de benză la cald (LBC) îl constituie cel de la Mizushima (Japonia), prezentat în lucrarea 11.3/A3. Sistemul este realizat pe mai multe niveluri de conducere. La nivelul superior, un calculator, cuplat on-line cu restul sistemului, realizează funcții specifice conducerii producției. La nivelul următor există o serie de subsisteme care realizează controlul cuptecelor, conducerea operativă și fixarea referințelor pentru procesele tehnologice, precum și subsisteme care controlează scuterea slaburilor din cuptecele de încălzire și pasarea lor pe linia de laminare. Un subsistem aparte asigură determinarea temperaturii de încălzire în cuptecare, minimizând consumul de energie.

Subordonat direct nivelului superior există un nivel de control on-line al calității, care are următoarele funcțiuni: determinarea și epurarea dimensiunilor și temperaturii slaburilor; estimarea proprietăților mecanice din datele operative și acceptarea/rejectarea calitativă a slaburilor; detectarea anormalilor slabului și adaptarea sistemului de laminare (viteză caje) în funcție de caracteristicile metalului. Bazat pe un mare număr de puncte de măsurare, se poate realiza un sistem de control on-line pe scară largă ca semifabricatului, asigurându-se în același timp și optimizarea operării. Sistemul de control al calității utilizează metode avansate de prelucrare matematică (filtrare, mediere, rivelare etc.) a semnalelor primite de la traducătoare. Citeva dintre aceste metode sînt prezentate în lucrarea 11.3/A3, în cazul determinării autentice a temperaturii, dimensiunilor, erorilor de formă, defectelor din structura metalului și estimării unor proprietăți mecanice ale laminatului.

Funcționarea acestor sisteme este coordonată de un mare volum de date, culese on-line din procesul de laminare. Aceste date sînt preluate prin intermediul unor microcalculatoare, care asigură prelucrările necesare interpretării datelor primite de la traductoarele din proces.

La combinatul siderurgic din Mizushima structura de conducere prezentată este suportată de două calculatoare universale (unul lucrînd pe loturi, altul on-line), 2 calculatoare de proces pentru funcțiile de supervizare, dintre care unul (cel pentru controlul calității) poate prelua la nevoie și funcțiunile celuilalt; funcțiile de colectare și interpretare primară date sînt realizate de microcalculatoare de proces, prin intermediul cărora se face și comandarea mărimilor de excoctie ale procesului controlate de sistem.

În cazul unui sistem integrat de conducere a activității de producție dintr-un combinat siderurgic subsistemele de mai sus aparțin conducerii de proces. Ele sînt subordonate subsistemelor de conducere a producției, care pot asigura mărimile de referință necesare proceselor de producție pentru funcționarea eficientă a întregului sistem, conform unor obiective globale.

Programarea producției într-un laminor, funcțiunea cea mai importantă de conducere a producției, implică realizarea unui compromis între diferite criterii de eficiență (cîteva mai practice: productivitatea laminorului, calitatea produsului final — urmărirea, de exemplu, în lucrarea 11.3/A3 — consumul de energie, funcționarea uniformă). În general programarea optimă a producției trebuie considerată inițial ca o problemă de optimizare multicriterială. Rezolvarea acestei probleme prin reducerea la un singur criteriu (fie prin comasarea ponderată a funcțiilor obiectiv, fie prin transformarea criteriilor suplimentare în restricții) nu dă în toate situațiile soluții acceptabile pentru toate criteriile, uneori rezultatul fiind chiar necorespunzător.

Programarea producției unui laminor se face în lucrarea 11.3/A1 printr-o metodă de programare dinamică multicriterială. Dintre cele trei criterii de optim considerate importante (consum de energie, productivitate, forma semifabricatului) au fost reținute primele două, al 3-lea fiind tratat ca o restricție. Din soluțiile obținute ca optim global în sens Pareto, operatorul poate selecta, în funcție de diferite preferințe de moment, varianta care se va implementa. Comparativ cu programele de producție empirice, obținute în sistemul manual, rezultatele obținute îmbunătățesc remarcabil funcționarea laminorului (dublarea productivității, reducerea consumului de energie etc.).

Abordarea amintită consideră procesul de laminare ca un proces de decizie în mai multe etape (N), corespunzătoare numărului de treceri ale metalului prin caje. Variabila de stare la pasul k, x^k , este grosimea de intrare a semifabricatului la pasul respectiv. Variabila de decizie la acest pas este un vector u^k cu patru componente (ajustarea secțiunii laminatului — d^k , viteza de laminare și forțele de împingere înainte, respectiv înapoi). Ecuația de stare este deci

$$x^{k+1} = x^k - d^k$$

Există un număr însemnat de restricții impuse de forma laminatului, forța de laminare, puterea motoarelor, viteza de laminare etc. Criteriul este un vector de două componente: consumul de energie și timpul total de prelucrare (care este reciproca productivității). Pentru un model de programare dinamică multicriterială care definește setul de puncte optime în sens Pareto pentru programele de laminare se folosește o metodă derivată din cele cunoscute în programarea dinamică convențională. Soluția, respectiv programul de laminare, indică valorile variabilelor de decizie pentru fiecare trecere; ea se obține într-un număr de iterații egal cu numărul de treceri pentru laminare N, astfel că din mulțimea de soluții fezabile se rețin cele acceptabile practic, pentru care $N \leq 5$. Pe baza experienței sale sau a unor cerințe de moment, operatorul selectează o variantă, înclinînd balanța către unul sau altul dintre cele două criterii.

Calculatorul a pătruns în majoritatea proceselor din industria siderurgică (fabricarea oțelului și fontei, laminare). O zecă mai puțin abordată, care oferă însă un potențial notabil de creștere a eficienței, este controlul calității și programarea tratării minereului pentru pregătirea materiei prime. Diferitele decizii lucrează de obicei independent, pe baza unor programe de producție empirice. Operațiile de prelucrare dintr-un depozit sînt variate și sînt greu de condus în cazul tratării lor interdependente, afectate de perturbații.

În combinatul de la Kagawa (Japonia), conform lucrării 11.3/D1, atît datele necesare, cît și programele de producție specifice depozitelor de materii prime din prelucrarea minereului de fier sînt furnizate de calculator. Obiectivele sistemului sînt reducerea costului prelucrării minereului și menținerea stabilității în procesele de producție din aval, printr-o alimentare corespunzătoare cu materie primă.

Sistemul de conducere a depozitelor este realizat pe mai multe nivele. Nivelul superior este constituit de un sistem de conducere pe termen lung a aprovizionării cu minereu și de planificare a mijloacelor de transport. Planul anual de aprovizionare este defalcat pentru perioade

mai scurte de timp utilizând programarea liniară pentru sisteme mari. Planul rezultat indică diferitele calități de minereu cu care trebuie alimentate furnalele, uzinele de sinterizare și de peletizare, precum și proporțiile în care trebuie amestecate aceste calități pentru asigurarea materiei prime. Al doilea nivel constă din sistemul de conducere a aprovizionării și formulare a cererii. Al treilea nivel asigură conducerea operațiilor din depozit și este în legătură directă cu diferitele procese de producție specifice prelucrării fontei și oțelului. La acest nivel funcțiile principale sînt de programare a operațiilor specifice, în care resursele urmărite sînt spațiile de depozitare și mijloacele de transport intern.

Programarea stocării minereului presupune dărmnarea timpului de descărcare pentru fiecare intrare de minereu, precum și a locației atribuite pentru fiecare sortiment de minereu descărcat. Datorită numărului mare de sortimente și de intrări de aprovizionare, utilizarea unui model este dificilă. Se folosește o tehnică de simulare cu increment variabil de timp, care presupune: preluarea datelor privind situația actuală a stocurilor și programul de aprovizionare și de producție; calcularea evenimentelor de timp care presupun mișcări în stocuri și ordonarea lor cronologică; efectuarea pas cu pas a mișcărilor în stocuri conform evenimentelor și unor criterii de selectare a variantei preferate bazate pe experiența operatorului. Rezultatele simulării prezintă, într-o formă ușor de interpretat, situația fiecărei locații din depozit la diferitele momente de timp ale programului și permite operatorului organizarea eficientă a celorlalte activități din depozit.

Programarea activităților de sfărîmare/sortare/amestec implică stabilirea sortului și cantității de minereu, precum și a timpilor de începere/terminare a operațiilor respective pentru a asigura fără întreruperi materia primă necesară proceselor de prelucrare a fontei și oțelului. Luînd ca exemplu operația de sfărîmare a unui lot compus din diferite sorturi de minereu, pentru programare se consideră o serie de variabile reale (exemplu, timpul de început și de sfîrșit al operației pentru fiecare sort) și o serie de variabile 0—1 (reprezentînd, de exemplu, opțiunea dacă un anumit sort de minereu trebuie sfărîmat sau nu, precum și ordinea de sfărîmare). Problema de programare se formulează ca un program linear cu variabile mixte. Criteriul urmărește maximizarea cantității totale de minereu sfărîmat în cadrul lotului în condițiile respectării unor restricții care asigură:

- un stoc suficient de materie primă;
- prelucrarea lotului printr-o singură operație;
- respectarea termenelor reparațiilor mijloacelor de transport intern;
- asigurarea spațiului de depozitare pentru minereul brut care urmează să sosească.

De o mare importanță în asigurarea unui timp de răspuns satisfăcător este reducerea numărului de variabile, ceea ce presupune reducerea numărului de sorturi de minereu considerate în problema de programare, care se poate realiza de către operator pe baza experienței sale. Considerînd, de exemplu, că pentru aplicația de la combinatul Kagogawa programul operațiv de sfărîmare (care a presupus circa 30 variabile reale și 50 variabile întregi), timpul de răspuns pe un calculator IBM 3033 a fost de circa 10 min, se consideră ca o valoare acceptabilă un număr de 15 sorturi de minereu.

Construite pe principiile sistemelor suport pentru decizie, subsistemele de programare a activităților din depozit permit operatorului selectarea din mulțimea soluțiilor fezabile a variantelor care corespund mai bine situațiilor particulare, alegere în care este implicată experiența operatorului.

Industria extractivă și în special cea a petrolului este de asemenea un domeniu în care și-au căutat aplicație metodele avansate de cercetare operațională. Motivul este evident, considerînd că datorită producțiilor mari în acest domeniu, îmbunătățirea metodelor de conducere chiar cu procente mici, atrage obținerea unor cîștiguri spectaculoase de materii prime deficiente.

În domeniul extracției petrolului obiectivul principal urmărit este îmbunătățirea factorului de recuperare a petrolului, respectiv a raportului dintre volumul de petrol recuperat și volumul total al bazinului exploatat. Tratarea problemei se poate face la nivelul unei unități de exploatare (ca în lucrarea 11.3/B3) sau pe ansamblul exploatărilor unui cîmp petrolifer (ca, de exemplu, în lucrarea 11.3/B2).

Modelul de exploatare a cîmpului de petrol conține: ecuațiile de filtrare pentru amestecul petrol-apă în mediul poros al cîmpului (care constituie un sistem dinamic de ecuații neliniare cu derivate parțiale care descriu distribuția de presiuni și saturația de petrol în cîmp); ecuațiile de mișcare a lichidului la urcarea în puțurile de extracție; ecuațiile care descriu raportul

dintre puțurile producătoare și sistemul de exploatare ; setul de restricții tehnologice. O particularitate privind aceste modele este că ele se pot construi după o exploatare de câțiva ani a sistemului respectiv, necesară pentru acumularea de date suficiente din activitatea geologică.

Variabilele de comandă sînt de două tipuri : continue (exemplu, ritmuri de funcționare pentru puțurile de producție și de pompare, adîncimea și presiunea în puțuri etc.) și binare (numărul, locul și timpul forării de noi puțuri, conectarea/deconectarea lor la/de la rețeaua existentă etc.).

Criteriul de optimizare ales este factorul de recuperare a petrolului, iar în cazul considerării unor criterii suplimentare (de exemplu, costul exploatării) ele sînt tratate ca restricții, problema rezolvată fiind în final monocriterială.

Dimensiunea mare a modelului (sute de variabile), nelinearitatea, existența simultană de variabile continue și discrete, precum și informația insuficientă generează în final o problemă de complexitate foarte mare. Ea se poate rezolva prin aproximații succesive (de exemplu, metoda variațiilor locale), la fiecare iterație soluționarindu-se o problemă de optimizare lineară auxiliară, în general de mare complexitate deoarece are variabile mixte și este de mari dimensiuni. Fezabilitatea procedurii este în mare măsură dependentă de precizia cu care se rezolvă problemele lineare auxiliare.

În lucrarea 11.3/B2 este sugerată o metodă de rezolvare a subproblemei lineare care folosește elemente ale teoriei sistemelor multivariabile și utilizează din plin specificitatea modelului cîmpului de petrol ca sistem multivariabil. Această metodă este analogă metodei simplex aplicată la spații funcționale, combinată cu metoda variațiilor locale. Astfel, dificultățile datorite dimensiunii sistemului sînt depășite, iar variabilele continue sînt separate de cele discrete. În continuare variabilele discrete sînt tratate cu metode combinatoriale în contextul abordării generale a proiectării optime a rețelei cu mai multe componente. Prin aceeași abordare, problema de optimizare lineară poate fi interpretată ca o problemă de planificare pe termen scurt.

Soluția problemei globale poate fi considerată ca proiectarea optimă a exploatării cîmpului petrolifer. Prin abordarea propusă în lucrarea 11.3/B2 conducerea tehnologiei de recuperare a petrolului este făcută compatibilă și poate fi corelată cu proiectarea construcțiilor de suprafață care urmăresc dezvoltarea sistemului de exploatare a cîmpului petrolifer.

Considerînd optimizarea completă a exploatării unui rezervor petrolifer ca un ansamblu de probleme de optimizare intercorelate, în lucrarea 11.3/B3 se abordează problema determinării bazei teoretice de calcul al celui mai bun mod de injectare a unui fluid de recuperare sporită a petrolului dintr-un rezervor. Criteriul de optimizare este minimizarea costului substanțelor chimice injectate și maximizarea petrolului recuperat. Obiectivele criteriului se exprimă ca o funcțională cost, sau indice de performanță, care trebuie minimizată. Variabila de comandă este compoziția sau starea fizică a fluidelor injectate. Astfel, problema de optimizare urmărește determinarea politicii de injectare care duce la minimizarea funcției de cost, respectînd restricțiile egalitate diferențiale care descriu dinamica sistemului. Calculul teoretic arată că utilizînd teoria sistemelor cu parametri distribuți este posibilă determinarea unei astfel de soluții.

În afara sistemului de control al calității într-un LBC referit anterior (11.3/A3), alte două comunicări prezentate în secțiunea 11.3 abordează conducerea unor procese tehnologice, din industria sticlei (11.3/A2) și pentru un accelerator complex (11.3/A5). Sînt de reținut atît metodele matematice utilizate — de mare eficiență pentru obținerea răspunsului în timp real — cît mai ales organizarea sistemelor de conducere care asigură cadrul necesar pentru succesul implementării.

În industria sticlei, după procesul de tragere în foi, tăierea la dimensiunile cerute prin comenzi poate fi optimizată în scopul eliminării defectelor de mici dimensiuni. Acestea sînt constatate prin dispozitive speciale, poziția lor fiind transmisă unui calculator de proces care controlează tăierea de așa manieră încît să elimine defectele minimizînd pierderile de material și asigurînd obținerea de plăci în dimensiunile cerute.

Tăierea foilor de sticlă se face în plăci dreptunghiulare de diferite dimensiuni, conform cererii. Operația presupune tăieri „verticale”, apoi „orizontale”, la dimensiunile solicitate. Neținînd seama de existența și poziția defectelor, un mare număr de plăci vor fi rebutate. O economie însemnată se face tăind plăcile așa încît defectele, sesizate prin detectoarele plasate înaintea dispozitivelor de tăiere, să fie evitate ; printr-o combinație optimă a diferitelor dimensiuni de plăci, precum și printr-o dispunere optimă a acestora se poate minimiza pierderea de material

eliminat odată cu defectele. Această problemă este tratată în lucrarea 11.3/A2 printr-o strategie ingenioasă de combinare a doi algoritmi într-un sistem condus de un calculator de proces. Unul este „algoritmul de tăiere optimă”, care așează optim, în spațiul unei foi la care au fost determinate defectele și poziția lor, două dimensiuni de plăci date (se demonstrează rezultate superioare față de considerarea unei singure dimensiuni a plăcilor). Așezarea se face astfel încât să se evite cu pierderi minime defectele, iar plăcile să se poată obține prin succesiuni de o tăiere verticală plus mai multe tăieri orizontale. Problema este complicată de curgerea semi-infinită a foii de sticlă.

Al doilea algoritm este un „algoritm de combinare optimă”, care asigură selectarea celei mai bune perechi de plăci pentru o anumită distribuție de defecte.

Datele de intrare folosite sînt : viteza și lățimea foii de material, distribuția de probabilitate a localizării defectelor, distanța între senzorii defectelor și dispozitivul de tăiere verticală, dimensiunile și numărul de plăci cerute. Pe baza acestor date algoritmul de combinare optimă determină o pereche de sortimente de plăci care va fi produsă la un moment dat. În funcție de localizarea reală a defectelor, aceste plăci sînt așezate în foie de către algoritmul de tăiere optimă, determinînd tăierile verticale/orizontale, care sînt transmise dispozitivelor de execuție. La epuizarea numărului solicitat de plăci dintr-o dimensiune a perechii, reintră în funcție algoritmul de combinare optimă. În afara prezentării algoritmilor, [11.3/A2] susține prin date experimentale valabilitatea și superioritatea algoritmilor propuși, față de alte metode abordabile.

Sistemul de conducere a unui accelerador complex de 28 GeV [11.3/A5] este compus dintr-o rețea de 20 minicalculatoare (calculatoare de control consolă, calculatoare frontale și pentru operații specializate, ex. gestiune mesaje) și 50 microcalculatoare cuplate la proces prin interfețe CAMAC (realizează în special funcții de control în interfețele specializate, activități de timp real, de prelucrare primară a datelor). Totul este operat de la 5 console generale. Software-ul utilizat este de două categorii : de sistem (sistem de operare, control rețea, limbaje de programare) și de aplicație (de la facilități dialog on-calculator pînă la algoritmi de control al hardware-ului de proces).

Sistemul este ierarhizat pe 6 nivele (de tip „layer”) și este realizat din module cu mare grad de independență. Cele 6 nivele sînt : module de interfață (protocoale de control standard pentru interfețele CAMAC proces-calculator) ; module de control al fiecărui tip de echipament (surse, pompe etc.) ; module de control pentru variabilele de proces complexe (exemplu, controlul fascicului de electroni) ; module pentru funcții de comandă a procesului în condițiile date de operator ; module pentru dialogul cu operatorul ; module specifice pentru controlere și prelucrarea primară de date. Aceste module acționează sub supervizarea unor funcții de conducere locală subordonate obiectivelor globale și asigură : activarea/oprirea modului, alocarea resurselor, prevenirea conflictelor, asigurarea sincronizării cu evenimentele specifice, afișarea mesajelor, controlul erorilor.

Sistemele de tipul celui prezentat sînt de o mare complexitate. Cum pot fi învățați inginerii să construiască astfel de sisteme ? Pînă la ce punct se poate metodologiza analiza și sinteza sistemelor complexe ? Acestea sînt întrebări la care lucrarea 11.3/A6 încearcă să dea un răspuns. Lucrarea urmărește dezvoltarea unei metodologii de înțelegere, standardizare și îmbunătățire a procesului de proiectare a sistemelor de comandă complexe. Plecînd de la premisa că cine poate analiza și înțelege structura de comandă a unui proces oricît de complex este capabil să dezvolte un mod bine organizat de sinteză a unui sistem de comandă asociat, autorii descompun în pași etapele de analiză și sinteză, exemplificînd pe cazul unui sistem industrial. Sistematizarea oferită este utilă în special inginerilor care fac primii pași în proiectare.

Care este însă efortul construirii unui sistem de conducere bazat pe utilizarea calculatorului ? Se poate estima eficiența așa-numitului software de aplicație ? Software-ul în general și software-ul de aplicație în particular suferă neavînd în spate experiența unei discipline ingineresti îndelung încercate. Așa cum se apreciază în lucrarea 11.3/A5, lipsa sa de tehnici riguroase de evaluare a costului care să cîștige încrederea utilizatorului face ca orice efort onest de dezvoltare a software-ului să fie privit cu o mare doză de scepticism. De aceea, considerăm utilă și instructivă evaluarea pe care o face lucrarea 11.3/A5 pentru un sistem de conducere complex pe baza unui formalism cunoscut sub numele de Barry Boehm. Acest model a fost aplicat satisfăcător la un mare număr de proiecte de dezvoltare software, lucru dovedit și de cazul

concret considerat, în care estimarea costului făcută prin diferite alte mijloace, inclusiv intuitiv, a fost ulterior confirmată atât de formalismul referit, cât și de costul real al proiectului. Formalismul presupune considerarea unui efort nominal bazat pe numărul estimat de instrucțiuni software precum și pe modul de organizare în echipa de programare. Acest efort nominal este apoi ponderat cu multiplicatori de efort pentru a se obține efortul efectiv. Acești multiplicatori particularizează proiectul relativ la obiectivul urmărit, hardware-ul implicat, personalul implicat și caracteristicile de implementare.

1. INTRODUCERE

Teoria sistemelor automate, în mod tradițional considerată ca parte a unei discipline analitice de sine stătătoare, a înregistrat în ultimii ani un proces de transformare calitativă, atât în ceea ce privește aparatul formal și procedural cu care se operează, cât și aria de cuprindere practică propriu-zisă. Dacă admitem că realizarea unui sistem de conducere automată presupune parcurgerea unor etape bine delimitate, cum ar fi : a) modelarea-identificarea procesului condus ; b) analiza-simularea modelului ; c) proiectarea legii de comandă și d) implementarea sistemului de conducere, atunci vom constata — prin prisma observației de mai sus — că mai ales etapele a, b, c au beneficiat de acumulări importante (idei, concepte, proceduri de proiectare, algoritmi de conducere automată avansată), în timp ce impactul asupra implementării (etapa d) și operării sistemului de conducere automată a rămas mai degrabă modest (cel mai notabil lucru fiind, probabil, tranziția de la implementările analogice la cele numerice).

Să considerăm exemplul regulatorului clasic proporțional-integrator-derivativ (PID). În acest caz valoarea mărimii de comandă $u(\cdot)$ este furnizată de o relație de tipul :

$$u(t) = \left[e_a^*(t) + \frac{1}{T_i} \int^t e(s) ds + T_d \frac{de(t)}{dt} \right] \quad (1)$$

Această este o ecuație lineară care permite studiul comportării sistemului de conducere automată și ai cărei parametri se pot determina prin proceduri bine stabilite.

Dar, în implementarea regulatorului PID, relația (1), deși absolut necesară pentru calculul mărimii de comandă, se dovedește insuficientă ; multe aspecte „practice“ importante nu sînt surprinse de aceasta, ca de exemplu :

- interfața cu operatorul tehnologic ;
- comutarea automat/manual ;
- regimurile tranziției care apar la modificarea parametrilor procesului (punctului de operare) ;
- efectele introduse de nelinearitățile elementelor de execuție ;
- saturarea termenului integral.

Asemenea probleme sînt de obicei scufundate în considerații practice de proiectare și numai foarte rar tratate și din punct de vedere teoretic. Realizarea unui regulator PID operațional presupune, desigur, implementarea formulei (1) dar și o anumită logică euristică pentru rezolvarea aspectelor menționate, aceasta constituint, de fapt, contextul logic al implementării relației (1). Un rol și mai important revine euristicii în cazul reguletoarelor multivariabile sau al reguletoarelor autocaldabile.

Această constatare se referă la activitățile asociate sistemelor de conducere automată de la nivel elementare ale unui sistem de cercetare complex, ierarhizat (adică la buclele izolate). Să observăm însă că euristica joacă un rol și mai important în rezolvarea problemelor de nivel superior (adică în conducerea unui proces cu toate caracteristicile de nivel inferior). La nivelul superior problema de conducere automată se poate formula în termeni de programare dinamică : stările sînt considerate în acest caz pe ansamblul procesului, iar stadiile reprezentate pe lângă modificările legilor de comandă aplicate subsistemelor de nivel inferior. Deciziile corespund selectării strategiilor de conducere automată (tipul legilor de comandă, de estimare etc.) impuse asupra caracteristicilor de nivel inferior. Criteriile de performanță de asemenea în brațul un caracter global și pot deveni multiple.

2. LOGICA EURISTICĂ ÎN CONDUCEREA AUTOMATĂ

Așa cum am menționat, recursul la logica euristică se impune cu necesitate pentru proiectantul de sisteme automate, fie că este vorba de regulatoarele PID obișnuite, de sistemele de reglare hibride (conținând circuite logice) sau de cele avansate.

În implementare, partea aferentă euristică se transpune prin bloouri de selecție („if-then-else”: „dacă-atunci-altfel”) sau de acțiuni dependente de situație („case”: „în cazul că”). Ca volum, codul (programul-sursă) respectiv poate depăși codul formulei (1); în plus, testarea, depararea și eventual modificarea părții din programul de conducere automată asociate „euristicii” se poate dovedi relativ complicată.

Din punctul de vedere al ingineriei programelor de conducere automată a proceselor apare deci justificată căutarea unor noi instrumente de implementare.

Înainte de a introduce un astfel de instrument vom mai prezenta câteva considerații decurgând dintr-o inversare de perspectivă: anume, dacă acceptăm că algoritmi de conducere automată includ logică euristică, să vedem ce ar putea aduce euristica în dezvoltarea sistemelor de conducere automată industriale.

Vom apela din nou la un exemplu: sistemele adaptive cu model de referință sau cele cu autoacordare. Algoritmi de comandă respectivi pot fi interpretați ca algoritmi de gradient local: pornind de la estimări inițiale suficient de bune ale ordinului sistemului, perioadei de eșantionare și ale parametrilor modelului, algoritmul poate ajusta parametrii regulatorului astfel încât să rezulte un sistem în circuit închis satisfăcător, cu condiția ca estimările inițiale să fie suficient de apropiate de valorile reale. Algoritmi menționați posedă chiar și proprietățile de urmărire, cu condiția ca parametrii să se modifice lent în timp. Algoritmi nu funcționează dacă estimările inițiale sînt depărtate de valorile reale (cazul cel mai tipic este perioada de eșantionare). De aici necesitatea unor algoritmi cu domeniu de operabilitate mai mare (mai siguri, într-un anumit sens), chiar în detrimentul proprietăților locale.

Din acest punct de vedere pare atractivă explicarea posibilității de a utiliza — într-o structură evoluată de conducere — algoritmi cu proprietăți diferite. Acest lucru reclamă facilități de combinare a diferiților algoritmi de conducere automată, astfel încât să se satisfacă criteriul de performanță adecvat fiecărui regim de funcționare.

Formulată în alți termeni — care se vor preciza în continuare — această problemă își găsește o rezolvare, cel puțin potențială, în aplicațiile sistemelor expert.

3. SISTEME EXPERT

Sistemele expert reprezintă un domeniu în plină dezvoltare al inteligenței artificiale; obiectivul acestora este de a modela cunoștințele și procedurile utilizate de experții umani în rezolvarea problemelor dintr-un domeniu bine stabilit. Pentru familiarizarea specialistului în automatică vom prezenta în continuare o scurtă introducere în sistemele expert. Ideile expuse sînt preluate după (I. Georgescu, 1983), unul din exponenții recunoscuți ai școlii românești de inteligență artificială.

Capabilitățile funcționale ale sistemelor expert pot fi clasificate după cum urmează:

- *funcții cognitive*, care oferă sistemului capacitatea de a trata cunoașterea de tip expert;
- *funcții rezolutive*, care se utilizează în procesul de rezolvare a problemelor pe baza cunoștințelor din domeniul de expertiză și a informațiilor furnizate de utilizator asupra contextului problemei considerate;
- *funcții explicative*, cu rolul de a explicita judecățile și inferențele care au condus la soluțiile preconizate;
- *funcții de comunicare*, care permit sistemului expert să comunice cu utilizatorul uman sau cu alte sisteme expert.

Potențialul de rezolvare a problemelor de către sistemele expert depinde de capacitatea de a memora, actualiza și manipula volumul considerabile de cunoștințe specifice domeniului, de flexibilitatea asigurată în reprezentarea cunoștințelor indiferent de complexitatea acestora, de posibilitatea de a apela la strategii de planificare și rezolvare evolute ca și de capacitatea de a-și îmbunătăți performanțele funcționale prin învățare din experiență.

Indiferent de cerințele particulare ale domeniului real de expertiză, un sistem expert se poate considera ca fiind alcătuit din următoarele elemente :

— *sistemul cognitiv*, destinat să memoreze cunoștințele într-un sistem de memorare special organizat denumit *baza de cunoștințe*, să caute cunoștințe parțiale direct specificate, prin identificarea de simboluri sau referite indirect, prin proprietățile asociate și valorile atribuite sau deduse din alte cunoștințe și de asemenea să întreprindă baza de cunoștințe conform evoluției domeniului de expertiză ;

— *sistemul rezolutiv*, destinat să rezolve problemele specificate de către utilizator prin aplicarea unor proceduri și operatori asupra obiectelor reale extrase din formularea problemei și din baza de cunoștințe, într-o secvență adecvată și pe baza unei strategii de control care furnizează finalmente soluția problemei ;

— *sistemul de comunicație*, care constituie interfața dintre utilizator și sistemul expert, prin intermediul unor procesoare adecvate de limbaj și al unor proceduri de intrare-ieșire.

Sprijinul obiectual al unui sistem expert, organizat în baza de cunoștințe, include cunoștințe *declarate*, care se autoexplicităză și cunoștințe *generative*, care sînt în esență mecanisme productive pentru obiecte, acțiuni, relații, stări și evenimente, capabile să genereze o entitate cu parametri specificați sau un set finit de entități cu proprietăți specificate.

Sprijinul procesual al unui sistem expert este organizat într-o ierarhie de patru nivele :

— nivelul mecanismelor de bază, care include mecanisme inferențiale pentru aspectele logice și mecanisme operaționale pentru aspectele algebrice ;

— nivelul procesoarelor de acțiune, care interpretează cunoștințele ce descriu acțiuni ;

— nivelul procesoarelor de control, concepute ca organe de decizie asupra strategiei de control pe care sistemul expert trebuie să o urmeze în rezolvarea unei probleme ;

— nivelul procesoarelor de metaccontrol, destinate să controleze, adapteze și actualizeze funcționarea procesoarelor de la nivelele subordonate.

Prin control în acest context se înțelege un procedeu de supervizare a proceselor de tip acțiune.

În ceea ce privește reprezentarea cunoștințelor, cea mai importantă opțiune conceptuală în proiectarea unui sistem expert, există mai multe metode, fiecare dintre acestea avînd limite și influențe asupra extinderii domeniului de expertiză, asupra complexității cunoștințelor, asupra flexibilității în manipularea cunoștințelor și finalmente asupra performanțelor sistemului expert ca atare. Dintre metodele de reprezentare a cunoștințelor vom menționa :

— limbaje bazate pe calculul predicatelor de ordinul întâi (formalism direct interpretabil de către componentele rezolutive ale sistemului expert) ;

— reprezentarea procedurală (în plus față de caracteristicile și proprietățile structurale se includ și descrieri ale modalităților de utilizare a cunoștințelor în acțiuni și procese) ;

— reprezentarea prin rețele semantice (prin excelență un instrument de reprezentare de tip relațional, în care nodurile se asociază obiectelor, acțiunilor, stărilor sau altor entități abstracte din domeniul de expertiză, iar arcele descriu relațiile dintre acestea) ;

— reprezentarea prin sisteme de producție (conceptual, un sistem de producție se compune din : a) un set finit de reguli de producție de tipul "DACA-condiție-ATUNCI-acțiune" ; b) un set finit de date specificînd contextul de aplicare a regulilor și c) un interpretor care decide inițierea, secvența de reguli și oprirea executării producțiilor) ;

— reprezentarea prin obiecte structurate („cadre“ reprezentînd descrierea unui obiect, acțiuni sau situații stereotipe pe care o anticipăm atunci cînd considerăm anumite cunoștințe).

Să examinăm acum domeniul de expertiză al inginerului automatist. Baza de date va conține informații factice, evidențe (în sensul unor înregistrări ale evoluției), ipoteze și obiective. Ca faptice vom considera, de exemplu, datele statice : toleranțele traductoarelor, limitele de operare, limitele de alarmă pentru parametrii procesului, restricțiile de înlănțuire pentru acțiunile de comandă, configurația de utilaje care concurează la realizarea procesului tehnologic respectiv. Ca evidențe se pot considera datele dinamice de la senzori, rapoartele și testele de laborator ș.a. Față de acestea vom observa următoarele particularități : diversitatea în tip, afectarea de zgomot, întârzierea (evidențierii în raport cu momentul producerii), incompletitudinea și chiar caracterul contradictoriu.

Desigur, inginerul automatist are capacitatea de a elabora soluții (ipoteze) pentru rezolvarea acestor complicații. În sistemele expert ipotezele se vor genera și memora în baza de date ; așa cum s-a menționat, ipotezele sînt astfel prezentate încît utilizatorul poate verifica rațional sistemului.

Obiectivele reținute în baza de cunoștințe pot fi atât statice, cât și dinamice în natură. Statice sînt, de exemplu, obiective de genul "optim staționar" sau "sistem stabil", în timp ce obiectivele dinamice sînt stabilite on-line prin comenzi externe de către sistemul de conducere automată.

Tipurile de cunoștințe necesare sînt :

- caracterizarea algoritmilor de comandă și estimare disponibili ;
- indicatoare pentru invocarea funcțiilor de supervizare, planificare sau diagnosticare ;
- instrucțiuni propriu-zise pentru supervizare, planificare sau diagnoză.

Pentru aplicațiile de conducere automată, regulile de producție și obiectele structurate par a fi cele mai potrivite metode de reprezentare a cunoștințelor. De exemplu, așa cum am menționat deja, o regulă de producție se descrie în general prin „dacă-situație-atunci-acțiune”.

„Situație” reprezintă un set de elemente ale bazei de cunoștințe, iar „acțiune” poate însemna activarea unui regulator, specificarea unui nou obiectiv (criteriu de performanță) etc. Obiectele structurate reprezintă structuri de date asignate situațiilor vizuale ; informațiile incluse se referă la instrucțiunile de utilizare (a obiectului structurat), ce se va împlini la momentul următor, ce trebuie făcut la momentul următor, precondiții necesare pentru o anumită operație și altele.

În funcționarea reală a sistemului de conducere automată un defect oarecare, o schimbare a parametrilor calitativi ai procesului de producție sau orice altă modificare necesită o secvență de acțiuni pentru a alinia procesul la noile cerințe. Fiecare pas al planului de modificare a evoluției procesului presupune o acțiune de ajustare, iar acțiunile stabilite la un anumit pas trebuie să nu interfereze cu precondițiile pentru acțiunile următoare.

Dacă sînt multe acțiuni și multe secvențe posibile, elaborarea planului de conducere automată presupune căutarea drumului care conduce la obiectivul fixat printr-o rețea relativ mare.

Planificarea acțiunilor în condițiile unui sistem complex cuprinde o serie de elemente specifice : incertitudine în starea și modelul procesului, în efectul acțiunilor de comandă aplicate.

Deși programarea dinamică statistică poate trata, în principiu, această problemă, totuși această abordare algoritmică de tip căutare globală se poate dovedi nerealizabilă din punct de vedere practic.

4. REGULATORUL ADAPTIV EXPERT

Pentru a ilustra conceptele introduse vom prezenta succint, ca exemplu, conducerea în regim staționar a unui proces industrial.

Fie, pentru simplitate, cazul unei singure bucle de reglare cu obiectivul de a menține fluctuațiile variabilei de ieșire, apropiate de o mărime de referință dată. Dacă dinamica procesului și perturbațiile ar fi cunoscute, atunci un regulator de minimă varianță (dispersie) ar constitui soluția de conducere automată. Într-adevăr dacă procesul este dat de modelul esanționat :

$$Ay(t) = Bu(t-d) + Ce(t) \quad (1)$$

unde $a(\cdot)$ este mărimea de comandă, $y(\cdot)$ — mărimea de ieșire, $e(\cdot)$ — un zgomot alb iar A , B , C sînt polinoame în operatorul de anticipare, atunci legea de comandă optimă este reprezentată de :

$$Ru = -Sy \quad (2)$$

unde polinoamele R , S sînt date de :

$$z^{d-1}CB = AR + BS \quad (3)$$

Operațiile necesare pentru a obține și menține legea de comandă de minimă varianță (dispersie), dată de relația (2), într-un mod sigur vor fi prezentate în lumina considerațiilor anterioare.

1) În terminologia sugerată, acțiunea "comandă de minimă varianță" constituie obiectivul principal al sistemului. Vom avea ca precondiții și cerințe : modelul procesului (1), modelul perturbațiilor, intervalul de eșantionare, condiția ca procesul să fie de fază minimă (toate zerourile să fie situate în interiorul cercului unitate).

2) Un aspect particular al legii de comandă de variație minimă îl constituie anularea zerourilor procesului, ceea ce poate conduce la un regim excitară dacă zerourile nu sînt suficient de amortizate. Apare astfel ca necesară includerea unui detector de oscilații în schema practică.

3) O manieră vizuală de a stabili dacă procesul tehnologic evoluează sub o comandă de minimă variație (dispersie), este de a verifica dacă ieșita este ca înșir și un proces de medie alunecătoare :

$$y(t) = \lambda [e(t) + \dots + f_{d-1}e(t-dh+h)]$$

unde $F=R/B$ iar h este intervalul de eșantionare. De aici necesitatea unui *supervizor de minimă variație*, care poate funcționa pe baza calculului funcției de corelație a ieșirii.

Să analizăm ce se întâmplă în cazul că unele condiții nu sînt îndeplinite.

a) Dacă modelul procesului nu este disponibil, atunci se poate folosi un regulator auto-acordabil, care este în esență un estimator (și care converge, în anumite condiții, către regulatorul de minimă variație). Prin urmare, un *estimator de parametri* va fi inclus ca operator. Acesta va funcționa corespunzător numai dacă datele experimentale sînt cîștigate atunci cînd procesul este bine excitat : este deci, nevoie de un *supervizor de excitare* (care va determina energia semnalului de intrare în domeniul util de frecvențe).

b) În cazul că excitația procesului nu este suficientă, sînt posibile două cîștiri :

b1) oprirea estimării și

b2) introducerea unor semnale perturbatoare (dacă procesul tehnologic permite acest lucru).

Acesta implică existența unui *generator de semnale de perturbație*, pentru care sînt necesare cel puțin specificarea domeniului de frecvențe și a nivelului admis al perturbațiilor. Aceste elemente se pot deduce din curcaterarea orizontului de predicție dh .

c) Anumite date a priori sînt necesare și pentru regulatorul autoacordabil :

h — perioada de eșantionare ;

d — întîrzierea (în multipli de h) ;

nr — gradul polinomului R ;

ns — gradul polinomului S ;

λ — factorul de uitare ;

0_0 — estimăția inițială ;

p_0 — covarianța inițială ;

uh, ul — limitele superioară și respectiv inferioară ale mărimii de comandă.

Acestea, la rîndul lor, antrenează și ele anumite condiții :

— d și h : sistemul în buclă închisă poate deveni instabil dacă acești parametri sînt subestimați. Rezultă așadar că este necesar un *supervizor de stabilitate* ;

— ns și nr : valorile gradelor trebuie să fie suficient de mari (validitatea lor se poate preciza prin calculul funcțiilor de covarianță $\gamma_{yy}(\tau)$ și $\gamma_{yu}(\tau)$). De aici necesitatea unui *supervizor de grad*.

— produsul dh : o estimare reușită a acestuia se poate obține pe baza amplificării critice K_c și a perioadei critice t_c , conform relației $d \cdot h = t_c/2$.

Vom adopta în acest caz un *estimator de tip K_c-t_c* (care furnizează date utile și pentru estimarea altor parametri).

d) Se pot lua în considerare și alte funcțiuni sau facilități. De exemplu, dacă dinamica procesului depinde de anumiți parametri (cum ar fi ritmul de producție), atunci se poate avea în vedere programarea amplificării și se pot deci introduce funcțiuni ca : "netezește și memorează parametrii regulatorului", "actualizează parametrii regulatorului" și "testează ipotezele de programare" (a amplificării). Un *supervizor instruit* va asigura deslășurarea corespunzătoare a tuturor acestor funcțiuni.

Rezumînd cele expuse, implementarea unui regulator activ cuprinde următoarele grupe de operatori :

Program de conducere general

Supervizor-stabilitate

Calculul-valori-medii-și-varianțe (dispersii)

Monitor conducere automată

Sistem de rezervă

Regulator PID

Estimator- K_c-t_c

Regulator de minimă varianță cu amplificare constantă

Reglare-de-minimă-varianță

Supervizor-de-minimă-varianță

Detector-de-oscilații

Supervizor-de-grad

Estimare :

Estimator-de-parametri

Supervizor-de-estimare

Supervizor-de-excitare

Generator-semnal-perturbații

Detector-de-salt

Regulator autoacordabil

Reglare-autoacordabilă

Instruire

Actualizare-parametri-regulator

Netezire-și-memorare-parametri-regulator

Testarea-ipotezelor-de-programare.

5. REALIZĂRI ȘI PERSPECTIVE

Pe baza lucrărilor celui de al 9-lea congres IFAC de la Budapesta, 1984, precum și a unor cercetări proprii (Vasiliu C. și Dumitru M., 1983 ; Ceangă E. și Vasiliu C., 1984) am încercat să prezentăm în paragrafele anterioare o soluție de perspectivă la o problemă tot mai stringentă a automatizării industriale : anume, realizarea de sisteme de automatizare flexibile, cu elemente de inteligență artificială.

Cu excepția considerațiilor privitoare la sistemele expert, expunerea a urmărit îndeaproape *lucrarea 11.4/B2*.

Vom trece în revistă alte probleme conexe abordate în lucrările congresului amintit (*secțiunile 11.4/A și 11.4/B*).

Aspecte legate de rezolvarea problemelor de modelare și comandă în sistemele om-mașină sînt examinate în *lucrarea 11.4/A1*. Natura interacțiunilor între comandă, planificare și detecția și anihilarea erorilor sînt exemplificate, pentru cazul unui autovehicul și al unei aeronave. Autorii susțin ideea unui sistem expert ca fiind cel mai adaptat cerințelor umane în situațiile examinate.

Conducerea în regim supervizor și dirijarea situațiilor anormale (*fault management*) fac de asemenea obiectul *lucrării 11.4/A3*. Se arată că un model al operatorului uman acționînd în regim supervizor depinde de modelul sistemului supervizat, de perturbațiile acestuia și de obiectivul procesului de supervizare ; un model analog în cazul unor situații anormale nu se poate construi. Soluția propusă de autori pentru completarea acestei lacune constă într-o abordare de tip normativ bazată pe teoria utilității.

Un aspect esențial care condiționează succesul în proiectarea sistemelor om-mașină (cum este și cazul unui operator uman față de un sistem de conducere automată) îl constituie înțelegerea modului în care omul formulează ipoteze și concluzii, identifică posibile variante de acțiune și analizează impactul acestor variante, precum și integrează consecințele acțiunilor în concordanță cu un sistem de evaluare. *Lucrarea 11.4/A4* conține o privire de ansamblu asupra eforturilor recente în domeniul reprezentării cunoștințelor, o atenție particulară fiind acordată sistemelor de producție.

Pentru a obține modele cât mai realiste ale operatorului uman aflat în postura de a prelucra informații și a lua decizii este necesar să se modeleze și memoria acestuia, fie că este internă procesului de decizie, fie că este externă acestuia (în forma unei baze de date, de exemplu). Un model de tip informațional al memoriei permanente este prezentat, împreună cu descrierea accesului la informațiile memorate, pentru un decident uman în *lucrarea 11.4/A5*.

Sistemele de decizie asistată de calculator sînt abordate în *lucrarea 11.4/A6*, în contextul unor concepte de psihologie, cibernetică și sisteme automate.

Pe aceeași linie de evoluție, adică în completarea abordărilor clasice, în *lucrarea 11.4/B1* se utilizează metodele teoriei mulțimilor vagi pentru descrierea strategiei unui operator uman.

Exemplele prezentate sînt simple, dar consacră abordarea "fuzzy" ca o soluție în cazul în care metodele clasice nu pot funcționa.

În sfîrșit, *lucrarea 11.4/B4* încearcă trasarea unui cadru conceptual pentru proiectarea sistemelor de automatizare complexe. Pornind de la necesitatea ca datele abstracte și adesea informale, generate pe parcursul etapelor preliminare de proiectare a unui sistem să fie astfel documentate încît să poată fi utilizate în etapele ulterioare (nu numai de proiectare, ci chiar de operare a sistemului respectiv) autorii propun ca instrument un limbaj dedicat, de nivel înalt, denumit SML (System Modeling Language), implementat pe un calculator VAX 11/750 cu sistemul de operare VMS.

BIBLIOGRAFIE

1. *Georgescu I.* (1983), *An Introduction to Expert Systems*, Report AI-FDO30, ICI.
2. *Vasilii C., Dumitru M.* (1983), *Tehnici de recunoaștere a formelor și de inteligență artificială în proiectarea sistemelor*, Simpozionul de Modelare și Optimizarea Sistemelor, Universitatea Galați.
3. *Ceangă E., și Vasilii C.* (1984), *Learning Systems Using Control Situations Recognition*, Raport intern, ICI.

CONDUCEREA DE CĂTRE OPERATOR UMAN A VEHICULELOR ȘI TELEMANIPULATOARELOR

Ing. A. Alexandru, Ing. L. Iacob.
Ing. P. Rădulescu-Banu

I.T.C.I.

Pentru a înțelege comportarea umană în sisteme conduse manual este din ce în ce mai importantă investigarea pornită din interior în locul descrierii exterioare. *Lucrarea 11.4/C4* este un pas într-o astfel de investigație, studiind ieșirea reglată manual a unui sistem comandat manual. Se consideră că ieșirea constă din 3 componente. Prima este componenta de intrare, care se presupune a fi o funcție continuă de timp, generată de modelul intrării în mintea operatorului. Cea de a doua este o componentă sistem, care depinde de starea curentă a erorii. Cea de a treia este zgomot. Modelul conceput pentru operatorul uman este prezentat în fig. 1.

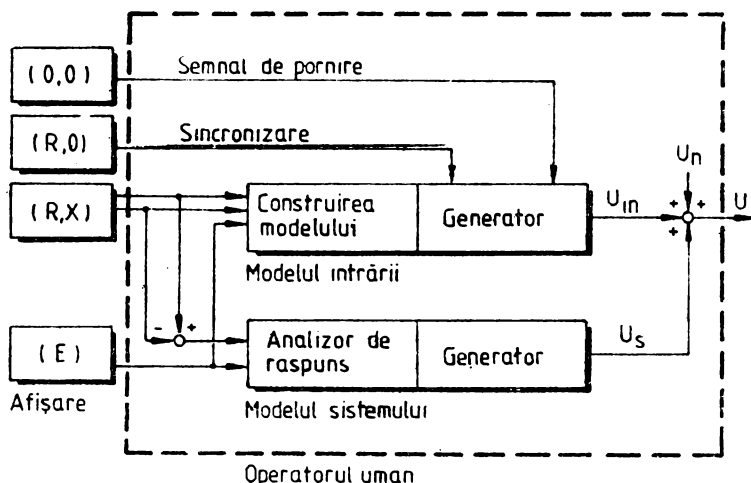


Fig. 1. Modelul conceput pentru operatorul uman.

Modelul intern este constituit mental de către operator prin identificarea caracteristicilor intrării și dinamicii echipamentului. El generează comanda ideală și este denumit modelul intrării, ieșirea lui fiind componenta de intrare U_{in} . Experimental, ținând cont de evoluția intrării de referință R și a variabilei reglate x , s-a arătat că modelul intrării are caracteristici de buclă deschisă de reglare.

Pe de altă parte, modelul sistemului este un model intern construit prin interacțiunea dintre operator și sistem în buclă închisă de reglare, ieșirea acestui model este componenta sistem U_s , care are rolul de a preveni creșterea erorii sau a stabili sistemul și de a compensa eroarea făcând-o să fie zero.

Se presupune că strategia de comandă a operatorului uman, însoțită de operația de recuperare, este reflectată de către componenta de zgomot U_n , datorită deci unor operații fizio-

logice și psihologice. Se consideră că această componentă există atât în cazul reacției negative, cât și în caz contrar.

Sistemul experimental de comandă manuală este prezentat în fig. 2.

Pentru experimentări, intrarea de referință este considerată sinusoidală $R = a \sin \omega t$. Pentru a simula dinamica echipamentului se folosește un factor de amplificare proporțional $G(s)$. Frecvența ω este 0,2; 0,5; 1,0; 2,0 [rad/s] și amplitudinea R este calibrată la 5 em pe un dispozitiv de afișare grafică localizat la 50 cm de operator. Controlerul este un potențiomtru rotativ (de la -60° la 60°). Fiecare experiment durează 60 de secunde.

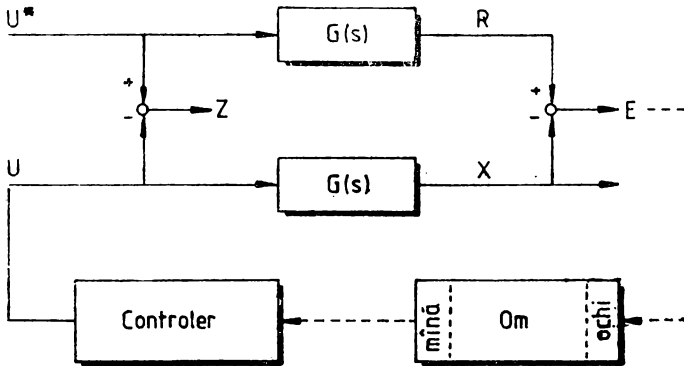


Fig. 2. Sistemul experimental de comandă manuală.

În primul rând este executat un proces de învățare folosindu-se un display de compensare (E). În timpul acestei faze, apar atât componenta de intrare, cât și componenta sistem. Pentru un operator neantrenat este dificil să execute o operație asupra componentei de intrare, astfel încât componenta sistem este dominantă. Se presupune că componenta de intrare devine dominantă atunci când, datorită antrenamentului, poate fi construit un model al intrării corespunzător comenzii ideale. În fig. 2 sistemul experimental este construit astfel încât comanda U a operatorului este comparată cu comanda ideală U^* .

Rezultatele experimentale arată că unii dintre parametrii modelului de intrare, constituit mental de către operator, necesită informații prin reacție negativă (de exemplu, amplitudinea și forma de undă) și alții nu (de exemplu, perioada și faza).

Pentru a stabili efectul informațiilor recepționate prin reacție negativă asupra componentei sistem, este necesară extragerea componentei sistem din ieșirea reglată. Dacă se folosește analiza spectrală, rezultatele obținute nu sînt satisfăcătoare, deoarece analiza spectrală este

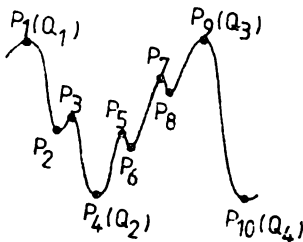
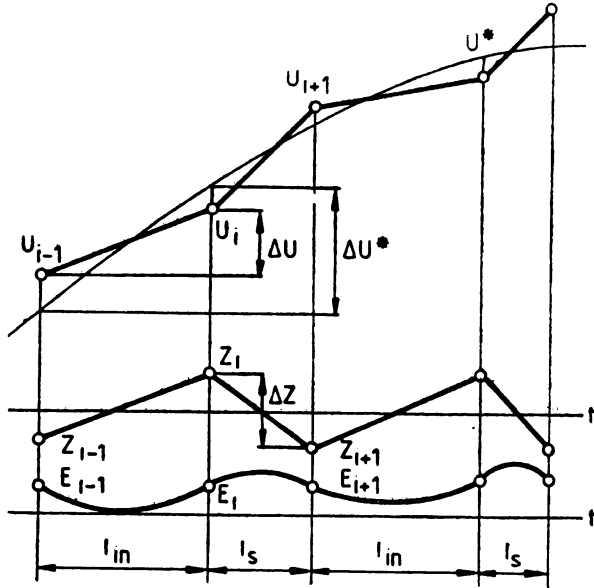


Fig. 3. Un exemplu de formă de undă constînd din 2 componente $\{P_i\}$, $\{Q_j\}$.

validă pentru unde staționare, iar ieșirea reglată nu este astfel. De aceea, se acordă atenție caracteristicilor de timp ale ieșirii reglate, reprezentîndu-se eroarea $E (= R - x)$ ca un șir de vîrfuri locale $\{P_i\}$, ca în fig. 3 și se definește perioada de timp T ca diferența $T = T_{i+1} - T_{i-1}$ între P_{i+1} și P_{i-1} .

Folosirea unui filtru pentru separarea componentelor de frecvență înaltă și joasă creează posibilitatea analizei în timp a fiecăreia din ele. Se arată că $\{Q_j\}$ — vîrfuri lente și $\{P_i\}$ sînt

legate de componenta sistem, respectiv de zgomot. Deoarece componenta sistem operează cu o frecvență mai ridicată decât intrarea de referință, ea este adesea privită drept zgomot care înrăutățește comportarea sistemului. Aplicând analiza în timp, virfurile $\{P_1\}$ separate de un filtru, se constată existența lor de la 0,3 la 0,4 secunde. Astfel, componenta care are virful în regiunea de la 0,3 la 0,4 secunde este superimpusă ieșirii reglate, independent de existența reacției negative sau intrării de referință. De aceea, această componentă este considerată drept zgomot cauzat de operațiile fiziologice și psihologice.



$$Z = U^* - U, E = R - X$$

Fig. 4. Intervalele în care acționează componenta de intrare și componenta sistem: I_{in} și I_s .

Considerăm $Z (=U^*-U)$ ca fiind diferența între comanda ideală U^* și comanda U , în care se reflectă și zgomotul. În fig. 4 este reprezentat Z printr-un șir de virfuri $\{Z_i\}$ și ieșirea reglată este împărțită în intervalul de acțiune a componentei de intrare I_{in} și intervalul de acțiune a componentei sistem I_s .

Fiecare interval este definit astfel:

$$I_{in} \text{ (tip 1): } (E_{i+1} + E_i)(Z_{i+1} - Z_i) \geq 0 \tag{1}$$

$$\Delta U^* \cdot \Delta U \geq 0$$

$$I_s \text{ (tip 2): } (E_{i+1} + E_i)(Z_{i+1} - Z_i) < 0 \tag{2}$$

$$\Delta U^* \cdot \Delta U \geq 0$$

$$I_s \text{ (tip 3): } \Delta U^* \cdot \Delta U < 0 \tag{3}$$

unde ΔU^* și ΔU sînt variațiile lui U^* și U în fiecare interval.

Exactitatea componentei de intrare este evaluată prin raportul $\Delta U/\Delta U^*$ în I_{in} . Pentru a descrie caracteristicile distribuției lui ΔU în raport cu ΔU^* în coordonate ortogonale este definită următoarea valoare medie pentru precizia trasării

$$h_{in} = 1 - \sqrt{\frac{\sum_i (\Delta U_i^* - \Delta U_i)^2}{\sum_i \Delta U_i^{*2}}} \tag{4}$$

În cazul în care acționează numai componenta de intrare, Z ar trebui să tindă spre zero. Dar Z este corectat de fiecare dată, fapt care conduce la o mică variație. De aceea se definește precizia trasării îmbunătățită de componenta sistem ca

$$h=1-\sqrt{\int_0^T Z^2 dt / \int_0^T U^2 dt} \quad (5)$$

Rezultatele experimentale au arătat că rolul componentei sistem este de a corecta eroarea, conducând la stabilitate.

Deoarece operatorul nu poate urmări intrarea de referință, își construiește mental un model de intrare corespunzător lui R într-un proces de învățare. De aceea, acțiunea componentei sistem și a componentei de intrare depind de gradul de antrenare.

Pentru studierea posibilității de conducere umană a vehiculelor și telemanipuloarelor au fost dezvoltate, testate în laborator și experimentate o serie de modele ale conducătorului uman. *Lucrarea 11.4/C5* dă o abordare unitară a modelării pilotului uman în scopul modificării modului de considerare a capacităților umane fundamentale și a limitărilor lor.

Baza de date necesară unor astfel de modele poate fi considerată completă din punct de vedere al efectelor dinamicii avioanelor și caracteristicilor perturbațiilor. În ciuda acestui fapt au apărut recent noi întrebări la care s-a răspuns sau nu s-a răspuns încă. Dintre aceste întrebări se pot aminti:

- (1) Prin ce mecanism pilotul uman generează egalizarea dinamică cerută în programele de urmărire?
- (2) Cum ar putea omul utiliza un model intern al mediului în aceste programe?
- (3) Ce cauzează comportarea nelineară a pilotului în timpul reglării fenomenelor cu dinamică ridicată?
- (4) Cum influențează caracteristicile manipulatorului dinamică și performanțele pilotului uman?
- (5) Cum își organizează omul percepțiile în programele de urmărire față de cele de compensare?
- (6) Cum își clasifică omul impresiile personale despre dificultatea programului sau abilitățile în manipulare?

Ținând cont de acestea este discutat un model structural al pilotului uman. Modelul are mai mult o natură descriptivă decât predictivă și este presupusă o structură cu reacție inversă implicată de sistemul uman neuromuscular (fig. 5).

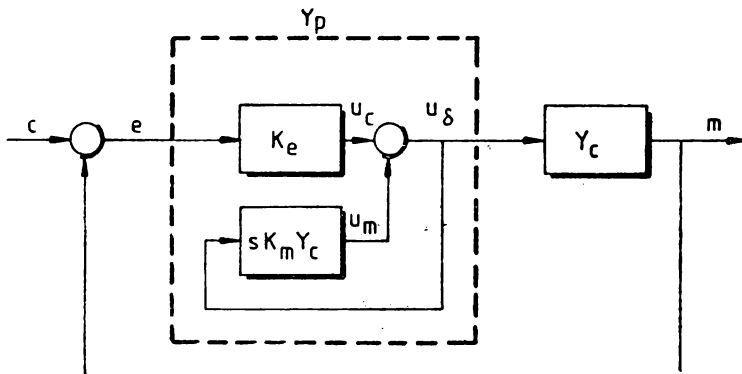


Fig. 5. Un model structural simplificat al operatorului uman pentru programe de compensare.

Modelul se bazează pe principii simple de reacție negativă. Omul realizează egalizarea cerută de orice program printr-o reacție negativă proprioceptivă și nu prin vreo operație serială directă asupra stimulului vizual $e(t)$. Caracteristicile generale ale acestei egalizări depind de

prelucrarea ulterioară a ieșirii manipulatorului $U_0(t)$: diferențiere, atenuare simplă sau integrare în frecvență în jurul frecvenței de ieșire. În bucla interioară pentru reacția negativă este utilizată ieșirea manipulatorului, care alimentează un element de compensare în care apare în mod explicit elementul comandat Y_c . Este interesant de utilizat modelul pentru a forma funcția de transfer a buclei deschise pentru trei elemente comandate stereotipe: K , K/s și K/s^2 .

$$\begin{aligned}
 Y_c = K; \quad Y_p \cdot Y_c &= \frac{K_o \cdot K}{K_m K S + 1} \\
 Y_c = \frac{K}{s}; \quad Y_p \cdot Y_c &= \frac{K_p \cdot K}{s (K_m K + 1)} \\
 Y_c = \frac{K}{s^2}; \quad Y_p \cdot Y_c &= \frac{K_o / K_m}{s [s (1/K_m K) + 1]}
 \end{aligned}
 \tag{6}$$

Modelul simplificat prezentat în fig. 5 poate fi dezvoltat sub forma unui model (fig. 6) care simulează operatorul uman în programe de urmărire cu compensare pe o singură axă. Figura 6 este împărțită în două componente: sistemul nervos central și sistemul neuromuscular, divizare care accentuează natura prelucrării semnalului implicată. Întârzierile de τ_0 secunde a fost inclusă ținându-se cont de întârzierile în perceperea vizuală, acționarea motorie etc.

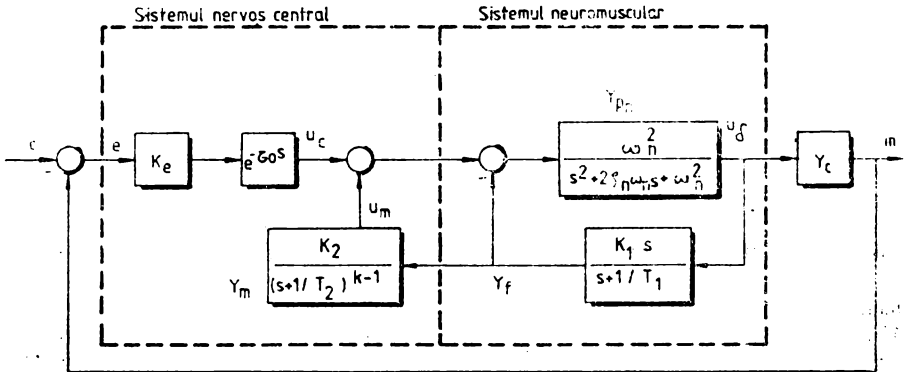


Fig. 6. Modelul structural al operatorului uman pentru programe de compensare.

Semnalul rezultat, $u_0(t)$, asigură comanda sistemului în buclă închisă, care conține modelul dinamicii neuromusculare în buclă deschisă a membrului care acționează asupra manipulatorului, Y_{pm} . Celelalte componente ale sistemului includ elementele Y_f și Y_m , care acoperă, cel puțin aproximativ, efectele combinate ale sistemului muscular, tendonul Golgi și dinamicile asociate prelucrării de nivel înalt a semnalului. Forma lui Y_m este determinată de natura dinamicii elementului comandat Y_o în regiunea frecvenței de tăiere în buclă deschisă. Deși funcția de transfer u_m/u_0 pentru fig. 6 nu este identică cu cea din fig. 5 pentru toate frecvențele, cele două sînt echivalente în regiunea de tăiere pentru valori ale parametrilor care conduc la comportări asemănătoare ale modelului și experimentului.

Din ceea ce se știe despre structura sistemului uman neuromuscular, se poate concluziona că este foarte probabil ca numai necesitățile procesului de integrare să implice „activitate de calcul” la nivelele superioare ale sistemului nervos central. Cînd sînt implicate elemente de comandă de ordin superior, cum ar fi K/s^2 , operatorul uman produce adesea o ieșire discretă sau pulsativă a manipulatorului. Intrările gaussiene produc adesea ieșiri ale manipulatorului, cu distribuții bimodale ale amplitudinii. Distribuția bimodală a amplitudinii este creată adesea, de comenzi de ieșire denumite comenzi „bang-bang”, adică asemănătoare ieșirii unui releu. Comportarea pulsativă sau impulsivă a operatorului uman în acționarea unor elemente de

comandă de ordin superior poate reprezenta o încercare de generare a unei ieșiri de comandă $u_c(t)$ mult mai ușor integrabilă decât cea existentă în absența pulsării, adică aplicabilă unor forme de undă a căror integrare necesită un minimum de activitate de nivel înalt a sistemului nervos central.

Scopul principal al introducerii modelului structural al operatorului uman este, după cum s-a menționat, de a modifica modul de considerare a capacităților umane fundamentale și a limitărilor lor în programele clasice de urmărire. Importanța acestui fapt a fost demonstrată de către rezultatele testelor experimentale de zbor desfășurate la NASA, care au arătat dependența dintre egalizarea realizată de către pilot și caracteristicile manipulatorului.

Dacă pe dispozitivul de afișare grafică sînt reprezentate atît eroarea cît și informațiile de intrare, atunci este denumit dispozitiv de afișare grafică de urmărire. În fig. 7 este prezentat modelul structural al operatorului uman pentru programe de urmărire (Hess, 1981), alcătuit similar celui prezentat în fig. 6. Linii îngroșate indică diferențele dintre fig. 6 și fig. 7.

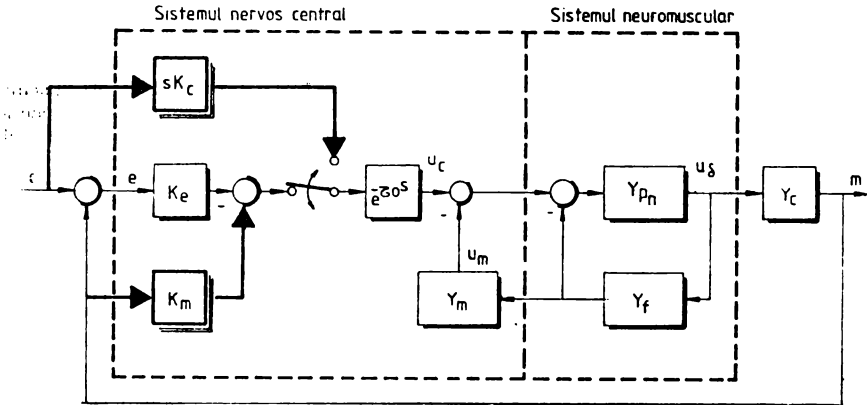


Fig. 7. Modelul structural al operatorului uman pentru programe de urmărire.

Modelul din fig. 7 este alcătuit făcînd următoarele presupuneri: permițînd constantelor de timp T_1 și T_2 să aibă o gamă suficient de largă de valori, funcția de transfer m/u_c poate fi considerată a fi un integrator pentru un domeniu larg de frecvență sub frecvența de tăiere. Mărirea acestor constante poate fi aleasă astfel încît căderea de fază la frecvențe joase, care în mod normal este asociată cu dinamica compensatorie a operatorului uman, să dispară. În plus, acceptînd izomorfismul modelului, dacă operatorul poate sesiza viteza de variație a intrării, dinamica elementului comandat poate fi efectiv inversată la frecvențe joase, permițînd ca $dc(t)/dt$ să fie intrarea primară a buclei din față. Limitarea lărgimii de bandă a elementului comandat inversat nu este prea importantă, deoarece operatorul uman selectează în general frecvența de tăiere sub lărgimea de bandă a intrării, dacă este posibil (McRuer și Krendel, 1974).

Utilizînd modelul structural din fig. 6 poate fi evaluată o medie care reflectă dificultățile programului sau calitățile în manipulare.

$$\sigma_{n_m} = \sqrt{\frac{1}{\pi} \int_0^{\infty} \left| \frac{u_m}{c}(j\omega) \right|^2 \Phi_{ce}(\omega) d\omega} \quad (7)$$

Mărirea lui σ_{n_m} variază cu aproape două ordine de mărime de la cele mai ușor la cele mai greu de manevrat elemente comandate.

Lucrarea 11.4/C3 propune studierea componentelor dinamice ale sistemelor de conducere a autovehiculelor în scopul asigurării siguranței traficului. Caracteristicile dinamicii autovehiculelor se schimbă brusc cînd automobilul trece de la o porțiune uscată la una umedă a drumului. În plus, la mersul în curbă, sistemului i se adaugă în plus o forță laterală și un moment. Comportările adaptive ale sistemelor de conducere a autovehiculelor în aceste situații critice

sînt studiate folosind pentru experimentare un simulator de autovehicul. Pentru explicarea comportărilor adaptive s-a folosit un nou model matematic pentru conducătorul de autovehicul, model pentru a cărui realizare a fost utilizată teoria reglării adaptive cu model etalon și a reglării predictive. În fig. 8 este prezentată schema bloc a modelului de reglare a sistemului de conducere a autovehiculului.

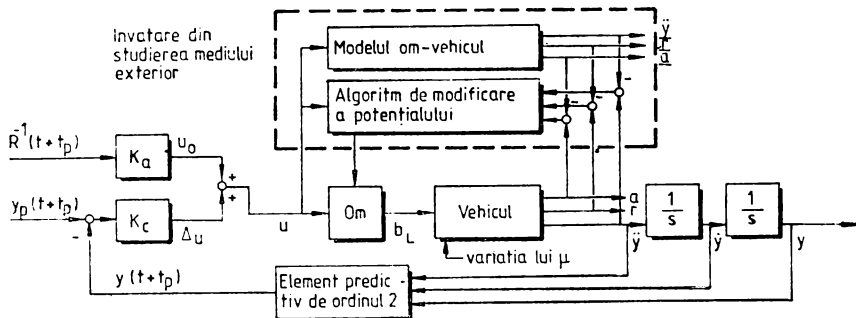


Fig. 8. Modelul de reglare adaptivă a sistemului de conducere a autovehiculului.

Nivelul de reglare predictivă. Funcțiile conducătorilor de autovehicule sînt perceperea variabilelor de ieșire ale autovehiculelor, compensarea prin reacție pozitivă sau negativă și întârzierile datorate sistemului neuromuscular.

(a) *Predicția de ordinul 2.* Conducătorul poate prevedea devierea viitoare a autovehiculului ținînd cont de forma curbei. Conducătorii experimentați pot prevedea devierea viitoare printr-o lege de predicție de ordinul 2.

$$y(t + t_p) = y(t) + t_p \dot{y}(t) + \frac{1}{2} t_p^2 \ddot{y}(t) \quad (8)$$

unde t_p este timpul de predicție.

(b) *Compensarea.* Conducătorii pot compensa anticipat unghiul volanului prin reacția pozitivă realizată prin perceperea razei viitoare a curbei $R(t + t_p)$ chiar cînd raza variază

$$u_0(t) = K_a \cdot R(t + t_p)^{-1} \quad (9)$$

unde K_a este o constantă oarecare sau o funcție. Presupunînd că autovehiculul merge pe o curbă circulară, R este constantă și u_0 este o valoare staționară. Dacă există o mică deviație față de punctul prezis, unghiul volanului este compensat folosindu-se această deviație drept reacție negativă

$$\Delta u(t) = K_o (y_d(t + t_p) - y(t + t_p)) \quad (10)$$

unde $y_d(t + t_p)$ este referința în punctul prezis pe curbă. Valoarea totală a unghiului volanului este suma dintre

$$u(t) = u_0 + \Delta u(t) \quad (11)$$

(c) *Sistemul neuromuscular.* Datorită întârzierii introduse de sistemul neuromuscular, valoarea reală a unghiului volanului nu poate fi realizată utilizînd ecuația (11). Se poate presupune că sistemul neuromuscular constă dintr-o întârziere de ordinul întâi și un timp mort, după cum urmează

$$b_L(t) + t_1 \dot{b}_L(t) = K_m(t) \cdot u(t - t_d) \quad (12)$$

unde $b_L(t)$ este unghiul real al volanului, t_1 este constantă de timp a filtrului de ordinul întâi, t_d este timpul mort și K_m este egal cu 1 dacă nu se face adaptare.

Nivelul de reglare adaptivă. După cum s-a menționat anterior, amplificarea buclei deschise a sistemului variază semnificativ la modificarea coeficientului de frecare și mai ales la adăugarea unei forțe laterale și a unui moment datorite mersului autovehiculului pe o curbă cu suprafață umedă. Ca urmare, autovehiculul deviază foarte mult spre exteriorul curbei, dacă nu se face sau nu se face suficient o compensare prin reacție negativă.

Ținând cont de studiile experimentale efectuate folosind un simulator, conducătorul poate manevra adaptiv volanul autovehiculului. Deoarece factorul de amplificare al buclei deschise a sistemului variază cel mai mult, cea mai importantă adaptare este modificarea factorului de compensare K_m descris în ecuația (12).

Pentru modificarea lui K_m se aplică teoria reglării adaptive cu model etalon. Blocul de reglare adaptivă este prezentat în partea superioară a fig. 8. Blocul de reglare adaptivă constă dintr-un model etalon al sistemului fizic om-vehicul, o parte de comparare a ieșirilor modelului etalon și sistemului real cu ajutorul căreia se detectează modificarea dinamicii reale, și un mecanism de modificare a parametrilor.

Se presupune că aceste părți se formează în mintea conducătorului într-o perioadă corespunzătoare printr-un proces de învățare. Utilizând sistemul adaptiv, conducătorul își modifică modul de acționare astfel încât ieșirile autovehiculului real pot urmări ieșirile modelului etalon.

Se presupune că pot fi percepute variabilele de ieșire, cum ar fi viteza de alunecare $r(t)$, unghiul de alunecare laterală $a(t)$ și accelerațiile laterală $y(t)$, ca și variabila de intrare în sistemul neuromuscular $u(t)$.

(a) *Viteza de alunecare și unghiul de alunecare laterală.* În blocul de reglare adaptivă din fig. 8, erorile vitezei de alunecare și unghiului de alunecare laterală

$$\begin{aligned}\Delta a(t) &= a(t) - \underline{a}(t) \\ \Delta r(t) &= r(t) - \underline{r}(t)\end{aligned}\quad (13)$$

sînt detectate într-o parte de comparare. Folosind teoria stabilității Liapunov se poate găsi un mecanism posibil pentru modificarea factorului de compensare K_m

$$K_m(t) = -(C_p + C_i/s) (h_r (\Delta r(t)) + h_a \cdot \Delta a(t)) \cdot u(t) \quad (14)$$

unde integrarea este esențială pentru a obține adaptarea stabilă a sistemului nelinear cînd factorul de amplificare al modelului dinamic al autovehiculului scade sub o anumită valoare pe drum umed, factorul de compensare $K_m(t)$ devine convergent spre o anumită valoare după un oarecare timp de adaptare.

(b) *Accelerația laterală.* Dacă unghiul de alunecare laterală este foarte mic, dinamica răspunsului vitezei de alunecare este aproape similară cu cea a accelerației laterale. În condiții experimentale, răspunsul unghiului de alunecare laterală este mult mai mic decît cel a vitezei de alunecare. Se selectează de aceea accelerația laterală $y(t)$ pentru a detecta variația parametrilor autovehiculului. Eroarea accelerației laterale

$$\Delta \bar{y}(t) = \bar{y}(t) - \underline{\bar{y}}(t) \quad (15)$$

este sesizată într-un bloc de comparare din fig. 8. Legea posibilă de adaptare care utilizează această informație este

$$K_m(t) = -(C_p + C_i/s) (h_r \Delta \bar{y}(t)) \cdot u(t) \quad (16)$$

Parametrii ecuațiilor (14) și (16) sînt identificați experimental folosindu-se un simulator.

Analizînd cu ajutorul simulatorului un accident datorit parcurgerii de către autovehicul a unei suprafețe umede în curbă, posibilitatea producerii accidentelor, inclusiv a cazurilor periculoase, cînd vehiculul de deplasează în apropierea axei drumului este de aproximativ 60—80%, în timp ce spre marginile laterale este de aproximativ 20—40%. Acest fapt nu depinde de direcția curbei. În lucrare sînt prezentate rezultatele experimentale și cele obținute simulîndu-se pe calculator comportarea modelului.

Viteza vehiculului și raza curbei sînt parametri foarte importanți la mersul în curbă, deoarece ecuațiile dinamicii vehiculului depind de viteza lui, iar forța laterală și momentul depind de viteza lui și forța centrifugă.

Conform rezultatelor experimentale și simulării pe calculator, rularea cu 80 km/oră este periculoasă. Adaptarea modului de conducere a autovehiculului este efectuată cu succes la 60 km/oră, dar este periculoasă la viteze mai mari de 70 km/oră.

Pentru a detecta acționarea defectuoasă a operatorului uman afectat de efectele nocive ale mediului exterior, consumului de alcool sau alte droguri au fost dezvoltate și validate dispozitive de testare a comportării umane. În lucrarea 11.4/C2 sînt prezentate un test cibernetic și o strategie de decizie care asigură mijlocul de detectare a acționării defectuoase a operatorului uman datorită alcoolului și altor droguri.

Operatorul uman trebuie să stabilizeze elemente reglate a căror dinamică este progresiv instabilă.

Teoria sistemelor și datele experimentale au verificat că utilitatea de acționare a operatorului uman în acest test este influențată de caracteristici cibernetică de bază și că performanțele obținute în timpul testării sînt influențate de efectele acționărilor defectuoase asupra acestor caracteristici.

Evaluarea abilității de acționare a operatorului uman este determinată cu ajutorul unei strategii de decizie statice.

Operatorul uman are posibilitatea acționării pe mai multe căi pentru a depăși un criteriu de performanță prestabilit. Sînt descrise diverse proceduri de stabilire a criteriului de performanță, proceduri care au fost testate experimental cu aparate instalate în automobile pentru a-i descuraja pe conducătorii de vehicule de a consuma alcool atunci cînd sînt la volan.

Testul implică două componente, și anume o componentă de comandă și o strategie de identificare. Componenta de comandă, denumită componentă de urmărire în regim critic (CTT), a fost utilizată pentru prima dată ca o metodă de detectare a acționării defectuoase de către compania General Motors în anul 1973.

Teoria de decizie statică pentru optimizarea strategiei de detectare a fost dezvoltată și validată în teste de laborator.

Componenta de urmărire în regim critic (CTT). Dinamica componentei constă dintr-un element de comandă instabil și un circuit de autoacordare. În fig. 9 a și b sînt prezentate elementele componente ale unui CTT și analiza stabilității locului rădăcinilor. Nu este necesară o intrare, deoarece zgomotul este suficient pentru a perturba sistemul.

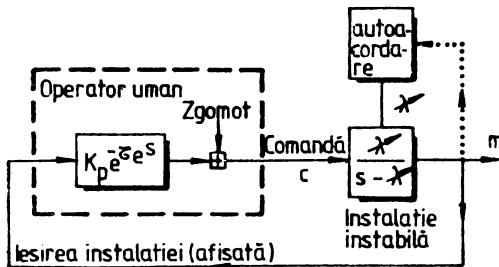


Fig. 9 a. Schema bloc.

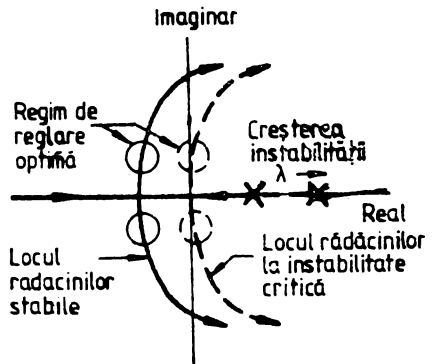


Fig. 9 b. Locul rădăcinilor.

Rădăcina instabilă λ are inițial o valoare mică. Subiectul începe să execute testul, instabilitatea unității crește (rădăcina se mută în planul din dreapta). Timpul total al testului este de 30 secunde.

După cum se vede din fig. 9, performanțele unui CTT pentru un subiect depind de o constantă de timp vizual/motor (τ_e), de o constantă de amplificare K_p și de zgomotul intern sau de acțiuni de comandă aleatoare. Constanta de timp de întârziere determină forma locului rădăcinilor, în timp ce constanta de amplificare K_p determină punctul curent pe locul rădăcinilor.

Creșterea instabilității (λ) translatează locul rădăcinilor la dreapta sau în direcția instabilității. Factorul de amplificare determină primele două rădăcini ale buclei închise (constanta

de timp de întârziere dă un număr infinit de rădăcini, dar perechea de la frecvența cea mai mică dictează caracteristicile de stabilitate).

Strategia optimă este de a da o valoare constantă K_p , astfel încât poli buclei închise să se afle pe axa imaginară. Acționările defectuoase pot să afecteze capacitatea de comandă a operatorului uman pe 3 căi: 1) prin creșterea timpului de întârziere vizual/motor; 2) diferențele în reglările precise ale lui K_p ; 3) creșterea zgomotului. Orice combinație a acestor 3 efecte ale acționărilor defectuoase pot să reducă raportul obținut de un CTT pentru λ_c .

Numeroase studii au demonstrat efectele alcoolului asupra lui λ_c .

Strategia de detectare a acționărilor defectuoase (IDS). Obiectivul unui IDS este de a mări șansa de a detecta acționările defectuoase ale unui operator cu un număr minim de CTT. Optimizarea și dezvoltarea unui IDS începe cu analiza proprietăților statistice ale performanțelor unui CTT (λ_c).

Bazat pe o analiză statistică a numeroase date, IDS a stabilit următoarele cerințe:

- 1) diferențele importante de performanță între operatori necesită criterii de performanță individualizate;
- 2) variația înregistrărilor performanțelor stabile și variația performanțelor încercărilor independente permit utilizarea unei strategii simple de eșantionare multiplă;
- 3) îmbunătățirea priceperii în termen lung ar necesita proceduri de eșantionare și înnoire periodică a criteriului de performanță.

Caracteristici statistice importante de performanță raportate la elaborarea unui IDS pot fi ilustrate cu funcții de distribuție cumulative. Distribuțiile sînt normalizate și mediate pe un număr mare de subiecți și reprezentate grafic pe formulare speciale (o distribuție gaussiană apare reprezentată ca o linie dreaptă). Datele sînt distribuite normal pe o gamă largă și efectele alcoolului sînt indicate clar.

Cerința de bază a unui IDS este că performanța eșantionată a subiecților trebuie să depășească un nivel de trecere prestabilit. Au fost analizate mai multe strategii de eșantionare cu baze de date vechi și prin diverse criterii a fost selecționat doar un prag din mai multe încercări admise. Cu această strategie și presupunînd încercări independente, probabilitatea de a greși testul este probabilitatea de a greși pentru o singură încercare ridicată la o putere egală cu numărul de încercări admise:

$$P_{\text{greșeală}}(N_{\text{încercări}}) = P_{\text{greșeală}}^N(\text{singură încercare}) \quad (17)$$

Performanța de control a operatorului uman măsurată cu o componentă de urmărire în regim critic (CTT) s-a dovedit a fi relativ sensibilă la efectul acționărilor defectuoase, în special alcoolul. Strategia de detectare a acționărilor defectuoase elaborată ulterior a arătat o capacitate rezonabilă de a decide între conducătorii de automobile care acționează defectuos și cei care acționează corect. Combinația CTT/IDS s-a dovedit a lucra în mod fiabil o perioadă de timp extinsă într-un mediu de lucru în care utilizatorii nu erau în mod necesar motivați să coopereze cu funcția de control. Au fost elaborate proceduri pentru instruirea operatorilor și utilizatorilor și pentru fixarea unor praguri de trecere bazate pe capacitatea individuală de performanță. CTT/IDS poate fi utilizat ca un dispozitiv cibernetic de filtrare și în alte servicii ca urmărirea zilnică a operatorilor de vehicule comerciale sau guvernamentale, urmărirea operatorilor din procesele industriale sau centrale electrice.

Strategia de detectare a acționărilor defectuoase (IDS) poate fi utilizată și în alte activități cibernetice care se dovedesc a fi sensibile la alte aspecte sau la acționări defectuoase ale operatorului uman.

Cercetări recente s-au făcut în domeniul telecomenzii vehiculelor roboți și a manipulatorilor, după cum se arată și în lucrarea 11.4/C6. Aceste cercetări, făcute în cadrul Laboratorului sistemelor om-mășină — MIT, analizează utilizarea calculatorului în ajutorul operatorului uman precum și afișările utilizate în cazul manipulatorilor utilizate în oceane la mare adîncime. Aceste sisteme sînt utilizate pentru a înlocui scafandri umani în cercetările la mare adîncime și de mari proporții și la operațiile care au tendința de a fi foarte riscante și costisitoare.

Operatorul uman îndeplinește funcția de controlor supervisor, comunicînd cu calculatorul care este în legătură cu un vehicul/manipulator îndepărtat; calculatorul interpretează și execută comenzi cu ajutorul unor traducătoare proprii ale vehiculelor și manipulatorilor.

Multe firme au procedus vehicule utilizate nu mai pentru prospectare, dotate cu camere video sau vehicule utilizate atât pentru prospectare, cit și pentru lucru, dotate cu camere video, brațe și dispozitive mecanice. Pentru condiționarea acestor vehicule de la suprafața vapoarelor,

operatorul uman urmărește un monitor video și comandă vehiculele și/sau manipulează brațe. O parte din sisteme au comandă master/slave cu multe grade de libertate. Dându-se vehiculelor câteva grade de libertate și permițând operatorului uman să comunice cu vehiculele intermi-tent printr-o legătură de comunicație acustică (lățime de bandă joasă), operatorul uman înce-tează să mai fie un „controlor în buclă”, deservind în schimb un operator supervisor de la distanță care transmite instrucțiuni la calculator. Calculatorul execută task-urile și închide bucla, utilizând senzorii proprii ai vehiculelor.

Pentru operatorul supervisor pot fi identificate 5 funcții :

- (1) planificarea misiunilor la fiecare nouă actualizare a instrucțiunilor către robot ;
- (2) comunicarea acestor instrucțiuni către calculator ;
- (3) afișarea de supraveghere care să asigure că programele se execută corect ;
- (4) intervenția pentru actualizarea programului sau pentru a prelua comanda în caz de avarie ;
- (5) folosirea experienței.

BIBLIOGRAFIE

- Hess, R.A. (1981) Pursuit tracking and higher levels of skill development in the human pilot. IEEE Trans. on Sys., Man, and Cybernetics, SMC-11, 262—273.
- Mc Ruer, D.T., și E. Krendel (1974) Mathematical models of human pilot behavior. AGAR-Dograph No 188.

SUPRAVEGHEREA PROCESELOR ÎN SISTEMELE OM-MAȘINĂ INDUSTRIALE

Funcționarea sistemelor om-mașină pentru supravegherea proceselor tehnologice ridică o serie de probleme legate de interacțiunile dintre tehnologie și structurile organizatorice. Importanța unor asemenea probleme crește în condițiile folosirii de sisteme multinivel, care îngrează supervizarea mai multor procese distincte și includ și alte funcții ale întreprinderii. *Luzarica 11.4/D2* analizează și evaluează asemenea interacțiuni pe baza unor studii de caz. Autorii lucrării ajung la concluzia necesității de a fi implementate relații modificate față de cele tradiționale între tipul de proces tehnologic, instrumentația de automatizare, echipamentul de calcul, configurarea camerelor de comandă și a consolelor operator, activitățile de supervizare și structurile organizatorice. Autorii se referă numai la procese continue și semicontinue, dar se poate considera că rezultatele studiului lor sînt utile și în cazul proceselor discontinue (exemplu construcția de mașini).

În practica actuală, operatorii lucrînd în echipe, în proiectarea unui sistem om-mașină este necesar să fie luate în considerare 2 etape în repartizarea de sarcini :

- repartizarea de sarcini sistem între mașină și om ;
- repartizare de sarcini „om” între operatori, cu divizarea corespunzătoare a responsabilităților și stabilirea unor reguli de comunicare, altfel spus determinarea unei structuri organizatorice.

Luarea menționează modul tradițional de abordare, în care organizarea muncii operatorilor este influențată univoc de următorii factori :

- tehnologia utilizată în procesul supervizat ;
- tradiția de organizare din întreprindere ;
- configurarea camerelor de comandă și a consolelor și comunicația om-mașină, dependentă univoc la rîndul lor de echipamentul de calcul și instrumentația de automatizare disponibile comercial.

În lucrare sînt menționate preocupările în acest domeniu ale altor specialiști. Există, în opinia autorilor lucrării, 4 categorii de asemenea preocupări :

- Experimente de laborator cu sisteme om-mașină, cu luarea în considerare a unor variabile organizatorice. Este menționată o lucrare a lui Terano, Murayama și Akiyama (1983) care, experimentînd sisteme om-mașină cu 1 și 2 operatori, au ajuns la concluzia că fiabilitatea unui operator scade atunci cînd el cooperează cu un al doilea operator, față de cazul concurenței sau al muncii individuale ;

— Încercări de a optimiza organizarea muncii operatorilor în condiții de tehnologie impusă. Este menționată o lucrare a lui Bailey (1983), care, în opoziție cu Taylor sau cu Weber, consideră că pentru creșterea eficienței muncii este necesară o creștere a gradului de satisfacție. O limită a cercetărilor lui Bailey ar consta în slaba utilizare a gradelor tehnologice de libertate susțin autorii lucrării 11.4/D2 ;

— Studii descriptive ale interacțiunilor dintre tehnologie și organizare. Sint amintite analizele, de acum clasice, ale lui Woodward (1965) și Perrow (1967), punându-li-se în evidență limitele (Perrow ajunge la o soluție weberiană pentru cazul tehnologiilor de rutină). Este menționată de asemenea o lucrare, mult mai recentă, avându-l ca principal autor pe Ekkers (1980), care se ocupă de relația dintre gradul de satisfacție și gradul de automatizare, cei doi factori fiind, de la un anumit nivel de automatizare, contradictorii ;

— Experimente de teren cu adaptarea mutuală a tehnologiei și organizării. Este citat aici Bibby (1975), cu un exemplu de minimizare a deranjamentelor într-o oțelărie, minimizare obținută nu prin îmbunătățirea instrumentației, ci prin schimbări organizatorice: fiecare operator răspunde nu numai de unitatea proprie, ci și de influența asupra unității din aval. Pentru ca noua responsabilitate să poată fi îndeplinită efectiv, operatorului i se furnizează informații din aval.

Autorii lucrării 11.4/D2 ne înfățișează apoi propriile lor preocupări. Ei pornesc de la definirea funcțiilor operaționale și a relațiilor dintre ele, sau, în exprimarea lor, a cadrului funcțiilor operaționale. Prin funcții operaționale autorii înțeleg categorii de sarcini atribuite omului și mașinii pentru asigurarea siguranței în exploatare, a nivelului calitativ și cantitativ al producției, al eficienței (definiția este reluată dintr-un articol anterior al unuia din autori, Rijnsdorp).

Cadrul funcțiilor operaționale constituie un sistem complex structurat pe mai multe nivele :

— nivelul de bază, reprezentat de bucla : proces (cu unități de corecție asociate) — măsurare-automatizare (protecție, automatizare secvențială, reglare) — proces ;

— nivelul de compensare/corecție, reprezentat de legătura măsurare-evaluare stare proces-diagnoză-compensare sau corecție a proastei funcționări ;

— nivelul de conducere adaptivă, reprezentat de legătura măsurare-evaluare stare proces-estimare a calității și eficienței-conducere adaptivă (prin ajustare de temporizări, de scheme de comandă, de mărimi de referință, etc.) — automatizare ;

— nivelul de optimizare, reprezentat de legătura măsurare-actualizare model-optimizare (prin rețete, valori critice, valori impuse)-conducere adaptivă. Menționăm că autorii au în vedere optimizarea atât a procesului, cât și globală, a secției/intreprinderii. De la secție/intreprindere se transmit spre proces valori critice, debite etc. ;

— nivelul întreprindere, reprezentat de legătura măsurare-informare operativă-programare producție și gestiune stocuri-optimizare.

Referitor la nivelul întreprindere, autorii menționează că legătura informare operativă-programare producție și gestiune stocuri se realizează atât direct, cât și prin intermediul funcției de planificare. Funcțiile de planificare, programare producție și gestiune stocuri interacționează cu alte funcții la nivelul întreprindere (livrări, întreținere etc.), care la rândul lor pot fi alimentate cu date de către informarea operativă. Existența acestor legături la nivelul întreprindere îi conduce pe autori (care îl citează în acest sens pe Forrester) la afirmarea necesității unui sistem integrat de conducere a întreprinderii, a cărui filozofie să fie bazată pe triada beneficiari-producție-furnizori, sistem care să fie alimentat de către informarea operativă.

Studierea acestui cadru complex al funcțiilor operaționale îi conduce pe autori la concluzia că maximizarea gradului de automatizare nu este în mod necesar echivalentă cu optimizarea conducerii procesului. Funcțiile operaționale vitale apar a fi măsurarea și evaluarea stării procesului. Un grad ridicat de automatizare reduce activitatea operatorului în principal la monitorizare. O asemenea situație (sunt citate, în sprijinul acestor afirmații, lucrări de van Drof-felar, 1975 ; Ekkers, 1980 ; și Bainbridge, 1983) este o sursă de stress, de degradare a satisfacției în muncă și este psihologic nepotrivită pentru comportarea în situații neprevăzute. Și autorii conchid, „nu este întotdeauna recomandabil să țintești spre cel mai înalt grad de automatizare posibil”.

În continuarea studiului lor autorii se ocupă de funcția de evaluare a stării procesului, funcție cheie în cadrul funcțiilor operaționale. „Cu instrumentația modernă”, spun ei, „informația este prezentată în camera de comandă pe dispozitive de afișare pentru a fi monitorizată și interpretată de către operatori. Totuși, în unele sectoare industriale, această informație

trebuie suplimentată prin observarea directă a condițiilor, de exemplu la benzi transportoare, uscătoare, centrifuge etc.“

Autorii disting 3 cazuri pe care le studiază separat :

— starea poate fi evaluată prin afișarea pe consola operator, eventual cu informații suplimentare obținute prin televiziune în circuit închis ;

— starea poate fi evaluată numai parțial prin afișarea pe consola operator, fiind necesară și o supraveghere locală ;

— evaluarea stării necesită supravegherea locală în timpul unor situații frecvente de porniri, opriri, purjări etc., dar poate fi monitorizată în camera de comandă în situații normale.

Primul caz se întâlnește în procese în care se prelucrează mase de lichid sau gaz (exemplu, petrochimia), al doilea caz se întâlnește în procese în care se prelucrează materiale solide sub formă pulverulentă sau granulară, iar al treilea caz se întâlnește în activități de deplasare/stocare. Cele 3 cazuri sînt luate în discuție în legătură cu activitățile operatorilor asociați cu console-operator și cu organizarea echipelor de operatori.

În primul caz, arată autorii, evaluarea stării este efectuată de operatori centrali, asistați însă de operatori de teren pentru relativ rarele acțiuni de comutare, pentru cooperarea cu echipele de întreținere și pentru supravegherea generală a echipamentelor. Procesul de prelucrare supravegheat fiind alcătuit din mai multe „unități-proces“, acestea sînt alocate uneia sau mai multor console, fiecare cu unul sau mai mulți operatori centrali, asistați de operatori de teren.

Din punct de vedere al organizării muncii operatorilor, autorii arată că trebuie rezolvate următoarele probleme :

— cite unități-proces să se aloce fiecărui operator ;

— cum să se mențină în munca operatorilor centrali un grad de încărcare relativ constant în timp.

Referitor la prima problemă, se obișnuiește ca volumul de muncă al unui operator central să fie exprimat în număr de bucle de reglare supravegheate. În cazul sistemelor automate moderne (în cascadă, multinivel și/sau care folosesc algoritmi multivariabili) această abordare este depășită, susțin autorii, care propun exprimarea volumului de muncă în organe de execuție supravegheate. Sînt menționate în context două aspecte care influențează volumul de muncă al unui operator :

— munca unui operator central este încărcată cu un volum important de comunicații, datorită contactelor frecvente cu operatorii de teren, alți operatori centrali, superiori, tehnologi, tehnicieni de întreținere ;

— este o mare diferență între operarea tradițională, în care se urmărește menținerea unor condiții prestabilite și operarea modernă, în care valorile impuse se modifică frecvent, iar limitele de încadrare ale valorilor critice sînt foarte strînse.

Referitor la problema menținerii în munca operatorilor centrali a unui grad de încărcare relativ constant în timp (ceea ce înseamnă evitarea diferenței de încărcare situație normală-situație de urgență), sînt propuse următoarele soluții :

— includerea de sarcini de tip „background“ în munca operatorilor centrali, care să poată fi neglijate pe durata situațiilor de urgență ;

— delegarea către operatorul central a responsabilității de coordonare a activității operatorilor de teren ;

— asistarea operatorului central pe durata situațiilor de urgență de către unul din operatorii de teren sau de către un alt operator central (ceea ce implică o cameră de comandă centrală, care să concentreze toți operatorii centrali). Pentru ca asistarea să fie de calitate este bine să se asigure o rotație periodică a operatorilor centrali între ei (deci de la un grup de unități proces la altul), precum și a operatorilor centrali cu operatorii de teren.

Autorii consideră că aspectele organizatorice au implicații asupra proiectării optimizate a camerei de comandă și a consolelor (forma și mărimea consolei, forma și mărimea caracterelor afișate etc.).

Este luat apoi în discuție cazul proceselor în care se prelucrează materiale solide sub formă pulverulentă sau granulară, la care sesizarea stării se face parțial prin afișare pe consolă operator, în camera de comandă, și parțial prin supraveghere directă. Aici autorii consideră că o structură organizatorică cu 2 operatori (central și de teren) pentru un grup de unități-proces este nepotrivită, implicînd un volum important de comunicații, cauză de erori, neînțelegeri, frustrare. Structura organizatorică optimă în acest caz presupune pentru un grup de unități proces un singur operator circulînd între camera de comandă și teren.

Necesitatea unei camere de comandă centrale, care să concentreze toți operatorii este argumentată astfel:

- permite cooperarea între operatori pentru optimizare globală;
- oferă posibilitatea asistenței mutuale.

În cazul în care distanța dintre camera de comandă centrală și proces ar fi prea mare, se impun camere de comandă locale, dar, susțin autorii, care să concentreze totuși cel puțin 2 operatori (supravegherea a 2 grupe de unități-proces, deci).

Autorii iau apoi în discuție cazul proceselor în care sînt frecvente porniri, opriri, purjări etc. (exemplu, activități de stocare/deplasare). În acest gen de procese este de efectuat un mare volum de supraveghere de teren, camera de comandă nefiind necesară.

Dacă însă procesul se desfășoară pe o suprafață foarte mare, camera de comandă devine necesară, pentru a putea supraveghea procesul în ansamblu.

După această discuție amănunțită a funcției de evaluare stare proces, autorii se referă la nivelele superioare ale cadrului funcțiilor operaționale. Ei arată că în prezent există o puternică tendință de integrare a proceselor individuale, prin schimbul de energie și materiale. Această integrare impune o bună calitate a comunicării dintre operatori, preferabil prin contacte directe vizuale și auditive (în special în situații de urgență), ceea ce este posibil numai în camere de comandă combinate și are implicații în proiectarea optimizată a consolelor (de exemplu, apare necesar ca dimensiunile consolelor să permită contacte vizuale între operatori).

În ceea ce privește nivelul de optimizare, autorii menționează un experiment dintr-o întreprindere chimică, care a dus la concluzia existenței unui set optimal de valori discrete ale debitelor de agenți energetici. Operarea este optimă cînd echipamentele energetice asigură numai aceste valori discrete, în funcție de necesitățile momentare ale întreprinderii. Cînd apare o mare cerere de energie, se impune funcționarea unor echipamente (turbine, cazane, etc.) la capacitatea maximă, cînd există o abundență de energie, echipamentele respective ar trebui să fie deconectate. Conducerea optimală este diferită de cea tradițională, iar echipei de operatori li revin deci sarcini suplimentare: supravegherea simultană a unor algoritmi de comandă divizați pe operatori, capacitatea de a răspunde rapid la schimbările în necesarul de agenți energetici. Aceasta impune adoptarea unor soluții de software „prietenos“ pentru console: posibilități eficiente de dialog, scheme adecvate etc.

Trecînd la nivelul întreprinderii, autorii pledază pentru un sistem integrat de conducere, implementat pe o rețea de calculatoare, deoarece, spun ei, „fără o rețea de calculatoare, o asemenea integrare ar necesita stringerea tuturor persoanelor cu atribuții de programarea producției într-o echipă bine îmbinată, ceea ce ar interfera cu restricțiile de timp de comunicare cu personalul operativ. O rețea bine proiectată permite fiecăruia să rămînă în sectorul propriu.“

În concluzie, abordarea preconizată de autori a interacțiunilor tehnologie-structură organizatorică diferă față de abordarea tradițională, menționată la începutul lucrării.

Organizarea muncii operatorilor nu trebuie să depindă de tradiția existentă în întreprindere, ci invers, într-o oarecare măsură organizarea muncii operatorilor ar trebui să influențeze tradiția.

Organizarea muncii operatorilor trebuie să impună soluția de proiectare a camerei de comandă și a consolei și nu invers. Aceasta impune minimizarea influenței echipamentelor de calcul și instrumentației de automatizare disponibile comercial asupra camerei de comandă și consolei. Sînt de dorit soluții de echipamente și programe „pe măsură“, care să rezulte din soluția de proiectare a camerei de comandă și a consolei.

Calitatea informației furnizate pe consolă determină așadar într-o măsură însemnată succesul unui sistem om-mașină. Aceasta este și opinia autorilor lucrării 11.4/D4 Tsuchiya de la Toshiba Corporation, Tokio și Uchida, care prezintă sistemele „prietenoase“ (operator friendly) folosite în termocentrale din Japonia.

Implementarea în termocentralele japoneze a sistemelor cu calculator reprezintă un succes incontestabil, datorită următorilor factori:

- necesitatea resimțită de utilizatori de reducere a cheltuielilor de exploatare și de minimizare a duratei procedurii zilnice de conectare/deconectare;

- gradul înalt de fiabilitate al sistemelor cu calculatoare de proces, observat în exploatare;

- volumul minim de probleme de interfață între beneficiari, furnizorii echipamentului energetic și cei ai echipamentului de calcul, avînd în vedere că uzual calculatorul este inclus în furnitura complexă;

- legătura funcțională care există între calculator și automatizările centralei ;
 - existența unui serios suport software de testare ;
 - creșterea în timp a complexității acestor sisteme, plecând de la un set minim de funcții.
- Ultimul factor apare, în opinia autorilor, ca fiind cel mai important. Evoluția sistemelor se prezintă astfel :

- comanda pornirii pentru turbine (1968) ;
- comanda conectării pentru cazane (1971) ;
- comanda conectării/deconectării pentru instalațiile auxiliare (1976) ;
- comanda deconectării întregii centrale (1976) ;
- operarea în situație normală (1980) ;
- operarea în situații de urgență (1980) ;

— aplicarea metodelor moderne de conducere automată. Autorii menționează în acest sens un articol al lui Uchida (1981), în care se prezintă sistemul de la termocentrala Buzen, unde metodele moderne de conducere automată au fost folosite la reglarea temperaturii din cazan.

În Japonia strategia de asigurare a necesităților de energie electrică acordă centralelor nucleare rolul de centrale de bază, termocentralelor rezervându-li-se rolul de centrale de vîrf. Pentru acest motiv s-au dezvoltat sisteme om-mașină care să asigure procedura de conectare/deconectare zilnică a termocentralei. Aceste sisteme își propun 4 obiective principale :

- conectare/deconectare sigură, fără șocuri, fiabilă. Autorii îl menționează pe Kato (1978) care într-un articol prezintă pachetul COPOS, destinat acestui obiectiv ;
- reducerea cheltuielilor de exploatare a echipamentelor, Karashima, într-un articol publicat în 1981, descrie implementarea în vederea realizării acestui obiectiv, pe sistemul de la termocentrala Hirono, a unei funcții de determinare a conectării/deconectării optime ;
- reducerea duratei conectării/deconectării ;
- reducerea gradului de încărcare în activitatea operatorului la conectare/deconectare.

Ultimelor 2 obiective le sînt dedicate noile sisteme TOSBAC, cu interfață „prietenosă,, (operator friendly), descrise de către autori.

Interfața om-mașină a calculatoarelor TOSBAC a evoluat astfel :

- consolă dedicată pentru conectări/deconectări (1968) ;
- display alb-negru (1971) ;
- display color alfanumeric, funcții de alarmare și informare (1976) ;
- display color grafic, sistem analogic de alarmare prin voce, funcții de software „prietenos“ (1980). Menționăm că pentru afișare de mesaje în limba japoneză sînt necesare facilități grafice :

- sistem digital de alarmare prin voce, rafinare a funcțiilor de software „prietenos“ pentru procedura de conectare/deconectare zilnică a termocentralei (1983).

Se observă corespondența cu etapele de evoluție a funcțiilor atribuite calculatorului. Menționăm că anii din paranteze indică implementări ale sistemelor în termocentrale.

Pentru a demonstra necesitatea unei interfețe care să asigure „prietenos“, autorii lucrării arată că sistemul nu este conceput în ideea de a opera fără intervenția omului. Mașina colaborează cu omul asistîndu-se reciproc. De aceea extinderea domeniului pe care îl controlează sistemul trebuie să implice creșterea calității informațiilor oferite operatorului.

Efectuarea unei operații de comandă asupra unui echipament se reduce la îndeplinirea a 3 categorii de condiții (autorii reiau aici concluziile unui studiu al lui Tanaka, 1975) :

- condiții care trebuie îndeplinite înainte de începerea operației ;
- condiții care trebuie îndeplinite pe durata efectuării operației ;
- condiții a căror îndeplinire semnifică încheierea operației.

Efectuarea operațiilor de conectare/deconectare a unei termocentrale necesită îndeplinirea secvențială de condiții din cele 3 categorii, condiții care se constituie într-o procedură complexă.

Funcțiile de software „prietenos“ dezvoltate pe sistemele TOSBAC oferă operatorului posibilitatea de a urmări procedura de conectare/deconectare prin următoarele tipuri de afișări pe display :

- indicarea stării unei operații prin afișarea de mesaje de informare asupra îndeplinirii/neîndeplinirii condițiilor ;
- afișarea de „hărți logice“ ale operațiilor cu indicarea îndeplinirii/neîndeplinirii condițiilor ;
- afișarea de scheme sinoptice ale instalațiilor cu indicarea locului unde s-a produs un deranjament ;
- afișarea de histograme.

Aceste funcții se adaugă funcțiilor disponibile de mai mult timp pe display (afișări de alarme, raportări la cerere etc.) și pe consola dedicată pentru conectări/deconectări.

Erorile tipice ale unui operator pot aparține uneia din următoarele 3 faze (după Oxby, 1982):

- percepție;
- decizie;
- operare.

Funcțiile prezentate în lucrare corespund fazei „operare”. În încheiere, se afirmă că tendința de evoluție a sistemelor om-mașină din termocentralele japoneze este spre faza „decizie”. În acest sens, într-un viitor apropiat vor apare sisteme om-mașină bazate pe cunoștințe.

Asistența operatorului în faza „decizie” este un subiect discutat în lucrarea 11.4/D3, care prezintă în acest sens un sistem în curs de testare într-un combinat siderurgic din R.D.G. (VEB Maxhütte Unterwellenborn).

Într-un mare număr de procese tehnologice complexe, arată autorii, omul trebuie să ia decizii de comandă pe baza observării procesului și pe baza propriei sale experiențe și intuiții. Aceste decizii sînt evident subiective (problema este dezbătută și de Brack și Sokollik, 1982; și de Förster și Starke, 1982).

Scopul pe care și l-au propus autorii lucrării 11.4/D3 este acela de a dezvolta instrumente de asistare a deciziei sub forma unor „unități consultative”, în special pentru procese cu următoarele caracteristici:

- informațiile despre proces nu se pot obține decât inexact sau sub formă de evaluări subiective;
- conexiunile interne din proces nu sînt suficient de cunoscute teoretic pentru elaborarea deciziei;
- procesul este nelinear, cu perturbații puternice;
- unele variabile esențiale ale procesului nu pot fi măsurate, sau pot fi măsurate numai indirect.

Aceste „unități consultative” trebuie să dea un suport obiectiv deciziei. În elaborarea propunerilor de decizii, unitățile consultative se bazează pe analiza procesului, pe tehnici de analiza erorilor, pe tehnici de optimizare și de asemenea pe experiența câștigată de colective largi de operatori.

O unitate consultativă își îndeplinește sarcinile în 2 etape:

- recunoașterea și clasificarea situației;
- elaborarea propunerii de decizie.

Cele 2 etape pot fi implementate pe unități de calcul diferite. În acest sens, una din unitățile consultative prezentate în lucrare este implementată pe 2 calculatoare, denumite calculatorul de analiză proces și calculatorul consultant (în sensul în care îl vede operatorul), amîndouă fiind subordonate unui calculator de conducere a procesului tehnologic, subordonat la rîndul lui calculatorului central al combinatului.

Sistemul în curs de testare la VEB Maxhütte Unterwellenborn cuprinde 5 unități consultative, situate pe lanțul convertizor-turnătorie-laminor. În lucrare este prezentată, succint, strategia de funcționare a unității consultative pentru afinare finală de la convertizor.

Asemenea „unități consultative” permit integrarea operatorului cu toate capacitățile sale în sistemul om-mașină și aduc beneficii economice importante (în cazul prezentat, după o perioadă de 2 ani, beneficiul obținut a fost de 10 ori mai mare decît totalul cheltuielilor, afirmă autorii).

Cele 3 lucrări prezentate pînă acum (11.4/D2, D3 și D4) se ocupă de sisteme om-mașină în industrie. Există și un alt domeniu în care sistemele om-mașină își găsesc aplicabilitate, cel al cercetării științifice, al experimentelor de laborator. Este cazul sistemului prezentat în lucrarea 11.4/D5, sistem folosit pentru măsurători în acceleratoarele de protoni.

Acceleratoarele produc particule încărcate, în serii (cicluri de accelerare) la intervale de miliseunde. Pentru determinarea performanțelor acceleratoarelor sînt necesare măsurări repetate asupra fasciculului de particule la momente arbitrare de timp în cadrul ciclului de accelerare.

Instrumentele de măsurat pot fi complet diferite între ele din punct de vedere al principiilor folosite, dar au 2 caracteristici comune: necesitățile de operare și necesitățile severe de timp real.

Necesitățile de operare se referă la folosirea instrumentelor de măsurat în regim multi-utilizator. În timpul fazii delicate de acordare a acceleratorului mai multe perscane pot dori

accesul la același instrument de măsurat în același ciclu de accelerare. Soluția multiplicării instrumentelor de măsurat ar prezenta 2 dezavantaje :

- cost ridicat ;
- lipsă de precizie : diferite instrumente prezintă ușoare diferențe între ele (calibrare, deriivă amplificator etc.), ceea ce reduce posibilitatea de comparare între măsurări.

Limitele de timp real sînt severe : intervalul minim între 2 măsurări este de 2—3 milisecunde. De menționat că o măsurare impune transferul cuvintelor de comandă și uneori a sute de cuvinte de date.

Existența acestor caracteristici comune a condus la soluția de a separa în fiecare echipament de măsurat partea specifică de partea standard (hardware și software). A rezultat o configurație hardware și software, care a fost denumită MTIM (Multiple Trigger for Measurements), la care se poate conecta un instrument de măsurat, independent de complexitate.

Sistemul MTIM este configurat într-un sertar CAMAC, mai multe asemenea sertare fiind cuplate la magistrala serială CAMAC a unui calculator de proces, conectat la rîndul lui la o rețea, prin care comunică cu 6 console de la care sînt accesate măsurările. Un sertar CAMAC conține componentele hardware ale MTIM (4 module) precum și modulele specifice instrumentului de măsurat respectiv.

Configurația software a MTIM este compusă din :

- un modul rezident în calculatorul de proces (comun pentru sistemele MTIM cuplate) ;
- un set de module rezidente în memoria RAM a controlorului sertarului (unul din cele 4 module microprocesor TMS 9900 pe 16 biți cu 16 K memorie RAM), la care se adaugă modulele software specifice instrumentului de măsurat respectiv. Modulele rezidente în memoria controlorului sînt încărcate din memoria calculatorului de proces prin magistrala serială.

Sisteme MTIM, menționează în încheiere autorii, sînt în funcțiune la Centrul European de Cercetări Nucleare de aproximativ 2 ani, avînd în sarcină 10 măsurări diferite.

BIBLIOGRAFIE

- Bailey, J. (1983). *Job design and work organization* ; Prentice Hall.
- Bainbridge, L. (1983). *Ironies of Automation* ; In : Proc. IFAC/IFIP/IFORS/IEA Conf. on Analysis, Design and Evaluation of Man-Machine Systems ; Baden-Baden, Pergamon.
- Bibby, K.S. (1975). *Human and organizational problems of production control* ; In : Proc. IFAC Workshop, Bad Boll (RWK, Frankfurt).
- Brack, G. și P. Sokollik (1982). *Die ingenieurtechnischen Aufgaben bei den operativen Lenkungen der Produktion*. *Chemische Technik*, 34, 2, p. 61—63.
- Droffelaar, H. van (1975). *A field study of stress, experienced by operators supervising a highly automated process* ; In : IFAC Workshop, Bad Boll (RWK, Frankfurt).
- Ekkers, C.L. ș. a. (1980). *Human control tasks* (în limba daneză) ; NIPG, Leiden.
- Förster, D. și J. Starke (1982). *Erfahrungen bei der Anwendung von Entscheidungshilfen für die operative Lenkung*. *Chemische Technik*, 34, 2, p. 64—66.
- Kato, N. (1978). *Application of Computer Control System COPOS to Fossil Fired Power Generation Plant*. 3-rd USA — JAPAN Computer Conference Proc., p. 505—509.
- Karoshima, N. (1981). *Newly Developed Comprehensive Automation Technique Applied to Hirono Thermal Power Station* ; Proc. of American Power Conference, 43-rd Annual Meeting, Chicago, p. 1—44.
- Oxby, K. (1982). *A Multi-Level Alarm Information Processing System Applied to Thermal Power Plant*. IFAC/IFIP/IFORS/IEA Conf. on Analysis, Design and Evaluation of Man-Machine Systems Proc., p. 89—94.
- Perrow, Ch. (1967). *A framework for comparative analysis of organization* ; *Am. Sociological Rev.* 32, p. 194—208.
- Tanaka, S. (1975). *New Concept Software System for Power Generation Plant Computer Control* — COPOS. Proc. of PICA Conference, p. 1—9.
- Terono, T., Y. Murayama și N. Akiyama, (1983). *Human reliability and safety evaluation of man-machine systems* ; In : Proc. IFAC/IFIP/IFORS/IEA Conf. on Analysis, Design and Evaluation of Man-Machine Systems ; Baden-Baden, Pergamon.
- Uchida, M. (1981). *Totally Computer Automated Control System in a Thermal Power Unit*. Proc. of IFAC 8-th Triennial World Congress, xx-135 — xx-141.
- Woodward, J. (1965). *Industrial organization, theory and practice* ; Oxford Univ. Press.

SISTEME DECIZIONALE IERARHICE ȘI METODE DE DECIZIE MULTICRITERIALE

Mat. Ovidiu Gheorghiu,
Mat. Tatiana Călin

I.T.C.I.

SISTEME SOCIO-ECONOMICE

În cadrul *colocviului 11.5* al celui de al IX-lea Congres IFAC, lucrările dedicate sistemelor decizionale ierarhice au fost grupate în două secțiuni. Vom prezenta în continuare prima secțiune, ce cuprinde 4 lucrări, a cincea [11.5/A3], figurând în cuprins, dar nefiind prezentă printre lucrări.

Vom face, pentru început, observația că lucrările secțiunii respective au toate un numitor comun, și anume substratul socio-economic al ideilor și rezultatelor prezentate. În altă ordine de idei, titlul secțiunii — Sisteme decizionale ierarhice — exprimă doar parțial caracteristica lucrărilor ce tratează într-adevăr probleme decizionale dar nu întotdeauna privesc ierarhizat. Trebuie evidențiată diversitatea unghiurilor de abordare a problemelor decizionale, articolele prezentând aspecte teoretice și practice generalizabile pe o arie largă de aplicații. În acest context se remarcă faptul că secțiunea cuprinde o dezvoltare a problemelor metodologice specifice ingineriei sistemelor socio-economice cu referiri la situații concrete din China [11.5/A2], precum și o abordare foarte riguroasă a problemelor de modelare a sistemelor socio-economice și a politicilor aferente, fiind date ca exemple modele ce reflectă particularitățile din Polonia [11.5/A4], Burkov în [11.5/A5], dezvoltând unele rezultate mai vechi, prezintă o serie de principii noi în conducerea economiei planificate, împreună cu exemplificări, în timp ce Zhao, în [11.5/A1] abordează o aplicație a metodei controlului optimal ierarhizat în analiza unui model dinamic de tip input-output.

Sistemele socio-economice sînt sisteme complexe de mari dimensiuni care surprind realități ale căror dinamică și legi de evoluție nu sînt decît în mică măsură cunoscute, dar problemele legate de conducerea acestor sisteme sînt vitale în contextul abordării unei planificări la nivel național. Apare astfel ca necesară o evaluare a potențialului metodologic și instrumentar, în vederea stabilirii unor politici convenabile în activitățile sociale și economice.

Se remarcă o ineficiență a metodologiilor existente în prezent pentru studiul sistemelor socio-economice, la care se adaugă aportul insuficient al instrumentelor matematice disponibile. Lipsa unor exprimări cantitative — sub forma unor ecuații matematice — asupra dependențelor și a dinamicilor disciplinelor specifice a condus la o alternativă mixtă, de combinare a descrierilor cantitative și calitative în cadrul sistemelor de tip om-mașină. Aceste noi structuri decizionale au în vedere utilizarea modelelor cu structură variabilă, crearea rapidă de modele de mici dimensiuni, dar conforme unor reguli predefinite de clasificare, preconizîndu-se de asemenea construirea de modele specifice fiecărui nivel ierarhic, adică modele interconectate ce agregă și dezagregă procesul decizional, într-un cuvînt se încearcă realizarea unei metodologii adecvate de modelare.

Orice știință tinde să încorporeze cât mai multă matematică și să-și crească numărul legilor specifice exprimate sub forma unor descrieri matematice. Privind retrospectiv la științele naturii, se remarcă un progres al metodelor cantitative (exprimate prin ecuații matematice). Se pune însă întrebarea dacă activitățile socio-economice pot fi suficient de bine explicate și analizate cu metode cantitative. Răspunsul dat de practică este negativ, impunându-se în acest sens noi metode, pe care le vom numi *calitative*. Pentru descrierile calitative se preconizează utilizarea tehnicilor „fuzzy” care permit în final o abordare matematică a informațiilor imprecise, vagi. Astfel poate fi măsurat gradul de posibilitate precum și gradul de satisfacere în cadrul unor discipline precum sociologia, economia, psihologia, politica etc., deci acolo unde se constată imposibilitatea unor exprimări cantitative. Se revine astfel la conceptul de satisfacere, dezvoltat de Simon în [1] ca o alternativă a conceptului de optimalitate.

Referindu-se la rezolvarea problemelor socio-economice, Liu semnalează în [11.5/A2] extinderea sistemelor om-mașină și este propusă o combinare a metodelor cantitative cu cele calitative. În mod cert s-a plecat de la observația că pe plan mondial a avut loc o explozie a sistemelor de asistare a deciziei, apărute în ultimii 10 ani, și care reprezintă soluția cea mai avansată în conducerea sistemelor mari și complexe. În cadrul sistemelor suport pentru decizie, modelelor li se acordă un loc important, dezvoltându-se tehnici specifice pentru construirea și rezolvarea lor.

Este de remarcat faptul că în [11.5/A2] sînt argumentate motivele pentru care cercetătorii chinezi, bazîndu-se pe realitățile concrete din țara lor — perioadă scurtă de tranziție la economia de tip socialist, diferențiată istoric în patru etape, suprafață geografică mare și eterogen dezvoltată etc. — au structurat o nouă metodologie de inginerie a sistemelor socio-economice, bazată pe modele de clasificare și modele cu structură variabilă, punîndu-se accentul pe coordonarea om-mașină în contextul sistemelor de asistare a deciziilor. Sînt semnalate aplicații în politica demografică a Chinei și în agricultură, iar actualmente, pe baza acestei noi metode, are loc fundamentarea planului național de învățămînt (educație).

R. Kulikowski, reprezentant de seamă al școlii poloneze de teoria sistemelor și modelare, prezintă în [11.4/A4] o încercare de modelare a sistemului socio-economic, care este privit ca un integrator al subsistemelor demografic, gospodăresc, forță de muncă și economic. Plecînd de la observația că majoritatea modelelor utilizate în planificare sînt concentrate pe procese și politici economice, substratul social fiind eventual reprezentat prin tehnici descriptive sau speculații bazate mai mult pe intuiție decît pe o analiză strictă, Kulikowski dezvoltă o paletă de modele ale sistemului socio-economic ce pot caracteriza nivelul de retribuție, nivelul de educație, nivelul asistenței sociale etc., deci standardul de viață al unei populații. Lucrarea pornește de la premisa că politica socială este responsabilă de distribuția veniturii naționale între diferitele grupuri sociale, precum și la nivelul fiecărei familii. Orice familie își adaptează structura, ca răspuns la politica socială, în așa fel încît resursele de muncă să se dividă între munca exterioră gospodăriei, munca din gospodărie, educarea copiilor etc. Deoarece venitul național depinde de rîndul său de nivelul forței de muncă, rezultă că procesele socio-economice sînt interdependente, factorul dominant fiind politica socială, care poate fi direcționată în vederea maximizării potențialului național și personal.

O primă parte a lucrării [11.5/A4] se ocupă de familie, gospodărie și participarea la muncă a membrilor familiei. Membrii familiei au fost împărțiți în susținători și persoane dependente — adică copiii pînă la 24 de ani, fiind prezentate, conform statisticilor, curbe reprezentînd relația dintre: x — participarea la muncă a părinților, c — raportul dintre numărul de copii și numărul total al membrilor familiei, și a — vîrsta mamei. Scopul acestor statistici este de a construi un model care să surprindă relația dintre x și c , precum și impactul politicii sociale, într-un mod formalizat. Este introdusă funcția utilitate, care caracterizează etalonul familiei medii, influențînd I — venitul total al familiei, precum și V — valorile familiale de tipul serviciilor gospodărești, educația copiilor etc. Aceste două calități I și V sînt exprimate sub forma unor ecuații lineare, iar pentru o funcție utilitate $U(I, V)$ oarecare, care verifică restricții de concavitate și diferențiabilitate natural verificabile, sînt date condițiile de optimalitate cu ajutorul cărora pot fi determinate cele mai bune strategii pentru a fi atinse valorile optime ale lui x și c .

În ceea ce privește modelul forță de muncă, se pleacă de la cererea și oferta existentă. Scopul său este de a ajuta la alegerea celei mai bune politici de distribuție a veniturilor între

diferitele sectoare ale economiei. Sînt exprimate diferențiat cantitatea de forță de muncă în mediul rural și în mediul urban, ținîndu-se cont de migrația sat-oraș și de rata transformării comunelor în orașe. Acești doi factori au fost la rîndul lor exprimați prin modele lineare, iar coeficienții au fost determinați prin regresie lineară. În urma efectuării de experimente se observă că, în general, nu poate fi obținută o stare de echilibru între forța de lucru rurală și cea urbană, datorită faptului că oferta este mai mică decît cererea în ambele sectoare. Pentru economia modelată cu ajutorul unei funcții de producție de tipul Cobb-Douglas, există posibilitatea determinării cererii optime de forță de muncă urbană.

Kulikowski expune în final o altă clasă de modele ce se referă la politicile de distribuire a veniturilor. Sînt considerate mai multe criterii: împărțirea veniturilor între generații (de exemplu, mărirea alocațiilor pentru copii față de mărirea pensiilor), între mediul urban și cel rural, iar în cadrul aceleiași mediu, între diferitele sectoare. Sînt luate în considerare migrarea sat-oraș și urbanizarea comunelor, fiind discutate urmările acestor transformări în dinamica costului vieții. Cîteva strategii de stabilire a raportului retribuțiilor dintre mediul urban și cel rural sînt discutate. Modelele sînt privite ca instrumente pentru definirea unor politici de dezvoltare a mediului social în conexiune cu cel economic. Aceste instrumente înglobate în sisteme interactive de asistare a deciziei permit trecerea de la adoptarea de hotărîri unilaterale la negocieri între puncte de vedere diferite, cu scopul găsirii unor soluții unanim satisfăcătoare.

Trebuie remarcat că articolul surprinde foarte clar și coerent aspecte socio-economice de importanță vitală în dezvoltarea unei țări, propunînd tehnici de abordare susținute cu exemple din statisticile poloneze. Alături de experiența bogată existentă la noi în problematica socio-economică din cadrul Sistemului Informatic Național și al Sistemului Informatic Teritorial, articolul aduce contribuții personale în modelarea sistemului social, idei și rezultate care merită să fie valorificate în vederea determinării unor politici optime la nivel național.

Modelele cu specific economic sînt prezentate în cadrul acestei sesiuni în (11.5/A1), unde este dezvoltat un model dinamic intrare-ieșire de tip Leontief, caracterizat prin ecuații diferențiale prospective, precum și algoritmul de rezolvare. Tehnica prezentată se constituie într-o metodă generală de planificare optimă a sistemelor dinamice mari cu retardări. Prin alegerea unui criteriu de performanță pătratic, se obține o problemă de conducere optimă, iar printr-o tehnică specifică de descompunere se ajunge la un șir de probleme de mici dimensiuni, a căror soluție rezultă iterativ. Problema economică este clasică, și după un număr de reformulări se prezintă astfel:

$$\text{Min } J = \sum_{k=0}^{K-1} \left\{ \frac{1}{2} \| X(k) \|_{Q(k)}^2 + \frac{1}{2} \| Y(k) - Y_0(k) \|_{R(k)}^2 \right\} \quad (1)$$

cu restricțiile

$$G_k X(k) = \sum_{j=1}^T \{ H_{j,k} X(k+j) + Y(k) \} \quad k=0, 1, \dots, K-1 \quad (2)$$

$$X_{\min}(k) \leq X(k) \leq X_{\max}(k) \quad k=0, 1, \dots, K-1 \quad (3)$$

$$Y_{\min}(k) \leq Y(k) \leq Y_{\max}(k) \quad k=0, 1, \dots, K-1 \quad (4)$$

$$X(0) = X_0 \quad (5)$$

unde:

- $k=0$ este anul inițial al planului;
- $k=K-1$ — anul final al planului;
- $R(k), Q(k)$ — matrice diagonale de ponderare, pozitiv definite;
- $X(k)$ — ieșirile;
- $Y(k)$ — intrările;
- $Y_0(k)$ — intrări precalculate;
- $G_k, H_{j,k}$ — matrici care conțin coeficienți de consum și investiții;
- K — orizontul dinamicii;
- T — întîrzierea maximă.

Algoritmul propus pentru rezolvarea problemei (1)–(5) apelează la funcția Lagrange, care printr-o rescriere convenabilă poate fi descompusă în K subprobleme independente. Deci determinarea soluțiilor X , Y ale problemei (1)–(5) se reduce la aflarea punctelor și ale lagrangeanului

$$L = \sum_{k=1}^{K-1} \left(\frac{1}{2} \|X(k)\|_{Q(k)}^2 + \frac{1}{2} \|Y(k) - Y_0(k)\|_{R(k)}^2 + p^T(k) \left[\sum_{j=1}^T H_{j,k} X(k+j) + Y(k) - G_k X(k) \right] \right) \quad (6)$$

unde $p(k)$ sînt multiplicatorii lui Lagrange.

Aflarea punctelor și ale funcției L revine la rezolvarea problemei

$$\max_p \min_{x, y} L(x, y, p) \quad (7)$$

Zhao propune o metodă iterativă în doi pași asemănătoare celei construite de Propoi și Yadjkin în [2]. Este de semnalat totuși că în algoritmul prezentat în [11.5/A1] nu sînt specificate condiții de convergență și finitudinea, care din punct de vedere practic au o importanță deosebită.

În [11.5/A5], Burkov propune o serie de rezultate aplicabile în îmbunătățirea controlului sectorial, bazate pe teoria sistemelor active. Sînt dezvoltate cîteva proprietăți specifice sistemelor active: optimalitatea principiului controlului perfect, condiții asupra mecanismelor funcționale, principiul reglării prin două canale, estimarea performanței de înțelegere, proprietăți ce sînt completate cu studii de caz pentru exemplificare. Ideile prezentate în acest articol sînt proprii școlii sovietice, și au fost demarate la Institutul de Știința Conducerii din Moscova încă în urmă cu 10 ani [3]. Este de semnalat faptul că a fost pus la punct un sistem de asistare a evaluărilor cantitative de înțelegere (comprehensive) a performanței — ACCORD, ce poate fi utilizat atît în fabrici, cît și în institute de cercetare.

O primă remarcă asupra lucrărilor prezente în secțiunea 11.5/A este aceea că articolele, plecînd de la considerații teoretice, converg spre aplicații practice ce nu se restrîng la un anumit domeniu, ci vizează întregul sistem socio-economic. În al doilea rînd se observă o încercare de exploatare a noilor tehnici informatice utilizabile în procesul decizional — sistemele suport pentru decizie, punîndu-se accent pe subsistemul modele și pe interactivitatea om-mașină. Nivelul științific ridicat al comunicărilor, utilizarea adecvată a exemplelor și a figurilor lămuritoare, furnizează tot atîtea argumente pentru lecturarea cu interes a acestora.

Modele pentru conducerea sistemelor mari

Cea de a doua secțiune a colocviului dedicat aplicațiilor tehnicilor specifice sistemelor mari în conducere și decizie este consacrată, ca și prima secțiune, sistemelor decizionale ierarhice, și cuprinde trei lucrări. Faptul că proceselor decizionale li s-au alocat mai multe secțiuni probează importanța de care se bucură acest subiect în domeniul aplicațiilor, atît din partea cercetătorilor, cît și din partea beneficiarilor potențiali. Sistemele socio-economice, sisteme de mari dimensiuni, complexe, cu dinamici dificil de formalizat și structură variabilă, sînt caracterizate de legi de guvernare ce nu sînt decît în mică parte descifrate. Decarece aceste sisteme constituie o componentă vitală a societății în care trăim, necesitatea abordării lor atît din punct de vedere cognitiv (descriptiv), cît și din punct de vedere decizional (conducere) apare ca o condiție sine qua non. În această secțiune sînt prezentate trei comunicări ce abordează probleme de modelare, analiză și conducere a sistemelor economice mari, interconectate [11.5/B1], o tratare metodologică a relației produs-mediu, ca o cerință a strategiei de cercetare și dezvoltare [11.5/B2], precum și o încercare de modelare a comerțului internațional și a încălzirii vapoarelor, cu aplicații la industria navală [11.5/B3]. Se evidențiază diversitatea problemelor dezbătute în aceste lucrări, numitorul lor comun fiind destinația socio-economică a metodelor și a rezoluțiilor propuse.

În opoziție cu „teoria modernă a reglării“, care vizează analiza și proiectarea sistemelor centralizate, cadrul nefavorabil nici din punct de vedere computațional și nici ca implementare, teoria sistemelor mari se preocupă de interacțiunile dintre subsisteme interconectate. În acest context, un sistem mare este descompus ierarhic (avîndu-se în vedere structura sa fizică sau matematică), ca o entitate multinivel ale cărei interacțiuni sînt caracterizate prin legături între nivelele superioare și cele inferioare. În [11.5/B1], Petrovič propune o metodologie de modelare, analiză și conducere a sistemelor economice interconectate, utilizînd modele bilineare de mari dimensiuni. Sînt utilizate metode ale geometriei diferențiale pentru studiul analizei structurale a sistemelor interconectate. În final sînt dezvoltate o serie de modele specifice și este prezentată problema reglării optime la două nivele.

De obicei în modelele economice comenzile sînt reprezentate prin termeni lineari aditivi incluși în ecuațiile sistemelor diferențiale. În realitate însă, multe comenzi, cum ar fi rata taxei sau rata creșterii stocurilor financiare, acționează natural într-un mod multiplicativ. În consecință, dacă sînt îndeplinite ipoteze convenabile, acest fapt generează modele bilineare. Mai mult, analiza și reglarea sistemului economic de mari dimensiuni pot fi realizate cu ajutorul modelelor bilineare, dacă interconexiunea dintre subsistemele microeconomice este modelată ca o variabilă de comandă lineară ce primește valori într-o mulțime compactă și este coordonată dinspre nivelul superior către cel inferior. Datorită faptului că bilinearitatea se prezervă prin dezagregarea sistemelor interconectate, se pot obține modele particulare bilineare de orice ordin și nivel de detaliere.

Vom prezenta în continuare modelul bilinear conform [11.5/B1]. Să presupunem că dinamicele subsistemelor sînt reprezentate de ecuații diferențiale lineare :

$$\dot{x}_i(t) = A_i x_i(t) + B_i^1 u_i^1(t) \quad i=1, \dots, N \quad (8)$$

$$u_i^1(t) = -K_i^1 x_i(t) \quad i=1, \dots, N \quad (9)$$

unde:

$x_i(t) \in \mathbb{R}^{n_i}$ — vectorul de stare pentru al i -lea subsistem S_i ;

$u_i^1(t) \in U$ — comenzile de feedback local.

Structura interconexiunilor este determinată de matricele A_{ij} și vectorii $v_{ij}(t)$, ($i, j=1, \dots, N$)
Fie $u^0(t)$ comanda dinspre nivelul coordonator către nivelul inferior. În acest context modelul se poate scrie :

$$\dot{x}(t) = Ax(t) + \sum_{\substack{i, j=1 \\ i \neq j}}^N v_{ij}(t) A_{ij} x(t) + Bu^0(t) \quad (10)$$

unde

$$A = \text{diag} (A_1 - B_1^1 K_1^1, \dots, A_N - B_N^1 K_N^1);$$

și $x \in \mathbb{R}^N$, $N_0 = n_1 + n_2 + \dots + n_N$ — vectorul de stare al întregului sistem.

Modelul de mai sus este bilinear, în sensul că este linear în stări și linear în comenzile $u^0(t)$, $v_{ij}(t)$, dar nu și împreună. Comenzile aditive $u^0(t)$ și cele multiplicative $v_{ij}(t)$ reprezintă coordonarea globală respectiv interacțiunile dintre subsisteme. Petrovič descrie un model de creștere multisectorial ierarhizat și un model economic interconectat, făcînd observații asupra posibilităților de aplicare efectivă a acestor modele.

Ideea de bază în proiectarea reglării este asigurarea stabilității asimptotice și a optimalității sistemelor cu ajutorul metodei dublu nivel, în care nivelul superior intervine asupra structurii procesului prin parametrii de interconexiune cu scopul de a permite stabilizarea și compensarea perturbațiilor prin feedback la nivelul inferior. În [11.5/B1] se face și o analiză structurală a sistemului interconectat utilizînd algebra Lie și metode ale geometriei diferențiale. Sînt descrise mulțimile accesibile și cele structural accesibile pentru sisteme ce pleacă dintr-o stare x la momentul t , Petrovič furnizînd o teoremă care caracterizează mulțimea structural accesibilă a sistemelor bilineare.

În literatură o atenție deosebită a fost acordată conducerii descentralizate a sistemelor mari, în special a celor lineare. În anumite condiții, pentru funcțiile bilineare, este dată forma comenzilor optimale atât pentru variabilele de coordonare, cât și pentru interacțiuni. Rezultă de aici că teoria sistemelor bilineare poate fi utilizată pentru a furniza o metodologie de abordare a sistemelor socio-economice. Chiar dacă aceste sisteme sînt puternic nelineare, incluzînd comenzi multiplicative și de tip histerezis, precum și întîrzieri, procesele fundamentale pot fi analizate cu ajutorul modelelor ierarhice binivel, în care comenzile aditive coordonează subsistemele dinspre nivelul superior, iar cele parametric (multiplicative) reprezintă interacțiuni între subsistemele nivelului inferior. Desigur, se poate ridica obiecția că, în timp ce Liu [11.5/A] și Kulikowski în [11.5/A4] argumentează și justifică din punct de vedere pragmatic necesitatea utilizării modelelor de mici dimensiuni, ușor de construit și de aplicat, folosirea modelelor mari și ierarhizate propuse de Petrovič apare greoaie și dificil de implementat. De asemenea, tehnicile de agregare-dezagregare trebuie să verifice condiții destul de restrictive pentru a fi consistente [4]. Dar utilizarea sistemelor bilineare pare interesantă și promițătoare din cel puțin două puncte de vedere. În primul rînd acestea descriu sisteme aproape neliniare, deci mai aproape de realitate decît sistemele lineare, și în al doilea rînd este de semnalat structura lor variabilă, nerigidă, fapt ce permite o abordare a modelării economice destul de realistă.

O interesantă pledoarie pentru studierea relației produs-mediu de utilizare, cu aplicații în domeniul cercetării și dezvoltării este făcută de Gerencsér în [11.5/B2]. Ideea de bază este că integrarea unui sistem complex în mediul utilizatorului presupune modificări retrospective în sistem și/sau în mediu. Noutatea metodologiei propuse constă în participarea activă a mediului la rezolvarea acestor cerințe. Fiind definită problema de rezolvat, mediul care primește un sistem complex trebuie ales și descris conform acestei probleme. Sistemul extins, care include și funcții ale mediului, trebuie analizat funcțional și divizat în subfuncții. Acestea trebuie clasificate conform considerațiilor tehnice-economice și a posibilităților de modularizare: configurația de bază a sistemului de implementat, opțiunile sistemului și cerințele de dezvoltare a mediului.

În timpul proiectării unui produs este necesar să fie luate în considerare aspectele tehnice precum și cerințele de tip marketing. Problema alocării resurselor în proiectele de cercetare-dezvoltare a fost abordată de Winkofski în [5], dar modelul binar dezvoltat de acesta nu lua în considerare relația produs finit-mediu de utilizare. Mediile caracteristice diferitelor nivele de utilizare a produselor necesită realizarea unor funcții specifice. Calea tradițională de rezolvare este modularizarea produsului. Problema poate fi abordată cu succes numai atunci cînd atît cerințele tehnice (funcții realizate de fiecare modul), cît și cele de tip marketing (selecția modulelor ce vor fi incluse în produs și a modulelor opționale ce sînt create de mediu sau achiziționate separat) sînt satisfăcute simultan.

O primă remarcă ce se desprinde din [11.5/B2] este aceea că, pentru realizarea aceluiași funcții, coeficienții de inteligență ai mediului (EIQ) și produsului (SIQ) variază invers proporțional. Sînt definite trei categorii de relații între un sistem și mediul în care el funcționează.

1) Sistemul corespunde unui singur tip de mediu.

2) Sistemul se potrivește cu mai multe tipuri de mediu. În acest context se pot dezvolta trei strategii:

a) SIQ minim și, deci, EIQ maxim, ceea ce presupune investiții mari la nivelul infrastructurii;

b) SIQ maxim și EIQ minim;

c) un compromis, bazat pe considerente economice, între SIQ și EIQ.

În domeniul informaticii, aceasta este situația cel mai des întîlnită, în sensul că se are în vedere atît o creștere a nivelului de inteligență a mediului în care se introduc sistemele de calcul, cît și o achiziționare de tehnică de calcul modularizată, cu o configurație care să corespundă cerințelor mediului.

3) Sistemul este creat ca un produs de piață, deci mediul este necunoscut și doar estimat.

Gerencsér descrie fazele elaborării strategiei de cercetare, dezvoltare și producere a unor sisteme de tipul celor din categoria a 3-a. Sînt discutate aceste faze în contextul creării și producerii unui obiect nou. În final este considerat un exemplu structurat pe metodologia propusă. Chiar dacă articolul are un caracter mai mult teoretic, vizînd punctarea unor principii specifice strategiei de cercetare și dezvoltare în orice domeniu, utilitatea sa constă în faptul că abordează probleme ce sînt destul de rar analizate, dar care au o importanță aparte în procesul decizional asociat creării și implementării noului.

Tot în cadrul aplicațiilor specifice sistemelor mari, cel de-al treilea articol din cadrul acestei sesiuni se referă la problematica comerțului naval din punctul de vedere al proprietăților și al constructorilor de vapoare. Industria navală a cunoscut în ultimii ani o serie de dificultăți datorate în primul rând ratelor mici de navlosire (încărcarea vaselor), scăderii valorii navelor și, deci, a creditelor acordate, costurilor tehnologice mari etc. Pau furnizează în [11.5/B3] un model al tendinței ratei de navlosire utilizabil în industria constructoare de vapoare și în comerțul marin. Indicii tendinței ratei de navlosire sînt determinați printr-un proces de „tatonare” bazat pe un model de dezechilibru cu o reeașare eficientă a capacităților de transport disponibile. Scopul acestui model este de a servi la fundamentarea deciziilor pe termen lung în ceea ce privește structura și mărirea flotei, vânzările, cumpărările și reparațiile.

Sistemul modelat gestionează o singură linie de vapoare ce operează asupra unei flote mixte formată din tancuri petroliere, vase cu încărcătură vrac („bulk carriers”) și cu încărcătură uscată (dry cargo). Ținîndu-se cont de tonajul vaselor, rutele pe care acestea pot fi alocate, cererea potențială de transport, capitalul disponibil și costurile specifice, Pau structurează un model al ratei de navlosire. Determinarea unei rate de navlosire satisfăcătoare se face pe baza unui algoritm de încercări succesive, fundamentat pe reguli specifice industriei navale și căruiua i se pot adăuga eventual condiții de optimizare. Este de remarcat faptul că modelul poate fi completat cu restricțiile financiare ale companiei, putînd fi corelat cu modelele de la nivel național. În urma coroborării cu o analiză financiar-economică și cu previziuni asupra comerțului mondial, modelul poate satisface trei tipuri de necesități:

- fundamentarea deciziilor asupra investițiilor pe termen lung (structura flotei, vânzări etc.);
- evaluarea structurii flotilelor și a rutelor concurenților;
- estimarea momentelor critice ale deciziilor strategice majore (finanțe, flotă, împărțirea piețelor).

Acest model este aplicat de către Battelle Memorial Institute, iar noutatea și ineditul său justifică lipsa bibliografiei.

În contextul în care congresul și-a propus să propulzeze diferite aplicații ale automaticii (A Bridge Between Control Science And Technology), lucrările cuprinse în secțiunea 11.5 B se subsumează acestui deciderat, iar interesul pentru subiectele tratate este cu atât mai mare cu cît domeniile abordate corespund unor cerințe reale, concret: furnizate de practica decizională

Sisteme suport pentru decizie

Presiunea continuă pentru îmbunătățirea eficienței proceselor decizionale i-a condus pe decidenți către tehnologiile informatice de vîrf. În general deciziile adoptate la un anumit moment sînt supuse la două tipuri de intrări. Pe de o parte judecata umană, concretizată prin experiență, pricepere, bun simț, și pe de altă parte ceea ce numim informație, adică un set de date la care trebuie să apelăm pentru a lua o decizie. A ne baza doar pe judecata noastră este evident un mod depășit de luare a deciziilor. A lua decizii bazîndu-ne numai pe informații, ceea ce revine la a apela la sisteme expert care hotărăsc independent de voința umană, este posibil, cel puțin la nivelul actual al informaticii în lume, numai pe domenii foarte restrictive din realitate. Apare ca un corolar al celor afirmate mai sus faptul că este necesară existența unei legături a celor două tipuri de intrări. Experiența a arătat că o clasă largă de probleme existente în conducerea diverselor sisteme nu se pretează unui suport informatic tradițional; această arie de probleme, implicînd luarea deciziilor la orice nivel, va putea beneficia de o recentă inovație în aplicarea ultimelor dezvoltări ale informaticii, și anume Sistemele Suport pentru Decizie (SSD).

Secțiunea a treia a colocviului dedicat aplicațiilor tehnicilor specifice sistemelor mari în procesul decizional și de conducere este consacrată sistemelor suport pentru decizie. SSD sînt sisteme de procesare a informațiilor care asistă, adesea interactiv, procesul decizional, constituînd sinapsa dintre informații, organizate sub forma unor baze de date, și judecata decidentului. Accentuăm că diferența dintre SSD și sistemele informatice operaționale constă tocmai în legătura explicită dintre SSD și procesul de decizie. Rolul acestor sisteme nu este de a înlocui decidentul ci de a-i spori eficacitatea, asistarea acestuia avînd o dublă motivație:

- SSD facilitează interacțiunea dintre date, proceduri de analiză a datelor, modele de decizie și utilizatorii acestora, recurgînd la tehnologii relevante de comunicare, limbaje cît mai apropiate utilizatorului pe domeniul de aplicație, grafică interactivă etc.;

— SSD asistă și decizii nestructurate, pentru care este dificil de furnizat date și proceduri relevante.

Secțiunea 11.5/C cuprinde un număr de cinci lucrări. Textul celei de a șasea comunicări [11.5/C2] nu este prezent în volum, dar pentru cei interesați menționăm că autorii au dezvoltat același subiect în două lucrări recente, [6] și [7]. Articolele sesiunii de față, având fiecare dezvoltări pe aplicații diferite, converg către un numitor comun și anume demonstrarea faptului că, în conformitate cu experiența ultimilor 10 ani, cea mai viabilă soluție în exploatarea tehnologiei actuale a calculatoarelor, având scopul de a veni efectiv în întâmpinarea procesului decizional, îl constituie SSD. Singh prezintă în [11.5/C1] cercetări asupra SSD utilizate în procesul de marketing precum și în conducerea strategică a corporațiilor multinaționale. Berezovskiy se ocupă în [11.5/C3] de etapa de concepție în proiectarea sistemelor tehnice complexe, iar Waśniowski în [11.5/C4] dezvoltă un sistem expert pentru viitorologie, evaluarea impactului și planificarea strategică cu aplicații în domeniul cercetării viitorului din punct de vedere socio-economic, tehnologic etc. În [11.5/C5], Aven descrie un SSD utilizabil în transporturile marinei comerciale și implementat la Baltic Shipping Agency, iar Baba abordează în [11.5/C6] trăsăturile structurale și comportamentale a două tipuri de corporații, „homeostatice” respectiv „dinamice”, cu scopul de a perfecționa strategiile de introducere a noului în corporațiile bazate pe tehnologii.

Pentru a caracteriza mai bine sistemele de asistare a deciziei trebuie menționat faptul că ele reprezintă o fuziune între sistemele informatice, tehnicile de inteligență artificială, cercetările operaționale și știința conducerii, [8]. În realizarea unui SSD, punându-se accentul pe descrierea procesului decizional, baza sa teoretică va fi atât informatică, cât și de cunoaștere psihologică, acesta având ca menire să asiste intuiția, să includă atât valorile, cât și logica realității, să facă față ambiguității, lipsei de informații, multiplicității și multidimensionalității. În acest context se situează și punctul de vedere al lui Singh, care în [11.5/C1] afirmă că SSD constituie inima noii generații de sisteme de birotică. Într-o primă parte a lucrării, Singh dezvoltă un model al legăturii producător-vinzător cu amănuntul - cumpărător, cu scopul optimizării lanțului de marketing. Facem observația că modelul considerat este static, în sensul că se bazează pe următoarele ipoteze: toată producția realizată într-o perioadă se contractează cu vânzătorii, care la rândul lor vînd totul consumatorilor. Deci în acest lanț nu există nici un fel de stocuri. Pe acest model sînt prezentate studii de simulare, făcîndu-se analiza rezultatelor obținute. Programul de suport al deciziei include facilități de schimbare interactivă a parametrilor, ceea ce permite decidentului, beneficiind de asemenea și de facilități grafice, să aleagă varianta cea mai convenabilă. Programul este implementat în FORTRAN 77 pe un minicalculator Prime 750. În a doua jumătate a comunicării, Singh atacă problematica corporațiilor multinaționale. Acestea sînt privite ca niște sisteme ierarhizate, pe nivelul superior aflîndu-se conducerea corporației, iar la nivelul inferior managerii diferitelor filiale, care sînt supuși la diverse presiuni atât din partea conducerii corporației, cât și din partea țărilor în care operează, presiuni care în general se manifestă antagonist și care au o puternică dinamică în timp. Pentru astfel de probleme, cu o structură complexă, există în literatură o clasă de modele conceptuale lingvistice, care descriu verbal sistemul și pentru care au fost create SSD specifice, [9]. Singh încearcă însă o abordare cu ajutorul modelelor matematice dinamice de optimizare cu restricții lineare și funcții obiectiv pătrate. Pentru rezolvarea acestora se recurge la funcția lui Lagrange asociată problemei inițiale și la teorema de punct șa. În procesul de determinare a punctelor șa ale langrangeanului se observă posibilitatea spargerii problemelor de maximizare ce apar în operații min-max, într-un număr de subprobleme independente egal cu numărul filialelor corporației. Este prezentată în final o argumentare detaliată a modelului. Desigur, cele două modele descrise au o importanță deosebită prin ele însele, dar utilitatea în contextul decizional se amplifică o dată cu încorporarea lor în instrumentele de suport al deciziei. Să observăm că Singh propune doar un cadru pentru SSD, care, prin menirea sa, trebuie să conțină o paletă diversă de modele ce asistă utilizatorul în luarea deciziilor, și nu numai un singur tip de modele. Dar, în același timp, să menționăm că dacă SSD se împart în sisteme orientate spre date și sisteme orientate spre modele, prima categorie furnizînd facilități de regăsire, analiză și prezentare a datelor, în timp ce a doua, mai sofisticată, oferă în special modele de optimizare și simulare, Singh preconizează un SSD care să aparțină celei de a doua categorii.

Berezovskiy, abordînd în [11.5/C3] domeniul proiectării asistate de calculator, se ocupă de faza de concepție și alegere a unei alternative satisfăcătoare, în contextul în care sînt numeroase criterii ce trebuie îndeplinite, dintre care multe dificil de formulizat. Este dezvoltată o

metodă de determinare a preferințelor în vederea alegerii unei opțiuni „nedominate”, adică pentru care nu există altă opțiune superioară din toate punctele de vedere. Sînt prezentate rezultate teoretice care se concretizează în două teoreme ce evidențiază tehnici specifice de descompunere în analiza opțiunilor.

Am arătat anterior că unul din obiectivele primordiale ale SSD îl constituie aplicarea intensivă a tehnicilor de inteligență artificială. În acest context Waśniowski propune în [11.5/C4] un sistem expert bazat pe cunoștințe destinat cercetării viitorului, evaluării impactului și planificării strategice. Sistemul expert se compune din două module : baza de cunoștințe și aparatul inferențial. Baza de cunoștințe conține informațiile despre domeniul unei problematice particulare, iar aparatul inferențial rezolvă problemele formulate de utilizator folosind baza de cunoștințe și generează explicații asupra soluțiilor adoptate. Deoarece în viitorologie intervin numeroase evenimente, tendințe și multe alte fapte dificil de identificat și de corelat, a fost dezvoltat un sistem denumit „FUTURIST” cu ajutorul căruia se pot construi modele calitative de simulare. Cunoștințele pentru rezolvarea problemelor sînt reprezentate în termenii unor probabilități condiționate, adică de tipul : anumite evenimente vor avea loc dacă în prealabil alte evenimente au avut loc. În acest fel pot fi create scenarii utilizînd diferite modele sau experiența umană, într-un mod interactiv. Programul, scris în PROLOG, a fost implementat pe un minicalculator PDP-11 sub un sistem de operare de tipul RSX-11M. Este de menționat faptul că sistemul utilizează raționamente cu predicat fuzzy, răspunde la întrebări de tipul „Why ?” și are o comandă „HELP”, deci poate fi considerat ca fiind un sistem orientat către utilizator. Trebuie să menționăm că și la noi în țară există experiențe asemănătoare [10], [11], în sensul că au fost create cîteva tipuri de sisteme experte dintre care unele, prin posibilitatea de a răspunde și la întrebări de tipul „How to ?” și „What if ?”, au facilități superioare celui descris de Waśniowski.

Dacă în [11.5/B3] Pau a prezentat doar un model al ratelor de navlosire (încărcarea vaselor) cu aplicații la industria navală, Aven descrie în [11.5/C5] un instrument complex pentru luarea deciziilor în conducerea transportului marin comercial. Transportul marin este analizat în conexiune cu celelalte sectoare ale economiei naționale, avînd atît trăsături comune cu acestea, cît și caracteristici specifice. Faptul că procesul este continuu, are un grad mare de variabilitate iar factorii exogeni sînt imprevizibili, precum și natura sa probabilistă obligînd la actualizări continue impun cu necesitate utilizarea unui SSD. Sistemul este destinat ordonanțării traficului naval, a reparațiilor, a operațiilor în uzinele de reparații etc. și este gîndit ca o interconectare de mai multe subsisteme. Două dintre acestea, sistemul de ordonanțare continuă și sistemul informatic pentru containere navale sînt prezentate în detaliu. Sistemul de ordonanțare continuă este dedicat conducerii traficului naval pe o durată pînă la 120 de zile. Actualizările se fac pe baza unei rețele de videoterminal, datele fiind gestionate de un SGBD de tipul INES-2M. Etapele dialogului dintre om și sistem constau în identificarea utilizatorului, specificarea scenariului cadru și efectuarea de simulări cu diferite scenarii utilizînd programe specifice de aplicații. Sistemul de ordonanțare cuprinde 15 baze de date cu aceeași structură logică. Sistemul „Container” este structurat pe trei nivele : containere, agenția navală și minister. Sînt luate în considerare toate operațiile pornind de la transportul pe cale ferată, încărcarea, descărcarea, distribuirea sosirilor containerelor la locurile de încărcare etc. Să semnalăm faptul că sistemul include pachete de programe pentru rezolvarea problemelor de optimizare, alături de facilitățile amintite pentru simulare. Acest produs informatic, care este implementat la Baltic Shipping Agency, este — atît prin facilitățile pe care le pune la îndemîna utilizatorului, cît și prin complexitatea și dificultatea problematice pe care o asistă în procesul decizional — unul din cele mai reprezentative sisteme de suport al deciziei create pe plan mondial. La noi în țară există încercări de a dezvolta un astfel de sistem cu facilități deosebite [12], sistem care include în plus și tehnici de inteligență artificială precum și elemente de știința conducerii.

Baba, în [11.5/C6] prezintă relațiile care există între două categorii de corporații : cele „homeostatice”, care caută să-și mențină structura invariabilă prin controlul ce-l impun asupra pieței, și cele „dinamice”, care încearcă să exploateze și să constituie avangarda eferescentei tehnologice. Schimbările tehnologice sînt privite ca forțe potențiale generatoare de perturbații în mediul corporațiilor și în acest context sînt avansate ipoteze asupra avantajelor corporațiilor tinere față de corporațiile mature. Analogia cu evoluția umană este folosită pentru a explica adaptările

corporațiilor la tehnologiile noi, fapt ce presupune riscuri mari. Acest articol, cu toate că vizează luarea deciziilor cu scopul îmbunătățirii structurii corporațiilor, ca și a politicilor și a strategiilor acestora, nu își propune să abordeze instrumentele practice necesare desfășurării procesului decizional, el încercând doar, pe baza unei documentații serioase, să prezinte o serie de principii și observații.

Dintre lucrările congresului, mai pot fi amintite și alte comunicări ce au avut ca obiectiv prezentarea unor sisteme de suport a deciziei. Astfel poate fi menționată comunicarea lui Amano [11.6/A4], în care este propus un sistem de planificare asistat de calculator, utilizabil în rețeaua de transport public urban, cu ajutorul căruia se poate predicționa fluxul traficului în rețea. Sistemul este utilizat în evaluarea impactului produs într-un mediu urban de construirea unui metrou. În [11.9/2] Sheridan pledează în favoarea SSD, subliniind rolul cercetării de laborator în domeniul sistemelor de reglare de tip om-mașină. În domeniul exploataării apelor sint de semnalat o serie de sisteme ce sint utilizate pentru tratarea apelor reziduale, Maeda în [11.7/5], sau pentru evaluarea lingvistică a resurselor de apă, Bečvăr în [11.8/A3], sistem ce este bazat pe teoria mulțimilor fuzzy. În sfârșit, Carmon propune în [11.8/C2] un sistem interactiv care realizează modele pentru diferite proiecte, un model de autoînvățare al sistemului, ce utilizează informația din timpul procesului decizional, un model de examinare a diferitelor politici, un set de proceduri în ajutorul utilizatorului (regăsirea informației relevante din proiecte, atenționări, actualizări, etc.), un algoritm de dialogare cu decidentul.

Chiar dacă primele SSD au fost construite cu numai 10 ani în urmă, succesul acestora a fost cel scontat, fapt demonstrat de numărul din ce în ce mai mare de astfel de sisteme utilizate astăzi. Să precizăm că SSD sint folosite în procesul decizional al unor domenii foarte diverse și asistă o paletă largă de decizii. Cele mai numeroase sisteme sint exploatate în probleme de planificare la nivel de corporații, sau la nivel guvernamental, precum și în domeniul militar, în medicină, în conducerea băncilor etc. La noi în țară există realizări în această direcție de vîrf a informaticii, dintre care menționăm sistemul „DISPECER“, utilizat cu succes în conducerea proceselor tehnologice [13], precum și sistemul „CAMDS“, un SSD bazat pe modele, dedicat în special conducerii sistemelor socio-economice [12].

BIBLIOGRAFIE

1. Simon, H.A. *The Sciences of the Artificial*, MIT Press, Cambridge, Massachusetts, 1969.
2. Propoi, A.I. Yadjkin, A.B. *Metodă iterativă parametrică pentru soluționarea problemelor de programare liniar-dinamice* (lb. rusă), *Avtomatika i Telemekhanika*, Nr. 2, 1976.
3. Burkov, V.N. Korotuev, V.V. *Quasioptimality of openhanded control in resource allocation*, Preprints "IFAC 6-th Congress", Pittsburg, 1975.
4. Sondermann, D. *Optimale Aggregation von grossen Gleichungen Systemen*, *Zeitschrift für Nationalökonomie*, 33, 1973.
5. Winkofski, E.P. Baker, N.R. Sweeney, D.J. *A Decision Process Model of R & D Resource Allocation in Hierarchical Organizations*, *Management Science*, Vol. 27, No. 3, 1981.
6. While, C.C. Sage, A.P., Scherer, W.T. *Decision Support with Partially Identified Parameters*, *Large Scale Systems*, Vol. 3, 1982.
7. Iagomasino, A., Sage, A.P. *A Learning Approach for Incorporation of Imperfect Knowledge in Decision Support. System Design*, First European Workshop on the "Real Time Control of Large Scale Systems", Patras, July 1984.
8. Gheorghiu, O. *Sistemele Suport Pentru Decizie - Instrumente Sofisticate În Ajutorul Procesului Decizional*, Sesiunea Națională de Comunicări Științifice, ICI-București, 1984.
9. Negoită C.V. *Fuzzy Sets in Decision Support Systems*, Working Paper, 1982.
10. Bărbuceanu, M. *An Object-centered Framework for Expert Systems in CAD*, *Proceedings of the IFIP WG. 5.2 Budapest, 1984*.
11. Georgescu, I., Holăran, A., Predescu, R., Vlasiu, M., Petrescu, F., Nicolăișă D., Nagy, A. *INTEXP - A Domain Independent Expert System*, Raport de cercetare, ICI, 1984.
12. Gheorghiu, O. *CAMDS - A Model Based DSS*, Raport de cercetare, ICI, 1984.
13. Filip, F.G., Donciuțescu, D.A., Orășanu, L. *Industrial Applications of Hierarchical Optimization Methods*, in A. Straszak (ed), *Preprints, 3-rd IFAC/IFORS Symp. Large Scale Systems Theory and Applications*, Warsaw, 1983.

METODE DE DECIZIE MULTICRITERIALE

Portofoliul de metode multicriteriale este din ce în ce mai vast, acest fapt fiind determinat de cerințele practice din ce în ce mai complexe ale modelelor reale.

Metodele multicriteriale se clasifică în: metode de decizie multiobiectiv (MODM) și metode de decizie multiatribut (MADM). În metodele de decizie multiobiectiv, obiectivele sînt explicite, iar alternativele sînt definite implicit prin restricții, și pot fi oricît de multe. În metodele de decizie multiatribut, obiectivele sînt implicite și greșit definite, alternativele sînt explicite și puține la număr.

Dintre metodele MODM, cele mai performante și mai des utilizate sînt metodele interactive. Aceste metode implică intervenția decidentului prin dialogul lui cu mașina/analistul la fiecare iterație. Aceasta conferă algoritmilor o mai mare flexibilitate și o mai corectă oglindire a procesului de decizie.

Din punct de vedere matematic, putem asimila decidentul cu un operator fuzzy, ceea ce, în anumite condiții, fie de situație "limită", „critică", fie din motive algoritmice, poate determina nonconvergența metodei. Pentru a evita aceasta, cît și alte inconveniente rezultate din aplicarea metodelor interactive, se prevede fixarea de către analist, prin programul software, a unui "domeniu de desfășurare", în cadrul căruia procesul este convergent și rezonabil.

Prezentăm cele cinci articole din secțiunea 11.5/D "Metode de decizie multicriteriale" prezentate la simpozionul IFAC 1984. Se remarcă o preocupare constantă în elaborarea unor metode interactive (cu un decident și/sau cu mai mulți decidenți) care să prezinte caracteristici de performanță superioare altor metode. De asemenea, în [11.5/D4 și 5] se face o analiză, vizînd globalul, a problemelor incertitudinii și a metodelor de decizie multicriteriale cu mai mulți decidenți.

În acest paragraf, încercăm să prezentăm pe scurt și să facem o analiză comparativă a metodelor prezentate de Nakayama și Sawaragi, Gomide, Tarvainen și Haimes, Stanoulov, Aleskerov și Vol'skiy, Takeguchi și Akashi la simpozionul IFAC 1984.

Astfel, Nakayama și Sawaragi prezintă în [11.5/D1] o metodă de decizie multiobiectiv iterativă, de tipul metodelor din familia STEM [1]. Metoda prezentată în [11.5/D1] este mai flexibilă și mai rapid convergentă decît metodele similare. Aceasta se realizează prin faptul că decidentul are posibilitatea să crească și/sau să relaxeze un număr oarecare de obiective, procedeul putîndu-se repeta, în funcție de un anumit test, repetarea lui micșorînd numărul de optimizări auxiliare și asigurînd o convergență mai rapidă. A se nota că metoda STEM [1] nu permite decidentului decît relaxarea unui singur obiectiv în cadrul unei iterații.

Față de metoda STEM, metoda "de satisfacere a compensărilor" a lui Nakayama și Sawaragi prezintă următoarele facilități:

- a) mulțimea restricției a problemei inițiale nu este modificată pe parcursul algoritmului;
- b) informațiile cerute decidentului nu necesită un efort prea mare din partea lui, asigurînduse totodată un proces de învățare pentru utilizator;
- c) în anumite condiții se poate obține o soluție mai bună decît în metoda STEM, și anume o soluție Pareto satisfăcătoare;
- d) decidentul poate să relaxeze și/sau să crească anumite obiective, după cum cere funcția sa de utilitate implicită, în timp ce metoda STEM permite decidentului doar relaxarea unui singur obiectiv;
- e) repetarea eventuală a procedurii de la punctul (d) asigură un număr mai mic de optimizări auxiliare, deși o convergență mai rapidă decît metoda STEM.

Față de alte metode din familia STEM, cum ar fi SEMOPT [6] care cere rezolvarea a $n \geq 2$ optimizări auxiliare la fiecare iterație, superioritatea metodei "de satisfacere a compensărilor" este evidentă.

În concluzie, metoda iterativă [11.5/D1] prezintă următoarele caracteristici:

- efort minimal al decidentului;
- informația furnizată decidentului este ușor de înțeles;
- convergență rapidă;
- număr mic de optimizări auxiliare;
- un proces de învățare pentru decident.

După cum este arătat în material, această metodă stă la baza unor programe de calcul și a fost aplicată în proiectarea construcției antiseismice a sistemului de piloni pentru un pod lung suspendat.

K. Tarvainen își continuă în [11.5/D2] împreună cu F. Gomide și Y.Y. Halmes, activitatea de cercetare relativ la problemele multicriteriale. De remarcat, dintre lucrările lui, teoria calității pentru problemele convexe multiobiectiv [9].

În [11.5/D2] se prezintă o metodă de negociere iterativă relativ la sistemele mari cu mai multe obiective. Această metodă face parte din clasa MODM cu $n \geq 2$ decidenți.

Autorii clasifică metodele de negociere multicriteriale astfel :

a) metode cu bază matematică (impun satisfacerea anumitor axiome, indiferent dacă sînt sau nu convenabile pentru decidenți) ;

b) metoda de generare (se generează toate sau aproape toate soluțiile Pareto, dintre care decidenții aleg, prin vot, de exemplu, cele mai bune soluții) ;

c) metode de decizie spațială (corespund metodelor cu preferințe a priori relativ la informații și un singur decident. Au fost concepute inițial de Robertson, 1976) ;

d) metode de negociere iterativă.

Metoda prezentată la [11.5/D2] prezintă similitudini cu metoda SEMOPT [6] — un singur decident, și anume :

1) la fiecare iterație se rezolvă un număr de $n \geq 2$ probleme auxiliare (dacă N este nu numărul de decidenți se rezolvă fie $N+1$, fie N optimizări auxiliare, în funcție de varianta folosită).

2) procedeul este convergent doar în cazul satisfacerii anumitor ipoteze.

De remarcat că această metodă este mai flexibilă, respectă opțiunile decidenților și reflectă mai corect procesul real decît metodele din clasele a), b), c), ceea ce îi conferă criteriul de performanță mai ridicat.

De asemenea, prin structura procedurii, poate fi ușor adaptată pentru cazul unui singur decident, ceea ce determină, din punct de vedere al argumentelor matematice, superioritatea acestei metode față de SEMOPT (numărul de optimizări auxiliare pentru fiecare iterație este mai mic). De altfel, autorii dau un exemplu numeric (rîul ipotetic Bow din [5]), care confirmă convergența mai rapidă decît cea a metodei SEMOPT.

Metodele MADM (metode de decizie multiatribut) sînt abordate în [11.5/D3] și [11.5/D4], ultimul avînd o viziune globală și de sinteză asupra lor.

În [11.5/D4], F.T. Aleskerov și V.I. Vol'skiy descriu regulile Pareto ca fiind clase de proceduri "de agregare" a criteriilor într-un singur criteriu "de selecție" rezultant.

Autorii fac o clasificare a acestor proceduri, astfel :

1) *Proceduri de tip I* — care transformă o totalitate de relații binare C_1 într-o relație "de selecție" C^* rezultantă.

Se definesc operatorii locali care transformă o colecție de relații binare slabe într-o relație binară aciclică/transitivă/de tranzitivitate negativă.

Se definește legătura existentă între acest tip de proceduri și metodele care folosesc matrici similare cu cele din teoria jocurilor.

2) *Proceduri de tip II* — transformă o totalitate de relații binare/criterii într-o funcție de selecție rezultantă $C^*(\cdot)$.

Autorii leagă rezultatele obținute în privința acestor proceduri cu cele obținute la punctul (1), făcînd referințe la metodele care selectează o submatrice "dihotomică" din matricea totală a alternativelor (vezi metoda prezentată în [11.5/D3]).

3) *Proceduri de tip III* — transformă o totalitate a funcțiilor de selecție $C_1(\cdot)$ într-o funcție de selecție globală $C^*(\cdot)$.

Acest tip de proceduri sînt foarte recente, primele rezultate fiind date de Grether și Ploft în 1982.

N. Stanoulov în [11.5/D3] descrie un procedeu de rezolvare a problemelor MADM pentru cazurile un singur decident și mai mulți decidenți, care aparține clasei procedurilor de tip II, descrise mai sus [11.5/D4].

Astfel, Stanoulov construiește un operator local care duce o mulțime de relații binare slabe într-o relație binară aciclică. Această observație plasează metoda în clasa procedurilor de tip I, dar faptul că din matricea alternativelor totale se selectează o submatrice (numită "dihotomică")

tomică") a alternativelor "preferabile", determină apartenența acestei proceduri la clasa celor de tip II, care include strict, într-un anume sens, pe cea a procedurilor de tip I.

Metoda DIMCO (multiple criteria assessment and choice of alternatives via matrix dichotomy [11.5/D3]) poate fi totodată asimilată cu un joc matriceal în care sînt determinate alternativele "cele mai bune", prin selectarea unui șir de "învingători" în cele trei faze ale algoritmului.

Față de alte metode similare (Toda, Amano, 1981) structura de preferință definită între alternative este diferită, ceea ce determină un grad de performanță mai ridicat.

Un studiu asupra incertitudinii deciziilor în condiții de risc este făcut în [11.5/D5], unde T. Takeguchi și H. Akashi presupun că există informații incomplete asupra probabilităților stărilor și, în funcție de tipul acestor informații, definesc dominanța statistică, stohastică sau S-S (statistică și stohastică) a alternativei A1 asupra alternativei A2.

Este, în fond, o tehnică auxiliară prognozei, vizînd o simulare a comportării sistemului real în condiții de incertitudine.

Autorii derivă din inegalitatea Fishburn două inegalități care definesc parțial probabilitățile stărilor:

$$(i) DH_h \geq P_h - P_{h+1} \geq DL_h, \quad h=1, \overline{m-1}$$

sau

$$(ii) RH_h \geq \frac{P_h}{P_{h+1}} \geq RL_h, \quad h=1, \overline{m-1}$$

unde P_h este probabilitatea stării h ;

DH_h, DL_h — margini superioare, respectiv inferioare, pentru diferența $(P_h - P_{h+1})$;

RH_h, RL_h — margini superioare: respectiv inferioare, pentru fracția $\left(\frac{P_h}{P_{h+1}}\right)$.

Se definește dominanța statistică a alternativei A1 asupra alternativei A2, ca fiind determinată de inegalitatea:

$$E[A1] \geq E[A2]$$

unde

$$E[A_i] = \sum_{j=1}^m P_j U_j^i, \quad i=1, 2$$

Autorii derivă din definiția de mai sus și din informațiile incomplete asupra probabilităților stărilor (i) sau (ii) rezultate echivalente pentru a exprima dominanța statistică.

Domanța stohastică se deosebește de cea anterioară prin faptul că, în locul probabilităților descrise mai sus, folosește distribuții de probabilitate ale consecințelor și introduce de asemenea clase de funcții de utilitate.

Funcțiile de utilitate pot fi clasificate după cum:

1) decidentul este împotriva riscului;

2) decidentul caută riscul;

3) decidentul dorește descreșterea riscului;

4) decidentul este împotriva riscului în "domeniul câștigului" și pentru risc în "domeniul pierderii".

Distribuția de probabilitate poate fi interpretată ca fiind "gradul de risc al alternativelor, din punct de vedere al consecințelor lor".

Spunem că distribuția de probabilitate F a alternativei A1 domină stohastic distribuția de probabilitate G a alternativei A2 dacă

$$E(u, F) \geq E(u, G) \quad \forall u \in U_j$$

unde

U_j = clasa j a funcțiilor de utilitate

și

$$E(u, F) = \int_{x \in X} u(x) dF(x)$$

$$E(u, G) = \int_{x \in X} u(x) dG(x)$$

Cu alte cuvinte, definiția de mai sus reflectă faptul că alternativa A1 este mai bună decât alternativa A2 din punctul de vedere al "gradului de risc al consecințelor".

Dominanța S-S (statistică și stohastică) poate fi exprimată în următorii termeni: În ce condiții alternativa A1 este mai bună decât alternativa A2 din punctul de vedere al "gradului de risc al consecințelor", când avem informații incomplete asupra probabilităților stărilor și funcția de utilitate implicită a decidentului aparține unei clase bine definite matematic?

Vizînd materialul [11.5/D3], putem aplica acest studiu metodei DIMCO, folosind această tehnică ca o prognoză și simulare a evoluției sistemului global în condiții de risc.

Rezultatele recente sintetizate aici sînt de două tipuri:

a) de analiză și sinteză a unor studii anterioare la care se adaugă contribuții originale [11.5/D4 și 5]

b) de prezentare a unor metode originale MODM sau MADM [11.5/D1, 2, 3] care extind, în sensul unor performanțe mai ridicate, metode similare deja existente.

Aceste studii devin utile cercetării în acest domeniu prin performanțele ridicate ale metodelor, care le fac mai ușor abordabile computațional și mai flexibile în utilizarea lor practică, sintezele, la rîndul lor, asigurînd totodată o deschidere mai largă a metodelor multicriteriale și o utilizare a lor vizînd globalul.

BIBLIOGRAFIE

1. Benayoun, R., de Montgolfier, J., Tergny, J., Larichev, O. — **Linear Programming with Multiplé Objective Functions : Step Method (STEM)**, Math. progr., V.1, No. 3, pp 366—375, 1971.
2. Blin, J.M. **A linear assignment formulation of the multiatribute decision problem**, RAIRO, Recherche opérationnelle, V. 10, No. 6, pp. 21—32, 1976.
3. Delalieux, Y. **Multiple criteria optimization with adaptive partition**, Preprints of the First European Workshop on the „Real Time Control of Large Scale Systems“ (Schmidt, C., Singh, M., Titli, A., Tzafestas, S., ed.), pp. 118—123, University of Patras, Greece, July 1984.
4. Hwang, C.L., Yoon, K. **Multiple Attribute Decision Making — Methods and Applications**, Lectures Notes in Economics and Mathematical Systems, Springer — Verlag, 1981.
5. Hwang, C.L., Masud, A.S.M. **Multiple Objective Decision Making — Methods and Applications**, Lectures Notes in Economics and Mathematical Systems, Springer — Verlag, 1979.
6. Monarchi, D.E. **Interactive Algorithm for Multiple Objective Decision Making**, Technical Reports on Hydrology and Water Resources (No. 6), University of Arizona, 1972.
7. Siskos, J. **Application de la methode UTA I à une problème de selection de points de vente mettant en jeu des critères multiples**, RAIRO, Recherche opérationnelle, 1983.
8. Steuer, R.E. **Goal Programming sensitivity analysis using interval penalty weights — Math. progr.**, 17, 1979.
9. Tarbainen, K. **A preference — oriented duality theory for multiojective optimization**, Report B70, System Theory Laboratory, University of Technology, Helsinki, 1982.
10. Vincke, Ph. **Une méthode interactive en programmation linéaire à plusieurs fonctions économique**, RAIRO, Recherche opérationnelle, V. 10, No. 6, pp. 5—20, 1976.
11. * * * **An International Journal Computers & Operations Research** (J. Raff, ed.), V. 7, No. 1—2, 1980 („Mathematical Programming with Multiple Objectives“).

OPTIMIZĂRI ÎN RAMURA TRANSPORTURILOR

Ing. M. Sirbu
I.P.A.

1. INTRODUCERE

Transportul are un rol vital pentru desfășurarea în bune condiții a vieții social-economice dintr-un oraș. În ultimii ani s-a observat că traficul a devenit din ce în ce mai aglomerat, acest lucru datorându-se în mare măsură creșterii numărului de mașini particulare. Studiile efectuate au pus în evidență faptul că mijloacele „convenționale” de transport (metrou, autobuze, tram-vale etc.) oferă încă posibilități mari de a fi îmbunătățite.

Lucrările prezentate în cadrul secțiunii 11.6 tratează câteva dintre problemele ridicate de aglomerarea circulației. Astfel, sînt propuse metode de descongestionare a traficului urban pe baza utilizării unor sisteme de planificare a rețelelor de transport public, pe baza utilizării unor sisteme de dirijare a circulației și a unor sisteme de comandă corelată a semafoarelor de circulație etc.

Unele lucrări propun automatizarea mijloacelor de transport (a mașinilor, a unor mijloace de transport naval), ca o modalitate de îmbunătățire pe ansamblu a traficului.

2. SISTEME PENTRU TRANSPORTUL TERESTRU

Lucrarea 11.6/A-1 prezintă un sistem ierarhic descentralizat pentru dirijarea circulației rutiere. Prima realizare fizică a sistemului propus, și care este prezentată pe scurt în lucrare, este sistemul „PROGETTO TORINO”.

Un astfel de sistem este necesar pentru satisfacerea următoarelor cerințe :

- a) prioritate absolută pentru liniile de transport public ;
- b) îmbunătățirea semnificativă a mobilității vehiculelor personale ;
- c) controlul continuu al vehiculelor publice aflate în acțiune ;
- d) informarea utilizatorilor care așteaptă în stații în legătură cu sosirea următorului vehicul public ;
- e) asigurarea unei fiabilități ridicate a întregului sistem, aceasta presupunînd autodiagnosticarea și informarea automată la postul central de dispecer în legătură cu defectele care pot apărea în orice punct al rețelei semafoarelor aflate sub control.

Pentru satisfacerea acestor cerințe se impun următoarele specificații de proiectare :

- exploatarea avantajelor oferite de coordonarea semafoarelor de circulație ;
- acordarea atenției cuvenite fenomenelor lente legate de traficul particular ;
- acordarea priorității unor linii de transport public pe străzile rezervate, prin intermediul pronosticării prin puncte a mișcării fiecărui vehicul ;
- libertatea controlului semafoarelor la fiecare intersecție prin intermediul unor operații locale în buclă închisă.

Cerințele de proiectare impuse mai sus conduc la realizarea unui sistem ierarhic distribuit. Modelul dinamic al traficului este descris prin intermediul a trei modele dinamice : modelul macroscopic al traficului privat, care descrie comportarea liniilor de trafic pe întreaga zonă aflată sub control ; modelul microscopic al traficului privat, care furnizează o reprezentare detaliată a traficului privat la fiecare intersecție și modelul traficului public, care descrie în detaliu comportarea fiecărui vehicul de pe liniile prioritare.

Strategia de conducere, aflată în strînsă conexiune cu medelele de mai sus, are ca scop minimizarea timpului total pierdut de vehiculele particulare, impunîndu-se în același timp restricția ca vehiculele publice de pe liniile cu prioritate să nu fie oprite la intersecții semaforizate. De asemenea, trebuie să se țină cont de restricțiile de siguranță a traficului și de durata maximă sau minimă a fazelor semafoarelor.

Deoarece apar perturbații puternice, restricțiile locale și globale fiind bazate pe variabile stohastice, va fi necesară o conducere în buclă închisă. Totodată, datorită complexității problemei, va fi necesară aplicarea unor metode de decompoziție.

Sistemul de conducere va fi ierarhizat pe două nivele. La nivelul superior se realizează conducerea globală, bazată pe informațiile furnizate de modelul global și se transmit la nivel inferior regulile care trebuie respectate pentru optimizarea performanțelor globale. Sarcinile la nivel inferior sînt realizate de fiecare dintre mecanismele de conducere bazate pe modelele locale corespunzătoare, utilizînd cu maximum de eficiență informația provenită de la nivelul superior.

Sistemul de conducere va avea următoarea structură :

— o rețea de mecanisme de conducere locale, conectate mutual, astfel încît să reproducă topologia conexiunilor dintre intersecții ;

— o rețea densă a detectoarelor pentru traficul privat și public, plasate astfel încît să furnizeze o estimare de stare ;

— un mecanism de conducere global, conectat la rețeaua mecanismelor de conducere locale.

Implementarea practică a acestui sistem de conducere cu două nivele ierarhice este reprezentată de sistemul „PROGETTO TORINO”. Elementele principale care intră în componența acestui sistem sînt : detectoare de vehicule particulare, detectoare de vehicule publice, elemente de acționare a semafoarelor, mecanismul de conducere locală și mecanismul de conducere globală.

Sistemul „PROGETTO TORINO” asigură verificarea în practică a conceptelor de proiectare descrise în lucrare. Instalarea acestui sistem a fost definitivată conform planului, în anul 1983. Primele rezultate obținute fiind bune, s-a trecut la aplicarea largă a sistemului în anul 1984.

Problema desconggestionării traficului urban este tratată și în lucrările 11.6/A-2, 11.6/A-3, 11.6/A-4, 11.6/A-5 și 11.6/A-7

Aspectele de o mai largă generalitate sînt abordate în cadrul lucrărilor 11.6/A-2 și 11.6/A-4, care se ocupă cu probleme de modelare și reglare a proceselor în sistemele de transport și respectiv cu descrierea unui sistem de planificare asistat de calculator pentru rețelele de transport public.

În lucrarea 11.6/A-2 a fost dezvoltată o versiune de bază a sistemului de modelare a transportului pe scară largă. Schema generală a procesului de transport este reprezentată cu ajutorul unui set de blocuri de conducere (BC), care definesc sistemul de conducere structurat și a setului de blocuri funcționale (BF), care definesc organizarea tehnologică de transport. Setul componentelor BF este divizat în subseturi neinteractive, în concordanță cu structura BC. Tranzacțiile dinamice (TD) asigură toate legăturile între BC și BF. Pe baza acestei scheme de modelare se realizează descrierea caracteristicilor dinamice.

A fost realizat un limbaj de simulare, care susține sistemul de modelare. Pentru reprezentarea caracteristicilor modelării simulate s-a utilizat o „metodă cadru”. Modelarea bazei de date a fost realizată pentru construirea și reprezentarea parametrilor de stare ai sistemului. Limbajul special de întrebare permite realizarea accesului la date și a unor manipulări cu datele modelării simulate.

Sistemul de modelare și mijloacele de programare prezentate au fost utilizate pentru predictarea transportului în condițiile în care interacționează diferite moduri de transport. Principalul scop al predicției transportului este de a face posibilă reglarea interacțiunii dintre diferitele moduri de transport. Un astfel de model de predicție a fost construit pentru sistemul de transport pe căi ferate. În acest caz, principalul scop al predicției este de a asigura o distribuție dinamică a fluxului de vehicule ajunse pentru perioade scurte în porturile marine.

În lucrare se tratează și problema modelării sistemului de conducere. Automatizarea planificării proceselor de conducere pentru sistemele de transport se face pe baza analizei informației de stare a sistemului, analiză care furnizează indicații în legătură cu strategiile de conducere, bazate pe model, care pot fi adoptate.

Ca o aplicație, s-a construit un model pentru sistemul de transport urban. Pentru aceasta s-au luat în considerare cîteva moduri de transport urban pentru pasageri, și anume : transportul pe căi ferate, căile ferate din metropolă, rețeaua de trafic și reglarea interacțiunilor. Sistemul liniilor de metrou a fost utilizat pentru optimizarea parametrilor traficului planificat.

Lucrarea 11.6/A-4 prezintă un sistem de planificare asistată de calculator pentru rețelele de transport public. Pentru început se face descrierea generală a sistemului, format dintr-un sistem de planificare a rețelei și dintr-un sistem asistat de calculator. Pentru planificare sînt necesare date socio-economice în legătură cu zona studiată, date privitoare la rețelele de transport și date în legătură cu cerințele de trafic. Pe baza acestor date se vor realiza mai multe

planuri, furnizându-se astfel mai multe alternative pentru planificare. Un astfel de plan este definit de configurația rețelei (rute, poziționarea stațiilor de autobuz etc.), de timpul necesar pentru parcurgerea fiecărei rute, de timpul de transfer, de frecvența serviciilor și de numărul de călători. După formularea planurilor alternative, urmează predictarea fluxului de trafic pe rețeaua de transport luată în considerare. Pentru aceasta se predictează comportamentul unui călător, iar estimarea volumului de trafic pe fiecare linie a rețelei de transport se va face prin compunerea predicțiilor comportamentelor individuale. Modelul de predicție utilizat în acest studiu poate fi foarte util pentru planificarea unei rețele unimodale. În continuare se face evaluarea planurilor alternative. Dacă este necesară modificarea acestor planuri sau adăugarea unor planuri noi, se va relua procesul descris mai sus pînă cînd nu mai există posibilitatea de a găsi planuri alternative mai bune.

Sistemul asistat de calculator este dotat cu facilități de gestiune a bazei de date, facilități de gestiune grafică, facilități de gestiune a programelor de aplicație și facilități de conducere interactivă. Sistemul asistat de calculator cuprinde un sistem gazdă și un sistem satelit. Sistemul satelit este dotat cu un microcalculator cu display pe bază de tub catodic, care este conectat la calculatorul gazdă împreună cu echipamentele periferice asociate (planșetă digitală, imprimantă serială și plotter digital). Sistemul gazdă construiește baza de date, memorează programele, realizează un volum mare de calcule, iar cifrele și tabelele obținute sînt furnizate ca ieșiri.

Descrierea sistemului de planificare asistat de calculator pune în evidență faptul că, pe baza unei metode interactive om-mașină, se poate asigura gestiunea unei cantități mari de date și a mai multor tipuri de programe. Ca urmare, se poate alocă mai mult timp pentru planificare ceea ce înseamnă că este posibilă evaluarea mai multor planuri alternative, deci planificarea este îmbunătățită.

În lucrare se face prezentarea unei aplicații a sistemului într-o zonă urbană reală. S-a avut în vedere evaluarea influenței pe care a avut-o construcția metroului asupra sistemului de rețele ale transportului urban.

Probleme vizate de lucrările 11.6/A-3, 11.6/A-5 și 11.6/A-6 se referă la aspecte care intervin în traficul urban fără a avea o generalitate atît de largă ca problemele abordate de lucrările prezentate pînă acum și care tratau aspecte legate de îmbunătățirea pe ansamblu a traficului urban.

Astfel lucrarea 11.6/A-3 urmărește îmbunătățirea traficului pe o arteră de circulație cu sens unic prin conducerea corelată a semafoarelor care reglează circulația la fiecare joncțiune de trafic de pe artera respectivă. Pentru aceasta se propune utilizarea unui regulator fazic în logică „fuzzy” (logică infinit valentă) pentru reglarea a două joncțiuni de trafic succesive, aflate pe aceeași arteră de circulație, în condițiile în care joncțiunea de trafic este străbătută de un număr relativ mare de vehicule. Folosirea acestui regulator este contraindicată în situațiile în care numărul de vehicule este foarte mare sau foarte mic. Această problemă este soluționată prin utilizarea unui regulator în logică „fuzzy” în cooperare cu un regulator fazic în logică „fuzzy”, cooperarea celor două regulatoare asigurînd o bună reglare a circulației și în cazurile în care intervin modificări în limite largi ale numărului de mașini, așa cum s-a constatat că se întîmplă înainte și imediat după orele de vîrf. Din acest motiv, reglarea semafoarelor de circulație instalate pe o arteră de circulație cu sens unic se va face prin cooperarea celor două tipuri de regulatoare, așa cum s-a arătat mai sus. Alternarea regulatoarelor se va realiza cu ajutorul unei metode de interschimbare bazată pe un algoritmul de tip „fuzzy”.

Pentru început, în lucrare se face prezentarea generală a unui regulator în logică „fuzzy”, după care se arată modul în care acesta este utilizat în cazul unor joncțiuni succesive de trafic. Urmează prezentarea generală a regulatorului fazic în logică „fuzzy”, după care se face descrierea modului în care cele două regulatoare se utilizează combinat.

În lucrare se prezintă un exemplu în care intervin doar două joncțiuni de trafic, însă metoda de interschimbare utilizată poate fi aplicată cu ușurință pentru un număr mare de joncțiuni de trafic de pe o arteră de circulație. Extinderea se face în felul următor: la prima joncțiune de trafic se utilizează un regulator în logică „fuzzy”; fiecare dintre următoarele joncțiuni de trafic poate fi controlată printr-o cooperare dintre un regulator în logică „fuzzy” și un regulator fazic în logică „fuzzy”. Algoritmul prezentat în lucrare este foarte simplu, ceea ce permite utilizarea unor microcalculatoare ieftine pentru realizarea sistemelor de reglare a semafoarelor.

Lucrarea 11.6/A prezintă un regulator ierarhizat destinat unui vehicul automatizat pe drumuri mari. În lucrare este propusă o structură de conducere ierarhizată pe două nivele,

cu supervisor unic la nivel superior și cu 3 subordonări la nivel inferior. Configurația a fost implementată și evaluată pe baza unei simulări de laborator pentru care s-au construit în mod empiric modelele dinamicii laterale și longitudinale ale vehiculului.

O soluție eficientă pentru descongestionarea traficului rutier constă în realizarea unor rețele de circulație care să cuprindă și linii automatizate. Aceasta asigură o serie de avantaje legate de îmbunătățirea securității traficului, utilizarea mai eficientă a șoselelor, economisirea de carburanți, reducerea stressului șoferului etc. Conducerea unui astfel de sistem este realizată pe baza unei configurații ierarhizate pe 4 nivele. Un regulator centralizat asigură supervizarea tuturor operațiilor din rețea. Al doilea nivel de conducere va fi la nivel zonal. Fiecare regulator zonal va superviza activitatea într-un anumit număr de sectoare și va conduce vehiculele din sectoarele respective pe baza comenzilor furnizate către calculatoarele de sector, care formează al treilea nivel de conducere. Fiecare calculator de sector va controla vehiculele din sectorul său. Al patrulea nivel de conducere este cel corespunzător fiecărui vehicul.

Lucrarea prezintă un regulator ierarhizat utilizat pentru conducerea unui vehicul automatizat. Principalul scop urmărit a fost acela de a îmbunătăți securitatea, eficiența și fiabilitatea unui astfel de regulator. Pentru aceasta s-a utilizat o configurație multiprocesor ierarhizată pe două nivele, avându-se în vedere asigurarea redundanței la conversia datelor. Toate funcțiile vehiculului vor fi supervizate de calculatorul de la nivelul superior, care va transmite comenzi spre calculatoarele de sector și va utiliza la rîndul său datele prelucrate furnizate de calculatoarele de sector. Calculatoarele de sector sînt dependente de aplicație. Astfel, calculatorul 1 asigură reglarea laterală, calculatorul 2 reglează viteza, iar calculatorul 3 supraveghează defecțiunile. Dacă este necesar, la nivelul 2 se mai pot adăuga și alte calculatoare.

Funcțiile de supervizare ale calculatorului de la nivelul superior și funcțiile de conducere ale calculatoarelor de la nivelul inferior necesită numai 10 ms pentru fiecare perioadă de eșanționare de 150 ms. Astfel 92% din timpul fiecărui calculator poate fi utilizat pentru realizarea altor funcții.

O problemă de generalitate mai largă decît cele abordate în ultimele două lucrări prezentate este tratată în lucrarea 11.6/A-5, care se ocupă cu problema dirijării direcției de circulație. Scopul urmărit constă în indicarea unei rute cît mai convenabile pentru un vehicul care se apropie de intersecție. Vehiculul trebuie să indice punctul de destinație, iar calculatorul local de la intersecție va răspunde prin specificarea rutei care trebuie urmărită.

Toate metodele de dirijare a direcției de circulație se bazează pe utilizarea unui tabel de dirijare a circulației. Acest tabel poate fi static (poate fi modificat numai în împrejurări critice, pe baza unei comenzi centrale), sau poate fi dinamic (modificarea tabelului se face în funcție de trafic). Dirijarea dinamică a circulației permite o rezolvare mai rapidă a situațiilor critice și o manipulare mai bună a modificărilor în circulație.

Implementarea sistemului de dirijare a circulației ridică probleme foarte mari din punct de vedere practic, problemele teoretice nefiind încă complet soluționate. Lucrarea abordează cîteva dintre problemele teoretice care nu au fost încă rezolvate. Se face o analiză a influențelor reciproce dintre rezultatele descentralizării, optimizării și prelucrărilor complexe.

Descentralizarea presupune utilizarea mai multor regulatoare în sistemul dat, fiecare regulator avînd rolul de a măsura numai anumite ieșiri, asigurînd reglarea unui număr restrîns de intrări. Această soluție este mai convenabilă deoarece, în cazul dat, centralizarea tuturor intrărilor și ieșirilor duce la scăderea fiabilității și la ridicarea costurilor.

Se utilizează un model în cadrul căruia întîrzierilor datorite congestionării circulației li se vor asocia cozi de așteptare în nodurile rețelei. Circulația revine la efectuarea unei mișcări cu viteză constantă între noduri. Timpul de trecere de la un nod la altul este dat de suma dintre întîrzierea constantă de circulație și întîrzierea datorată timpului de așteptare în coada formată la nodul de rețea. Acest model evită dinamica nelineară care ar rezulta dacă s-ar lua în considerare un profil de viteză dependent de densitatea de trafic.

Selectarea soluției optime se face avînd în vedere comportarea pe ansamblu a sistemului. Opțiunea făcută trebuie să țină cont de cheltuielile posibile și de implicațiile unor noi sarcini de calcul care se impun în cazul în care ar fi necesară adăugarea unor noi canale de informare.

O altă problemă importantă se referă la complexitatea efortului de calcul impus de obținerea unei soluții optime. Se prezintă o metodă de optimizare ierarhizată pentru rezolvarea problemei de reglare optimă globală, ținîndu-se seama de restricțiile legate de lungimea cozilor de așteptare în noduri și de întîrzierile din sistem. Deși metoda asigură anumite avantaje din punct de vedere numeric, este totuși prea lentă pentru lucrul în timp real.

3. SISTEME PENTRU TRANSPORTUL NAVAL

Lucrarea 11.6/B-2 abordează câteva aspecte în legătură cu proiectarea unui pilot automat (autopilot) adaptiv pentru nave. Modelul matematic al autopilotului descris este dezvoltat pe baza unor principii de natură fizică și hidrodinamică. Autopilotul bazat pe un astfel de model prezintă o serie de avantaje față de autopilotul bazat pe modelele cutiei negre, și anume:

- permite estimarea apriorică a parametrilor pe baza unor legi fizice și hidrodinamice;
- pentru un anumit ordin al modelului, numărul parametrilor care trebuie estimați este mai mic;
- autopilotul descris în lucrare asigură minimizarea acțiunilor de cîrmă nedorite, utilizînd pentru aceasta un model al mișcării de deviație induse de valuri.

Modelul matematic al autopilotului este utilizat într-o structură de conducere adaptivă. Pentru estimarea stării se va utiliza un filtru Kalman cu amplificare constantă. Matricia de amplificare a filtrului Kalman este calculată o singură dată la început, după care rămîne neschimbată. Matricia de amplificare a filtrului poate fi modificată pentru o anumită navă pe baza datelor cunoscute în legătură cu noua lungime a navei. Calculul matricii de amplificare se poate face cu o precizie suficient de mare utilizînd amplificarea filtrului corespunzătoare unei nave de lungime egală cu unitatea. Pentru implementare s-a utilizat versiunea discretizată a filtrului.

Algoritmului de estimare a parametrilor se bazează pe calculul erorii medii pătratice de predicție a filtrului Kalman. După inițializare, funcționarea algoritmului se caracterizează prin „uitarea” unei cantități de informație (referitoare la diferiți parametri) egală cu cantitatea de informație furnizată de măsurările efectuate la un moment dat. Pentru analiza acestui algoritmu s-au făcut simulări pe calculator și au fost realizate experimente la bordul navei de coastă norvegienne MIDNATSOL. Testările au fost făcute în timpul operațiilor normale ale navei, iar rezultatele obținute au fost bune. Pilotul automat adaptiv va fi introdus în cel mai scurt timp în producția comercială.

Lucrarea 11.6/B-3 prezintă un sistem pentru conducerea încărcării unei nave cu pernă de aer. Pentru aceasta se utilizează un sistem cu ventilator activ. Principalul element al sistemului este reprezentat de fluxul axial de ridicare al ventilatorului. Caracteristicile acestuia pot fi modificate prin intermediul paletelor ale căror unghiuri de înclinare față de planul de rotație sînt modificate continuu pe durata funcționării. Mișcarea unghiulară a paletelor este obținută cu ajutorul unui element de execuție hidraulic, care la rîndul său este reglat cu ajutorul unei bucle cu reacție după poziție. Reglarea unghiurilor paletelor ventilatorului și reglarea accelerației de ridicare se realizează prin intermediul buclei interioare și respectiv prin intermediul buclei exterioare.

Funcțiile de transfer asociate dinamicii pernei de sprijin s-au obținut pe baza identificării parametrilor cu ajutorul unor modele ale planului coeficienților. Coeficienții, la rîndul lor, au fost estimați cu ajutorul unui algoritmu de optimizare nelinier.

Experimentele efectuate au pus în evidență faptul că sistemul asigură un mijloc rapid și eficient de reglare a accelerației de ridicare și, în plus, este asigurată o reducere importantă a efectului de frinare în timpul traversării valurilor. Rezultatele obținute nu sînt optime, însă se încearcă în continuare îmbunătățirea lor.

4. CONCLUZII

Lucrările prezentate în cadrul secțiunii 11.6 au tratat probleme legate de transportul terestru și naval, oferind diferite soluții pentru îmbunătățirea traficului.

Cîteva dintre lucrările prezentate au abordat probleme care privesc rezolvarea traficului urban pe ansamblu. Astfel *lucrarea 11.6/A-4* propune un sistem de planificare asistată de calculator pentru rețelele de transport public, *lucrarea 11.6/A-5* propune un sistem de dirijare a circulației, *lucrarea 11.6/A-1* propune un sistem de conducere ierarhizată descentralizată a semafoarelor de circulație, iar *lucrarea 11.6/A-2* propune un sistem de modelare pentru procesele de conducere a sistemelor de transport.

Probleme de o mai mică amploare sînt tratate în cadrul *lucrărilor 11.6/A-3 și 11.6/A-6*, care prezintă un regulator fazic în logică infinit valentă pentru conducerea sincronizată a joncțiunilor de trafic de pe arterele de circulație cu sens unic, respectiv un regulator ierarhizat pentru un vehicul automatizat.

În cadrul *secțiunii 11.6/B* au fost abordate probleme legate de transportul naval. Astfel *lucrarea 11.6/B-2* prezintă aspecte legate de proiectarea unui pilot automat adaptiv, iar *lucrarea 11.6/B-3* prezintă aspecte legate de automatizarea încărcării unei ambarcațiuni cu pernă de aer.

HIDROLOGIA, PROTECȚIA CALITĂȚII AERULUI ȘI A APEI, GOSPODĂRIREA RESURSELOR DE APĂ *

Ioan Dima, Violeta Vișan
I.C.P.G.A.

Viorel Al. Stănescu
I.M.H.

Gospodărirea apelor și protecția mediului, condiții intrinseci ale progresului economic și ale creșterii calității vieții, intervin tot mai pregnant ca o componentă majoră în activitatea cotidiană precum și în planurile sau programele de dezvoltare, ca și în studiile de prospectare a viitorului pe diferite orizonturi de timp.

Prin specificul lor multi și interdisciplinar, și dată fiind complexitatea și diversitatea problemelor de gospodărire a apelor și de protecție a mediului, rezolvarea lor reclamă eforturi conjugate din partea specialiștilor din practic toate sectoarele științei și tehnologiei.

Faptul că mare parte din problemele de gospodărire a apelor și de protecție a mediului, indiferent de aria aferentă și de scara de timp, implică luarea de decizii ierarhizate, în a căror pregătire intervine prin conexiune inversă rezultatul aplicării deciziei anterioare, justificând utilizarea conceptelor și metodelor teoriei reglării (controlului) optimale sau adaptive, își găsește confirmare în numeroasele referate prezentate pe aceste teme la congres.

În cadrul *colocviilor 11.7 și 11.8 (secțiunile 11.8/A, 11.8/B, 11.8/C)* s-au abordat aspecte privind hidrometeorologia, protecția calității aerului și apei și gospodărirea resurselor de apă.

Cele 18 referate prezentate se pot grupa în funcție de aspectele tratate astfel :

- aspecte de hidrometeorologie și protecția calității aerului, un număr de 3 referate [1, 2, 3] ;
- aspecte de protecție a calității apelor inclusiv epurarea apelor uzate, un număr de 7 referate [4, 5, 6, 7, 8, 9, 10] ;
- aspecte de gospodărire a resurselor de apă în bazine amenajate, inclusiv de alimentare cu apă, un număr de 8 referate [11, 12, 13, 14, 15, 16, 17, 18].

În cele ce urmează se prezintă, pe cele 3 subgrupe, o sinteză conținând elementele de esență ale referatelor, din cadrul secțiunilor simpozionului interesând domeniul apelor.

1. PERFECȚIONAREA MODELĂRII ÎN HIDROLOGIE ȘI PROTECȚIA CALITĂȚII AERULUI

Referatele din această subgrupă privesc optimizarea proiectării rețelelor de supraveghere a calității aerului [1], analiza consistenței și criteriile de calibrare a parametrilor modelelor matematice în hidrologie [2]. În referatul [3] se expun rezultate din experiența specialiștilor din R.P.U. în utilizarea modelelor recursive la elaborarea prognozei operative a debitelor pe fluviul Dunărea.

1.1. Proiectarea rețelelor de supraveghere pentru controlul poluării aerului [1]

Supravegherea continuă a poluării aerului este necesară atât în scopul cunoașterii evoluției calității acestuia, cît și pentru luarea unor decizii de reglare asupra surselor de poluare. Aceasta, în condițiile în care, pe de o parte, fondurile alocate acestei activități sînt limitate, iar pe de altă parte, modelarea fizico-matematică a dispersiei poluanților în zonele urbane nu

* Ioana Dima și Violeta Vișan, I.C.P.G.A. ; Viorel Al. Stănescu, I.M.H. (coordonatori). Rezultatele pe paragrafe : D. Mișu [1.1], V. Al. Stănescu [1.2] și I. P. Zlate [1.3], I.M.H. Ioan Dima [2.1, 2.2, 2.6, 2.7, 3.1, 3.4, 3.6], C. Al. Negulescu [2.3], Th. Ognean [2.4, 2.5], V. Vișan [3.2]. V. Vișan și Carmen Călătan [3.3], R. Drăgășanu [3.5], V. Rojanschi [3.7, 3.8], I.C.P.G.A., Institutul de cercetări și proiectări pentru gospodărirea apelor, București. I.M.H. — Institutul de meteorologie și hidrologie, București.

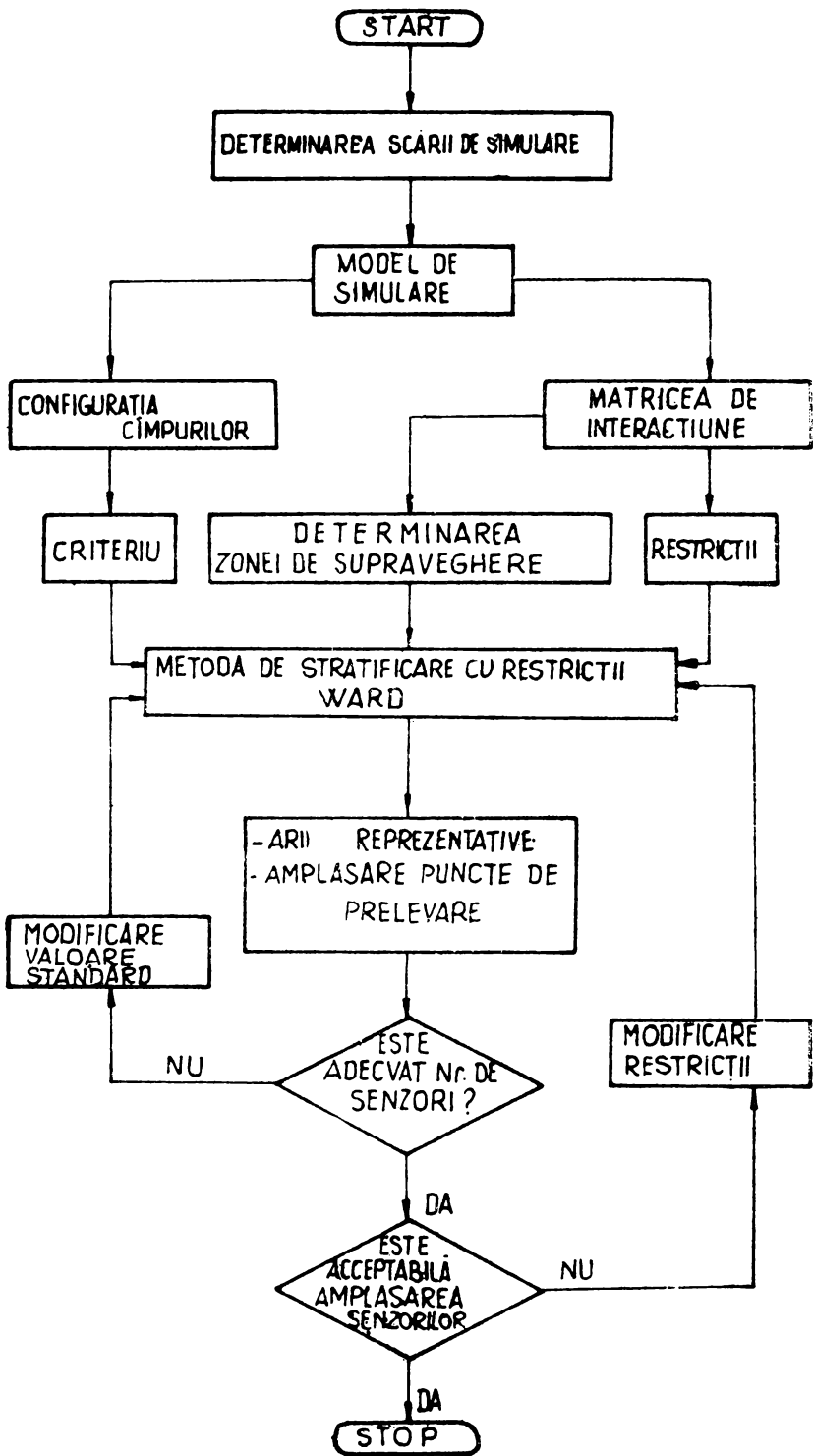


Fig. 1. Schema logică a procesului interactiv de proiectare a rețelelor de supraveghere.

poate oferi o predicție spațio-temporală a evoluției timpurilor de concentrații de poluanți ca urmare a variabilității extrem de mari a emisiilor și a factorilor meteorologici.

În proiectarea optimă a rețelelor de supraveghere a poluării aerului se cerc să se amplaseze punctele de prelevare de pe teritoriul unei zone de supraveghere astfel încât ariile de reprezentativitate oferite să fie două câte două adiacente.

Accastă formulare generală cuprinde în sine trei aspecte majore :

- numărul punctelor de prelevare ;
- reprezentativitatea punctelor de prelevare ;
- amplasarea fiecărui punct de prelevare în cadrul zonei de supraveghere.

Tratarea acestor aspecte se poate realiza prin participarea nemijlocită a factorului uman, în dialog cu calculatorul, în cadrul unui proces interactiv de rezolvare a problemei de proiectare optimă a rețelelor.

Lucrarea folosește un model de simulare a procesului de dispersie de largă aplicabilitate, capabil să evalueze concentrații medii pe termen lung (o lună, un sezon, un an etc.).

În fig. 1 este prezentată schema logică a desfășurării procesului interactiv.

Datorită faptului că aria de reprezentativitate a fiecărui punct de prelevare devine variabilă în condiții meteorologice diferite, în fiecare caz în parte se poate introduce, în locul reprezentativității legate de punctul de prelevare, noțiunea de *arie de acoperire*, legată de o sursă sau un grup de surse care pot influența într-un anumit grad concentrațiile din această arie.

Articolul prezintă o aplicare a procedurii prezentate pentru aria orașului Kyoto. Sistemul interactiv propus realizează o compatibilitate între resursele avute la dispoziție pentru o rețea de supraveghere, nivelul de cunoaștere generală a fenomenului de dispersie și necesitatea conservării calității aerului. Acest sistem permite oricărui utilizator o rezolvare globală și rapidă a problemei proiectării rețelelor de supraveghere a calității aerului.

1.2. Consistența parametrilor modelelor matematice în hidrologie și criterii de calibrare [2]

Procedurile de calibrare ale modelelor matematice în hidrologie bazate pe tehnici de optimizare a funcțiilor nelinare conțin anumite imperfecțiuni datorite în special următoarelor cauze :

- existența posibilă a unor zone optime locale ;
- inexistența unei strategii suficient fundamentate pentru trecerea de la soluția inferioară la cea mai bună soluție de optimizare ;
- unii parametri nu sînt independenți unul de altul, existînd astfel o interacțiune de natură fizică între aceștia :

- erori în datele măsurate care servesc la comparare ;
- insuficiențele criteriilor de calibrare.

Lucrarea prezintă o metodologie de rezolvare a ultimei probleme.

În general, într-un procedeu ideal de calibrare, rezidualii (diferențele între valorile măsurate și cele calculate) trebuie să formeze serii temporale necorelate, pentru ca valorile parametrilor să fie consistente. În caz contrar înseamnă că nu a fost folosită întreaga informație de care se dispune și ca atare calibrarea nu are un caracter optimal. Pornind de la modelul descris în fig. 2 se propune utilizarea unui număr de 4 criterii de optimizare, astfel :

- criteriul funcției *ad-hoc*, F_1 , exprimată pe baza debitelor de apă măsurate și respectiv calculate din ziua i , și a numărului de zile utilizate pentru operația de calibrare ;
- criteriile exprimate de funcțiile de verosimilitate F_2 (funcția autoregresivă a rezidualilor), F_3 (funcția heteroscedastică a rezidualilor), F_4 (funcția autoregresivă — heteroscedastică a rezidualilor).

Pentru un bazin folosit ca test s-au selectat 7 seturi de valori inițiale ale parametrilor modelului și a fost aplicată procedura de calibrare pentru toate cele 4 funcții criteriu menționate, rezultînd în final 28 de seturi de valori finale ale parametrilor. Procedura de optimizare s-a aplicat pentru o perioadă de 350 zile.

Procedura aplicată are ca scop determinarea aceluia criteriu care să asigure următoarele :

- cea mai bună consistență a setului final de parametri ;
- valorile cele mai de încredere ale parametrilor modelului, care să fie capabile să simuleze cât mai corect scurgerea din perioada care nu a servit la calibrare.

În ceea ce privește consistența setului de parametri s-a constatat că funcția „ad-hoc” estimează consistent numai parametrul x_1 , capacitatea de înmagazinare în stratul superior al solului, în timp ce toate celelalte funcții de verosimilitate, și în special F_2 și F_3 , sînt mai eficiente în estimarea celorlalți parametri importanți ai modelului.

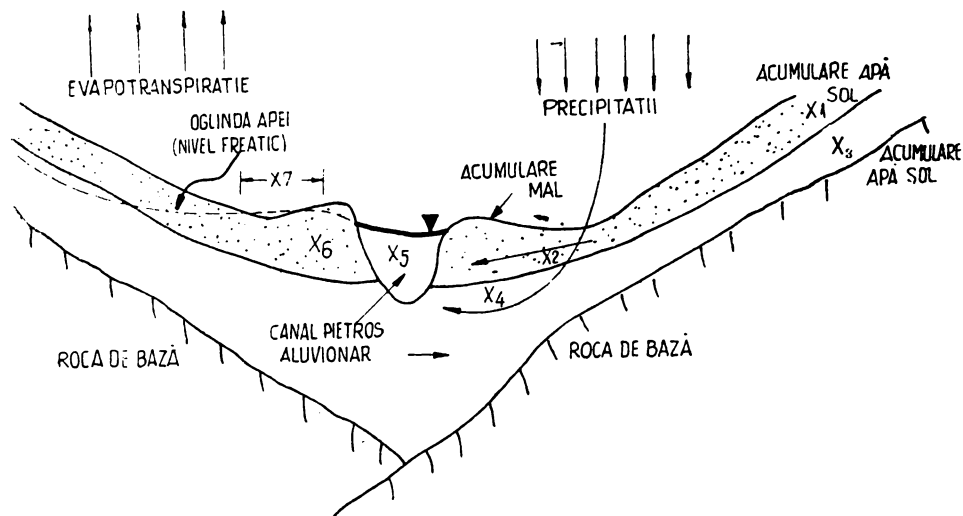


Fig. 2. Schema de principiu a modelului conceptual :

X_1 , X_3 — capacități de înmagazinare în stratul superior, respectiv inferior al solului; X_2 — cantitate de apă nereținută de sol; se scurge spre albie și se infiltrează în subsol; X_4 — scurgerea din subsol spre albie; X_5 , X_6 — scurgere, respectiv infiltrație dinspre mal spre albie și invers X_7 — fracțiune care suplimentează înmagazinarea laterală

1.3. Utilizarea algoritmilor recursivi la prognoza hidrologică în timp real [3]

În scopul obiectivizării prognozelor hidrologice, în R.P. Ungară au fost elaborate [3] două modele matematice (în care incertitudinile sînt considerate explicit ca tip, mărime și mod de propagare), pe care Serviciul Hidrologic Național de Prognoze din Ungaria le folosește în activitatea zilnică pe Dunăre în modul interactiv *om-mașină*. Se îmbină astfel informațiile obținute pe următoarele două căi, și anume: model combinat structural-stohastic cu actualizarea prognozei prin filtru liniar Kalman; model stohastic integral cu actualizarea prognozei pe principii autoreglării, cu experiența prognozistului.

Ambele modele au structură modulată pentru flexibilitate, putînd fi adaptate condițiilor particulare din bazinele mari sau mici.

Primul model pornește de la aproximația de ordinul I a ecuațiilor Saint-Venant, sub forma sectoarelor caracteristice obținută de Kalinin-Miliukov, cu soluție similară de altfel modelului cascadei Nash. Dezvoltarea modelului este bazată pe principiul analizei mulțimii stărilor.

Definind pentru un sector — numărul „ n ” de sectoare și „ K ” timpul de propagare sînt obținute printr-o tehnică de optimizare — volumul „ x ” ca variabilă de stare și considerînd intrarea „ u ” în primul sector, iar „ y ” ieșirea din ultimul, se poate scrie un sistem dinamic de ecuații lineare caracterizat de matricile F , G și H , unde F este operator și matrice de selectare, G — vectorul de selecție al termenului de comandă, sau excitația sistemului natural, iar H — operator și vector de selecție a observației. Prin modelul prezentat se obține eroarea de 13 cm pe 10 mai 1979 în prognoza cu anticipare de 5 zile la Mohaci pe Dunăre.

Al doilea model (integral stohastic adaptiv cu actualizare pe principiul autoreglării) realizează predicția în funcție de erorile de estimare anterioare.

Variabilele de intrare se referă la stațiile din amonte și de pe afluenți. După recepționarea ultimelor date măsurate se reactualizează vectorul parametrilor și matricea de pondere exprimată prin matricea de covarianță a erorilor de estimare a parametrilor.

Modelul este derivat din ARMA și realizează prognoza în 4 pași :

1) Realizarea corecției predicției $\hat{y}(t/t-K)$, făcute la timpul $t-K$ pentru timpul t , în funcție de măsurările efectuate la timpul t ,

$$\varepsilon(t) = y(t) - \hat{y}(t/t-K) \quad (1)$$

2) Actualizarea vectorului parametrilor constanți dar necunoscuți $\hat{a}(t)$ prin algoritmul recursiv de tipul celor mai mici pătrate, funcție de estimația la momentul $t-1$ a vectorului \hat{a} și de matricca de covarianță a erorilor de estimare pentru momentul t .

3) Actualizarea matricii de covarianță;

4) Predicția cu noii parametri și vectori de date:

$$\hat{y}(t+K/t) = x^T(t) \hat{a}(t) \quad (2)$$

Al doilea model este utilizat în paralel cu primul, iar prognosticul ia o decizie în mod subiectiv în funcție de estimațiile prin modele, de performanțele recente ale acestora, modelele fiind considerate factori ajutători de decizie la experiența sa personală.

Deși primul model este mai eficient, prospectarea la momentul t pe două căi diferite este necesară, dat fiind că ambele moduri de aproximare a naturii introduc erori inerente, fapt care determină ca experiența prognosticului să nu poată fi neglijată niciodată din procesul elaborării prognozei hidrologice.

2. ABORDĂRI MAI EFICIENTE ÎN CONCEPEREA ȘI APLICAREA MĂSURILOR DE PROTECȚIE A CALITĂȚII APELOR

În cadrul acestei grupe de referate sînt expuse rezultate și abordări în curs pentru îmbunătățirea și extinderea aplicării conceptelor, metodelor și procedeele bazate pe teoria mulțimilor vagi și pe teoria reglării optimale în activitatea practică de supraveghere și protecție a calității apelor, precum și în exploatarea curentă a stațiilor de epurare.

Rezultatele și preocupările legate de utilizarea teoriei mulțimilor vagi și a modelelor ierarhizate în analiza și luarea deciziilor de protecție a calității apelor fac obiectul referatului [4].

Aspecte legate de folosirea unui *estimator de stare* pentru supravegherea poluării apelor unui rîu avînd regim de curgere nestaționar sînt redată în referatul [5].

Referatul [6] tratează optimizarea rețelelor de canalizare prin controlul funcționării pe mai multe nivele.

În referatele [7, 8] se expun aspecte legate de aplicarea modelării matematice la procesele de epurare a apelor prin procedeul cu nămol activat, respectiv în cazul procesului de neutralizare a apelor uzate.

Prezentarea unui sistem de dirijare a funcționării unei stații de epurare prin utilizarea unor scenarii de decizie face obiectul referatului [9].

Referatul [10] analizează critic stadiul aplicării teoriei reglării optimale la problemele de protecție a calității apelor. Se evidențiază posibilitățile de extindere a utilizării conceptelor acestei teorii în condițiile favorabile ale dezvoltării tehnicii și de accentuare a cerințelor privind protecția calității apelor, subliniindu-se totodată necesitatea unor reorientări asupra rolului exploatarea stațiilor de epurare.

2.1. Utilizarea teoriei mulțimilor vagi și a modelelor ierarhizate la protecția calității apelor în cazul unor impurificări neconcentrate [4]

În studiul măsurilor privind majoritatea sistemelor tehnice este posibilă optimizarea unor parametri de performanță, definiți și măsurabili, în condiții impuse de restricții definibile și măsurabile.

În cazul sistemelor de gospodărire a apelor și, în particular, al celor pentru protecția calității apelor, definirea funcției obiectiv și a restricțiilor este o problemă deosebit de complexă.

Astfel, analiza măsurilor de protecție a calității apelor, legat de sursele de impurificare neconcentrate din centrele populate, implică aspecte care nu pot fi cuantificate sau măsurate cu precizie. Două asemenea aspecte privesc participarea populației la elaborarea măsurilor și la supravegherea calității apei, precum și estimarea eficienței tehnice și economice a variantelor de soluționare.

Pentru considerarea celor două aspecte s-au elaborat procedee de studiu bazate pe teoria mulțimilor vagi și a modelării ierarhice. Mulțimile vagi sînt utilizate la generarea și analiza datelor de bază și totodată oferă posibilități sporite de interpretare a eficienței măsurilor studiate. Modelul preconizat este de tip *agregat*, reprezentînd o combinație a abordării ingineresti clasice cu un proces de ierarhizare pe mai multe nivele. Modelul matematic se axează pe analiza eficienței procesului, determinînd *valoarea optimistă* și *valoarea pesimistă* a acestuia cu ajutorul teoriei mulțimilor vagi.

Pentru aplicarea modelului la *analiza eficienței participării publicului* în planficarea măsurilor de protecție a calității apelor se consideră următorii factori cu un pronunțat caracter vag ;

- nivelul influenței publicului asupra deciziei, determinat de gradul de participare, nivelul impactului publicului asupra deciziei și de gradul de reprezentativitate a participanților ;
- gradul de performanță a soluției adoptate, caracterizat de efectul acesteia în realizarea obiectivelor și de nivelul de impact asupra mediului ;
- atitudinea cetățenilor față de măsurile adoptate.

Modelul de ierarhizare a fost utilizat de asemenea pentru *analiza eficienței măsurilor de protecție a calității apelor*. S-a întrebuițat un algoritm euristic și algoritmul de programare dinamică în condiții vagi, determinîndu-se *valoarea pesimistă* și *valoarea optimistă* pentru eficiența măsurilor analizate, eficiență exprimată atît sub aspect tehnic, cît și economic.

Acest mod de abordare face posibilă analiza cantitativă a unor măsuri/proiecte depinzînd de parametri cu caracter vag datorit impreciziei în determinare. (Se subliniază diferența principală între caracterul vag datorit impreciziei în determinare și incertitudinea datorită în principal naturii aleatoare a elementelor analizate). *Valoarea pesimistă* și cea *optimistă* a funcției de eficiență oferă planficatorului o imagine mai reală asupra limitelor cîmpului determinat de diferite atitudini privind diferiții factori sau parametri, decît în cazul calculării unei singure valori. Se consideră de asemenea utilă determinarea unei valori intermediare, ceea ce ar ajuta la alegerea soluției făcînd uz și de procedul Delphi.

2.2. Supravegherea poluării apelor într-un rîu influențat de fenomenul de maree, folosind un estimator de stare cu structură comutabilă [5]

Înrăutățirea calității apei rîurilor datorită impurificării de natură organică este o problemă actuală și urgentă în multe țări industrializate. Această situație accentuează necesitatea realizării de sisteme de supraveghere, în scopul obținerii de informații pentru decizii de intervenție care să prevină depășirea concentrației admisibile.

Preocupările de pînă acum ale specialiștilor atît privind modele de tip cu debit constant, cît și modele de tip dinamic nu se referă la situații de nestaționaritate hidromecanică.

Referatul prezintă un procedeu de determinare denumit *estimator de stare* pentru supravegherea variației calității apelor într-un rîu cu regim de nivel variabil (sub influența fluxului și refluxului mării în care deversează). Procedeu permite luarea în considerare a condițiilor de nestaționaritate și de fluctuație a regimului curgerii apei și este utilizabil la supravegherea impurificării organice în rîurile cu nivelul apei influențat de fenomenul de maree, atunci cînd se dispune numai de măsurări de la senzori situați la mare distanță.

Estimatorul constă din două componente de bază : *modelul procesului analizat* pentru generarea de estimări ale variabilelor de stare și o *bucleă de corecție* prin care se semnalează abaterile dintre valorile estimate de model și valorile determinate efectiv, reprezentînd conexiunea inversă pentru îmbunătățirea estimărilor viitoare.

Modelul de rîu este de tip unidimensional, în care timpul este considerat ca o variabilă continuă, iar spațiul ca variabilă cu valori discrete. În model intervin pentru un sector de rîu următoarele variabile :

- valori medii pe sector pentru secțiunea de curgere (secțiunea vie), concentrația oxigenului dizolvat și consumul biochimic de oxigen ;
- debitul de apă și de impurificări efluent din sectorul considerat, în sectorul din aval.

După încercări, în care s-a utilizat *filtrul Kalman extins*, acesta a fost înlocuit cu un filtru mai simplu de tip *matrice de amplificare* folosind un microcalculator la locul de observare, în teren. Pentru a lua în considerare inversarea direcției de curgere a apei, *matricea de amplificare* din alcătuirea estimatorului are structură comutabilă.

Rezultatele simulărilor efectuate arată că acest procedeu permite determinări cu suficientă precizie în condiții de instacionaritate a regimului apei în riuri.

2.3. Conducerea multinivel a exploatării rețelelor de canalizare [6]

Conducerea multinivel se referă la sistemele unitare de canalizare alcătuite din tronsoane de conductă, bazine tampon și elemente de deversare. În absența ploilor apele uzate menajere sînt epurate și apoi se descarcă în receptor. În caz de ploaie debitele din rețea depășesc considerabil capacitatea stației de epurare, rezultînd descărcarea în emisar a unei cantități de ape neepurate. În ultima perioadă s-a manifestat un interes crescînd pentru reducerea efectelor deversărilor de ape de ploaie neepurate din sistemele unitare de canalizare.

Reducerea descărcărilor de apă neepurată în emisar se poate realiza prin reținerea în bazinele tampon a unei părți din volumul de apă scurs în timpul ploii. Imediat după ploaie, apa din bazinele tampon se trece prin stația de epurare pentru ca aceste bazine să poată fi utilizate la întreaga capacitate în cazul unei alte ploi.

Sistemul de conducere multinivel combină evacuarea debitului optim pentru minimizarea deversărilor de apă neepurată în emisar cu costuri moderate de realizare. Structura pe mai multe nivele este alcătuită dintr-un nivel de optimizare, un nivel de reglare directă și un nivel de adaptare.

Într-o situație dată de debite din precipitații reglarea optimă a scurgerii necesită considerarea sistemului de canalizare în ansamblu. Procedura de optimizare elaborată în acest scop consideră sistemul de canalizare format din subrețele cu și, respectiv, fără capacități de stocare.

Problema de optimizare poate fi rezolvată principial prin aplicarea algoritmului de programare dinamică, dar memoria necesară devine foarte mare dacă problema include mai mult de 3—4 variabile de stare și peste 10—15 intervale de timp. De aceea, a fost aplicat *principiul maximului discret*, care permite rezolvarea unei probleme de optimizare cu mai mult de 8 variabile de stare și peste 60 intervale de timp de calcul.

Pentru aplicarea principiului *maximului discret* a fost necesară înlăturarea unor inconveniente specifice, astfel :

— unele variabile de stare exprimate ca restricții duc la dificultăți deosebite. Pentru depășirea acestei dificultăți s-a adoptat metoda *funcțiilor de penalitate* ;

— problema de optimizare este rezolvată prin exprimarea condițiilor necesare de optimalitate și formularea acesteia ca o problemă de valori limită în 2 puncte ; soluționarea problemei astfel formulate poate conduce la un minim local în locul celui global. Dat fiind însă că *funcția obiectivă* este convexă și restricțiile sînt liniare, soluția problemei de optimizare reprezintă minimul global ;

— problema se rezolvă folosind un algoritm numeric a cărui convergență trebuie examinată. Aplicînd tehnica gradientilor conjugați se garantează o convergență rapidă chiar și pentru matrici mari de penalitate.

Soluționarea problemei de optimizare necesită cunoașterea următoarelor elemente :

- debite afluate ;
- decalajul de timp determinat de tronsoanele de canalizare ;
- condiții inițiale ;
- matrici de ponderare.

Nivelul de reglare directă are ca scop modificarea comportării dinamice a procesului astfel încît să se apropie cît mai mult de modelul simplificat utilizat în nivelul de optimizare. Aceasta se realizează prin menținerea variabilelor de stare în vecinătatea traiectoriilor de referință optime determinate în cadrul nivelului de optimizare.

Nivelul de adaptare are ca sarcină :

- să asigure date bazate pe măsurări efective ;
- să inițieze o nouă analiză de optimizare ori de cîte ori datele respective au devenit necorespunzătoare.

2.4. Modelarea și reglarea procesului de epurare cu nămol activ utilizînd un model autoregresiv [7]

Pentru modelarea procesului de epurare în condiții dinamice se face în general apel la modelele matematice mecaniciste bazate pe ecuații diferențiale care exprimă bilanșurile de materiale din cadrul procesului. Asemenea modele au dezavantajul că includ o serie de coeficienți cinetici, a căror valoare se modifică o dată cu schimbarea parametrilor de funcționare ai stației

de epurare — funcționarea stației în regim *dinamic*. Modelul autoregresiv elaborat își propune să înlăture acest dezavantaj.

Modelul autoregresiv este un model statistic din categoria modelelor „cutie neagră”. Un asemenea model se definește în felul următor : starea prezentă a unui sistem este o sumă a combinațiilor lineare a variabilelor sistemului din trecut și a zgomotului alb. Aceasta se exprimă cu ajutorul ecuației :

$$X(n) = \sum_{m=1}^M A(m) X(n-m) + U(n)$$

în care : $A(m)$ este matricea coeficienților ;

$U(n)$ — zgomotul alb ;

M — ordinul modelului.

Schema generală după care se aplică modelul autoregresiv cuprinde următorii pași :

- culegerea de date dintr-o stație de epurare reală ;
- tararea/calibrarea modelului definit prin relația de mai sus ;
- evaluarea (calculul contribuției zgomotului și predicția) ;
- obținerea de date din instalația pilot ;
- calibrarea modelului pentru analize de sistem și proiectarea sistemului de reglare.

Modelul autoregresiv este capabil să reproducă datele obținute în determinările experimentale din stația de epurare. Prin utilizarea lui în cadrul unei stații pilot s-a putut prevedea modul de schimbare a variabilelor de operare a procesului. Modelul se poate implementa într-o stație de epurare propriu-zisă folosindu-se metoda reglării linear pătratice.

2.5. Reglarea adaptivă a unui proces de neutralizare a apelor uzate [8]

Se prezintă un procedeu nou pentru reglarea adaptivă a pH-ului în procesul continuu de neutralizare dintr-o instalație de epurare a apelor.

Ca urmare a limitărilor de spațiu și a altor considerente impuse de procesul tehnologic, neutralizarea are loc într-un mic reactor cu timp de staționare mic, de ordinul a 30 s.

Comportarea statică a procesului de neutralizare este determinată prin curbele de titrare a apei uzate afluate. Compoziția și concentrația apelor uzate și deci și curbele lor de titrare variază foarte repede. Variația rapidă a parametrilor de proces și marile perturbări ale procesului fac ca mijloacele de reglare convenționale să nu mai fie satisfăcătoare. Pentru a depăși aceste dificultăți s-a apelat la conceptul de *reglare adaptivă*, în care bucla (ciclul) principală de reglare este condusă de o buclă suplimentară de reglare adaptivă.

Bucla de reglare adaptivă manevrează rezultatul/avansul algoritmului de reglare principal astfel încât în cazul unor abateri, pH-ul efluentului să se apropie de punctul de neutralizare prin urmărirea *traectoriei de referință* prescrise, care constituie criteriul de adaptare. Prin acest mod de adaptare, orice gen de abateri poate fi controlată/corectată cu suficientă rapiditate.

Sistemul de reglare este constituit din următoarele blocuri :

- algoritmul de reglare ;
- viteza efectivă a abaterilor de reglare ;
- calculul vitezei de referință ;
- corecția globală.

Sistemul de reglare adaptivă a fost testat într-o stație pilot. În acest scop a fost realizată o stație pilot de neutralizare cu un mic nou reactor cu o capacitate de 1,5 m³/h și timp mediu de staționare de 25 s.

Sistemul de reglare adaptivă este implementat pe un minicalculator și numeroase experiențe au arătat utilitatea sa pentru o largă gamă de abateri.

În stația centrală de epurare sistemul se va implementa pe un microprocesor de reglare. Se subliniază pe cazul prezentat că aplicarea metodelor moderne de reglare în procesele ingineresti poate conduce la importante beneficii.

2.6. Sistem bazat pe cunoaștere pentru conducerea procesului de epurare a apelor uzate [9]

Procesul de epurare a apelor uzate la o stație de epurare existentă poate fi caracterizat prin reglarea concentrației oxigenului dizolvat (OD), a concentrației materiilor în suspensie (MTS) și a cantității de nămol cu un calculator numeric.

Sistemul bazat pe cunoaștere propus cuprinde trei părți :

- baza de date ;
- interpretor ;
- memorie.

Setul regulilor de operare se referă la :

- 1) Identificarea situației ;
- 2) Planificare și control ;
- 3) Diagnostică și procedee de tratare ;
- 4) Instrucțiunile de exploatare.

Spre exemplificare, pe baza experienței în exploatarea bazinelor cu nămol activ la scară naturală pe mai mulți ani, se definesc șase situații tipice clasificate ca :

- situația optimă ;
- 4 situații suboptimale ;
- situația anormală.

Aceste situații sînt evaluate pe baza timpului de retenție al solidelor (TRS), consumului biologic de oxigen (CBO) și temperaturii, răspunsurile fiind date de operator. Cel care evaluează situația (optimă, suboptimă, anormală) decide asupra operațiilor necesare ca urmare a perturbărilor datorate încărcării organice.

În mod asemănător poate fi evaluată situația funcționării bazinelor de fermentare a nămolului, a platformelor de deshidratare, a stațiilor de pompare etc. Din inventarul datelor de bază furnizate de operator se deduc modificările care trebuie făcute în funcționarea instalației pe baza răspunsurilor furnizate automat de calculator, care are în memorie un număr mare de instrucțiuni.

Sistemul expert propus poate ajuta la rezolvarea problemelor ivite în exploatarea stațiilor de epurare, în luarea deciziilor pentru îmbunătățirea funcționării. Avantajul sistemului derivă din faptul că datele de bază sînt furnizate pe baza experienței operatorului, deciziile fiind luate de calculator.

Preocupările în continuare au în vedere îmbunătățirea performanțelor și sporirea încrederii în acest sistem, de exemplu prin determinarea automată a regulilor de exploatare.

2.7. Reglare și evaluare operativă în protecția calității apelor [10]

În referat se analizează mai întîi rezultatele obținute în aplicarea teoriei reglării la două probleme clasice de protecție a calității apelor : reglarea oxigenului dizolvat și reglarea procesului cu nămol activat în instalațiile de epurare a apelor. Legat de aceasta se menționează că aplicarea reglării automate nu a înregistrat succese deosebite și s-a axat mai ales pe problema reglării valorii de referință în condițiile perturbațiilor frecvente.

În fig. 3 este redată schema conceptuală a unui sistem de reglare pentru mai mulți indicatori. Se menționează că simplitatea inevitabilă a figurii 3 maschează caracterul pregnant ierarhic al gospodăririi integrate a resurselor de apă, pe ansamblul bazinelor ca și localizat, de exemplu, în cazul conducerii funcționării rețelelor de canalizare.

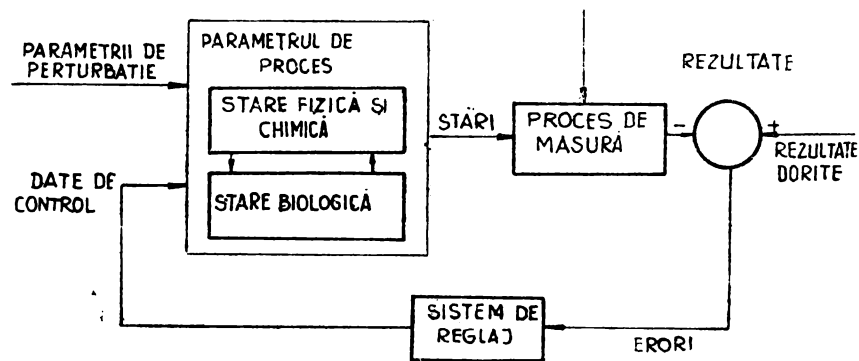


Fig. 3. Schema conceptuală a unui sistem de reglare pentru mai mulți indicatori.

Se consideră în general că reglarea în probleme de poluare a apelor privește, sau se referă la, exploatarea stațiilor de epurare și a stațiilor de tratare a apelor. Deși acesta este cazul general, contribuții importante la gospodărirea calității apei pot fi aduse prin utilizarea lacurilor de acumulare, a schemelor cu pompare din apa subterană ca și a altor sisteme de exemplu centuri (perdele) din stuf.

Problema reglării oxigenului dizolvat a cunoscut progrese, inclusiv sub forma reglării dinamice a procesului. Dintre dificultățile întâmpinate se menționează imposibilitatea determinării *on-line* a consumului biochimic de oxigen (CBO), iar condițiile tehnologice actuale nu permit reglarea valorii concentrației CBO a apelor evacuate din stațiile de epurare.

Totodată, recent a fost remarcat faptul că preocupările anterioare nu au luat în considerare gradul foarte înalt de incertitudine în comportarea sistemelor de resurse de apă.

În ce privește reglarea procesului de epurare cu nămol activat, stațiile de epurare oferă o flexibilitate în funcționare satisfăcătoare, urmărindu-se în special reducerea concentrației de substanțe solubile, a CBO și a substanțelor în suspensie din apele uzate. A devenit larg utilizabilă în practică reglarea funcționării aeratorului în legătură cu concentrația oxigenului dizolvat, printr-o schemă de tip cu o singură buclă închisă, folosind un regulator de tip cu *acțiune proporțională și integrală*. De asemenea sînt preocupări în aplicarea reglării între conținutul apei reziduale neepurate și viteza de recirculare a nămolului.

Se evidențiază însă posibilitatea unor rezultate necorespunzătoare în reglarea procesului cu nămol activat, ca de exemplu în cazul depunerii nămolului și al distrugerii populației de microorganisme în prezența unor concentrații deosebit de mari de substanțe toxice.

Condițiile favorabile oferite de dezvoltarea tehnicii și de accentuarea cerințelor privind protecția calității apelor oferă posibilități de extindere a utilizării conceptelor teoriei reglării optime.

Recunoașterea variabilității performanțelor este fără îndoială o schimbare pozitivă în favoarea aplicării conceptelor *reglării*. Caracterizarea performanței ca „satisfăcătoare” în sens *vag* este bine venită, complementar celei asociate utilizării cuantitativ de 95%.

Dat fiind caracterul problemelor de protecția calității apelor se consideră de asemenea posibilă obținerea unor bune rezultate prin aplicarea atentă a unor metode din teoria sistemelor ierarhice. Unele din aspectele menționate vor implica însă reorientări în considerarea rolului exploatarea stațiilor de epurare.

3. PROGRESE PRIVIND APLICAREA TEHNICILOR DE ANALIZĂ A SISTEMELOR LA FUNDAMENTAREA DECIZIILOR ÎN GOSPODĂRIREA RESURSELOR DE APĂ

În cadrul acestei categorii de probleme sînt de scos în evidență preocupările legate de extinderea utilizării tehnicilor de analiză a sistemelor și de perfecționare a utilizării posibilităților oferite de mijloacele moderne de calcul.

Referatul [11] analizează critic unele neajunsuri privind stadiul de utilizare a tehnicilor de modelare în studiul problemelor de gospodărire a apelor și punctează unele modalități de obținere a unor modele mai robuste pentru reprezentarea fenomenului hidrologic într-un bazin; se subliniază necesitatea utilizării conceptului de *optim global* bazat pe determinarea *frontierei de negociere* în pregătirea deciziilor privind planificarea și exploatarea amenajărilor de gospodărire a apelor.

O contribuție vizînd extinderea posibilităților de interpretare fizică a *regulilor de decizie de tip linear* în exploatarea unui lac de acumulare este expusă în referatul [12].

Prezentarea unui concept de abordare de tip *anti-risc* în stabilirea regulilor de exploatare a unui lac de acumulare cu folosințe multiple face obiectul referatului [13].

Referatul [14] scoate în evidență, pe un exemplu concret, posibilitățile oferite de asigurarea unor informații hidrologice *on-line* pentru creșterea eficienței în exploatarea a unui lac de acumulare avînd funcțiune dublă: combaterea inundațiilor și regularizarea debitelor în scopul producerii de energie hidroelectrică.

Aspectele legate de utilizarea *teoriei mulțimilor vagi* în probleme de alocare a resurselor fac obiectul referatului [15].

Referatul [16] prezintă un sistem de analiză bazat pe modelarea matematică și pe folosirea unui procedeu *interactiv* (conversațional) om-calculator, în planificarea proiectelor de amenajare în domeniul apelor.

Utilizarea modelării matematice și a lucrului conversațional om-calculator legate de extinderea rețelelor de alimentare cu apă și respectiv, în exploatarea stațiilor de pompare, fac obiectul referatelor [17], respectiv [18].

3.1. Cvasi-optimalitate în modelele de gospodărire a apelor și în cele fizice [11]

Se analizează aspectele legate de utilizarea conceptului de *cvasi-optimalitate* și de *optim global* în domeniul gospodăririi apelor.

Sînt menționate trei cauze principale care împiedică utilizarea pe scară largă a tehnicilor de analiză a sistemelor în modelele privind sistemele de gospodărire a apelor: (1) existența unor obiective (cerințe) contradictorii, (2) existența unei *suprafețe de răspuns* relativ plate în vecinătatea optimului global și în consecință posibilitatea acceptării unei game de variante de acțiune, și (3) caracterul imprecis al datelor și nesiguranța rezultatelor. Toate acestea conduc la rezultate *negociabile* și nu la prescripții lipsite de ambiguitate. Efectuarea unor asemenea negocieri în sectorul public nu se desfășoară în condiții foarte bune, astfel că uneori este mai simplu și mai comod să se adopte procedeu „soluției prestabilite”.

Hidrologia s-a dezvoltat ca o ramură a ingineriei, deoarece inginerii au avut dintotdeauna răspunderea pentru prevenirea efectelor inundațiilor și secetelor. Aceștia completau golurile în datele hidrologice, în general pe baze empirice.

Modelele convenționale *precipitații-curgere* multiparametrice, elaborate în ultimii 10—15 ani, nu se dovedesc utilizabile în practică deoarece mici variații ale numeroaselor elemente componente conduc la modificări pronunțate ale elementului final, curgerea rezultată; aceste modele sînt greu de calibrat și aplicat în calculele curente de proiectare.

Astfel, dată fiind varietatea fenomenelor și a ariilor lor se preconizează dezvoltarea de modele la scară medie care să asigure o reflectare fizică cît mai plauzibilă și care să păstreze ceva din natura *stohastică* și din caracterul *vag* al proceselor implicate. Cu alte cuvinte, este mai bine să lucrăm cu modele cu cîtiva parametri cît mai robuști (care nu sînt supuși unor modificări prea mari), decît cu modele foarte sensibile la perturbări și la erori de calibrare.

De aceea, pentru modelarea fenomenelor într-un bazin hidrografic se preconizează utilizarea a 4 variabile de bază: precipitațiile, înmagazinarea (în sol), evapotranspirația și scurgerea.

În aplicarea tehnicilor de analiză a sistemelor la proiectarea și exploatarea unor sisteme mari și complexe, problema poate prezenta un mare număr de soluții relativ apropiate din punct de vedere tehnic. Uneori, soluții diferite de soluția optimă sub aspect tehnic pot fi mai acceptabile din punct de vedere instituțional și politic. Astfel barajul Hoover din S.U.A. este un triumf al tehnicii construcțiilor, dar constituie o eroare din punct de vedere al gospodăririi apelor; trebuie însă amintit că, dată fiind situația de depresiune economică din acea perioadă, alegerea soluției s-a bazat pe criteriul bunăstării economice, respectiv pentru ocuparea forței de muncă.

În alegerea soluțiilor, în cazul amenajărilor cu cerințe competitive, este nevoie să se negocieze asupra obiectivelor și a restricțiilor. Conceptul de *optim global* devine necesar deoarece decizia finală implică afectarea parțială sau totală a intereselor uneia sau alteia dintre cerințele competitive.

Pareto-optimalitatea, ca mod de rezolvare a unor asemenea probleme, caută să identifice soluții „de compromis”, de etapă, determinînd *frontiera de negociere*, cu cîteva soluții posibile, alegerea finală a soluției urmînd a se face în cadrul procesului instituțional-politic.

3.2. Analiza și interpretarea fizică a regulilor de decizie privind regimul de exploatare a unui sistem de gospodărire a apelor [12]

În legătură cu proiectarea și exploatarea sistemelor de gospodărire a apelor se cunosc numeroase încercări bazate pe utilizarea *regulilor de decizie lineare*.

Procedeele elaborate pînă în prezent se limitează totuși la formularea problemei capacității minime a acumulării. Problema regimurilor de exploatare a unei acumulări existente pe baza unor asemenea reguli nu și-a găsit încă aplicare.

Referatul prezintă un mod nou de abordare care permite utilizarea regulilor de decizie lineare atât la proiectare, pentru determinarea capacității acumulării, cât și pentru studiul exploatării unei acumulări existente.

Pentru aceasta se formulează așa-numita *problemă operativă*, care se rezolvă prin repetiție pe baza informațiilor curente și a alegerii adecvate a formei *indicelui de performanță*.

Ca indice de performanță pentru un sistem din două lacuri s-a utilizat **printre expresia** :

$$\max_{S^*} \left(\sum S_i^{*1} + 2 \sum S_i^{*2} \right)$$

reflectând condiția de a stoca în lac o cantitate cât mai mare de apă. În această expresie S_i reprezintă valoarea probabilă (valoarea medie așteptată) a volumului de apă acumulat în lac, S_i^1 ,

Analizele efectuate pe exemple numerice au condus la următoarele concluzii :

— regulile de decizie lineară au o interpretare fizică și permit formularea problemei de proiectare și de exploatare ca probleme de programare lineară ;

— aceste reguli sînt deosebit de utile în cazul sistemelor complexe, în care alte metode devin inaplicabile ca urmare a dimensiunilor excesiv de mari ale problemei ;

— inconvenientul rezultînd din ipoteza că regulile sînt lineare poate fi neglijat, deoarece în fiecare sezon, sau interval de calcul, problema se poate rezolva repetitiv luînd în considerare condițiile curente.

3.3. Abordare de tip anti-risc în exploatarea lacurilor de acumulare [13]

Lucrarea prezintă o abordare *deterministă* a problemelor de exploatare a lacurilor de acumulare cu scopuri multiple, menită să conducă la evitarea unor situații extreme, determinînd întreaga mulțime de decizii posibile care garantează funcționarea eficientă a sistemului. Aceasta permite organului de exploatare atât să aleagă cea decizie care se potrivește cel mai bine cu informațiile suplimentare sau cu prognoza de care dispune, cât și să o adapteze la obiectivele secundare care nu au fost considerate în formularea inițială a problemei.

Regula de exploatare este de tipul $r_\tau^i = r(\tau, x_\tau^i, a_\tau^i)$, unde x_τ^i reprezintă nivelul (volumul) în lac la începutul zilei τ a anului i și a_τ^i afluxul în lac.

Regula optimă de exploatare $r(\cdot)$ va fi selectată cu referiri expresie la performanțele pe care le permite în anumite situații specifice definite printr-un set $I = \{(a_\tau^i) ; \tau = 1, \dots, 364, i = 1, \dots, n\}$ — set de referință.

Problema se formulează ca o problemă de optimizare cu dublu obiectiv, în care urmează să se minimizeze cel mai mare deficit de apă ($\max D^1$) și cel mai mare număr de zile cu inundații ($\max F^1$)

$$\left\{ \begin{array}{l} \min \\ X_0, r(\cdot) \end{array} \right\} \left| \begin{array}{ll} \max D^1 & \max F^1 \\ 1 \leq i \leq n & 1 \leq i \leq n \end{array} \right. \quad (1)$$

X_0 — set al valorilor inițiale.

Restricțiile problemei sînt :

$$x_{\tau+1}^i = x_\tau^i + a_\tau^i - r_\tau^i \quad \text{— ecuația de continuitate} \quad (2)$$

$$0 \leq r_\tau^i \leq S(x_\tau^i) \quad \text{— restricție fizică} \quad (3)$$

$$S(x_\tau^i) \quad \text{— funcția de descărcare a lacului.}$$

$$x_\tau^i \geq \underline{x} ; r_\tau^i = S(x_\tau^i) \text{ dacă } x_\tau^i > \bar{x} \quad \text{— restricție legală (reglementare)} \quad (4)$$

$$x_{365}^i = x_0 \quad \text{— restricție terminală} \quad (5)$$

Soluționarea ambelor probleme conduce la reguli de exploatare de forma :

$$r_\tau^i = r(\tau, x_\tau^i, a_\tau^i, D_\tau^i, F_\tau^i, D^*, F^*) \quad (6)$$

în care D^* și F^* sînt valorile de referință (limite superioare considerate).

Pentru aceste seturi de reguli de exploatare, o procedură simplă va determina pe acelea care garantează valoarea minimă a lui F^* pentru orice valoare dată a lui D^* .

În general, aceste reguli de exploatare eficiente nu sînt unice. Dată fiind valoarea curentă a informațiilor (τ , x_τ^i , a_τ^i , D_τ^i , F_τ^i), algoritmul sugerează o gamă întreagă de debite defluate posibile r_τ^i (fig. 4).

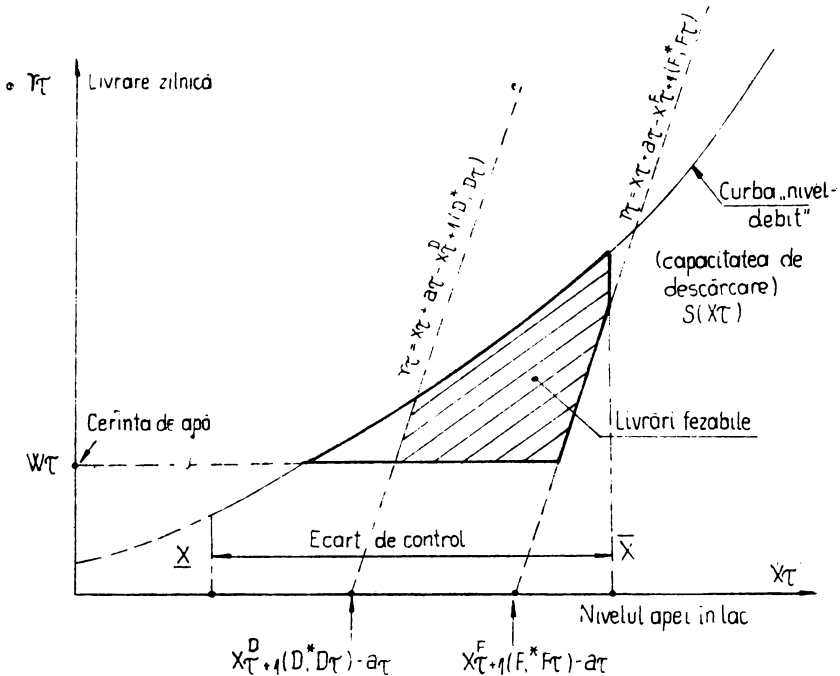


Fig. 4. Livrări fezabile pentru probleme de exploatare a unui lac de acumulare pentru combaterea inundațiilor și satisfacerea cu apă a folosințelor.

Rezultă un domeniu de debite defluate fezabile care satisfac condițiile impuse de cele două probleme. Deci, în condiții normale, există o oarecare libertate în alegerea deciziei finale, putînd fi luate în considerare și obiectivele secundare care au fost neglijate inițial în descrierea problemei. Cînd condițiile hidrologice sînt critice, această libertate dispare.

3.4. Utilizarea on-line a informațiilor hidrologice pentru exploatarea unui lac de acumulare în scopuri multiple [14]

În general, preocupările privind optimizarea exploatarea lacurilor de acumulare iau în considerare o cunoaștere „perfectă” a elementelor hidrologice de referință (debite, precipitații etc.) sub formă *deterministă* sau *probabilistă*.

Această ipoteză este nerealistă și de aceea sînt de subliniat eforturile pentru realizarea de sisteme informaționale care să asigure, în timp util, colectarea, prelucrarea și analiza datelor hidrologice. Nu s-a acordat însă suficientă atenție unor analize vizînd beneficiul obținut din îmbunătățirea exploatarea lacurilor de acumulare pe baza perfecționării sistemelor informaționale hidrologice.

Se prezintă rezultatele privind evaluarea beneficiilor obținute din sistemele care asigură informații hidrologice *on-line* în timpul perioadelor de viitor, în cazul unui lac de acumulare din bazinul rîului Mondego (Portugalia). Evaluarea se referă atît la exploatarea curentă a lacu-

lui de acumulare în timpul producerii viiturilor, cît și la reglarea adaptivă a volumului de apă din lac pentru atenuarea viiturii. În legătură cu acesta din urmă se preconizează înlocuirea curbei „ghid superior“ utilizată în graficul de exploatare a acumulării, cu conceptul de funcție „ghid superior“. Folosind conceptul menționat, informația obținută din prognoza hidrologică permite, pentru un volum de lac pentru atenuare V , diminuarea riscului anual de inundare r ; pentru un r dat este posibilă reducerea substanțială a volumului pentru atenuare V și deci, creșterea eficienței în utilizarea lacului pentru celelalte folosințe.

Exploatarea lacului de acumulare se formulează, din punct de vedere al estimării valorii informațiilor hidrologice, ca un proces în două faze :

- determinarea volumului de atenuare V , în afara perioadelor de ape mari ;
- controlul debitului defluent în timpul perioadelor de ape mari (viitură).

Analiza efectuată în cazul prezentat s-a axat pe evaluarea efectului informației hidrologice în cazul considerării volumului de atenuare V ca parametru fix și respectiv în condițiile reglării adaptive a lui V . În ambele cazuri s-au considerat trei situații privind informațiile hidrologice, și anume :

- situația 1, în care nu se dispune de loc de date hidrologice *on-line* ;
- situația 2, în care se dispune parțial de informații *on-line* privind afluxul în lac și pe zonă necontrolată aval de lac ;
- situația 3, în care se dispune de informație hidrologică perfectă *on-line*, sub forma deterministă.

Funcția „ghid superior“ pentru reglarea adaptivă a lui V s-a determinat sub forma unei familii de curbe calculate pe baza afluxului mediu în lac în ultimele 20 zile.

Ca rezultate mai importante ale analizei efectuate se menționează următoarele :

— riscul de inundare, în fiecare perioadă de ape mari considerată, este redus sensibil prin asigurarea prognozei *on-line* a debitelor de viitură provenite din zona de bazin aval de lacul de acumulare ;

— volumul de atenuare poate fi redus în mare măsură fără a afecta sensibil riscul anual de inundare în zona apărată ;

— prin utilizarea prognozei hidrologice de scurtă durată privind afluxul de apă în lac și prin reglarea adaptivă a volumului de atenuare se obține un spor de producție de energie electrică de 15 GWh, în condițiile unui an hidrologic mediu.

3.5. Utilizarea informațiilor cu caracter vag în probleme de alocare a resurselor [15]

Materialul abordează probleme de alocare a resurselor de apă și alte resurse naturale, care pot fi tratate sub formă de probleme de programare matematică cu parametri *vagi* și sugerează un mod de abordare pentru acele tipuri de probleme care permit să se determine *alternative nedominante*.

Referindu-se la modelarea matematică a unor sisteme reale, se arată că modelele tradiționale nu pot cuprinde decît un număr limitat de factori ai sistemelor sau fenomenelor studiate, fapt care poate fi ameliorat prin utilizarea unor parametri neprecizați, mai bine zis printr-o modelare care ține seama de caracterul *vag* al acestor parametri. Se arată că teoria *mulțimilor vagi (fuzzy)* aduce elementele necesare în acest sens, cu mențiunea că un sistem real poate fi obiectul unei modelări cu ajutorul unor mulțimi și al unor parametri *vagi*, cu particularitățile specifice domeniului, deci cu definițiile și terminologia adecvată.

Pentru ilustrare se prezintă un exemplu de problemă de programare matematică liniară :

Pentru n tipuri de culturi diferite se caută repartizarea parcelelor de pe o suprafață totală A , astfel încît beneficiul total să fie maxim.

Problema, în formulare tradițională, este exprimată de relația :

$$\sum_{j=1}^n p_j c_j x_j \rightarrow \max$$

cu condițiile :

$$\sum_{j=1}^n w_j x_j \leq W, \quad \sum_{j=1}^n x_j \leq A, \quad x_j \geq 0, \quad (j=1, \dots, n).$$

unde :

- p_j este beneficiul pe unitatea de producție a culturii j ;
- x_j — suprafața alocată pentru cultura j ;
- c_j — productivitatea culturii j ;
- w_j — cantitatea de apă necesară irigației culturii j pe unitatea de suprafață ;
- W — cantitatea totală de apă disponibilă pentru irigații.

În problema enunțată intervin parametrii c_j , w_j și W , care în mod natural depind de : conținutul nutritiv al solului, tehnologiile de tratare a solului, activitatea solară etc., factori ce nu sînt întotdeauna neglijabili.

De aceea, un model mai apropiat de realitate ar fi cel în care :

— parametrii respectivi se consideră nu ca numere, ci ca intervale de variații ;

— se iau în considerare unele informații suplimentare date sub forma unor coeficienți de pondere.

Se ajunge astfel la o formulare a problemei de programare matematică de tip *fuzzy*, impunându-se ca cerințele din condiția de mai sus să fie respectate cu valori ale coeficienților c_j ($j = 1, \dots, n$) și w_j date ca mulțimi *vagi*.

Problema se formulează ca o problemă de programare matematică *fuzzy*, cu funcția obiectiv reprezentată de o aplicație *fuzzy*. Se stabilește apoi o relație de preferință de tip *fuzzy* între alternativele realizabile (posibile) și este definită funcția de apartenență a submulțimii de *alternative nedominate*.

Considerînd gradul dorit de *nedominanță* α și nivelul de *fezabilitate* β , alternativa care îndeplinește aceste cerințe se poate determina rezolvînd problema de programare matematică :

$$\sum_{j=1}^n p_j c_j x_j \rightarrow \max$$

x_j, c_j, w_j

cu condițiile :

$$\sum_{j=1}^n w_j x_j \leq W, \quad \sum_{j=1}^n x_j \leq A$$

$$x_j(c_j) \geq \alpha ; \quad v_j(w_j) \geq \beta \quad j=1, \dots, n$$

$$x_j \geq 0$$

Prin modificări ale vectorilor α și β se pot determina alternative cu diferite echilibre între gradul de *nondominanță* și cel de *fezabilitate*.

În final se evidențiază avantajul introducerii unor asemenea raționamente de tip *fuzzy* pentru apropierea între limbajul folosit în modelele matematice și limbajul mai puțin precis, mai *vag*, folosit de utilizatorii potențiali ai acestor modele.

3.6. Sistem de analiză și fundamentare a deciziilor pentru programele de amenajare în domeniul apelor [16]

Pentru probleme de decizii privind eşalarea proiectelor, decalarea sau eliminarea din programul de lucrări, se propune o combinație de modele matematice, precum și de cunoștințe și intuiții ale decidentului într-un *sistem suport pentru decizii* „om-calculator“.

Sistemul de analiză și fundamentare propus este destinat să ajute pe decident în următoarele direcții :

- a) definirea exactă a politicii/strategiei pentru care urmează a se lua decizia și mai buna înțelegere a organizării ;
- b) considerarea majorității datelor semnificative care caracterizează proiectul ;
- c) calculul metodic al importanței datelor proiectului supus deciziei, în raport cu politica globală, luînd în considerare parametrii calitativi ;
- d) gradarea proiectului în concordanță cu datele strategice și cu deciziile anterioare și alegerea variantei ;
- e) examinarea relației dintre strategia preconizată inițial și strategia adoptată ;

f) definirea strategiei în luarea de decizii fundamentale; stabilirea nivelului de însemnătate și definirea detaliată a unei variante se pot face de către tehnicienii de nivel mai puțin ridicat.

Sistemul este alcătuit din: echipamentul de calcul, un set de programe și proceduri de calcul și decident. Ca o parte specifică, sistemul include algoritmul Saaty pentru definirea cantitativă a strategiei.

Modelul pentru ierarhizarea preliminară a proiectelor este alcătuit din două submodele:

— *submodelul 1* privește proiectele în curs de execuție și calculează pagubele care ar decurge din decizia de a opri sau de a amâna execuția proiectelor respective. Pagubele se calculează în funcție de o serie de parametri (investiția neutilizată, ponderea în amenajare, efortul de investiție imobilizat, prioritatea funcțională, complexitate, prioritate pe termen lung, siguranța balanței apei, pagube cauzate subcontractanților etc.), atașându-le coeficienți de pondere relativă definiți de către decident;

— *submodelul 2* privește proiectele care se găsesc într-un stadiu preliminar și calculează utilitatea anticipată, ca un număr cu valoare între 0 și 1, în funcție de aportul lor sub aspect tehnic, de economicitate, nivelul de risc și de considerente generale. Parametrii cu diferite dimensiuni se *normalizează* printr-o relație de conversie, pentru comparabilitate.

După ierarhizarea preliminară, valoarea *utilității* fiecărui proiect se corectează, în cursul desfășurării procesului de decizie, pe măsura obținerii de precizări privind parametrii luați în considerare. În acest scop se folosesc probabilitatea *a priori*, precum și probabilitatea *a posteriori*, calculată cu ajutorul teoremei lui Bayes.

Pentru ca decidenții să poată „dialoga” direct cu calculatorul, s-au realizat dispozitive de afișaj cu posibilitate de reglare a regimului de afișare, în funcție de experiența utilizatorului,

3.7. Descompunerea, coordonarea și agregarea în conducerea optimală a unei mari rețele de alimentare cu apă [17]

Se prezintă problema de conducere optimală, în timp real, a unei rețele relativ mari de alimentare cu apă, care include peste 20 de rezervoare. Prin exploatarea particularităților acestui sistem, format din subrețele interconectate cu interconexiuni controlabile, este posibilă cooperarea acestora într-o structură ierarhică complexă pentru a realiza optimizare și reglare directă. Deși în prezent se asigură urmărirea funcționării printr-un sistem de teletransmisie, reglajul trebuie considerat manual, deoarece actul decizional se sprijină doar pe experiența operatorului.

În 1974, societatea SLEE (Société Lyonnaise des Eaux et de l'Éclairage) a decis să cerceteze posibilitățile oferite de teoria optimizării în vederea producerii unui pachet de programe care să permită luarea sistematică de decizii cu ajutorul calculatorului. Studii preliminare efectuate pe o subrețea de 3 rezervoare principale au dus la concluzii pozitive în 1977. Din punct de vedere matematic, s-au utilizat în principal idei de centralizare/descentralizare, care s-au transformat într-o procedură euristică iterativă, obținându-se rezultate satisfăcătoare (comparativ cu comportamentul simulat al operatorului uman).

Noul studiu comentat în articol a început în 1980 și intenționa să producă în final un pachet de programe operațional. În acest scop o firmă particulară de software (SINAC) a fost asociată cu proiectul. Software-ul care a rezultat este foarte modular și se poate adopta multor alte cazuri, deoarece rețeaua considerată oferă majoritatea dificultăților care se pot întâlni la problemele de acest nivel.

Problema a fost împărțită în subprobleme mai mici, fără a renunța la optimizarea globală, care depinde de asemenea de exploatarea optimă a interconexiunilor (pe lângă controlul optim al fiecărei subrețele). În ciuda înfrângerii, unele subprobleme rămân complexe și scumpe pentru a fi rezolvate de mai multe ori, așa cum este procesul de coordonare. S-a făcut deci apel la idei de centralizare pentru a simplifica rezolvarea acestor subprobleme.

Întregul procedeu de optimizare *indirectă* este efectuat în timpul nopții, pe un orizont de o zi, bazat fiind pe consumul de apă preliminar. Deoarece consumul real, evident, va diferi de valorile prevăzute, și pentru că există și alte inexactități ale modelului, este necesară de asemenea computerizarea *directă*. O parte din pachetul de programe pentru optimizarea *indirectă* poate fi utilizată direct pentru a compensa aceste abateri, dar cu un orizont limitat de unul sau de câteva intervale de timp, evitându-se astfel partea cea mai scumpă a computerizării, adică optimizarea *dinamică*.

Simulări de câteva zile s-au efectuat utilizându-se atît strategia computerizată prin program, cît și cea folosită de operator pentru zilele corespunzătoare. Toate rezultatele au fost în favoarea programului. Mai mult, strategia generată de program a dus la o bună comportare calitativă.

Deși nu se așteaptă mari economii pe termen scurt, principalele avantaje pe care le oferă software-ul sînt cele ale oricărui automat: performanțe uniforme și reproducibile, adaptare imediată la orice schimbare provizorie sau definitivă în rețea și de asemenea o mai bună înțelegere a limitărilor sistemului și deci un ajutor pentru planificarea viitoare, posibilitatea de a evalua mai rațional viitoarele investiții etc.

Pachetul de programe este planificat să devină operațional la începutul anului 1984. Sistemul se va utiliza ca un sistem de luare a deciziilor cu ajutorul calculatorului și va oferi următoarele date operatorului:

- optimizare *indirectă*, cu cererea prevăzută;
- optimizare *directă* de „închidere a gurilor“ (în fiecare oră);
- simulare directă a viitoarelor funcționări ale rețelei la cererea operatorului, cu strategii de reglare optimizate sau cu date *a priori*;
- înmagazinare de date și evaluare a consumurilor reale;
- evaluarea *a posteriori* a performanțelor indirecte ale strategiilor operatorului și calculatorului față de consumul real etc.

Pe lângă utilizarea curentă zilnică, softul realizat va putea fi utilizat în optimizări și simulări ale funcționării rețelei de alimentare cu apă în scopuri de planificare.

3.8. Optimizarea pompării în sistemele de alimentare cu apă [18]

Sistemele de alimentare cu apă tipice cuprind un mare număr de conducte interconectate, precum și ventile și pompe. Rețelele sînt prevăzute deseori cu capacitate de stocare sub forma unor rezervoare de serviciu. Date obținute din practică arată că o proporție semnificativă din cheltuielile totale de exploatare se datorește costului energiei electrice pentru pompare. Aceasta furnizează un motiv puternic pentru optimizarea operațiunilor de pompare. Local, obiectivul optimizării este de a selecta o combinație a pompării în scopul minimizării cererii de *putere* necesară. Global, obiectivul principal este de a determina strategiile ale pompării pentru minimizarea costurilor cu energia de pompare.

De aceea, în condiții specifice, problema generală a pompării optimizate în sistemele de alimentare cu apă se poate împărți în două direcții principale: utilizarea optimizată a pompării și programarea optimizată a pompării.

Utilizarea optimizată a pompării este o problemă de optimizare *statică*, în care se caută un regim adecvat al pompării, cu consum minim de putere, pentru a satisface condițiile instantanee de exploatare a sistemului.

Programarea optimizată a pompării este o problemă de optimizare *dinamică*, în care se ține seama de stocajul de apă din sistem și de variația în timp a tarifelor de electricitate, în scopul minimizării costurilor totale de exploatare într-o anumită perioadă de analiză. Ambele aspecte trebuie avute în vedere pentru studiul optimizării sistemelor de alimentare cu apă.

Determinarea unor scheme de acțiune îmbunătățite necesită modele corecte ale rețelelor de distribuție. Totuși, interacțiunea puternică între componentele neliniare ale rețelei și numărul mare de variabile ale sistemelor impun severe restricții teoretice și de computerizare în aplicarea tehnicilor de optimizare pentru analiza funcționării sistemelor. S-au făcut mai multe propuneri de optimizare a anumitor sisteme. În cele mai multe cazuri, s-au făcut presupuneri restrictive, care ori simplifică considerațiile de optimizare reținînd modelul general al sistemului, ori încearcă optimizarea completă dar cu o reprezentare simplificată a sistemelor.

Stările de pompare au, de obicei, combinații în paralel de pompe cu caracteristici diferite. Aceasta permite o reglare de tip *discret* sau o reglare de tip *mixt* cu variabile atît *discrete*, cît și *continue*. Obiectivul primordial în optimizarea pompării este de a determina combinații potrivite între aceste variabile. Formularea care rezultă formează baza de dezvoltare a procedurilor de programare optimizată; acestea sînt utilizate împreună cu tehnica de programare dinamică.

Programarea dinamică este utilizată ca procedeu de bază în optimizarea planificării pompării, cu selectarea combinațiilor de pompare bazate pe considerații de folosire a pompelor. Soluția definește acțiunile necesare atît *discrete*, cît și *continue*, corespunzînd combinațiilor de funcționare-nefuncționare ale pompelor și vitezelor de pompare.

Pentru calculul pompării optime momentane și a programului optim de pompare a fost elaborat un program de calculator pentru lucru *interactiv*, cu ilustrări grafice, program aplicat din anul 1982 la un sistem existent de alimentare cu apă.

Rezultatele obținute privesc în special:

- creșterea pompării în timpul nopții;
- utilizarea celor mai eficiente pompe;
- reducerea comutărilor de pe o pompă pe alta;
- limitări ale variației nivelelor din rezervoare;
- limitări ale puterii maxime cerute;
- diminuarea debitelor necesare la sursa de alimentare.

Se pot obține astfel reduceri ale costului de ordinul 5—10 %, reduceri care sînt semnificative avîndu-se în vedere că schemele tipice de alimentare cu apă din Anglia au costuri de exploatare anuale care depășesc 1 milion lire sterline.

BIBLIOGRAFIE

1. Y. Nakamori, Y. Sawaragi (Japonia) **Surveillance network design for air pollution control** (Proiectarea rețelelor de supraveghere pentru controlul poluării aerului). Indicativ 11.7-2.
2. R.P. Ibbitt, P.D. Hutchinson (Noua Zeelandă) **Model parameter consistency and fitting criteria** (Consistența parametrilor modelelor matematice în hidrologie — și criteriile de calibrare). Indicativ 11.8/B-3.
3. A. Szöllösi-Nagy, ș.a. (R.P.U.) **On the use of recursive algorithms in real-time river flow forecasting — Hungarian experiences** (Asupra folosirii algoritmilor recursivi în prognoza hidrologică în timp real — Experiența ungară). Indicativ 11.8/C-5.
4. A. Esogbue (S.U.A.) **Using fuzzy sets and hierarchical models in non point source water quality management** (Folosirea teoriei mulțimilor vagi și a modelelor ierarhizate la protecția calității apelor în cazul unor impurificări neconcentrate). Indicativ 11.8/A-6.
5. M. Cremer, J. Kappenberg (R.F.G.) **Surveilling water pollution in a tidal river by a state estimator with a switching structure** (Supravegherea poluării apei într-un riu influențat de fenomenele de maree, folosind un estimator de stare cu structură comutabilă). Indicativ 11.7-3.
6. M. Papageorgiou (R.F.G.) **Multilayer control of sewer networks** (Conducerea multinivel a exploatării rețelelor de canalizare). Indicativ 11.7-4.
7. M. Hiraoka, K. Tsumura (Japonia) **Modelling and control of the activated sludge process by use of autoregressive model** (Modelarea și controlul procesului cu nămol activ utilizînd modelul autoregresiv). Indicativ 11.8/C-1.
8. H. Kurz (R.F.G.) **Adaptive control of a waste water neutralization process** (Reglarea adaptivă a unui proces de neutralizare a apelor uzate). Indicativ 11.7-6.
9. K. Maeda (Japonia) **A knowledge based system for the wastewater treatment process** (Sistem bazat pe cunoaștere pentru conducerea procesului de epurare a apelor uzate). Indicativ 11.7-5.
10. M.B. Beck **Operational control and estimation in Water Quality Management** (Reglare și evaluare operativă în gospodărirea/protecția calității apelor).
11. Myron B. Fiering (S.U.A.) **Near optimality in management and physical models** (Cvasi-optimalitate în modelele de gospodărire a apelor și în cele fizice). Indicativ 11.8/B-1.
12. H. Pietkiewicz-Saldan (Polonia) **Analysis and physical interpretation of decision rules of operative water system management** (Analiza și interpretarea fizică a regulilor de decizie privind regimul de exploatare a unui sistem de gospodărire a apelor). Indicativ 11.8/C-6.
13. G. Guariso (It.), S. Orlovski (Austria), S. Rinaldi (It.) **A risk-averse approach for reservoir management** (Abordare de tip anti-risc în exploatarea lacurilor de acumulare). Indicativ 11.8/C-3.
14. L. Valadares Tavares (Portugalia) **Value of on-line hydrologic information for the operation of a multipurpose reservoir** (Utilizarea „on-line“ a informației hidrologice pentru exploatarea unui lac de acumulare în scopuri multiple). Indicativ 11.8/B-5.

15. *S.A. Orlovski* (Austria) **Fuzzy information in problems of resources allocation** (Informații cu caracter vag în probleme de alocare a resurselor). Indicativ 11.8/A-1.
16. *Y. Carmon, M. Ben-Bassat* (Israel) **A decision support system for large scale water development programs** (Sistem suport pentru decizii în programele de amenajare în domeniul apelor). Indicativ 11.8/C-2.
17. *P. Carpentier, C. Cohen* (Franța) **Decomposition, coordination and aggregation in the optimal control of a large water supply network** (Descompunerea, coordonarea și agregarea în conducerea optimă a unei mari rețele de alimentare cu apă). Indicativ 11.8/C-4.
18. *B. Coulbeck, G.H. Orr* (Anglia) **Optimized pumping in water supply systems** (Pompare optimizată în sistemele de alimentare cu apă). Indicativ 11.8/B-4.

APLICAȚII SPAȚIALE

Ing. M. Șirbu
I.P.A.

1. INTRODUCERE

Lucrările *secțiunii 12.0* se ocupă cu aplicații în domeniul astronauticii. Cunoașterea universului a reprezentat din totdeauna un domeniu de mare atracție pentru oamenii de știință, însă numai revoluția tehnologică actuală a făcut posibilă studiarea eficientă a acestuia. Ceea ce putem cunoaște despre univers depinde de mijloacele de observare și de metodele de măsurare pe care actuala dezvoltare tehnică ni le pune la dispoziție.

Limitele universului observabil se situează astăzi în jurul a 15 miliarde de ani lumină, ceea ce înseamnă că dacă am face o hartă a universului observabil pe care Luna s-ar găsi la 1 mm de Terra, această hartă ar acoperi o suprafață de 500 miliarde kilometri pătrați.

Cîteva din lucrările prezentate în cadrul secțiunii 12.0 analizează soluțiile la care s-a recurs pentru efectuarea observațiilor astronomice. Alte lucrări propun metode noi, mai eficiente, de investigare a spațiului cosmic. O problemă importantă abordată de lucrările acestei secțiuni se referă la poziționarea navelor cosmice, prezentându-se sistemele de reglare automată utilizate în acest scop, analizându-se măsura în care acestea satisfac necesitățile ridicate de cercetarea științifică a spațiului cosmic.

2. REGLAREA POZIȚIEI NAVELOR SPAȚIALE

Telescopul spațial descris în *lucrarea 12.0/A-1* este cel mai mare și mai puternic observator optic astronomic în ultraviolet care va funcționa în spațiu și care va permite obținerea unor imagini din spațiul cosmic de o calitate inegalabilă. Distanța nominală a orbitei pe care se situează telescopul pentru efectuarea observațiilor asupra spațiului cosmic este de 550 km. În lucrare sînt descrise și instrumentele complementare asociate observatorului, sistemul optic și sistemul de control al vizării.

Spre deosebire de telescoapele relativ mici care l-au precedat, telescopul spațial al NASA are gabaritul și durata de viață mult mai mari. Lansarea în spațiu, desfășurarea, deservirea și readucerea pe Pămînt a telescopului se vor realiza cu ajutorul sistemului de transport al navei spațiale. După lansare se manevrează pentru atingerea orbitei și poziției corespunzătoare, după care urmează separarea telescopului de sistemul de transport al navei spațiale. În timpul conectării telescopului la sistemul de separare, sau imediat după lansarea sa în spațiu, are loc desfășurarea antenei de înaltă frecvență și a aripilor solare și totodată sînt inițiate testările funcționale de sistem în scopul de a verifica dacă observatorul funcționează corespunzător. Din acest moment începe misiunea telescopului, iar sistemul de transport al navei spațiale este readus pe Pămînt. Deoarece treptat telescopul părăsește orbita stabilă, este necesară executarea unor manevre de menținere pe orbită. Acestea vor fi efectuate de către Orbiter în punctul de întîlnire. Este posibilă aducerea observatorului pe Pămînt pentru realizarea unor verificări amănunțite.

Telescopul spațial este o structură cilindrică cu lungimea de 13,1 metri, cu diametrul de 4,27 metri și cu greutatea de 11 000 kilograme. Elementele de bază din configurația telescopului spațial sînt ansamblul telescop optic (ATO), un modul de protecție a sistemului (MPS), instrumentele științifice (IS) și aripile solare (AS). Cele două aripi solare pot fi desfășurate sau retractate la comandă și pot fi rotite în jurul axelor transversale ale observatorului pentru menținerea poziției perpendiculare pe linia soarelui. ATO este format dintr-un telescop de tip Cassegrain, care este plasat în interiorul unui scut meteoric și al unei apărătoare contra soarelui. MPS este modulul de protecție, care cuprinde în interiorul său ATO și IS, asigurînd în același timp legătura între Orbiter și operațiile executate la sol. MPS furnizează elementele de bază de care depinde funcționarea corectă și precisă a observatorului. Instrumentele științifice sînt

plasate în planul focal, în spatele oglinzii primare și sînt menținute în poziție fixă. Telescopul are 5 instrumente științifice și un instrument pseudoștiințific, fiecare dintre acestea fiind un modul separat care poate fi deplasat în spațiu.

O problemă importantă abordată în cadrul lucrării se referă la reglarea vizării și stabilității. Elementele care condiționează proiectarea sistemului de reglare a stabilității depind de calitatea sistemului optic al telescopului spațial și de sensibilitatea instrumentelor științifice asociate telescopului. În cadrul metodei de proiectare pentru care s-a optat există două posibilități de reglare a poziției de funcționare: reglarea poziției în raport cu soarele (R.P.S.) și reglarea operațională a vehiculului (ROV). RPS este utilizată în timpul desfășurării inițiale a observatorului, cînd au loc testările de echipament înainte de misiune, precum și în timpul operațiilor neprevăzute. Modulul de reglare ROV asigură orientarea și menținerea axelor optice ale telescopului spre coordonate de interes științific, pentru efectuarea unor observații ce pot dura de la cîteva minute pînă la cîteva zile.

Telescopul spațial va fi un instrument puternic pe durata celor 50 sau mai mulți ani de viață și va fi accesibil oricărui cercetător științific calificat. Operațiile efectuate cu acest telescop vor fi în cea mai mare parte asemănătoare cu cele efectuate de alte centre astronomice cu baza la sol, exceptînd acele operații care se referă la conducerea unui telescop situat la distanță mare în spațiul cosmic.

Cercetările nu au fost încă definitivate, astfel încît, ținînd seama de structura sistemului telescopului spațial și de problemele ridicate de funcționalitatea acestuia, se impun atenției problemele ridicate de lucrarea 12.0/A-5, în ceea ce privește panourile solare și problemele ridicate de lucrările 12.0/A-2, 12.0/A-3, 12.0/A-4, 12.0/A-6 în legătură cu reglarea poziției. Deoarece se preconizează ca pe durata celor 50 de ani de viață telescopul să nu fie adus la sol, sau să fie adus foarte rar, o problemă importantă care se ridică este aceea de a asigura energia necesară pentru funcționarea întregului sistem pe această durată. Aripile cu baterii solare pot constitui o sursă de energie convenabilă.

Controlul flexibilității aripilor cu baterii solare constituie tema lucrării 12.0/A-5. Rolul acestor aripi este de a colecta în mod eficient energie solară prin controlul precis al orientării către soare. Limitările impuse în legătură cu greutatea rachetelor lansate în spațiu impun ca aripile să fie construite din materiale ușoare și foarte flexibile. Sistemul format de aceste aripi flexibile va fi un sistem cu parametri distribuți, deci va avea mai multe moduri de vibrație. Aripile sînt modelate ca un sistem în consolă flexibilă, vibrînd numai prin încovoiere, neglijîndu-se vibrațiile prin torsiune. Soluțiile formulării matematice nu sînt prezentate în această lucrare datorită spațiului limitat. Dinamica sistemului este analizată pe baza modelului în consolă al aripilor cu baterii solare.

În prezent se lucrează la o metodă de reglare pe baza estimării modurilor de vibrație ale aripilor. Există două tehnici de estimare corespunzătoare celor două tipuri de senzori care pot fi utilizați în spațiu: aparatul de măsurare a deformării la încovoiere și celulele instrumentului cu baterii solare. Cei doi senzori pot măsura momentele de încovoiere și respectiv unghiurile de deviație. Metoda de estimare propusă este cea corespunzătoare celui de al doilea tip de senzor, aceasta fiind mai adecvată condițiilor din spațiu.

O altă problemă tratată în lucrare se referă la reglarea maximală a ieșirii. Aceasta este una dintre cele mai importante probleme, deoarece este legată de orientarea precisă a aripilor în funcție de soare pentru a maximiza cantitatea de lumină ce cade pe aripă. Spre deosebire de metoda de reglare convențională, metoda de reglare modificată propusă în lucrare ia în considerare flexibilitatea aripii.

Fie S_i și S_{i+1} ieșirile aripii cu baterii solare la momentul i și respectiv după efectuarea unei mici rotații către soare. Schema de reglare statică convențională va fi:

- (1) Dacă $S_i \geq S_{i+1}$, atunci aripa se rotește în continuare în aceeași direcție, cu un unghi precris, pentru a mări cantitatea de lumină ce cade pe aripă;
- (2) În caz contrar, rotația se face în direcția opusă;
- (3) Se continuă rotația în sensul pentru care s-a luat decizia mai sus, pînă cînd ieșirea devine maximă.

Metoda modificată de reglare pornește de la estimarea, pe baza datelor obținute prin măsurarea cantității de lumină ce cade pe aripa nedeviată. Cu această estimăție, reglarea se implementează utilizînd exact același algoritim convențional descris mai sus.

O altă problemă importantă este ridicată de necesitatea de a asigura un sistem de reglare tolerant la defecte, care este absolut necesar atunci cînd se lucrează în spațiu. Se va utiliza conceptul de reglare duală cu detectarea defectului și diagnosticare. Dacă un senzor se defectează, defectul poate fi detectat și izolat în cadrul algoritmului de reglare, evitîndu-se în acest

fel ca reglarea să se facă pe baza unor informații greșite. Detectarea defectului și diagnostica carea sînt realizate cu un sistem de reglare biprocesor. Experimentele făcute au dat rezultate bune. La apariția unui defect sistemul de reglare va lucra în mod recuperativ, adică reglarea se modifică astfel încît să anuleze efectul produs de apariția defecțiunii.

În lucrarea 12.0/A-5 s-a pus în evidență necesitatea de a lua în considerație elasticitatea elementelor constructive ale navei spațiale. Această problemă este abordată și în cadrul lucrării 12.0/A-6, aici însă centrul de interes cade pe proiectarea regulatorului care controlează poziția navei spațiale cu structură elastică. Proiectarea se realizează pe baza metodelor frecvențiale. Se începe cu descrierea sistemului prin ecuații diferențiale parțiale cu model de amortizare intern. Structura elastică este constituită din doi arbori elastici atașați de o parte și de alta a unui disc rigid. Funcția de transfer care descrie comportarea acestei structuri este următoarea :

$$\theta_1 = [1/s^2 (I_1 + I_2 + I_3)] G_m(s) - T_1 \quad (1)$$

unde :

$$G_m(s) = \left[1 + \sum_{i=1}^3 \delta_i \right] / \left[1 + \sum_{i=2}^3 \delta_i (th Y_{1s}) / Y_{1s} \right] / Y_{1s}$$

este funcția de transfer a sistemului mecano-elastic,

- $Y_1 = \sqrt{I_1 [K_1 + s d_1(s)]}$ și $\delta_i = I_i / I_1$ — raportul de inerție ;
 I_1 — momentul de inerție al discului rigid [kgm^2] ;
 I_i — momentul de inerție total al fiecărui arbore [kgm^2], $i=2,3$;
 T_1 — torșiunea de reglare exercitată asupra discului [N] ;
 θ_1 — unghiul de rotație al arborelui flexibil.

Acest model poate fi adus la următoarea formă mai simplă :

$$s^2 I_1 \left[1 + \sum_{i=2}^3 \delta_i (th Y_{1s}) / Y_{1s} \right] \theta_1 = T_1 \quad (2)$$

În lucrare se face analiza răspunsului frecvențial și se definesc cele trei caracteristici de bază ale sistemului elastic, și anume : funcția medie, marginea superioară și marginea inferioară.

Pentru determinarea funcției medii se pleacă de la termenul tangentei hiperbolice $G_1(s)$. Avem :

$$G_1(s) = th (Y_{1s}) / (Y_{1s}) \quad (3)$$

unde :

$$Y_1 = \sqrt{I_1 [k_1 + s d_1(s)]} = (1/\omega_1) \sqrt{1 + \{\zeta_1(s)/\omega_1\} s}$$

Caracteristicile frecvențiale ale termenului tangentei hiperbolice vor fi :

$$\omega_1 = \sqrt{k_1 / I_1} ; \quad \zeta_1 = \{d_1(s)/2\} \sqrt{k_1 / I_1}$$

unde $d_1(s)$ reprezintă constanta de amortizare

$G_1(s)$ se poate aproxima printr-o funcție mai simplă astfel :

$$G_1(s) = \lim_{I_1 \rightarrow 0} 1/G_m(s) \text{ pentru un singur arbore}$$

$G_1(s)$ este reciproca funcției de transfer a sistemului mecano-elastic pentru cazul în care discul este mic și se atașează un singur arbore la disc. Structura elastică va fi descrisă prin combinarea a citorva sisteme cu parametri distribuți de tipul lui $G_1(s)$. Se obține funcția de transfer a structurii elastice care este exprimată prin intermediul funcției medii, marginii superioare U_a („upper bound“) și marginii inferioare L_a („lower bound“). Proiectarea regulatorului se va face pe baza acestor caracteristici generale. Înainte de a începe proiectarea, structura elastică ce trebuie reglată va fi analizată din 4 puncte de vedere și anume :

- (1) dacă co-locăția este sau nu satisfăcută ;
- (2) dacă frecvența de rezonanță este suficient de mare, este de același ordin, sau este de ordin mai mic față de lățimea de bandă a sistemului de reglare ;
- (3) dacă raportul de inerție este mare sau mic ;
- (4) dacă reglarea arborelui este sau nu necesară.

Există trei categorii de reglatoare :

- cu compensarea amplitudinii ;

- cu compensarea fazel în cadrul lăţimii de bandă a elementului de execuţie şi compensarea amplitudinii în afara acesteia ;
- cu dublă compensare a avansului de fază.

În lucrare se indică situaţii în care se recurge la una sau alta dintre tipurile de reglare menţionate mai sus, această alegere fiind făcută în funcţie de rezultatele analizei structurii elastice ce urmează să fie reglată. Se constată că sarcinile de reglare a structurii elastice se împart astfel : sarcini rezolvate de reglarea activă, care acoperă lăţimea de bandă a elementului de execuţie şi sarcini rezolvate de reglarea pasivă, care acoperă domeniul situat în afara lăţimii de bandă a elementului de execuţie. Din acest motiv caracteristicile de amortizare ale materialelor vor avea aceeaşi importanţă ca şi regulatorul şi vor fi tratate ca parte a regulatorului.

Această metodă de proiectare, bazată pe caracteristicile generale ale structurii elastice este foarte robustă şi are avantajul insensibilităţii la variaţii ale parametrilor. Exemplul de proiectare prezentat în lucrare pune în evidenţă că în cazul oscilaţiilor datorate torsiunii, compensatorul optim va cuprinde o buclă de viteză şi un filtru trece-jos.

Se preconizează ca, pe viitor, legătura dintre proiectantul structurii şi proiectantul sistemului de reglare să se realizeze prin intermediul funcţiei medii $G_1(s)$ şi a caracteristicilor de amortizare $\zeta_1(s)$ ale matricelor.

Probleme legate de reglarea poziţiei mai sînt tratate în cadrul *lucrărilor 12.0/A-2, 12.0/A-3 12.0/A-4*. Acest grup de lucrări se ocupă însă cu reglarea poziţiei unor sateliţi, spre deosebire de celele lucrări care aveau în vedere aceeaşi problemă, însă pentru cazul navelor spaţiale. *Lucrarea 12.0/A-2* propune o metodă de reglare a poziţiei unui satelit astrometric cu rotire stabilizată, pe baza schemei cu cuplu de torsiune magnetic, iar *lucrarea 12.0/A-3* propune reglarea numerică în buclă închisă a poziţiei utilizînd jeturi de gaze.

Legea de reglare a poziţiei pe baza cuplului de torsiune magnetic propusă în *lucrarea 12.0/A-2* se bazează pe o reacţie lineară după abaterea de poziţie şi asigură atât reglarea vitezelor unghiulare cît şi amortizarea nutaţiei. Lucrarea pune în evidenţă posibilitatea de a utiliza în mod avantajos sistemul pe bază de momente de polarizare în combinaţie cu cuplu de torsiune magnetic pentru sateliţii a căror poziţie este reglată cu precizie mare în raport cu un sistem de referinţă inerţial. Se mai propune o nouă schemă pentru manevrele cu unghiuri mari, schemă care se bazează de asemenea pe principiul cuplului de torsiune magnetic. Această schemă încearcă să realizeze o deplasare cuasistatică a referinţei de poziţie astfel încît să determine o schimbare a poziţiei în jurul axei de rotaţie. Se presupune că senzorii de poziţie la bord sînt giroscopice integrate de viteze şi magnetometre pe 3 axe. Se mai presupune că la bord există echipament cu microprocesoare. Este important de subliniat că schema pentru manevrele cu unghiuri mari se caracterizează prin simplitatea hardware-ului la bord şi a implementării software, cantitatea de prelucrări pe care trebuie să le realizeze calculatorul de la bord fiind mică. Analiza stabilităţii legii de reglare a poziţionării se face prin metoda locului rădăcinilor.

S-au realizat simulări pe calculator pentru cazul satelitului ASTRO-C, care urmează să fie lansat în februarie 1987. Rezultatele pun în evidenţă faptul că stabilitatea sistemului este asigurată şi că precizia de poziţionare se poate obţine cu ajutorul unui echipament relativ simplu la bord. Rezultatele simulării indică de asemenea că schema propusă pentru manevrele cu unghiuri mari funcţionează satisfăcător şi că este corespunzătoare pentru aplicarea în cazul sateliţilor cu rotire stabilizată.

Lucrarea 12.0/A-3 propune o metodă de reglare numerică în buclă închisă a poziţiei unui satelit astrometric de tip Hipparcos, acţionarea fiind asigurată pe bază de jeturi de gaz.

Satelitul Hipparcos, care urma să fie lansat pe o orbită geostaţionară la sfîrşitul anilor 1980, este destinat să observe 100 000 de stele cu ajutorul a două telescoape PT şi FT, care utilizează în comun acelaşi plan focal. Scopul misiunii Hipparcos este acela de a construi un vast catalog al corpurilor cereşti, cu o limită de precizie de 0,002" secunde de arc ; (1 arc sec = 1°/3 600).

Înainte de abordarea principalelor aspecte ale problemei de reglare, în lucrare se prezintă cîteva generalităţi în legătură cu structura şi mişcarea satelitului Hipparcos, şi anume se face o scurtă prezentare a sistemului optic, a modului în care decurge observarea corpurilor cereşti şi mişcarea nominală a satelitului.

Problemele ridicate de reglarea poziţiei se referă la estimarea poziţiei, alegerea elementelor de execuţie şi la legea de reglare în buclă închisă. Procedura de observare în timp real a imaginilor corpurilor cereşti impune ca estimarea poziţiei să se facă la bord, în mod continuu.

cu o precizie de 1 arc sec. Pentru această problemă, un grup de autori (Carlucci și Donati) propune un algoritm care satisface în același timp și cerințele legii de reglare. La alegerea elementelor de execuție trebuie să se țină seama de faptul că nu se admit surse de vibrație care să producă perturbații de poziție persistente cu o amplitudine mai mare de câteva miliardecunde. Mai mult, este necesar ca evoluția poziției să poată fi modelată cu o precizie de câteva miliardecunde cu ajutorul unui model matematic caracterizat de cei mai mici parametri. În aceste condiții, cea mai convenabilă soluție de element de execuție este cea bazată pe jeturi de gaz, deoarece sînt eliminate vibrațiile care apar între două activări consecutive ale limitatorului de cursă.

La alegerea legii de reglare trebuie să se țină cont de următoarele cerințe :

- (i) activarea limitatoarelor de cursă să fie simultană pentru axele controlate ;
- (ii) legea de reglare operează cu interval de timp T constant între două manevre ;
- (iii) numărul N de manevre de reglare să fie cît mai mic pe durata unei perioade corespunzătoare parcurgerii unei orbite.

Aceste cerințe impun alegerea unei structuri de reglare numerică în buclă închisă cu perioada de eșantionare T fixată și cît mai mare posibilă. Scopul sistemului de reglare a poziției este acela de a menține eroarea de poziționare a axei Z și eroarea mișcării de spin în limitele de toleranță precise în condițiile în care asupra satelitului acționează perturbații necunoscute datorate momentului de torsiune, despre care se știe că au un caracter periodic. Principala problemă care apare în proiectare constă în evitarea efectului erorilor care apar datorită eșantionării cu T mare a răspunsului sistemului la aplicarea unei reglări cu momente de torsiune în impulsuri. Soluția originală pe care o dau autorii pentru rezolvarea acestei probleme constă în însumarea semnalelor de referință pentru poziție cu efectele calculate asupra poziției satelitului, efecte care apar datorită unor componente de înaltă frecvență ale impulsurilor de reglare. Amplificarea pe reacție datorată acestor operații va fi luată în considerare la proiectarea legii de reglare generală.

Se impune ca legea de reglare aleasă, aplicată în prezența unor perturbații periodice necunoscute, să asigure următoarele performanțe :

- eroarea absolută de poziționare a axei Z să fie mai mică de $10'$ (minute de arc) ;
- eroarea vitezei unghiulare de spin să fie mai mică de 2% ;
- eroarea unghiulară a axelor de spin, în raport cu mișcarea nominală, să fie mai mică de $10'$;
- să avem valoare mare a lui T într-un domeniu corespunzător lui $N=8, 10, 12$, în condiții de operare normale.

Perturbațiile datorate momentului de torsiune, care apar în condiții normale de funcționare, apar datorită presiunii solare și giroscopelor.

Admițînd ipoteza simetriei cilindrice în raport cu axa Z și a unor perturbații mici pe timpul mișcării nominale, ecuațiile dinamice ale satelitului se descompun după următoarele mișcări independente :

- rotația de spin Ψ în jurul axei Z
- interacțiunea mutuală φ și rotațiile θ în jurul axelor X , respectiv Y .

S-a proiectat cite o lege de reglare pentru fiecare dintre aceste mișcări. Reglarea lui Ψ este mai simplă decît reglarea unghiurilor (φ, θ), deoarece în primul caz sistemul dinamic este de ordin mai mic, iar perturbațiile datorate momentului de torsiune au un număr mai mic de armonice.

În continuare se face o prezentare comparativă a performanțelor în buclă închisă evaluate prin simulare pe calculator, cu performanțele legii de reglare în buclă deschisă, care au fost deduse admițînd ipoteza că perturbațiile sînt perfect cunoscute. Rezultatele obținute, precum și faptul că reglarea optimală în buclă deschisă se sprijină pe o ipoteză care în condițiile date nu este valabilă, pun în evidență faptul că metoda de proiectare propusă în lucrare este cea mai bună metodă aplicabilă în practică.

Un obiectiv similar cu cel al lucrării 12.0/A-3 are lucrarea 12.0/A-4, care preconizează întocmirea unui catalog stelar de mare precizie al întregului cer. Lucrarea propune observarea timp de 2.5 ani a 100 000 de corpuri cerești, cu ajutorul unui telescop special, montat pe un satelit aflat pe o orbită staționară în jurul Pământului. Datele furnizate de telescop vor fi prelucrate la sol.

Scopul misiunii Hipparcos este de a determina parametrii astrometrici : poziții (longitudine și latitudine), mișcări particulare (în longitudine și latitudine) și paralaxă trigonometrică pentru fiecare corp ceresc observat.

Observațiile asupra stelelor vor fi efectuate cu ajutorul unui instrument optic special, format din două telescoape care fixează cerul ca o pereche de compase, și care au un singur plan focal și un tub proiector de imagine. Datele colectate de acest instrument vor fi transmise la sol, iar prelucrarea lor va fi efectuată de două institute științifice independente. Rezultatul acestei prelucrări va fi Catalogul Hipparcos, care cuprinde estimarea a 5 parametri astrometrici pentru fiecare stea din cele 100 000 care apar în catalog.

Observațiile se efectuează prin baleierea continuă și sistematică a întregului cer de către telescopul dublu, care va măsura cu precizie unghiul dintre două stele îndepărtate. Prin intermediul unui set complex de oglinzi, telescopul va reflecta două câmpuri de vedere asupra cerului separate printr-un unghi γ , în ansamblul planului focal unic care conține o grilă periodică și un tub proiector de imagine. Tubul este situat dincolo de planul focal și are o arie mică de sensibilitate, permițând observarea unei singure stele, din mulțimea de stele vizibile simultan prin intermediul telescopului CVI, care se mută rapid de la o stea la alta în cele două câmpuri de vedere ale telescopului pentru a face observații simultane. Prelucrarea datelor necesită transformarea coordonatelor corpului ceresc, pentru trecerea din planul focal al telescopului într-un sistem de referință inerțial obișnuit. Pentru a realiza corect această transformare, trebuie reconstituită poziția satelitelui în raport cu cele 3 axe pe baza datelor furnizate de senzorii de poziție de la bord: cartograful stelar și giroscopale. Reconstituirea poziției este formulată ca o problemă de estimare de tip Gauss-Markov, care va fi rezolvată numai pe baza prelucrării datelor furnizate de cartograful stelar.

Reconstituirea poziției a fost testată prin simularea numerică a dinamicii satelitelui Hipparcos și aplicarea a 10 pulsuri de reglare pe fiecare rotație de spin. Poziția a fost modelată pe baza răspunsurilor date de sistem la impulsurile de reglare aplicate, și utilizând serii Fourier de ordinul 20. Estimatorul Gauss-Markov a fost implementat utilizând tehnica Householder. Pentru prelucrarea unui set de date obținute în urma unor observații de durată mai lungă (mai mult de o rotație de spin), se vor folosi serii Fourier cu număr mai mare de termeni, acesta fiind un instrument pentru dezvoltarea ulterioară a acestei teme de cercetare științifică.

Rezultatele finale pun în evidență precizia foarte bună cu care se realizează reconstituirea poziției pe baza estimății Gauss-Markov.

3. ALGORITMI ȘI SOFTWARE PENTRU APLICAȚII SPAȚIALE

Lucrările prezentate pînă acum au abordat probleme legate de reglarea poziției navei spațiale. Un alt grup de lucrări, care abordează de asemenea probleme din domeniul aplicațiilor spațiale, este constituit de lucrările care au fost prezentate în cadrul secțiunii 12.0/B. Acestea prezintă algoritmi și software pentru aplicații spațiale.

Astfel, *lucrarea 12.0/B-1* se ocupă cu problema reglării antenelor destinate transferului de date prin intermediul sateliților și se prezintă modul în care se face estimarea elementelor orbitale cu ajutorul filtrului Kalman.

Aceste antene asigură urmărirea semnalului luminos al satelitelui cu ajutorul unui receptor urmăritor, lucrînd în modul de urmărire automată. Antenele pentru transmisie-recepție de date prin intermediul sateliților artificiali ai Pămîntului sînt proiectate sub forma unui reflector parabolic montat astfel încît să se poată roti în jurul a două axe (azimut și elevație). Datorită lățimii reduse a fasciculului de raze, antena trebuie poziționată cu o precizie foarte mare. De aici decurge necesitatea de a asigura un grad mare de rigiditate a antenei și o reglare exactă a poziționării, obținîndu-se în acest fel o stabilitate bună în raport cu factorii perturbatori (vîntul etc.).

O problemă tehnică importantă a constituit-o sistemul de reglare pentru poziționarea antenei. Inițial antenele erau echipate cu un servosistem numeric pentru reglarea exactă a poziționării, dar recent a fost adăugat un nou calculator, denumit calculator orbital.

Antena trebuie să asigure localizarea satelitelui la orizont, urmărirea acestuia pe orbită și regăsirea satelitelui dacă este pierdut din vedere datorită unor perturbații. Pentru aceasta antena este prevăzută cu următoarele moduri de operare:

- 1) PRESET — antena este adusă într-o poziție de referință pe baza unui algoritm optimal în timp.
- 2) RATE — antena primește referință de viteză pentru fiecare dintre cele două axe (azimut și elevație), pînă la apariția unei comenzi de stop.

- 3) SEARCH — antena face mișcări în spirală în jurul unei poziții de pornire în scopul de a localiza satelitul.
- 4) AUTO-TRACK — în conformitate cu semnalul luminos sosit de la satelit, antena va primi informații exacte în legătură cu deviația dintre propria sa poziție și cea a satelitului. Pe baza acestor informații se asigură închiderea unui ciclu de reglare pentru urmărirea antenei.
- 5) PROGRAM-TRACK — antena urmărește o curbă de poziție stabilită de către calculatorul gazdă cu ajutorul unei funcții de timp.

Comanda antenei este realizată prin așa-numitul servocalculator. Modurile de operare ale antenei pot fi selectate de la un panou de comandă, sau prin intermediul calculatorului gazdă. Valorile dorite pot fi introduse manual. Transferul datelor de PROGRAM-TRACK de la calculatorul gazdă la servocalculator se face cu ajutorul unei interfețe.

Calculatorul gazdă are posibilitatea de a obține în fiecare moment date referitoare la poziția antenei iar în modul de operare AUTO-TRACK, poate obține date referitoare la poziția satelitului. Calculatorul gazdă poate de asemenea să precalculeze poziția satelitului utilizând modelul orbitei și să furnizeze aceste valori servocalculatorului sub forma datelor de PROGRAM-TRACK. Combinarea acestor două senzuri de transfer permite operarea autonomă a antenei. În acest caz, calculatorul conectat la servocalculator se va numi calculator orbital.

Modelul orbital folosește un set de elemente orbitale care se pot modifica în timp. Dacă la un moment dat sînt cunoscute elementele orbitale, calculatorul orbital va putea calcula poziția corespunzătoare a satelitului cu ajutorul modelului orbitei. Cînd antena a localizat satelitul, pozițiile măsurate ale azimutului și elevației vor fi utilizate pentru corectarea elementelor orbitale cu ajutorul filtrului Kalman.

Orbita satelitului este caracterizată de următoarele elemente orbitale: înclinarea, ascensiunea dreaptă, argumentul periheliului, anomalia medie în raport cu perioada, axele principale și excentricitatea elipsei pe care se deplasează satelitul.

În modul de operare AUTO-TRACK se asigură în timp real îmbunătățirea continuă a elementelor orbitale cu ajutorul unui filtru Kalman. Se furnizează în acest fel datele de referință necesare pentru poziționarea antenei. Un astfel de calcul al datelor de referință devine necesar în cazul în care se cercetează orizontul în vederea localizării satelitului, cînd orbita satelitului este foarte aproape de zenitul antenei, precum și în cazul în care trebuie asigurată redundanța sistemului de urmărire automată. Programul a fost testat pe un minicalculator.

Lucrarea 12.0/B-2 se ocupă cu formularea filtrării prin metoda rădăcinilor de ordinul 2 pentru navigarea autonomă a satelitului.

Problema de determinare a efemeridelor și poziției sateliților orbitali ai Pămîntului este formulată ca o problemă de filtrare cu rădăcini de ordinul 2 utilizîndu-se pentru aceasta un vector de baleiere dinspre satelit spre PCS (puncte de control la sol). Acest vector este obținut pe baza datelor furnizate de imaginile detectate la mare distanță. Efemeridele și poziția sateliților orbitali ai Pămîntului au fost estimate pe baza datelor furnizate de stațiile de urmărire radar de la sol, de urmăriitorii de stele, de senzorii solari și/sau tereștri etc. Precizia de estimare, este mai mare în cazul în care sînt disponibile date furnizate de imagini detectate la distanță, PCS sînt puncte de reper cu localizare precis cunoscută pentru datele furnizate de imagini detectate la mare distanță și sînt utilizate pentru corectarea geometrică a acestora. Procesul de găsimă a PCS din datele detectate la mare distanță, poartă numele de armonizare a datelor. După ce s-a efectuat armonizarea datelor, se obține determinarea unui vector de baleiere pe baza căruia se va realiza o metodă de navigare autonomă.

În lucrare este prezentată o metodă de estimare a stării utilizînd informația PCS obținută pe baza datelor detectate la mare distanță în conjuncție cu ieșirile giroscopelor de la bord. Se face analiza performanțelor sistemului propus pentru diferite poziții ale PCS. Pentru astfel de probleme se poate utiliza filtrul Kalman extins. În lucrare se utilizează un algoritm al rădăcinilor pătrate, deoarece algoritmi de filtrare linearizați tind să devină instabili. Rezultatele obținute pun în evidență faptul că filtrul poate avea performanțe foarte bune și că algoritmul rădăcinilor pătrate este avantajos din punct de vedere numeric.

Lucrarea pune în evidență faptul că desfășurarea PCS în imaginea baleiatorului multispectral are un efect considerabil asupra preciziei de estimare. Metoda prezentată poate fi utilizată în diferite probleme de navigare autonomă a sateliților.

4. CONCLUZII

În cadrul secțiunii 12.0 s-au prezentat lucrări ale căror teme constituie aplicații în domeniul astronauticii.

Lucrările prezentate în cadrul primei sesiuni 12.0/A au abordat probleme legate de reglarea poziției navelor spațiale. Astfel lucrarea 12.0/A-2 propune un sistem de reglare automată a poziției unui satelit astrometric cu rotire stabilizată, pe baza schemei momentului de torsune magnetic, în timp ce lucrarea 12.0/A-3 prezintă un sistem de reglare numerică în buclă închisă a poziției unui satelit astrometric, acționarea fiind asigurată pe bază de jeturi de gaz.

Lucrările 12.0/A-5 și 12.0/A-6 tratează probleme legate de reglarea poziției pentru cazul particular al navelor spațiale cu structură elastică.

Lucrarea 12.0/A-4 are ca temă problema reconstituirii la sol a poziției unor corpuri cerești pe baza datelor colectate în timpul unei misiuni în spațiu. Se prezintă și instrumentul optic utilizat pentru observarea corpurilor cerești, în timp ce lucrarea 12.0/A-1 are ca temă principală prezentarea unui nou observator astronomic de mare capacitate și durată mare de viață.

În cadrul sesiunii 12.0/B s-au prezentat algoritmi și software pentru aplicații spațiale.

Lucrarea 12.0/B-1 tratează problema reglării unei antene destinate transferului de date prin intermediul sateliților, prezentându-se și aspecte legate de estimarea elementelor orbitale cu ajutorul filtrului Kalman. Lucrarea 12.0/B-2 prezintă formularea filtrării prin metoda rădăcinilor de ordinul 2 pentru navigarea autonomă a sateliților.

Lucrările prezintă realizări practice sau experimentale de laborator care dau rezultate concludente cu privire la eficacitatea soluțiilor propuse. Se face de asemenea o testare a rezultatelor prin metode de simulare pe calculator.

Ciclul
„Service pentru calculatoare”

TESTĂRI AUTOMATE PENTRU MINICALCULATORI

Ing. *Toma Geber*
(Coordonarea ciclului)
Ing. *Ioana Dobre*
Ing. *Camelia Busuioc*
Ing. *Dan Horhoianu*
Ing. *Mariana Niță*
Ing. *Gina Olaru*
(I.T.C.)
Ing. *Mihai Stănculescu*
(I.I.R.U.C.)

I, SISTEM DE GENERARE DE SECVENȚE DE TEST (SGST)

CUPRINS

1. Introducere
2. Descrierea SGST
 - 2.1. Funcțiile SGST
 - 2.2. Structura SGST
 - 2.3. Programele utilizator
 - 2.4. Gestionarea memoriei
 - 2.5. Alte facilități de scriere a programelor
 - 2.5.1. Switch-uri de operare
 - 2.5.2. Operanți predefiniți
 - 2.6. Raportarea erorilor
3. Componente SGST
 - 3.1. Monitorul
 - 3.1.1. Executivul
 - 3.1.2. Procesul de comenzi
 - 3.1.3. Compilatorul
 - 3.1.4. Dispecerul
 - 3.2. Rutine de control
 - 3.2.1. Rutina de control RK05
 - 3.2.2. Rutina de control TM02
 - 3.2.3. Rutina de control RX11/RX01
 - 3.2.4. Rutina de control DH11
 - 3.2.5. Rutina de control DL11
 - 3.2.6. Rutina de control LP11
4. Procedura de lansare în execuție și operare
 - ANEXA A Comenzile SGST
 - ANEXA B Instrucțiuni de uz general pentru scrierea programelor sursă
 - ANEXA C Exemple de secvențe de test
 - ANEXA D OPSW
 - ANEXA E Operanți predefiniți

1. INTRODUCERE

Sistemul de Generare de Secvențe de Test (SGST) este un instrument comod și util pentru întreținerea și depanarea echipamentelor periferice ale minicalculetoarelor din familiile INDEPENDENT și CORAL. Pund la dispoziția utilizatorului un set puternic de comenzi și un

limbaj flexibil și ușor de învățat, sistemul permite definirea, introducerea și executarea diferitelor secvențe de test pe care depanatorul și le dorește.

SGST poate fi inclus în sistemele de testare din familia XXDP, cu care este perfect compatibil. Față de programele de test autonome care testează un singur echipament la un moment dat, SGST prezintă avantajul că poate gestiona inițierea de schimburi simultane pe mai multe periferice, simulând condiții de test foarte apropiate de cele din exploatarea reală. De fapt, este unicul instrument cu care se pot scrie, simplu și rapid, secvențe de test specifice unei situații reale.

Printre alte avantaje, menționăm faptul că SGST se dovedește deosebit de util în depanarea echipamentelor periferice ale minicalcutoarelor lipsite de becuri și switch-uri pentru date și adrese pe panoul frontal. Sistemul permite atât definirea unor bucle de test scurte, destinate vizualizării pe osciloscop a unor semnale, cât și scrierea unor programe mai complexe, care să interacționeze, pentru același periferic sau pentru periferice diferite, punând în evidență defecte mai subtile, care se manifestă sub forma conflictelor de acces.

Configurația hardware minimală necesară folosirii SGST constă din :

- unitate centrală ;
- memorie ;
- consolă ;
- periferic suport pentru SGST (disc, bandă magnetică) ;
- echipamente periferice de testat.

2. DESCRIEREA SISTEMULUI DE GENERARE DE SECVENȚE DE TEST

2.1. Funcțiile SGST

Pentru introducerea secvențelor de test, SGST furnizează utilizatorului un limbaj compus din instrucțiuni de uz general și instrucțiuni de Intrare-Ieșire de nivel înalt.

Cu ajutorul acestora din urmă se pot scrie secvențe de test specificându-se numai funcția de I/E dorită, controlul la nivelul interfeței perifericului rămânând transparent utilizatorului. Cu ajutorul instrucțiunilor de uz general, utilizatorul poate exercita efectiv controlul asupra echipamentului, la nivel de interfață.

Pe lângă acest limbaj utilizatorul dispune de un set de comenzi care îi permit folosirea funcțiilor SGST :

1. — introducerea programelor sursă ;
2. — compilarea programelor sursă ;
3. — executarea programelor obiect ;
4. — modificarea programelor sursă ;
5. — salvarea programelor sursă pe un suport extern, în vederea utilizării lor ulterioare ;
6. — încărcarea programelor sursă de pe un suport extern pentru a fi modificate, executate etc.

2.2. Structura SGST

Structura SGST este următoarea :

1. — monitorul SGST, constând din :
 - a) — executiv ;
 - b) — procesor de comenzi ;
 - c) — compilator ;
 - d) — dispecer ;
2. — rutine de control specifice echipamentelor periferice.

2.3. Programele utilizator

Un exemplu de program utilizator este prezentat în ANEXA C.

Limbajul folosit pentru scrierea programului constă din instrucțiunile de uz general (ANEXA B) și instrucțiunile de Intrare-Ieșire specifice echipamentului.

Numerotarea liniilor este utilă pentru modificări ulterioare și pentru salturi în program. Utilizatorul poate numerota linia în mod explicit sau SGST va incrementa automat numărul liniei cu 10, la introducerea unui blank drept prim caracter.

Programele se individualizează printr-un număr și, opțional, prin nume.

Fiecare program își exercită controlul asupra unui singur periferic (la un moment dat) și este tratat ca o entitate separată de către SGST.

Programului îi pot fi asignate mai multe echipamente de același tip și, printr-o singură comandă RUN, testul va fi executat pentru toate echipamentele, în mod secvențial.

2.4. Gestionarea memoriei

SGST asigură posibilitatea executării simultane a 16 programe utilizator prin încărcarea acestora în 16 partiții distincte, a căror dimensiune este stabilită dinamic.

Structura de principiu a unei partiții :

Tabela Informații Program (TIP)
Rutina de control a echipamentului (RCE)
Zona de Date
Program utilizator format sursă
Program utilizator format executabil

TIP (Tabela Informații Program) conține :

1. — vectorul de stare al programului (PFLGWD) ;
2. — switch-urile de operare pentru program (POPSW) ;
3. — numărul și numele programului ;
4. — tipul echipamentului ;
5. — numărul echipamentului curent testat ;
6. — numerele echipamentelor asignate spre a fi testate ;
7. — celulele de lucru TEM0-TEM15 ;
8. — alte celule de lucru.

Rutina de control a echipamentului îndeplinește funcția unui driver, permițând execuția instrucțiunilor de Intrare-Ieșire de nivel înalt, specifice perifericului. În plus, asigură afișarea erorilor (prin mesaje explicite), a registrelor de stare și a unor informații statistice.

Deoarece multe instrucțiuni de Intrare-Ieșire nu depind de o anumită zonă de memorie, SGST asigură două buffer-e de date cu nume simbolice predefinite (RDIO -citire, WRIO -scriere), localizate în zona de date din partiție. Dimensiunile buffer-elor au valoarea implicită de 256 bytes, dar pot fi schimbate, în momentul introducerii programului, cu orice valoare între 0 și 32776.

Utilizatorul mai dispune de o altă zonă de I/E, cunoscută sub numele de FREE, care se extinde de la sfârșitul SGST până la sfârșitul memoriei. Ea se folosește pentru operații de I/E de mare anvergură și ca zonă comună pentru transfer de date între programe diferite (vezi exemplul din ANEXA C).

2.5. Alte facilități de scriere a programelor

2.5.1. SWITCH-URI DE OPERARE

Switch-urile de operare, grupate într-un vector numit OPSW, se referă la funcții comune tuturor programelor scrise cu SGST :

- controlul execuției programului în diverse situații (în caz de eroare, când programul nu s-a executat complet, când programul este întrerupt etc.);
- controlul tipării mesajelor (mesaje de eroare, informații statistice);
- execuția unor funcții speciale (lucrul în modul maintenance etc.).

Semnificația biților OPSW este dată în ANEXA{D.

2.5.2. OPERANZI PREDEFINIȚI

SGST pune la dispoziția utilizatorului posibilități de generare a unor cîmpuri de date, de lungime variabilă, prin folosirea diferitelor combinații de operanzi predefiniți, identificați prin nume simbolice (ANEXA E) sau prin folosirea operandului RANDOM.

2.6. Raportarea erorilor

Mesajele de eroare pot fi grupate în trei categorii :

1. Mesajele de eroare ale procesorului de comenzi, care se referă la :
 - a) erori de sintaxă ale comenzilor ;
 - b) erori detectate în timpul execuției comenzilor ;
 - c) erori detectate în timpul execuției programelor.
2. Mesajele de eroare ale compilatorului, care se referă la sintaxa incorectă a liniilor din programul sursă al utilizatorului.
3. Mesajele de eroare ale rutinelor de control, care se referă la :
 1. erori de date ;
 2. erori propriu-zise de Intrare-Ieșire.

3. COMPONENTELE SGST

3.1. Monitorul

3.1.1. EXECUTIVUL

Executivul realizează următoarele :

- inițializarea memoriei și a tabelului SGST ;
- încarcă fișierul TVDAnn.SGT, care conține lista perifericelor admise de SGST ;
- tipărește informații ca :
 - punctele de restart ;
 - limitele zonei FREE.
- transferă controlul procesorului de comenzi.

3.1.2. PROCESORUL DE COMENZI

Procesorul de comenzi realizează următoarele :

- analizează linia de comandă ;
- detectează și semnalează eventualele erori de sintaxă ;
- execută comanda (comenzile RUN, STOP, CONT presupun cedarea controlului între procesorul de comenzi, programul utilizator, și dispecer) ;
- semnalează eventualele erori survenite în timpul execuției comenzii.

Comenzile pot fi împărțite în patru categorii:

1. comenzi asociate programului utilizator — de introducere, lansare în execuție, oprire, relansare, ștergere etc. ;
2. comenzi de generare de cimpuri de date ;
3. comenzi de lucru cu programele sursă — editare, salvare, încărcare, afișare ;
4. comenzi utilitare — conversii, calcul de offset, afișare și modificare conținut celule de memorie.

Lista comenzilor este dată în ANEXA A.

3.1.3. COMPILATORUL

Compilatorul asigură traducerea liniilor sursă în cod obiect, direct executabil. Detectează și afișează eventualele erori de sintaxă din programul sursă.

Instrucțiunile recunoscute de compilator sînt cele de uz general (ANEXA B) și cele de I/E asigurate de rutinele de control (vezi secțiunea 3.2).

3.1.4. DISPECERUL

Dispecerul asigură execuția simultană a maximum 16 programe utilizator, permițînd evidențierea eventualelor conflicte de acces la resursele sistemului sau depanarea simultană a mai multor periferice.

Dispecerul scanează partițiile, folosind o tehnică tip polling, pentru a afla dacă un program este prezent, a fost specificat pentru a fi executat și îndeplinește condițiile necesare lansării în execuție. Dacă programul nu a fost oprit de utilizator, nu a detectat o eroare și nu așteaptă terminarea unei operații de I/E, atunci dispecerul transferă controlul programului obiect la adresa lui curentă. Programul păstrează controlul pînă cînd apare unul din evenimentele următoare, caz în care el va reda controlul dispecerului :

1. a fost inițiată o operație de I/E și se așteaptă sfîrșitul ei ;
2. se încearcă lansarea unei operații de I/E, dar perifericul e ocupat ;
3. s-a detectat o eroare pe durata unei operații de I/E ;
4. programul utilizator conține o instrucțiune LET GO (cedează controlul) ;
5. programul utilizator s-a terminat.

După ce a reprimat controlul, dispecerul realizează operațiile necesare pentru a suspenda programul și testează următoarea partiție. Cînd a terminat scanarea ultimei partiții va reveni la prima.

Dacă toate programele așteaptă terminarea unor operații de I/E, dispecerul va continua, în buclă, testarea partițiilor cîutînd un program care a devenit gata pentru reluarea execuției.

Dacă o operație de I/E se termină, programul care a lansat-o va fi reluat la următoarea scanare a partiții asociate. Dacă se încearcă lansarea unei operații de I/E pe un periferic care este deja ocupat sau programul execută instrucțiunea LET GO (cedează controlul), scanarea pentru lansarea în execuție a altor programe se reia cu partiția următoare, aceasta permițînd ca fiecare program să primească controlul măcar o dată.

Scanarea partițiilor continuă pînă cînd fie toate programele s-au terminat, fie ele au fost oprite de utilizator sau de apariția unei erori.

3.2. Rutine de control

3.2.1. CARACTERISTICI GENERALE

Execuția instrucțiunilor de Intrare-Ieșire de nivel înalt, specifice perifericului, este asigurată de prezența rutinei de control respective. În lipsa acesteia, întreaga pregătire a execuției unei operații de Intrare-Ieșire este asigurată de utilizator în programul său, folosind setul de instrucțiuni de uz general.

Rutina de control este o colecție de tabele și subrutine care gestionează echipamentul la nivel de interfață.

INSTRUCȚIUNI

Pe lângă instrucțiunile specifice fiecărui periferic, care vor fi descrise în paragrafele următoare, toate rutinele de control asigură execuția instrucțiunilor de mai jos :

NOWAIT

După inițierea unei operații de Intrare-Ieșire nu se așteaptă terminarea ei, ci se continuă execuția programului utilizator cu instrucțiunea următoare. Acest lucru permite lansarea altei operații de I/E, vizualizarea registrelor și/sau alte prelucrări de date în acest timp. Pentru periferice de teletransmisie, acest mod de lucru permite execuția simultană de citiri și scrieri pentru verificarea tuturor liniilor.

WAIT

După lansarea unei operații de Intrare-Ieșire se așteaptă terminarea acesteia înainte de executarea instrucțiunii următoare din programul sursă.

COUNTS

Se afișează informațiile statistice.

STATUS

Se afișează conținutul curent al registrelor perifericului și conținutul lor la ultima întrerupere.

INFORMAȚII STATISTICE

Informațiile statistice sînt stocate în niște celule de memorie și contorizează :

- numărul de octeți transferați ;
- numărul de comenzi de același fel executate de program ;
- numărul de intrări în rutina de întrerupere ;
- numărul de erori.

CUVINTE DE INTERFAȚĂ CU PROGRAMUL UTILIZATOR

Fiecare rutină de control are la dispoziție un număr de opt cuvinte, dintre care șase pot fi folosite în funcție de necesitățile proprii rutinei de control (CYL, HEAD, SECT pentru disc ; EOT, EOF pentru banda magnetică ; număr de reluări ale unei operații de I/E etc.). Celelalte două cuvinte au asignate nume simbolice și sînt comune tuturor rutinelor de control :

SIZE

Numărul de octeți transferați efectiv de o operație de I/E.

ERR

Flag de eroare, resetat la inițierea fiecărei operații de I/E.

RAPORTAREA ERORILOR

În momentul detectării unei erori, minimul de informații afișate va fi :

- identificatorul erorii ;
- numărul liniei din programul sursă generatoare de eroare.

În funcție de tipul erorii pot fi afișate și alte informații ca :

- numărul unității curente ;
- conținutul registrelor perifericului ;
- mnemonicele biților de eroare.

După raportarea erorii nu se mai reia execuția programului decît dacă bitul 15 din OPSW este setat sau dacă se dă comanda CONT.

3.2.1. Rutina de control RK05

RUTINA DE CONTROL TRKAnn.SGT gestionează operațiile de intrare/ieșire pentru discul RK05 conectat prin cuplor de tip RK11 (cartridge).

Informațiile specifice echipamentului sînt :

- mnemonica : RK11 sau RK05 (se acceptă ambele) ;
- număr de unități : 0—7 ;
- adresa de bază : 177400 ;
- adresa vectorului de întrerupere : 220 ;
- prioritate : 5.

Numele simbolice ale registrelor cuplorului și deplasamentele lor față de adresa de bază sînt :

NUME	DEPLASAMENT	FUNCȚIE
1. RKDS	0	Drive Status Register
2. RKER	+2	Error Register
3. RKCS	+4	Control and Status Register
4. RKWC	+6	Word Count Register
5. RKBA	+10	Bus Address Register
6. RKDA	+12	Disk Address Register
7. RKDB	+16	Data Buffer Register

Instrucțiunile de nivel înalt specifice echipamentului RK05 se împart în :

- a) instrucțiuni prin care se lansează o operație de I/E ;
- b) instrucțiuni prin care nu se lansează o operație de I/E.

a) Instrucțiunile de nivel înalt prin care se lansează o operație de I/E se pot efectua cu valori implicite sau cu valori definite de utilizator (pentru lungimea blocului de date transferat, pentru adresa discului și adresele de memorie între care are loc transferul). Dacă se folosesc valorile implicite, atunci citirea se face în bufferul de citire (RDIO), iar scrierea în bufferul de scriere (WRIO) din partiție, pe o lungime de 256 octeți. Adresa discului este indicată în cuvintele de interfață CYL, HEAD, SECT.

Instrucțiunile sînt următoarele :

```
READ (D256 INTO RDIO)
READ n (INTO RDIO)
READ n INTO a
```

Se transferă n octeți (implicit 256) de pe disc în memorie, la adresa a (implicit în bufferul RDIO).

```
WRITE (D256 FROM WRIO)
WRITE n (FROM WRIO)
WRITE n FROM a
```

Se transferă n octeți (implicit 256) din memorie de la adresa a (implicit buffer WRIO) pe disc.

SEEK

Se lansează funcția SEEK la adresa de disc indicată în cuvintele de interfață CYL, HEAD, SECT. Se așteaptă întreruperea „Search Complete“.

WRLOCK

Unitatea testată va fi protejată la scriere.

RDFMT n INTO a

Se lansează funcția READ cu bitul de format FMT=1. Se transferă deci doar cîte 2 octeți pe sector (header). Sintaxa nu permite valori implicite pentru n și a.

WRFMT n FROM a

Se lansează operația WRITE cu bitul FMT=1. Se scriu cîte 2 octeți pentru fiecare din cele n sectoare, de la adresa a din memorie. Nu se acceptă valori implicite pentru n și a.

RDCK

Se lansează funcția READ CHECK la sectorul a cărui adresă este în cuvintele de interfață CYL, HEAD, SECT.

WRCK n AT a

Se lansează operația WRITE CHECK cu n octeți la adresa de memorie a. u se admit valori implicite.

CRESET

Se dă comanda CONTROL RESET cuplorului.

DRESET

Se dă comanda DRIVE RESET unității curent testate și se așteaptă întreruperea "Search Complete".

u) Instrucțiunile de nivel înalt care nu lansează o operație de intrare/ieșire sînt :

STEPUP n

Se crește adresa disc cu n sectoare, furnizîndu-se totodată o adresă de disc validă (cilindru, cap, sector)

STEPDN n

Se scade adresa disc cu n sectoare, generîndu-se totodată o adresă de disc validă (cilindru, cap, sector).

3.2.2. RUTINA DE CONTROL TM02/TU16

RUTINA DE CONTROL TMMAn.SGT gestionează operațiile de intrare/ieșire pentru banda TU16 conectată prin cuplor de tip TM02.

Informațiile specifice echipamentului sînt :

- mnemonica : TM02 sau TU16 (se acceptă ambele) ;
- număr de unități : 0—77 (8) ;
- adresa de bază : 172440 ;
- adresa vectorului de întrerupere : 224 ;
- prioritate : 5.

Numele simbolice ale registrelor cuplorului și deplasamentele lor față de adresa de bază sînt :

NUME	DEPLASAMENT	FUNCȚIE
1. MTC1	+0	Control Status 1 Register
2. MTWC	+2	Word Count Register
3. MTBA	+4	Bus Address Register
4. MTFC	+6	Frame Count Register
5. MTC2	+10	Control Status 2 Register
6. MTDS	+12	Drive Status Register
7. MTER	+14	Error Register
8. MTAS	+16	Attention Summary Register
9. MTDB	+22	Data Buffer Register
10. MTMR	+24	Maintenance Register
11. MTDT	+26	Drive Type Register
12. MTSN	+30	Serial Number Register
13. MTTC	+32	Tape Control Register

Instrucțiuni de nivel înalt specifice :

READ (D256 INTO RDIO)

READ n (INTO RDIO)

READ n INTO a

- transferă n octeți de pe bandă în memorie la adresa a ; banda merge înainte.

WRITE (D256 FROM WRIO)	
WRITE n (FROM WRIO)	
WRITE n FROM a	— se transferă n octeți din memorie, de la adresa a, pe bandă.
READRV n INTO a	— se transferă n octeți de pe bandă în memorie, la adresa a ; banda merge înapoi.
WRITM	— scrie TAPE MARK.
WCKFWD n FROM a	— compară n octeți de la adresa a cu cei din blocul curent de pe bandă, aceasta deplasându-se înainte.
WCKREV n FROM a	— similară cu WCKFWD, banda se mișcă înapoi.
SPWFD n	— salt înainte peste n blocuri.
SPREV n	— salt înapoi peste n blocuri.
REWIND	— rebobinare pînă la BOT.
RWDOFL	— rebobinare cu trecere OFF-LINE.
DCLEAR	— inițializare cuplor-formatter-drive.
ERASE	— șterge aproximativ 7 cm de bandă.
ODD	— seteaza mod paritate impară
EVEN	— seteaza mod paritate pară.
BPI v	— modifică densitatea de scriere pe bandă, conform valorii v :
	200— 200 bpi
	556— 556 bpi
	800— 800 bpi
	1 600—1 600 bpi
FMT v	— modificarea formatului de scriere pe bandă, conform valorii v :
	N — NORMAL
	C — CORE DUMP
	M — 15 MODE
MWRITE v	— verificarea lanțului de scriere în mod maintenance, folosind cuvîntul de date de la adresa v.
MREAD v	— verificarea lanțului de citire în mod maintenance, folosind cuvîntul de date de la adresa v.

3.2.3 RUTINA DE CONTROL RX11/RX01

Rutina de control RTXAnn.SGT gestionează operațiile de intrare/ieșire pentru discul flexibil simplă densitate RX01 conectat prin cuplor de tip RX11.

Informațiile specifice echipamentului sînt :

- mnemonica : RX11 sau RX01 (se acceptă ambele) ;
- număr de unități : 2 ;
- adresa de bază : 177170 ;
- adresa vectorului de întrerupere : 264 ;
- prioritate : 5.

Numele simbolice ale registrelor cuplorului și deplasamentele lor față de adresa de bază sînt :

NUME	DEPLASAMENT	FUNCȚIE
1. RXCS	0	Control Status Register
2. RXDB	+2	Data Register
3. RXTA	+2	Track Address Register
4. RXSA	+2	Sector Address Register
5. RXES	+2	Error & Status Register

Zona de interfață cu programul utilizator constă din următoarele locații :

TRAK	Un cuvînt conținînd numărul pistei în biții 0—6, număr ce va fi folosit în următoarele operații de I/E.
SECT	Un cuvînt conținînd numărul sectorului în biții 0—4, număr ce va fi folosit în următoarele operații de I/E.
RTRY	Numărul de reluări ale operației de I/E înainte de a decide eroare nerecuperabilă.
SIZE, ERRI	Implementare standard.

Instrucțiunile de nivel înalt specifice echipamentului RX01 sînt următoarele :

READ (D256 INTO RDIO)	
READ <i>n</i> (INTO RDIO)	
READ <i>n</i> INTO <i>a</i>	Se citesc <i>n</i> octeți (implicit 256) la adresa de memorie <i>a</i> (implicit RDIO).
WRITE (D256 FROM WRIO)	
WRITE <i>n</i> (FROM WRIO)	
WRITE <i>n</i> FROM <i>a</i>	Se scriu <i>n</i> octeți (implicit 256) de la adresa de memorie <i>a</i> (implicit WRIO).
CRESET	Controller Clear (RESET).
STEPUP <i>n</i>	Incrementează valoarea SECT cu <i>n</i> , generînd o adresă validă de sector și pistă.
STEPDN <i>n</i>	Decrementează valoarea SECT cu <i>n</i> , generînd o adresă validă de sector și pistă.
WRDDS	Marchează sectorul șters.
FILBF <i>n</i> FROM <i>a</i>	Se scriu <i>n</i> octeți de la adresa <i>a</i> în SILO (max. 128 octeți).
EMPTYB <i>n</i> INTO <i>a</i>	Se citesc <i>n</i> octeți din SILO în memorie la adresa <i>a</i> .
WRSEC	Se scrie un sector pe discul flexibil. Adresa discului este specificată de locațiile TRAK și SECT.
RDSEC	Se citește un sector de pe discul flexibil. Adresa discului este specificată de locațiile TRAK și SECT.
RDSTAT	Se afișează registrul de stare al echipamentului.
RDERR	Se afișează codul de eroare.

Observație : Dacă bitul 7 (200) din OPSW este setat, codul erorii va fi afișat (cu mesajul explicativ) la orice eroare nerecuperabilă.

3.2.4 RUTINA DE CONTROL DH11

Rutina de control TDHAnn.SGT gestionează operațiile de intrare/ieșire pentru multiplexorul asincron programabil pe 16 linii DH11.

Informațiile specifice echipamentului sînt :

- mnemonica : DH11 ;
- adresa de bază : 160020 ;
- adresa vectorului de intrerupere : 300 ;
- prioritate : 5 ; 5.

Numele simbolice ale registrelor cuplorului și deplasamentele lor față de adresa de bază sînt :

NUME	DEPLASAMENT	FUNCȚIE
1. SCR	0	System Control Register
2. NRC	+2	Next Received Character
3. LPR	+4	Line Parameter Register
4. CAR	+6	Current Address Register
5. BYCR	+10	Byte Count Register
6. BAR	+12	Buffer Active Register
7. BCR	+14	Break Control Register
8. SSR	+16	Silo Status Register

Instrucțiunile de nivel înalt specifice echipamentului DH11 sînt următoarele :

READ (D256 INTO RDIO)

READ n (INTO RDIO)

READ n INTO a FROM u

READ n INTO a

Recepție n caractere (implicit 256).

WRITE (D256 FROM WRIO)

WRITE n (FROM WRIO)

WRITE n FROM a FROM u

WRITE n FROM a

Emisie n caractere (implicit 256).

BREAK t

BREAK t ON i1 [, ... i1 ...]

Se generează o intrerupere de durată t pe liniile i1, ... i1.

CRESET

Inițializare cuplor.

ALARM v

Se încarcă nivelul de alarmă SILO (v poate fi 0, 1, 2, 4, 10, 20, 40).

SETUP 1 [i1, ...]

Selectează liniile pentru care se vor stabili parametri în LPR cu instrucțiunile de mai jos.

RBAUD v

Selectează viteza de recepție v :

v=0	0
v=1	50
v=2	75
v=3	110
v=4	134,5
v=5	150
v=6	200
v=7	300
v=10	600
v=11	1200
v=12	1800
v=13	2400
v=14	4800
v=15	9600
v=16	EXT A
v=17	EXT B

TBAUD v	Selectează viteza de transmisie v.
BAUD v	Selectează viteza de transmisie/recepție v.
EVEN	Selectează paritate pară.
ODD	Selectează paritate impară.
NOPAR	Dezactivează paritatea.
ONESTP	Setează numărul de biți de stop = 1.
TWOSTP	Setează numărul de biți de stop = 2.
BITS n	Selectează numărul de biți pe caracter (n poate fi 5, 6, 7, 10 sau D8).
EGHO	Activează auto-ecoul.
NOECHO	Dezactivează auto-ecoul.
PRESET	Selectează ca parametri: 8 biți pe caracter, fără paritate, 2 biți de stop, 110 bauds, fără ecou, full duplex.
FDUPLX	Selectează mod full duplex.
HDUPLX	Selectează mod half duplex.

Observație : Dacă bitul 7 (SPOPER) din OPSW este setat, se va pune pe 1 bitul de maintenance (bitul 9) din SCR.

3.2.5. RUTINA DE CONTROL DL11

Rutina de control TDLAnn.SGT gestionează operațiile de intrare/ieșire pentru interfața de linie simplă serială asincronă DL11.

Informațiile specifice echipamentului sînt :

- mnemonica : DL11 ;
- adresa de bază : 175610 ;
- adresa vectorului de intrerupere : 300 ;
- prioritate : 4 ; 4.

Numele simbolice ale registrelor cuplorului și deplasamentele lor față de adresa de bază sînt :

NUME	DEPLASAMENT	FUNCȚIE
1. RCSR	0	Receiver Status Register
2. RBUF	+2	Receiver Data Buffer
3. XCSR	+4	Transmitter Status Register
4. XBUF	+6	Transmitter Data Buffer

Instrucțiunile de nivel înalt specifice echipamentului DL11 sînt următoarele :

READ (D256 INTO RDIO)

READ n (INTO RDIO)

READ n INTO a

Recepție n caractere (implicit 256).

WRITE (D256 FROM WRIO)

WRITE n (FROM WRIO)

WRITE n FROM a

Transmisie n caractere (implicit 256).

BREAK t	Generează întrerupere de durată t.
CRESET	Inițializare cuplor.
CALL	Inițiază un apel către DL11.
LISTEN	Așteaptă un apel de la DL11.
ANSWER	Răspunde unui apel de la DL11.
HANGUP	Termină un apel
SEND	Setează linia pentru transmisie.
RECV	Setează linia pentru recepție.
RDRON	Setează bitul 0 din RCSR care deschide releul de pornire al benzii de hirtie (unde e posibil).
RDROFF	Șterge bitul 0 din RCSR.

Observație: Dacă bitul 7 (SOPER) din OPSW este setat, atunci se va seta bitul 2 din XCSR (maintenance).

3.2.6. RUTINA DE CONTROL LP11

RUTINA DE CONTROL TLPAnu.SGT gestionează operațiile de intrare/ieșire pentru imprimanta de tip LP11.

Informațiile specifice echipamentului sînt:

- mnemonica: LP11;
- adresa de bază: 177514;
- adresa vectorului de întrerupere: 200;
- prioritate: 4.

Numele simbolice ale registrelor cuplorului și deplasamentelor față de adresa de bază sînt:

NUME	DEPLASAMENT	FUNCȚIE
1. LPS	0	Control Status Register
2. LPB	+2	Data Buffer Register

Instrucțiunile de nivel înalt specifice echipamentului LP11 sînt următoarele:

WRITE (D256 FROM WRIO)

WRITE n (FROM WRIO)

WRITE n FROM a

Se transferă n octeți (implicit 256) din memorie de la adresa a (implicit buffer WRIO) la imprimantă. După date se inserează automat CR/LF, dacă bitul SOPER din OPSW=0.

SPACE n

Se avansează n linii (prin scrierea în bufferul de date a două combinații CR/LF).

TOF

Top of Form. Se trimite la imprimantă caracterul FORM FEED (cod 014).

EOT

End of Transmission. Se trimite caracterul EOT (cod 004) către imprimantă.

Observație: Dacă bitul 7 (SOPER) din OPSW=1, se inhibă inserarea automată a caracterelor CR/LF după datele transmise spre imprimantă.

4. PROCEDURA DE LANSARE ÎN EXECUȚIE ȘI OPERARE

Sistemul de Generare de Secvențe de Test este încărcat de monitorul XXDP sau de utilitarul UPD2, la fel ca orice diagnostic. Este încărcat înfi Executivul, care realizează o inițializare a locațiilor 0-276 (trap catcher), determină dimensiunea memoriei prezente fizic, șterge zona de memorie de deasupra lui și determină prezența unității de management a memoriei, prin adresarea registrului MMRO. Dacă unitatea de management există, Executivul va întreaga operatorul asupra folosirii variantei cu sau fără management a SGST. În conformitate cu răspunsul primit de la acesta, Executivul va încărcă monitorul TMGANm.SGT (fără management) sau TMMANm.SGT (cu management).

Monitorul va inițializa o serie de locații și contoare, își va scrie numele, versiunea și adresa de restart și va încărcă apoi Tabela de Echipamente Permise (TVDANm.SGT). În continuare, monitorul va afișa harta ocupării memoriei (Memory Map), va evalua zona de memorie disponibilă (FREE) și va tipări mesajul "ENTER CMND" și preceptorul * (asterisc), așteptând introducerea uneia din comenzile de editare program sursă, modificare, execuție, salvare sau restaurare de programe (vezi Anexa A).

Execuția unui program poate fi întreruptă, revenind în monitor, prin apăsarea tastelor CTRL-C. În mod „întrerupere”, monitorul acceptă doar un subset limitat de comenzi, care nu modifică programele existente, ci doar raportează erorile, schimbă switch-urile de execuție a unui sau a mai multor programe sau oprește execuția unui sau mai multor programe. După ce s-a dat o nouă comandă RUN, execuția se reia, prin scararea succesivă a partițiilor. Se continuă astfel până când s-au terminat toate programele și s-au executat toate comenzile.

În Anexa C am prezentat câteva exemple de introducere și de execuție de secvențe de test (pentru discurile magnetice RK05, banda magnetică TM02 și imprimanta LP11).

ANEXA A

Setul de comenzi SGST

a) Comenzi asociate programului utilizator.

ENTER p [AS nume]

— trece în mod introducere linii sursă pentru programul p cu "nume" opțional.

RUN [p1 [AT ln] [, ... [, pn [AT ln]]]]

— lansează în execuție programele utilizator pn, de la prima linie sursă sau de la cea specificată prin ln.

STOP p1 [, ... , pn]

— oprește unul sau mai multe programe utilizator.

CONT p1 [, ... , pn]

— reia execuția unui sau mai multor programe utilizator oprite de o comandă utilizator sau de apariția unei erori. Execuția efectivă nu este reluată decât la introducerea unei comenzi RUN sau a unei linii nule.

KILL p1 [, ... , pn]

— forțează terminarea unui program. Programul poate fi restartat cu comanda RUN.

DELETE p1 [, ... , pn]

— șterge programele specificate și eliberează memoria alocată lor.

ASSIGN mdl [, u1, ... , un] TO p

— permite schimbarea numelor de unități asignate unui program existent și/sau schimbarea constantelor perifericului.

ASSIGN {mdl, u TO {SAVE
{NONE {FETCH
{LIST

— asignare, deasignare a perifericului curent drept dispozitiv LIST, SAVE, FETCH.

REPORT p [, p STATUS ,p]

REPORT p [,p COUNTS ,p]

— forțează tipărirea tuturor registrelor echipamentului, informațiile de eroare și informațiile statistice pentru programul (programele) specificate. Are efect doar asupra programelor scrise pentru echipamente care au rutine de control.

OPSW [p] oooo

— folosit pentru a modifica switch-urile de operare. Dacă numărul programului nu este specificat, noua valoare a OPSW va fi valabilă pentru toate programele rezidente în memorie, altfel doar OPSW-ul programului p va fi modificat.

b) Comenzi de generare de timpuri de date.

FILL BFnn WITH {oooo, oooo, ..., etc
 { *aaa ... a
 { #aaa ... a

— permite încărcarea celor 16 buffer-e comune de date fie cu numere octale, fie cu caractere ASCII. Valoarea nn poate fi între 00 și 15, specificând care buffer va fi încărcat. Codurile *, # pentru formarea caracterelor ASCII permit introducerea sau nu a caracterelor de control <CR>, <LF>.

FILL BUF WITH {PATn
 {oooo, oooo, ..., etc.

— întreg buffer-ul de 256 bytes este încărcat fie cu o configurație de date predefinită, fie cu cuvinte în octal.

FILL COM[n] WITH {oooo
 { *aaa ... a
 { #aaa ... a
 { PATn

— aceasta este o altă formă a comenzii FILL care permite accesul la cele 10 cuvinte comune COMO—COM9 ca și la zonele BUF00—BUF15.

c) Comenzi de lucru cu programele sursă.

MODIFY p

— următoarele linii sursă vor fi considerate corecții pentru programul p, existent deja.

DISPLAY p [, p CODE ,p]

— afișează liniile sursă ale programului specificat cu sau fără codul obiect generat.

/SAVE p [AS name]

— liniile sursă ale programului p vor fi scrise pe dispozitivul SAVE curent, cu un identificator de 1—6 caractere furnizate de "name". Dacă acesta nu este specificat, programul va fi salvat sub numele asignat acestuia în memorie. Programul nu va fi șters din memorie.

/FETCH name AS p

— citește un program salvat anterior.

/DELETE name [, name, ..., name]

— șterge programele utilizator specificate, salvate anterior pe dispozitivul SAVE curent.

/LIST_{ALL

 _{FAST

 _{ALL, FAST

— listează fișierele de pe dispozitivul FETCH.

ALL — se listează toate fișierele.

FAST — se listează doar numele fișierelor.

/ZERO

— inițializează dispozitivul SAVE curent. Când se încearcă inițializarea dispozitivului suport pentru SGST este afișat un mesaj de avertizare care așteaptă un răspuns afirmativ pentru a efectua inițializarea.

/BOOT

— execută funcția de bootstrap-are de pe dispozitivul FETCH curent.

d) Comenzi utilitare.

FM

— formatează memoria eliminând orice spațiu nefolosit care a apărut între SGST și programele utilizator.

RDM aaaa [, cccc]

— afișează conținutul celei de memorie de la adresa octală aaaa. Dacă și a doua adresă este specificată se afișează întreaga zonă de memorie de la aaaa la cccc, inclusiv.

WRM aaaa, oooo [, oooo, ..., etc.]

— scrie un cuvânt sau un șir de cuvinte octale începând de la adresa octală de memorie aaaa.

BOC aaaa, cccc

— calculează valoarea deplasamentului dintre două adrese pare de memorie.

ADD oooo, oooo [, oooo, ..., etc.]

— adună două sau mai multe numere octale și afișează rezultatul.

SUB oooo, oooo

— scade primul număr octal din al doilea și afișează rezultatul.

CBD oooo

— convertește un număr octal într-unul zecimal și afișează rezultatul.

CDB dddd

— convertește un număr zecimal într-unul octal și afișează rezultatul.

ANEXA B

Instrucțiuni de uz general pentru scrierea programelor sursă

LOAD a WITH v

— încarcă celula de memorie de la adresa a cu valoarea v.

INCR a [BY i]

— incrementează valoarea de la adresa a cu i. Dacă i nu e specificat incrementarea se face cu 1.

DECR a [BY i]

— decrementează valoarea de la adresa a cu i. Dacă i nu e specificat decrementarea se face cu 1.

MOVE

— această instrucțiune, fără parametri, implică un „move“ indirect. Numărul de bytes de transferat e conținut de locația NBR, iar adresa de unde încep în locația SRC. Adresa destinației se găsește în locația DST.

MOVE [n AT] a1 TO a2

— se mută n bytes de la adresa a1 la adresa a2. Dacă n nu e specificat se mută un cuvânt.

ADD [n AT] a1 TO a2

— adună n bytes de la adresa a1 și cu n bytes de la adresa a2 și pune rezultatul la adresa a2. Dacă n lipsește se adună doar cite un cuvânt.

SUB [n AT] a1 TO a2

— se efectuează scăderea pe n bytes, descăzutul fiind la adresa a2, iar scăzătorul la adresa a1. Rezultatul este pus la adresa a2. Dacă n lipsește se efectuează scăderea pe cuvânt.

NEGATE a

— se efectuează complementul față de 2 al conținutului celei de memorie de la adresa a.

- SET a BIT n [THRU n [AND n]]**
 {— &
 — se setează bitul n în locația a. Pot fi specificate combinații de biți (AND, &) sau grupuri de biți (THRU, —).
- CLEAR a BIT n [THRU n [AND n]]**
 {— &
 — se șterge bitul n de la locația a. Ca și la SET se pot specifica combinații și grupuri de biți.
- IF a BIT n [AND n ... AND n [THRU n]] {SET GO TO ln
 {CLEAR**
 — dacă bitul sau combinația de biți specificată îndeplinește condiția, atunci se execută salt la linia sursă ln.
- IF [n AT] a1=a2 GO TO ln**
 >
 <
 ⇄
 ⇄
 < >
 — dacă relația specificată între conținutul celulelor a1 și a2 este adevărată se execută salt la linia sursă ln; n specifică numărul de bytes pe care se face comparația.
- GO TO ln**
 — salt necondiționat în program, la linia sursă ln.
- LINK ln**
 — salt la subrutina care începe de la adresa ln.
- RETURN**
 — reîntoarcere din subrutină.
- PRINT * text**
 #
 — se tipărește textul specificat la dispozitivul de tipărire asigurat programului. După text urmează sau nu <CR>, <LF>, după cum se folosesc * sau #.
- PRINT [n AT] a IN {BINARY
 {OCTAL
 {DECIMAL
 {ASCII**
 — se convertește n bytes conform formatului cerut și se tipărește la dispozitivul de tipărire asigurat programului. Dacă n lipsește se tipărește un singur cuvânt.
- FILL [n AT] a WITH {RANDOM
 {ASCII
 {v
 {PATn**
 — se umplu n bytes, începând de la adresa a, cu date generate aleator, cu caractere ASCII, cu combinația de biți specificată sau cu pattern-ul n. Când n lipsește, valoarea sa este implicit 256.
- ROTATE [n AT] a**
 — se rotesc la dreapta cu o poziție, n bytes de la locația a; a este implicit adresa buffer-ului WRIO pentru acest program.
- VECTOR a TO ln**
 — se memorează adresa liniei sursa ln la adresa a.
- VERIFY [n AT] a1 WITH a2**
 — se face o comparație pe n bytes a două zone de memorie, prima începând de la adresa a1, a doua de la adresa a2. Se tipăresc, în octal, perechile care nu concordă.
- PAUSE**
 este compilată ca un WAIT.

DELAY v	— se așteaptă v microsecunde. Variabila v trebuie să aibă o valoare zecimală.
ENTRY	— salvează în stivă registrele R0—R5 și e folosită pentru începerea rutinei de întrerupere scrisă de utilizator.
EXIT	— reface registrele R0—R5 și permite ieșirea din rutina de întrerupere scrisă de utilizator.
LET GO	— prin această instrucțiune programul utilizator redă controlul dispecerului.
RESET	— instrucțiunea RESET.
WORD v	— se rezervă în programul utilizator un cuvânt a cărui valoare este v.
END	— marchează sfârșitul programului. Trebuie să fie instrucțiunea cu cel mai mare număr de linie.

ANEXA C

Exemple de secvențe de test

```

*ENTER 1 AS DISK1
?ASGN DEV : RK11, 0
?RDIO = 256/0
?WRIO = 256/0
?DEV REG = 177400/
?INT VEC = 000220/
?BUS REQ = 5/
ENTER STMTN'TS
>0010 LOAD COM0 WITH 0
>0020 READ D2048 INTO FREE
>0030 LOAD COM0 WITH 1
>0040 LETGO
>0050 IF COM0 = 1 GO TO 40
>0060 STEPUP 4
>0070 IF CYL > 0 GO TO 20
>0080 IF HEAD > 0 GO TO 20
>0090 IF SECT > 0 GO TO 20
>0100 END
>DONE
"PGMS COMPILED/MEM REFORMATTED"
*ENTER 2 AS DISK2
?ASGN DEV : RK11, 1
?RDIO = 256/0
?WRIO = 256/0
?DEV REG = 177400/
?INT VEC = 000220/
?BUS REQ = 5/
ENTER STMTN'TS
>0010 IF COM0=1 GO TO 40
>0020 LETGO
>0030 GO TO 10
>0040 WRITE D2048 FROM FREE
>0050 STEPUP 4
>0060 LOAD COM0 WITH 0

```

```

>0070 IF CYL > 0 GO TO 20
>0080 IF HEAD > 0 GO TO 20
>0090 IF SECT > 0 GO TO 20
>0100 END
>DONE
"PGMS COMPILED/MEM REFORMATTED"
*ENTER 3 AS MGTAPE
?ASGN DEV : TM02, 0
?RDIO = 256/1000
?WRIO = 256/1000
?DEV REG = 172440/
?INT VEC = 000224/
?BUS REQ = 5/
ENTER STMT'S
>0010 LOAD TM04 WITH D100
>0020 FILL D1000 AT WRIO WITH PAT 9
>0030 WRITE D1000
>0040 DECR TM04
>0050 IF TM04 > 0 GO TO 30
>0060 REWIND
>0070 LOAD TM05 WITH D100
>0080 READ D1000
>0090 VERIFY D1000 AT RDIO WITH WRIO
>0100 DECR TM05
>0110 IF TM05 > 0 GO TO 80
>0120 REWIND
>0130 END
>DONE
"PGMS COMPILED/MEM REFORMATTED"
*ENTER 4 AS PRINTER
?ASGN DEV : LP11
?RDIO = 256/132
?WRIO = 256/132
?DEV REG = 177514/
?INT VEC = 000200/
?BUS REQ = 4/
ENTER STMT'S
>0010 LOAD TM00 WITH D96
>0020 FILL D132 AT RDIO WITH 401
>0030 FILL D132 AT WRIO WITH 20040
>0040 WRITE D132 FROM WRIO
>0050 ADD D132 AT RDIO TO WRIO
>0060 DECR TM00
>0070 IF TM00 > 0 GO TO 40
>0080 EOT
>0090 END
>DONE
"PGMS COMPILED/MEM REFORMATTED"
*RUN 1, 2, 3, 4

```

ANEXA D

Switch-uri de operare

Valoarea prestabilită este : 100000

Semnificația biților de control :

- 15 0 — continuă execuția la eroare ;
 1 — oprește execuția programului.

- 14 0 — dă controlul echipamentului următor, la sfârșitul execuției;
1 — rela execuția pentru aceeași unitate.
- 13 0 — tipărește mesaj eroare;
1 — inhibă tipărirea.
- 12 x — nefolosit.
- 11 x — nefolosit.
- 10 0 — oprire program la terminarea listei de echipamente;
1 — ciclează pe lista de echipamente.
- 9 0 — continuă execuția pe același echipament în caz de eroare;
1 — rela execuția pentru echipamentul următor, la eroare (bitul 9 este efectiv doar dacă bitul 15 are valoarea 0).
- 8 0 — verificare erori;
1 — fără verificare erori.
- 7 0 — nu sînt operații speciale;
1 — operații speciale.
- 6 x — nefolosit.
- 5 0 — activează timeout de I/E;
1 — fără timeout de I/E.
- 4 0 — afișarea automată a informațiilor statistice conform valorii bitului 3 din OPSW;
1 — nu se afișează inf. statistice.
- 3 0 — afișează inf. statistice la sfârșitul fiecărui pas;
1 — afișează inf. statistice numai după pasul final.
- 2 0 — șterge zona de inf. statistice după fiecare pas, exceptînd pasul final;
1 — nu se inițializează zona de informații statistice decît la începutul execuției
- 1 0 — tipărește toate diferențele rezultate în urma instrucțiunii VERIFY;
1 — tipărește doar prima diferență detectată de instrucțiunea VERIFY.
- 0 0 — tipărește mesajul "PROGRAM COMPLETED" după pasul final.
1 — nu tipărește mesajul de terminare a execuției programului.

ANEXA E

Operanzi predefiniți

- PAT0** 100000 — walking ones
040000
020000, etc.
- PAT1** 077777 — walking zeroes
137777
157777, etc.
- PAT2** 100000 — expanding ones
140000
160000, etc.
- PAT3** 077777 — expanding zeroes
037777
017777, etc.
- PAT4** 125252 — alternate ones (horiz.)
125252
125252, etc.
- PAT5** 052525 — alternate zeroes (horiz.)
052525
052525, etc.

PAT6	177777 — alternate ones (vert.) 000000 177777 000000, etc.
PAT7	000000 — alternate zeroes (vert.) 177777 000000 177777, etc.
PAT8	070707 — octal checkerboard 107070 070707 107070, etc.
PAT9	125252 — binary checkerboard 052525 125252 052525, etc.
PATA	000000 — count words 000001 000002, etc.
PATB	165555 — RP04 disk serial data 133333 155555 133333, etc.

II. UTILIZAREA MONITORULUI XXDP PENTRU SERVICE MINICALCULATOARE

1. MONITORUL DE TESTE XXDP

1.1. Generalități

Pentru manipularea programelor de test sau a diverselor fișiere în format specific pe suporturi magnetice, există programe de tip monitor de teste, specifice fiecărui tip de periferic.

Denumirea generică a acestor monitoare este XXDP (există și XXDP + în diverse variante), unde XX are următoarele semnificații:

- XX=RK (pentru disc cartridge DK)
- XX=RX (pentru disc floppy simplă densitate DX)
- XX=RP (pentru disc de masă DP)
- XX=TH, TM (pentru banda magnetică MM).

Dacă acest program este salvat în mod specific (vezi 2.4/4) la începutul fizic al suportului magnetic respectiv, el este încărcabil și executabil printr-o procedură de bootstrapare a perifericului.

1.2. Comenzile monitorului

Indiferent de perifericul de pe care este încărcat, monitorul XXDP are următoarele comenzi:

D (CR) — tipărește la consolă lista programelor („directory“) de pe suportul magnetic;

D/I (CR)	— tipărire la imprimantă a listei programelor de pe suportul magnetic ;
R_NAME (CR)	— încărcarea de pe suportul magnetic a programului NAME și autostartarea acestui fișier de la adresa de memorie 200 ₈ ;
L_NAME (CR)	— încărcarea de pe suportul magnetic a programului NAME în memorie ;
S (CR)	— lansarea (startarea) programului încărcat anterior în memorie, de la adresa implicită 200 ₈ ;
S_ADDR (CR)	— lansarea (startarea) programului încărcat anterior în memorie, de la adresa ADDR (introdusă în octal) ;
C_NAME (CR)	— încărcarea și lansarea în execuție a lanțului de programe NAME. Acest lanț este de fapt un fișier de comenzi indirecte pentru monitorul XXDP (vezi 2.4/9) ;
C_NAME/QV (CR)	— încărcarea și lansarea în execuție a lanțului de programe NAME. Execuția se face numai cîte o dată pentru fiecare program din lanț.

OBSERVAȚII :

a. Programele (fișierele) exploatabile cu monitorul XXDP au următoarul indicativ :

NAME.EXT, unde
 NAME = numele programului
 EXT = extensia (tipul) programului

Pentru EXT se folosesc următoarele coduri mai frecvent :

EXT = **BIN** — fișier binar (imagine memorie executabilă)
 = **BIC** — fișier binar înlănțuibil (în lanț de teste)
 = **TXT** — fișier text ASCII
 = **LST** — fișier listă
 = **CCC** — fișier lanț de programe

Se pot declara și alte extensii (de ex. MAN pentru manual, CMD pentru fișier de comenzi etc.)

b. Programele (fișierele) exploatabile cu comanda R NAME trebuie să fie binare (BIN, BIC)

c. Programele (fișierele) executabile în lanț trebuie să aibă extensia BIC.

1.3. Lansarea monitorului XXDP.

Dacă monitorul XXDP se află pe un suport magnetic, în formă bootstrapabilă, procedura este următoarea :

1. Se lansează emulatorul de consolă ;
2. Se bootstrapează perifericul pe care avem suportul magnetic cu programele de test și XXDP ;
3. În urma bootstrapării se tipăresc la consolă cîteva mesaje ajutătoare (listarea comenzilor XXDP) după care se obține prompter-ul XXDP „“. Din acest moment se pot introduce comenzile XXDP necesare utilizatorului.
OBS. Se poate inhiba tipărirea mesajului ajutător pentru obținerea mai rapidă a prompter-ului, cu ajutorul CTRL-C la consolă, după comanda de bootstrap-arc.

1.4. Lansarea programelor în XXDP.

1. Se lansează monitorul XXDP
2. Se lansează programul NAME dorit prin una din metodele :
 - a. cu comanda R_NAME (CR)
 - b. cu comenzile L_NAME (CR) și apoi cu
 - b.1. S (CR) start de la adresa 200₈
 - b.2. S_ADDR (CR) start de la adresa ADDR₈
 - b.3. comenzi de panou sau emulator de consolă, după ieșirea din monitorul XXDP (procedură) folosită de obicei pentru a încărcă SWR înainte de startarea testului).

2. Programul utilitar UPD

2.1. Generalități programe utilitare

Există câteva programe utilitare, cu ajutorul cărora se pot manipula programe în format XXDP.

UPD1 } UPD2 } UPD3 }	Programe utilitare generale
XTECO	Program editor de texte
COPY1 } COPY2 }	Programe de copiere între periferice identice

Cele mai folosite sînt UPD2/UPD3, variante îmbunătățite ale UPD1. Pentru informații despre celelalte utilitare se listează HELP.TXT (vezi 2.4/10) care e fișierul XXDP USER MANUAL M-11-DZQXA.

2.2. UPD2 – Comenzi și periferice suportate

Lista comenzilor UPD2 cuprinde multe	comenzi, din care se descriu cele mai folosite
BOOT_DEV : (CR)	— bootstrapează perifericul DEV (la fel ca în emulatorul de consolă);
DIR_DEV : (CR)	— listează la consolă cuprinsul suportului de programe de pe perifericul DEV (la fel ca D din XXDP);
DIRLP_DEV : (CR)	— listează la imprimantă cuprinsul suportului magnetic de pe DEV;
ZERO_DEV : (CR)	— inițializează suportul de pe DEV;
TYPE_DEV : NAME.EXT (CR)	— tipărește de la consolă conținutul fișierului NAME.EXT din suportul de pe DEV;
PRINT_DEV : NAME.EXT (CR)	— tipărește la imprimantă conținutul fișierului NAME.EXT din suportul de pe DEV;
LOAD_DEV : NAME.EXT (CR)	— încarcă de pe DEV, în memorie, în format absolut (BIN, BIC) programul NAME.EXT, făcînd verificarea CHECKSUM;
DUMP_DEV : NAME.EXT (CR)	— încarcă pe DEV, cu numele NAME.EXT, conținutul memoriei în format absolut;
PIP_DEV0 : NAME0.EXT0 = DEV1 : NAME1.EXT1 (CR)	— copiază de pe DEV1 fișierul NAME1.EXT1, pe DEV0 cu numele NAME0.EXT0, fără a verifica CHECKSUM;
FILEF_DEV0 : = DEV1 : NAME1.EXT1 (CR)	— copiază rapid de pe DEV1 pe DEV0, păstrînd numele fișierului;
FILED_DEV : NAME.EXT (CR)	— șterge pe DEV, fișierul NAME.EXT;
REN_DEV : NAME2.EXT2 = DEV : NAME1.EXT1 (CR)	— schimbă pe DEV, numele fișierului NAME1 în NAME2, și extensiile respective;
CORE (CR)	— tipărește limitele memoriei afectate de transferul în memorie anterior;
LOCORE_ADDR (CR)	— fixează limita inferioară de memorie pentru un ulterior DUMP;
HICORE_ADDR (CR)	— fixează limita superioară de memorie pentru un ulterior DUMP;
MOD_ADDR (CR)	— examinează conținutul locației de memorie ADDR, cu posibilitatea modificării ei;
DO_DEV : NAME.EXT (CR)	— execută un fișier NAME.EXT de comenzi UPD2;
AVSE_DEV : NAME.SAV (CR)	— creează un bloc autoîncărcabil pe DEV (de tip bandă magnetică) cu numele NAME;

SAVM_DEV : (CR)	— creează un bloc autoîncărcabil pe DEV (de tip disc magnetic), cu numele din operația de LOAD anterioară ;
TEXT DEV : NAME.EXT (CR)	— creează pe DEV un fișier listing sau de comenzi (XXDP, UPD) ;
^C (CTRL-C)	— termină comanda în curs, cu revenire în UPD2 ;
^Z (CTRL-Z)	— termină comanda TEXT.

Pentru unele comenzi (PIP, FILEF) comenzile acceptă caracterul „*“ în diverse combinații, cu semnificația :

- *.BIN = toate fișierele cu extensia BIN ;
- NAME.* = toate fișierele cu numele NAME, cu toate extensiile existente ;
- *.* = toate fișierele cu toate extensiile ;
- NA*.BIN = toate fișierele cu numele începînd cu NA, cu extensia BIN

Perifericele suportate de UPD2, cele mai frecvent folosite, sînt :

- PR = lector bandă perforată
- PP = perforator bandă hirtie
- DKN = discul cartridge N ($N=0\div 3$)
- DXN = discul floppy N ($N=0\div 1$)
- DPN = discul de masă N ($N=0\div 1$)
- MMN = unitatea de bandă N ($N=0\div 3$)

Aceste mnemonice sînt folosite în timpul DEV din comandă. În orice caz care conține unitatea N, aceasta trebuie indicată, chiar dacă $N=0$ (nu este considerată implicit unitatea zero în UPD2).

2.3. Lansarea utilitarului UPD2

1. Se lansează monitorul XXDP.
2. Se dă comanda R_UPD2 (CR).
3. Programul își tipărește numele și apoi cere data, care trebuie scrisă în format DD—MMM—YY (CR).

DD — ziua

MMM — luna (primele trei litere din numele în engleză)

YY — anul (de ex. 85).⁹⁴

După care este tipărită adresa de restart (utilă pentru relansarea directă a UPD2 încărcat în memorie) și în final se obține prompter-ul „*“ pentru UPD2, așteptînd linii de comandă.

2.4. Exemple de folosire UPD2

Exemplele următoare sînt liniile de comenzi folosite cel mai frecvent în diferitele cazuri de lucru cu XXDP, programele de test sau utilitarul UPD2 :

1. Bootstraparea unui periferic (DEV)

BOOT_DEV : (CR) — pentru periferice cu suporturi magnetice

2. Listarea cuprinsului unui suport de pe perifericul magnetic DEV

DIR_DEV : (CR)

3. Ștergerea fișierelor de pe un periferic magnetic DEV

FILED_DEV : NAME.EXT (CR)

4. Crearea unui suport magnetic autoîncărcabil pe DEV2

— presupunem că programul XXDP specific perifericului pe care vrem să creăm acest suport bootstrapabil se află pe un suport magnetic pe DEV1

a. pentru benzi magnetice

ZERO__DEV2 : (CR)
 LOAD__DEV1 : THDP.BIN (CR)
 SAVE__DEV2 : THDP.SAV (CR)
 LOAD__DEV1 : TMDP.BIN (CR)
 SAVE__DEV2 : TMDP.SAV (CR)

unde DEV2 este $MM0 \div 3$

b. pentru discuri magnetice

ZERO__DEV2 : (CR)
 LOAD__DEV1 : XXDP.BIN (CR)
 SAVM__DEV2 : (CR)

unde DEV2 = DXN/DKN/DPN, iar

XXDP = RXDP/RKDP/RPDP, corespunzător lui DEV2.

6. Copierea fișierelor de pe un suport magnetic (pe DEV1) pe un alt suport magnetic (pe DEV2)

a. fișier cu fișier, cu verificarea CHECKSUM și actualizarea datei cu cea introdusă la lansarea UPD2

LOAD__DEV1 : NAME1.EXT1 (CR)
 DUMP__DEV2 : NAME2.EXT2 (CR)

b. fișier cu fișier, fără verificarea CHECKSUM

PIP__DEV2 : NAME2.EXT2 = DEV1 : NAME1.EXT1 (CR)

c) mai multe fișiere, cu o singură comandă

- c.1. PIP__DEV2 : *.BIN = DEV1 : *.BIN (CR) — toate fișierele cu extensia BIN
 c.2. PIP__DEV2 : NA *.BIN = DEV1 : Na*.BIN (CR) — toate fișierele cu nume începând cu NA, și extensie BIN
 c.3. PIP__DEV2 : *.* = DEV1 : *.* (CR) — toate fișierele cu toate extensiile
 c.4. FILEF__DEV2 : = DEV1 : *.BIN (CR) — toate fișierele cu extensia BIN, copiere rapidă
 c.5. FILEF__DEV2 : = DEV1 : NA*.BIN (CR) — toate fișierele cu nume începând cu NA, extensie BIN, copie rapidă;
 c.6. FILEF__DEV2 : = DEV1 : *.* (CR) — toate fișierele cu toate extensiile, copiere rapidă.

Observații : cu cazurile a și b se pot schimba la transfer numele și extensia ; în toate cazurile se păstrează numele și extensia.

6. Încărcarea unui fișier de pe bandă perforată, scris în format recunoscut de ABSOLUTE LOADER

a. încărcare în memorie

LOAD PR : (CR)

b. încărcare pe un suport magnetic DEV

b.1. LOAD__PR : (CR)
 DUMP__DEV : NAME.EXT (CR)
 b.2. PIP__DEV : NAME.EXT = PR : (CR)

Se poate folosi b.1. dsau b.2., fișierul căpătînd numele NAME.EXT.

7. Scoaterea fișierelor pe bandă perforată (în format recunoscut de ABSOLUTE LOADER)

a. duplicare bandă perforată

PIP__PP : = PR : (CR)

b. perforare fișier de pe un suport magnetic DEV

b.1. LOAD__DEV : NAME.EXT (CR)
 DUMP__PP : (CR)
 b.2. PIP__PP : = DEV : NAME.EXT (CR)

8. Crearea unui fișier (program) și salvarea lui pe un suport magnetic aflat pe DEV

a. Se intră în UPD2, ținînd minte adresa de restart

b. Se iese din UPD2, intrînd în emulatorul de consolă

c. Se introduce programul în memorie cu emulatorul de consolă sau cu funcțiile de panou

d. Se reintră (**restartare**) în UPD2 de la adresa de restart

LOCORE_INF (CR) } Se introduc limitele de memorie ale unei zone în care am
 HICORE_SUP (CR) } introdus programul
 DUMP_DEV : NAME.EXT (CR) se salvează pe suportul magnetic cu numele dorit

Observații , 1) Presupunem un program care nu afectează zona de memorie la care am introdus UPD2 ;

- 2) De obicei programele se scriu la zone mici de memorie (peste 1 000₈) ;
- 3) În cazul în care dorim posibilitatea de lansare automată cu XXDP, programul trebuie să aibă prima instrucțiune la adresa 200₈ ;
- 4) Programul se poate perfora și păstra pe bandă perforată, conform procedurii de mai sus.

9. Editarea de fișiere listing, fișiere text, fișiere de comenzi.

a. Se intră în programul utilitar UPD2

b. Se dă comanda TEXT_DEV : NAME.EXT (CR), după care programul tipărește mesajul :

MAKE OUTPUT READY.TYPE (CR) WHEN READY.

După ce perifericul pe care se editează textul (DEV) este Ready, se apasă (CR), programul tipărește prompter-ul " " , așteptând introducerea textului.

c.1. Editarea unui fișier listing (text pentru tipărire)

- Extensia este dată, în general, de forma TXT, LST, MAN ;
- Textul se introduce în paginația dorită, trecerea la linie nouă făcându-se cu (CR), obținind prompter-ul pentru linia următoare ;
- Buffer-ul de caractere ASCII format în memorie este transferat pe suportul de pe DEV cu comanda CTRL-Z dată la sfârșitul introducerii textului ;
- Acest fișier se poate lista ca la paragraful 10.

c.2. Editarea unui fișier de comenzi XXDP pentru execuția unui lanț de teste

- Testele înlanțuibile trebuie să aibă extensia BIC ;
- Extensia acestui fișier trebuie dată de forma CCC ;
- Scrierea comenzilor se face în forma R_NAME/N (CR) unde N = numărul de treceri ale testului NAME la rularea lanțului de teste ;
- Se acceptă introducerea de linii de comentariu, care se recunosc dacă au primul caracter " ; " .
- Ieșirea din comanda TEXT se face cu CTRL-Z, iar listarea unui fișier de comenzi XXDP se face ca la paragraful 10 ;
- Rularea acestui fișier se face cu comenzile C_NAME (CR) sau C_NAME/QV (CR) explicate anterior

c.3. Editarea unui fișier de comenzi UPD2.

- EXT poate fi de exemplu de forma CMD ;
- Se pot introduce în text linii cu comenzi UPD2, și în plus linii începând cu caractere speciale, interpretate astfel :
 - Începând cu " ; " — linie de comentariu de tipărit în timpul execuției fișierului de comenzi ;
 - Începând cu „S“ — linie ca mai sus, dar la sfârșitul tipăririi procesorul intră în HALT, repornirea făcându-se cu CCNTINUE de la panou.
- Introducerea liniilor de comandă și salvarea pe suportul de pe DEV se face la sfârșit, cu CTRL-Z ;
- Execuția unui astfel de fișier se face cu comanda UPD2 : DØ_DEV : NAME.EXT (CR)
- Listarea conținutului fișierului se face ca la paragraful 10.

10. Listarea unui text sau a unui fișier de comenzi.

Se pot lista la consolă sau imprimantă fișiere cu extensia TXT, LST, MAN, CMD, CCC sau altă extensie declarată la crearea fișierului, cu comandă TEXT (din UPD2)

Comenzile sînt :

TYPE_DEV : NAME.EXT (CR)

listare la consolă

PRINT_DEV : NAME.EXT (CR)

listare la imprimantă

„Calculatoare personale și programarea lor“

FAMILIA DE CALCULATOARE PERSONALE ROMĂNEȘTI PRAE ȘI LIMBAJUL SĂU BASIC

ing. *Patrubany Nicolae*

ing. *Pop Baldi Nicolae*

mat. *Kiss Alexandru*

mat. *Socaciu Nicușor*

fiz. *Szász Detre*

I.T.C., Filiala Cluj-Napoca

0. CALCULATOARE PRAE. CARACTERISTICI FUNCȚIONALE

0.1. Introducere

Cuvîntul PRAE (a se citi pre) este un prefix latin și înseamnă un început. Numele sugerează convingerea autorilor că familia de calculatoare care poartă acest nume reprezintă un preludiv, o deschidere către un mod nou, calitativ superior, de utilizare a calculatoarelor în România.

Această familie de calculatoare a fost elaborată în întregime la Filiala din Cluj-Napoca a Institutului de Cercetări pentru Tehnică de Calcul și este produs în serie la Fabrica de Memorii Timișoara.

Prototipul PRAE (vezi figura) a fost prezentat în public la Sesiunea din Bușteni a Cercului Utilizatorilor de Microcalculatoare și Terminale Programabile din noiembrie 1983.

Familia de microcalculatoare personale PRAE este clădită pe o unitate centrală de 8 biți (microprocesorul Z 80), ea acoperind o gamă largă de aplicații, începînd cu PRAE-T care este un bun de larg consum pînă la PRAE-M un calculator profesional.

PRAE-T este destinat în primul rînd tinerilor, pentru care el poate reprezenta un excelent mijloc de aprofundare a cunoștințelor în orice domeniu. El poate fi utilizat și ca mijloc de agrement elevat. PRAE-T folosește la maximum aparatele electronice existente în gospodăriile cetățenilor :

— televizorul servește ca dispozitiv de afișare iar castofonul audio ca memorie externă pentru programe și date. Prin orientarea sa către bunurile de larg consum, acest calculator reprezintă un prim pas către o societate în care informația devine avere națională, o societate în care fiecare individ va fi capabil să folosească calculatorul în vederea prestării unei munci calitativ superioare.

PRAE-L este un calculator ce poate fi utilizat cu succes în laboratoare de cercetare și învățămînt precum și în atelierele de proiectare.

PRAE-M (M specifică faptul că este varianta maximă a familiei) este un microcalculator profesional competitiv cu produse ca APPLE II, COMMODORE 64, M 118 etc.

PRAE-M este dotat cu o unitate duală de disc flexibil de 5" 1/4 și cu un sistem de operare orientat pe disc. Astfel s-a creat o ambianță de calcul în care utilizatorul dispune de majoritatea limbajelor de programare de nivel înalt.

Familia de calculatoare PRAE prezintă o compatibilitate de sus în jos astfel încît aplicațiile elaborate pe membrii inferiori ai familiei să poată fi transpuse fără modificări pe membrii superiori.

Arhitectura hardware permite transformarea membrilor inferiori ai familiei în oricare membru superior prin simpla adăugare de elemente funcționale, servindu-se astfel interesele beneficiarilor.

Ca limbaj de programare de nivel înalt comun tuturor membrilor familiei folosește PRAE-BASIC, un interpretor deosebit de performant, compatibil cu standardurile internaționale (ANSI, CAER). Precizia de calcul este de 11 cifre semnificative.

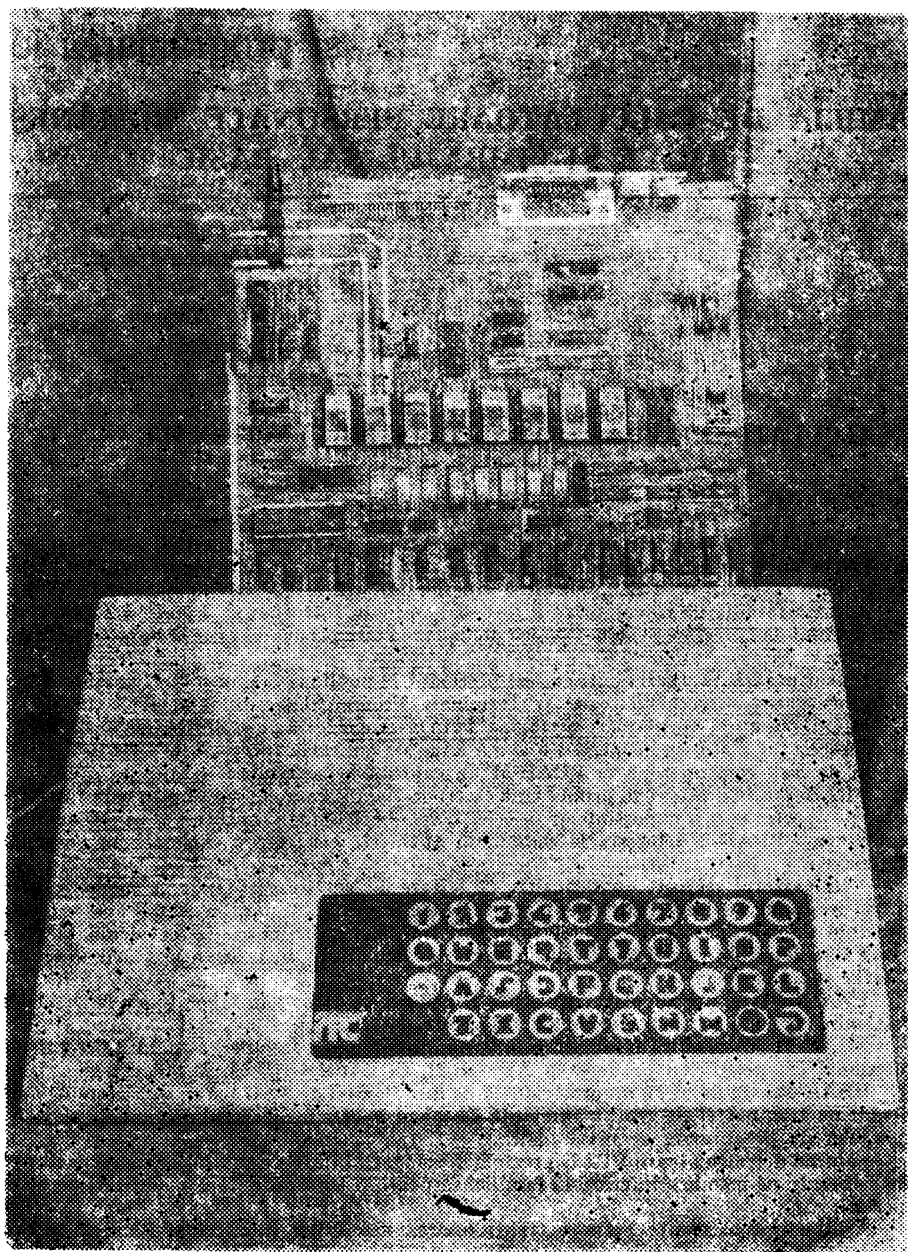


Fig. A.

Timpul de rulare pentru programul de test BM-7 acceptat internațional pentru calificarea interpretoarelor BASIC este de 53 s.

Prezenta lucrare cuprinde manualul de utilizare a componentelor software, monitorul și interpretorul BASIC V3.5 cum și anumite variabile sistem folosite de utilizatorilor. Arhitectura hardware va face obiectul unei alte lucrări.

Cea mai mare parte a lucrării este dedicată învățării limbajului BASIC-PRAE.

0.2. Software rezident

În cei 16 k EPROM se află un monitor, interpretorul PRAE-BASIC 3.5, interfață tastatură, mașină de scris și casetofon, generator de caractere.

Monitorul are următoarele comenzi :

F (Fill)	FAdr1 Adr2 Valoare	(Între Adr1 și Adr2 se umple cu valoarea Valoare)
G (Go)	GAdr1 Adr2 GAdr1 G Adr2	(Se folosește în timpul depanării unui program)
L (List)	L (CR) L (A, B, C, D, E, H, I, M, SP)	Listarea și/sau modificarea registrelor)
M (Move)	MAdr1 Adr2 Adr3	(Se mută conținutul dintre Adr1 și Adr2 începând de la Adr3)
Q (Quit)	Q (CR)	(Salt în Basic cu distrugere program Basic)
R (Return)	R (CR)	(Return în Program Basic în locul întrerupt)
S (Substitute)	SAdr	(Substituie conținutul memoriei)

Adrese mai semnificative din monitor :

01EA	Convertește o valoare binară pe 4 biți în hexa extern Valoarea binară se află în A, caracterul imprimabil se returnează în C. Nici un alt registru afectat.
01F2	Retur de car. Regiștri afectați : A, C.
0241	Convertește o valoare binară între 0—15 în hexa extern și imprimă. Valoarea se află în A. Regiștri afectați A, C.
024C	Convertește o valoare binară între 0—1255 în hexa extern și imprimă. Valoarea în A, registru C afectat.
0247	Convertește o valoare binară între 0—65535 în format hexa extern și îl imprimă.
2D5	Intrare de la consolă ; caracter recepționat în registru A.
1F9	Ieșire la consolă ; caracter de transmis în registru C.
1E5	Tipărire spațiu ; BLK
25C	Decodifică un caracter hexa extern în valoarea lui binară ; intrare în registru A ; în cazul unui caracter eronat $C_v=1$.

Interpretorul BASIC primește spre interpretare programe de la tastatură sau de la perifericul extern (casetă/bandă magnetică). Calculatorul funcționează și în regim de calculator de buzunar. Se pot executa comenzi, calcule imediate, ceea ce se identifică prin faptul că linia respectivă nu conține număr de ordine.

În cazul cînd o linie conține un număr de ordine între 1—65535 ea se structurează într-un program și se lansează în execuție prin comanda RUN.

Calculule în BASIC se fac în dublă lungime. Reprezentarea internă este flotantă. Se reprezintă pe 6 octeți în care 1 octet este exponentul binar (caracteristica) și 5 octeți mantisa.

Este permis lucrul cu masive de orice dimensiuni, se pot folosi funcții definite. Există variabile de tip șiruri etc.

Biblioteca BASIC conține în afara funcțiilor matematice și funcții referitoare la operații cu șiruri de caractere.

PRAE-BASIC are rutine pentru grafică, foarte rapide și ușor utilizabile.

Pentru a satisface nevoile celor mai diverși utilizatori amintim posibilitatea compunerii de melodii pe acest calculator avînd la dispoziție instrucțiunea BEEP.

Toate aceste posibilități oferite de BASIC vor fi detaliate la descrierea limbajului.

0.3. Software pe casetă. Extinderea interpretorului PRAE-BASIC

Pentru a nu se mări necesarul de memorie fixă (16 cocteți) anumite funcții ale interpretorului PRAE-BASIC se livrează pe casetă. Aceste funcții rulează în RAM.

Amintim două din cele mai importante module:

- a) IOARRAY — permite salvarea și recitirea de pe casetofon a variabilelor masive de tip numeric și/ sau alfanumeric.
- b) MATRIX — permite operații pe matrici cum ar fi: matricea inversă, matricea transpusă, determinantul unei matrici, adunarea, scăderea și înmulțirea matricilor, inițializarea, citirea și imprimarea matricilor.

0.4. Adrese de subprograme utile

3151	Sistem Console Input	(Caracter în A)
3154	Sistem Console Output	(Caracter în C)
3169	Sistem Console Status	(FF sau caracterul în A)
316C	Serial OUT	
317B	Serial INPUT	
31A8	Test EPROM	
3187	Scrierea pe casetă a conținutului de la 4080—40FF H	
318A	Citirea unui bloc (curent) de pe casetă la adresa 4080 H	
318D	Intrare tastatură proprie (reg A)	
3190	Ieșire pe ecran (reg C)	
319F	Console Status Serial	
31A2	Console Status tastatură proprie	
31A5	Desen miră (HL distrus)	
31A8	Test EPROM	
31AE	Test RAM (Revine după terminal afișarea adresei primei celule defecte).	

0.5. Celule de lucru, variabile sistem mai importante

BEG : 4004—4005 H	Adresă început ecran pentru rutinele de tipărire
BEGPLT : 401A—401B H	Adresă început ecran pentru rutinele de grafică
FIN : 4008—4009 H	Adresă sfârșit ecran, coincide cu adresa sfârșit RAM.
PNT : 4006—4007 H	Adresă curentă (pontor) de afișare text pe ecran.
FONDLI : 400AH	Starea SHIFT S tastatură
TIPCON : 400BH	Tip consolă (tastatură proprie sau serială)
SWITCH : 400CH	Starea după instrucțiunea SWITCH Basic (Valoare 0, 1 sau 2)
ULTCAR : 4019H	Ultimul caracter tastat în timpul execuției unui program BASIC (simulare INKEY).
STARES : 4020 H	Validarea sunetului la tastare (=0 — tastatură fără sunet =1 — tastatură cu sunet)
UPAD : 4010 H	Interfața serială
+0	Starea curentă a portului de ieșire
+1	Adresă port ieșire (80 H)
+2	Număr biți pe unitate de transfer (5, 6, 7, sau 8)
+3	Viteza (254 ₁₀ pentru 300 baud 127 ₁₀ pentru 600 baud 64 ₁₀ pentru 1200 baud etc.)
+4	Paritatea b ₀ =0 transfer fără paritate b ₀ =1 transfer cu paritate b ₁ =0 paritate pară b ₁ =1 paritate impară b ₀ = bitul de paritate rezultat în cazul transferului cu paritate
+5	Număr biți STOP (1 sau 2)
DENSIT : 4018 H	Densitate de înregistrare pe casetă (10 ₁₀ stand, DENSIT ≥ 6)
LONG : 403C—403D	Lungime bloc casetă (LONG ≤ 134 ₁₀)

1. INTRODUCERE ÎN BASIC

BASIC-ul este un limbaj de programare conversațional folosind cuvinte cheie sugestive din limba engleză și notațiile matematice familiare, ceea ce îl recomandă ca unul din limbajele cele mai ușor asimilabile.

În același timp, pentru programatorii experimentați, limbajul pune la dispoziție tehnici avansate pentru rezolvarea eficientă a problemelor de orice natură.

1.1. Convenții de notații

În cuprinsul acestui manual sînt utilizate unele convenții de notații pentru explicarea detaliată a sintaxei instrucțiunilor și a elementelor compuse ale limbajului.

Pentru descrierea sintaxei instrucțiunilor se folosesc următoarele notații:

1. Elementele scrise cu litere mari (ex. LET, PRINT, IF etc.) trebuie să apară în instrucțiuni așa cum sînt notate, ele constituind vocabularul limbajului (cuvinte cheie);

2. Elementele scrise cu litere mici și închise între paranteze unghiulare indică elementele sintactice ale unei instrucțiuni sau comenzi.

Exemplu :

```
[LET] <variabilă> = <expresie>
```

3. Parantezele drepte indică posibilitatea prezenței sau absenței elementului cuprins între ele (element opțional);

4. Parantezele acolade indică posibilitatea alegerii uneia din variantele indicate, separate prin semnul "/".

Exemplu :

```
IF <expresie> THEN {<instr.> / <nr. linie>}
[ELSE <instr.> / <nr. linie>]}
```

1.2. Programul BASIC-PRAE

Un program constă din una sau mai multe „linii program“. Din punct de vedere fizic, linia program este echivalentă cu linia terminal utilizator. „Linia program“ este definită ca avînd structura

```
<nr. linie> <lista de instrucțiuni>
```

Fiecare linie program începe cu un număr de linie care va identifica linia în contextul întregului program și indică ordinea de execuție a instrucțiunilor. O linie program poate conține una sau mai multe instrucțiuni separate prin simbolul „:“. Lista de instrucțiuni a unei linii poate fi și vidă.

Fizic, linia program reprezintă totalitatea caracterelor cuprinse între promterul emis de sistem și terminatorul unei linii terminal (CR sau "RETUR DE CAR").

Lista de instrucțiuni are forma :

```
<instr.> : <instr.> : <instr.> ...
```

Fiecare instrucțiune începe cu un cuvînt cheie exprimat în limba engleză care indică tipul operației implicate de respectiva instrucțiune.

Capitolul 3 conține descrierea tuturor instrucțiunilor BASIC-PRAE.

1.3. Număr de linie

Fiecare linie program BASIC-PRAE începe cu un număr de linie.

Numărul de linie :

- indică ordinea de execuție a instrucțiunilor programului ;
- permite schimbarea ordinii normale de execuție prin instrucțiuni de salt și cicluri ;
- permite punerea la puncta unui program dînd posibilitatea schimbării oricărei linii fără afectarea restului programului ;
- permite indicarea instrucțiunii eronate în timpul execuției programului.

Numărul de linie este un număr întreg format din 1 pînă la 5 cifre zecimale puțină avea valoarea cuprinsă între 1 și 65 529.

Programul se execută în ordinea crescătoare a numerelor de linie (ordine logică).

Remarcă : la prima scriere a unui program este recomandată numerotarea liniilor folosind rația 5 sau 10 pentru a permite, în faza de punere la punct a programului, inserarea unor instrucțiuni între instrucțiunile existente ale programului.

Introducerea unei linii avînd un număr de linie deja existent în program, provoacă ștergerea liniei vechi și înlocuirea sa prin linia nou introdusă.

Introducerea unei linii în care numărul de linie este imediat urmat de caracterul CR (RETUR DE CAR) produce ștergerea liniei cu același număr de linie dacă aceasta a existat.

1.4. Linie simplă. Linie multiplă

Instrucțiune multiplă

O linie program poate conține una (linie simplă) sau mai multe (linie multiplă) instrucțiuni separate între ele prin caracterul ' '.

Remarcă : promterul componentei BASIC care se imprimă la începutul unei linii terminal în așteptarea introducerii unei noi linii terminal este simbolul '—'.

Exemplu :

```
10 LET A=0.1
50 FOR I=1 TO 5 : PRINT I ^ 2 : NEXT I
```

Remarcă : transferul controlului poate fi executat numai la prima instrucțiune a unei linii multiple.

Orice program poate fi salvat, listat, modificat sau executat prin comenzile proprii componente.

1.5. Utilizarea caracterelor ' ' (spațiu) și TAB (CTRL I)

Caracterul ' ' (spațiu) poate fi utilizat singur sau în mod repetat oriunde în textul unei instrucțiuni pentru a servi estetica textului sursă.

De exemplu linia :

```
100 FOR I = 1 TO 10 : PRINT I 2 : NEXT I
```

poate fi scrisă :

```
100 FOR I = 1 TO 10 : PRINT I ^ 2 : NEXT I
```

mărind evident posibilitatea remarcării rapide a elementelor componente ale liniei.

Caracterul TAB poate fi de asemenea folosit oriunde în textul unei instrucțiuni pentru a fi ușor citit. Textul liniei în care se folosește caracterul TAB apare identic la listare cu textul introdus.

Exemplu de utilizare :

```
10 FOR X=1 TO 10
20 FOR Y= 1 TO 10
30 A(X, Y)=1/(X+Y-1)+A(X, Y)
40 NEXT Y
50 NEXT X
```

2. ELEMENTELE LIMBAJULUI BASIC-PRAE

2.1. Principii de realizare

Alfabetul limbajului BASIC-PRAE conține :

- a) litere : A, B, ..., Z
- b) cifre : 1, 2, ..., 9, 0
- c) caractere speciale

În setul caracterelor speciale intră toate caracterele care posedă un cod imprimabil.

Pornind de la caracterele de bază ale limbajului, folosind reguli sintactice precise (detaliat descrise la timpul potrivit) se obțin elementele compuse ale limbajului.

2.2. Elemente aritmetice

2.2.1. Constante numerice

În limbajul BASIC-PRAE toate numerele sînt tratate ca numerele zecimale conform notației matematice uzuale.

Pentru numere este folosită denumirea de constantă numerică, pornind de la faptul că acestea pentru programul în sine au tot timpul aceeași valoare.

O constantă numerică este un șir de cifre zecimale a cărui valoare reprezintă o valoare numerică (în sens obișnuit matematic). Valoarea constantelor numerice acceptate de limbaj trebuie să fie cuprinsă în intervalul real (10^{-38} , 10^{+38}).

În limbaj se recunosc trei formate de exprimare a constantelor numerice :

— format întreg 123

— format real 123.456

— format exponențial (cu puterile lui 10). 12.3E6

Constantele pot fi reprezentate în oricare din cele trei formate.

Sistemul BASIC consideră constantele ca fiind reale reprezentate în format flotant.

..Dacă constantele sînt scrise cu caracterul & în fața, ele sînt constante hexazecimale (&04AC).

2.2.2. Variabile simple

Variabila este un simbol algebric reprezentînd un număr.

Numele de variabilă este format dintr-o literă urmată de oricîte caractere alfanumerice, dintre care numai primele două sînt luate în considerare.

Valoarea variabilei este reprezentată pe 6 octeți în flotant.

De exemplu :

CORECT	INCORECT
I	2A
A8	2B
X	12

Unei variabile I se poate atribui o valoare prin :

— instrucțiunea LET ;

— instrucțiunea INPUT sau READ.

Valoarea unei variabile rămîne constantă pe intervalul cuprins între două atribuiri.

În momentul lansării în execuție a unui program, toate variabilele acestuia sînt puse la 0.

2.2.3. Variabile indexate

Pentru prelucrarea listelor, tabelelor sau matricilor limbajul posedă un element special numit „variabilă indexată”. BASIC-PRAE permite prelucrarea variabilelor indexate avînd unul, doi sau pînă la 255 indici.

Numele unei variabile indexate este format dintr-un nume admis de variabila simplă urmat de una sau două expresii de indici în paranteze.

Exemplu :

A(I), B(I, J), C (I+K, J+K)

Astfel o listă de 6 elemente căreia i se atribuie numele A poate fi reprezentată prin variabila indexată A (I) unde I poate lua valorile 0, 1, 2, ..., 5.

A(0), A(1), A(2), A(3), A(4), A(5)

Prin această convenție fiecare element al listei poate fi reprezentat printr-o variabilă indexată avînd indicele egal cu ordinul elementului în listă. O matrice (tablou) bidimensională cu numele B se poate defini și reprezenta grafic astfel :

B (0, 0), B (0, 1), . . . , B (0, N)
 B (1, 0), B (1, 1), . . . , B (1, N)
 B (2, 0), B (2, 1), . . . , B (2, N)
 B (M, 0), B (M, 1), . . . , B (M, N)

Limitele între care pot varia indicii unei variabile indexate pot fi declarate într-un program :

- implicit, folosind prima reprezentare legală a variabilei indexate ;
- explicit, prin instrucțiunea de declarare DIM.

Într-un program este posibilă definirea prin același nume a unei variabile simple și a unei variabile indexate.

Exemplu :

A — variabilă simplă reală
 A(X) — variabilă indexată reală

2.2.4. Expresii numerice

Expresia numerică este definită ca un grup de simboluri, elemente numerice ale limbajului, care poate fi evaluat. Expresiile numerice sînt compuse din constante numerice, variabile simple, variabile indexate sau combinații ale acestora separate prin operatori aritmetici, operatori de relație sau operatori logici.

2.2.4.1. Expresii aritmetice

Expresia aritmetică este compusă din elemente numerice (constante, variabile simple sau indexate), separate între ele prin operatori aritmetici.

Reprezentarea unei expresii aritmetice este similară notației matematice uzuale.

Exemplu :

$A + 0.5 * B(I, J) / 2$

Operatorii aritmetici admiși sînt :

Operator	Exemplu	Semnificație
+	A+B	adună A cu B
-	A-B	scade B din A
*	A×B	înmulțește A cu B
/	A/B	împarte A prin B
^	A^B	calculează A la puterea B
+	+A	Operația unară +
-	-A	Operația unară -

Dacă într-o expresie aritmetică apar mai mulți operatori, evaluarea expresiei se va face conform priorității matematice uzuale a acestor operatori.

Într-o expresie oarecare data, ordinea de execuție (prioritatea) este următoarea :

1) evaluarea expresiilor cuprinse între paranteze

2) în absența parantezelor ordinea este :

- a) operatori unari
- b) ridicarea la putere
- c) înmulțirea și împărțirea
- d) adunarea și scăderea

3) între operatori cu același nivel de prioritate, relația de ordine este indicată prin regula de la stînga la dreapta.

De exemplu în expresia:

$$A = B * ((C ^ 2 + 4) / X)$$

Ordinea de evaluare este :

- P1—C ^2
- P2 — rezultat P1+4
- P3 — rezultat P2/X
- P4 — rezultat P3•B
- P5 — rezultat P4 se atribuie lui A

2.2.4.2. Expresii de relație

Expresia de relație definește prin intermediul operatorilor de relație legătura între două entități.

În forma cea mai simplă, expresia de relație este compusă din două expresii aritmetice separate între ele printr-un operator de relație.

Operatorii de relație admiși sînt :

Simbol matematic	Simbol BASIC-PRAE	Exemplu	Semnificație
=	=	A=B	A egal cu B
<	<	A<B	A mai mic decît B
>	>	A>B	A mai mare decît B
<=	<=	A<=B	A mai mic/egal cu B
>=	>=	A>=B	A mai mare/egal cu B
≠	<>	A<>B	A diferit de B

Evaluarea expresiei de relație se execută prin :

- evaluarea expresiilor aritmetice componente,
- evaluarea relației dintre valorile obținute.

Valoarea unei expresii de relație este o valoare logică de adevăr („ADEVĂRAT” sau „FALS”).

Expresia de relație redusă la o expresie aritmetică exprimă în fapt relația dintre valoarea expresiei aritmetice și valoarea 0, și are valoarea însăși a expresiei aritmetice.

2.2.4.3. Expresia logică

Expresia logică este compusă din una sau mai multe expresii de relație separate între ele prin operatori logici.

Operatorii logici simbolizează operațiile logice din logica matematică obișnuită. Operatorii logici admiși sînt :

Simbol	Exemplu	Semnificație
NOT	NOT L	operația logică „NOT”
AND	L1 AND L2	operația logică „SI”
OR	L1 OR L2	operație logică „SAU”

Tabelele de adevăr ale operațiilor sînt cele din logica matematică :

L	NOT L	L1 I L2	L1 AND L2	L1 L2	L1 OR L2
A	F	A A	A	A A	A
F	A	A F	F	A F	A
		F A	F	F A	A
		F F	F	F F	F

Operația se execută bit cu bit dacă operanzii sînt în intervalul 0, 65535, altfel funcția este refuzată cu mesajul de eroare corespunzător.

Ordinea de evaluare a unei expresii logice este :

- evaluarea expresiilor de relație,

— evaluarea operațiilor logice în ordinea :

NOT
AND
OR

— între operații cu același nivel ierarhic evaluarea se face de la stînga la dreapta.

2.2.5. Funcții matematice standard

Multe din funcțiile matematice comune (sinus, funcția putere, funcția logaritmică) pot fi executate în sistemul BASIC prin intermediul unor funcții aritmetice speciale, recunoscute în limbajul BASIC-PRAE.

De exemplu, pentru calcularea valorii $\sin(75^\circ)$ se creează o expresie simbolică formată din numele simbolic al funcției respective (SIN) urmat de valoarea parametrului în radiani închis între paranteze :

SIN (75*PI/180)

Este prezentat în continuare tabelul funcțiilor matematice recunoscute ca elemente ale limbajului BASIC-PRAE :

Notăție	Simbol	Semnificație matematică
x	ABS (X)	Valoare absolută
sgn x	SGN (X)	Generează : 1 dacă $x > 0$ -1 dacă $x < 0$ 0 dacă $x = 0$
[x]	INT (X)	Parte întreagă din x
cos (x)	COS (X)	Cosinus de x (radiani)
sin (x)	SIN (X)	Sinus de x (radiani)
tg (x)	TAN (X)	Generează o valoare reală egală cu valoarea funcției tangente pentru argumentul x dat în radiani
arctg (x)	ATN (X)	Arctangenta lui x
\sqrt{x}	SQR (X)	Rădăcină pătrată din x
e^x	EXP (X)	Calculează e^x
ln x	LOG (X)	Calculează logaritmul natural din x

2.2.6. Funcții aritmetice definite

În practica de programare se observă adeseori necesitatea executării unei secvențe de instrucțiuni sau a unei expresii în mai multe puncte ale programului.

Limbajul BASIC-PRAE pune la dispoziția utilizatorilor săi posibilitatea definirii unice a unei astfel de expresii sau secvențe de instrucțiuni și simplul apel al acestora în diferite puncte ale programului.

Aceasta înseamnă că limbajul permite utilizatorului să definească funcțiile sale proprii și să apeleze apoi aceste funcții în același mod în care se apelează funcțiile aritmetice standard.

Numele unei funcții utilizator este format din maximum 4 caractere :

— primele două caractere standard egale cu „FN“

— un nume de variabilă corect (o literă urmată de maximum 1 caracter alfanumeric).

Exemplu :

CORECT
FN1
FNAB

INCORECT
FN2
FN1B

Definiția unei funcții utilizator se introduce printr-o instrucțiune DEFFN.

Apelul unei funcții se realizează prin introducerea numelui de funcție urmat de parametrul (parametrii) de apel.

La apelul unei funcții se face o corespondență strictă între numărul și tipul parametrilor formali și numărul și tipul parametrilor de apel (actuali).

Conform structurii definiției, funcțiile definite utilizator se împart în două categorii :

— funcție simplu-definită, a cărei definiție constă dintr-o expresie numerică (aritmetică, de relație sau logică)

— funcție multilinie, a cărei definiție constă dintr-o secvență de instrucțiuni cuprinse între instrucțiunile DEFFN și FNEND.

Pentru înțelegerea mai detaliată, a se vedea paragraful „Instrucțiunea DEFFN“.

2.3. Elemente nenumerice

Limbajul BASIC-PRAE permite manipularea unor informații numerice sub forma șirurilor de caractere.

Prin șir de caractere se înțelege o secvență de caractere BASIC-PRAE admise, primită ca o unitate.

Un șir de caractere poate conține litere, cifre sau caractere speciale. Lungimea unui șir de caractere se definește ca fiind numărul de caractere BASIC-PRAE componente.

În cele ce urmează vom numi „șir vid“ șirul de caractere având lungimea 0.

Pe mulțimea șirurilor de caractere se pot defini operații ca.:

— adunarea (concatenarea) a două șiruri

— extragerea unui subșir dintr-un șir dat

— modificarea elementelor componente ale unui șir etc.

Toate elementele numerice BASIC-PRAE au corespondente în setul elementelor nenumerice :

Exemplu :

constante numerice — constante șir
variabile numerice simple — variabile simple șir
masive numerice — masive șir
funcții standard numerice — funcții standard șir.

2.3.1. Constante șir de caractere

Prin constanta șir de caractere se înțelege orice șir închis între simbolurile "" (ghilimele).

O singură excepție de la această definiție există și apare în cazul constantelor șir de caractere dispuse într-o instrucțiune DATA (a se vedea paragraful „Instrucțiunea DATA“).

Exemplu :

"123-45"
"LET'S BEGIN"

2.3.2. Variabile simple de tip șir

Variabila simplă de tip șir este un simbol reprezentând un șir de caractere.

Numele unei variabile simple șir este format din :

— nume variabilă (litera urmată opțional de maximum 5 caractere alfanumerice)
— caracterul „\$“

Exemplu :

Variabile simple numerice	Variabile simple șir
A	A \$
B2	B2 \$
M	M \$

Într-un program pot exista variabile simple numerice și variabile simple șir cu același nume.

Exemple :

A, A\$

Ca și în cazul variabilelor numerice, unei variabile simple de tip șir i se poate atribui o valoare prin instrucțiunile LET, INPUT, READ.

Variabilele de tip șir sînt inițializate la execuția unui program cu valori egale cu „șirul vid”.

2.3.3. Variabile indexate de tip șir

BASIC-PRAE permite definirea variabilelor indexate de tip șir pentru rezolvarea prelucrării listelor și masivelor de șiruri de caractere.

Variabilă indexată de tip șir este un simbol care constă din :

- un nume corect de variabilă
- caracterul „\$”
- unul sau pînă la 255 indici între paranteze

Exemple :

A\$(5)
A1\$(I, J)

De reținut că expresiile de indici nu pot fi decît expresii aritmetice.

Limitele între care pot varia indicii unei variabile indexate de tip șir pot fi declarate în program :

- implicit prin prima referință
- implicit prin instrucțiunile de declarare DIM

Este posibilă utilizarea simultană a unei variabile simple șir și a unei variabile indexate șir cu același nume.

Exemplu :

A\$ și A\$(10)

De asemenea este permisă utilizarea simultană a aceluiași nume pentru variabilele numerice și variabilele șir.

Exemplu :

A, A\$, A(I), A\$(I)

2.3.4. Expresii nenumerice

Expresia șir este definită ca un grup de elemente nenumerice care poate fi evaluat.

Expresiile șir sînt compuse din constante șir, variabile simple sau indexate de tip șir sau combinații ale acestora separate prin operatori :

- operatorul de concatenare (simbol +)
- operatori de relație (expresie condiție șir)

Operatorul de concatenare marchează operația prin care din două șiruri date se poate obține un nou șir prin alipirea șirurilor date.

Operatorul de concatenare se notează prin simbolul matematic „+”. El operează numai asupra șirurilor și nu este comutativ.

Exemplu :

„ABC”+„123” produce șirul „ABC123”

Operatorii de relație operează asupra șirurilor cu respectarea ordinii alfabetice. Compararea a două șiruri se face caracter cu caracter de la stînga la dreapta.

Operatorii de relație permiși și interpretarea lor figurează în tabelul următor :

Operator	Exemplu	Semnificație
=	A\$=B\$	Șirurile A\$ și B\$ sînt identice
<	A\$<B\$	A\$ mai mic alfabetic ca B\$ sau A\$ subșir al lui B\$
>	A\$>B\$	A\$ mai mare alfabetic ca B\$ sau B\$ subșir stînga al lui A\$.

Operator	Exemplu	Semnificație
<=	A\$<=B\$	A\$ mai mic sau egal alfabetic cu B\$
>=	A\$>=B\$	A\$ mai mare sau egal alfabetic cu B\$
≠	A\$<>B\$	A\$ diferit de B\$

Expresia de relație redusă la un singur element marchează în fapt relația între elementul și "șir vid".

2.3.5. Funcții standard pe șiruri de caractere

BASIC-PRAE recunoaște ca și în cazul funcțiilor matematice standard, un set de funcții asupra șirurilor de caractere.

Apelul unei astfel de funcții se poate face în orice punct al unui program prin indicarea numelui simbolic al funcției și a parametrului sau parametrilor de apel.

Funcțiile standard pe șirurile de caractere acceptate sînt :

Cod apel	Semnificație
LEFT\$(A\$,N)	Extrage subșirul stîng de lungime N al șirului A\$. LEFT\$("ABCD",2) produce "AB".
RIGHT\$(A\$,N)	Extrage subșirul drept de lungime N din șirul A\$. RIGHT\$("ABCD",2) produce "CD".
MID\$(A\$,N1,L)	Extrage din șirul A\$ un subșir de lungime L începînd cu al N1-lea caracter. MID\$("ABCD",2,2) produce "BC".
LEN(A\$)	Generează o valoare întreagă egală cu lungimea șirului A\$. LEN("ABCD") produce 4.
CHR\$(N)	Produce un șir de lungime 1 care conține codul ASCII de valoare N. CHR(42) produce "D".
ASC(A\$)	Generează valoarea zecimală întreagă a codului ASCII a primului caracter al șirului A\$. ASC("ABC") generează 65.
INSTR(A\$,B\$,S,L)	Generează o valoare întreagă ca rezultat al căutării subșirului B\$ de lungime L în șirul A\$ începînd cu caracterul al S-lea din A\$. Dacă B\$ nu se află în A\$ se produce valoarea întreagă 0. Dacă B\$ se află în A\$ produce valoarea întreagă egală cu indicele caracterului din șirul A\$ începînd cu care cele două șiruri coincid Instr("ABCDEF", "X",2) produce 0 Instr("ABCDEF", "EF",2) produce 5. Prin convenție "șirul vid" este subșir stîng al oricărui șir.
SPC(L%)	Generează un șir de blankuri de lungime L%.
STR\$(N)	Generează un șir de caractere numerice reprezentînd valoarea de editare a numărului N. STR\$(1234567) produce "1234567".
VAL(A\$)	Generează valoarea flotantă a șirului de caractere numerice A\$. VAL("123") generează valoarea în reprezentare a numărului 123.

2.3.6. Funcțiile definite pe șiruri de caractere

Limbajul BASIC-PRAE pune la dispoziția utilizatorilor posibilitatea definirii unor funcții proprii de tip șir de caractere.

Indicația de tip se marchează prin introducerea caracterului '\$' imediat după numele funcției.

Definiția și apelul funcțiilor definite de tip șir respectă aceleași reguli ca cele de tip numeric. Funcția definită de tip șir poate fi simplă sau multilinie. Într-un program este permisă utilizarea simultană a unei funcții numerice și a unei funcții de tip șir cu același nume.

Exemplu :

FNA, FNA, FNA\$

2.3.7. Funcții grafice

Interpretorul BASIC-PRAE recunoaște 6 funcții grafice pentru a face posibilă desenarea pe ecran. Ecranul este considerat ca fiind compus din 256 linii și 256 coloane (64*1 024 puncte).

Pentru stabilirea fondului ecranului există o directivă : SWITCH valoare

Dacă valoarea este 0, ecranul are fond alb ; dacă are valoarea 1, fondul este negru ; dacă valoarea are valoarea 2 se realizează o intrare-ieșire serială.

Funcțiile grafice sînt :

Cod apel	Semnificație
CIRCLE (x, y, r)	desenează un cerc avînd centrul în punctul (x,y) de rază r
CIRCLEC (x, y, r)	șterge de pe ecran cercul cu centrul (x,) și de rază r
PLOT (x, y)	desenează un punct pe ecran de coordonare (x,y)
PLOTC (x, y)	șterge punctul de coordonate (x,y)
DRAW (x, y)	leagă cu o dreaptă ultimul punct desenat pe ecran] cu punctul de coordonate (x, y)
DRAWC (x, y)	șterge dreapta desenată
x, y, r	sînt expresii numerice, cu valori între 0 și 255.

3. INSTRUCȚIUNI BASIC-PRAE

În acest capitol este dată o descriere detaliată a instrucțiunilor admise în limbajul BASIC-PRAE.

Sînt folosite următoarele convenții și notații :

<notație>	Semnificație
<nr. linie>	Orice număr de linie admis
<instr.>	Orice instrucțiune admisă
<expr.>	Orice expresie admisă indiferent de tip (numerică sau șir)
<expr. num.>	Orice expresie numerică admisă (aritmetică, de relație, logică)
<expr. șir>	Orice expresie șir admisă
<expr. aritm.>	Orice expresie aritmetică admisă
<expr. rel.>	Orice expresie de relație admisă
<expr. log.>	Orice expresie logică admisă
<var.>	Orice variabilă admisă indiferent de tip (simplă, încetată, întregă, șir, reală)
<mesaj>	Lanț de caractere admis ca și comentariu
<cond.>	Orice expresie logică admisă, care se poate reduce la o expresie de relație sau aritmetică
<const.>	Orice constantă numerică sau șir admisă
<șir>	Orice constantă șir admisă
<var. num.>	Orice variabilă numerică admisă indiferent de tip (întreg sau real).

3.1. Instrucțiunea REM

Practica programării arată că un program bine comentat este cu mult mai ușor de testat și pus la zi decît unul necomentat.

În plus, un program bine comentat poate fi ușor înțeles și utilizat de orice alt programator decît autorul.

De remarcat că textul comentariului poate conține orice caractere, acesta fiind în întregime ignorat de tratare.

De reținut că un comentariu pe linie poate fi introdus numai după ultima instrucțiune a liniei ; în caz contrar, textul instrucțiunii este considerat ca făcînd parte din textul comentariului.

Instrucțiunea REM are formatul general :

REM [ARK] <mesaj>

unde prin mesaj se înțelege orice șir de caractere care va fi ignorat la tratare.

Exemplu :

10 REM LINIE DE COMENTARIU

Pe o linie multiplă, instrucțiunea REM trebuie să fie ultima instrucțiune a liniei.

Exemplu :

```
10 L=2*PI*R : REM LUNGIME CERC
10 L=2*PI*R : LUNGIME CERC
```

Linia :

100 REM LUNGIME CERC : L=2*PI*R

va fi tratată în întregime ca o linie comentariu.

3.2. Instrucțiunea LET

Instrucțiunea LET permite atribuirea la o variabilă simplă sau indexată a unei valori numerice sau șir.

Formatul general al instrucțiunii este :

$$[\text{LET}] \langle \text{var} \rangle = \langle \text{expr} \rangle$$

Execuția acestei instrucțiuni constă în evaluarea expresiei din dreapta semnului "=" și atribuirea valorii astfel determinate variabilei marcate în stînga semnului "=".

Instrucțiunea LET poate fi plasată oriunde în cadrul unei linii multiple.

Exemplu :

100 A=SQR (B^2+C^2) : A1=SQR (B^2-C^2)

3.3. Instrucțiunea DIM

Limajul BASIC-PRAE permite utilizarea variabilelor indexate punînd la dispoziția utilizatorilor mijloace de prelucrare a listelor și matricilor.

În BASIC-PRAE se pot trata variabile indexate cu unul, doi sau pînă la 255 indici.

Într-un program declararea masivelor se poate realiza pe două căi :

- declararea explicită prin instrucțiunea DIM
- declararea implicită prin prima referință la un element de masiv.

La declararea implicită numărul de dimensiuni a masivului declarat este egal cu numărul indicilor variabilei indexate, fiecare dimensiune fiind egală cu 10. Dacă masivul A este declarat implicit printr-o referință de forma :

$$A(\langle \text{expr.num.} \rangle)$$

atunci va avea o singură dimensiune egală cu 10. Dacă declarația implicită se realizează prin :

$$A(\langle \text{expr.num.1} \rangle, \langle \text{expr.num.2} \rangle)$$

atunci masivul A va fi un masiv bidimensional avînd dimensiunile (10,10).

Instrucțiunea DIM servește la declararea explicită a dimensiunilor unui masiv.

Formatul general :

$$\text{DIM} \langle \text{var} \rangle (\langle \text{dim1} \rangle, \langle \text{dim2} \rangle), \langle \text{var} \rangle (\langle \text{dim1} \rangle, \langle \text{dim2} \rangle)$$

unde :

var. reprezintă numele masivului și poate fi orice nume de variabilă indexată admis în limbaj.

dim1 și dim2 reprezintă dimensiunile masivului și pot fi orice expresii numerice admise de limbaj. Numărul dimensiunilor este limitat la 255.

Exemplu :

```
10 N=20 ; M=15
20 DIM A(25), B(N,N), C (10,M)
```

Prin execuția liniei 10 se declară masivul A avînd dimensiunea (25), masivul bidimensional real B avînd dimensiunile (20, 20) și masivul bidimensional șir C avînd dimensiunile (10, 15).

Posibilitatea declarării unui masiv cu dimensiuni variabile mărește mult gama utilizării acestora permițînd ca la execuții diferite a unui program spațiul alocat masivului să aibă lungimi diferite.

Fie programul :

```
10 INPUT N
20 DIM A (N, N), B(N)
100 END
```

Se observă că dimensiunile masivelor A și B și deci spațiul alocat este diferit la execuții diferite fiind indicate de utilizator prin execuția instrucțiunii din linia 10.

Declararea explicită a masivelor efectuîndu-se la execuția instrucțiunii DIM rezultă că aceasta trebuie plasată logic în program înainte de prima referință la masivele conținute, adică înainte alocării masivului ; în caz contrar, declararea explicită este tratată ca o redimensionare.

De exemplu, fie liniile program :

```
10 A(I+1)=I+1
20 DIM A(100)
```

La execuția liniei 10 masivul A cu dimensiunea implicită (10) este alocat ; execuția liniei 20 produce abandonarea execuției programului cu indicarea mesajului de redimensionare încorectă.

Se impune deci :

```
10 DIM A(100)
20 A(I+1)=I+1
```

prin care A este declarat cu dimensiunea 100.

Rezultă că este indicată plasarea instrucțiunilor DIM la începutul programelor (ordine logică).

3.4. Instrucțiunea DATA

Instrucțiunea DATA utilizată împreună cu instrucțiunea READ servește la introducerea datelor în timpul execuției programului.

De remarcat că prin acest cuplu introducerea datelor se face fără intervenția utilizatorului, fiind dispuse în memorie în blocuri de date odată cu introducerea programului.

Formatul general al instrucțiunii :

```
DATA <const.>, <const.> ...
```

Exemplu :

```
DATA BASIC, 'BASIC'
```

Instrucțiunile DATA pot fi introduse în orice loc într-un program cu condiția respectării ordinii crescătoare a numărului de linie în funcție de care se construiește și se exploatează blocul de date al programului.

Practica recomandă gruparea instrucțiunilor DATA la sfîrșitul programului pentru verificarea rapidă la testarea și execuția programului.

La comanda RUN se inițializează un pointer către prima dată din blocul de date asociat primei instrucțiuni DATA.

La fiecare moment acest pointer va indica data posibilă de citit prin READ.

3.5. Instrucțiunea READ

Formatul general al instrucțiunii :

READ <var.>, <var.>

unde :

<var.> poate fi o variabilă simplă sau indexată de tip numeric sau șir.

La execuția instrucțiunii READ se face o verificare strictă în privința corespondenței de tip dintre variabila var și data curent disponibilă.

Dacă tipurile coincid, primei variabile din listă i se atribuie data curentă după care pointerul va indica următoarea dată disponibilă din blocul de date. Dacă în blocul de date al unei instrucțiuni nu mai sînt date disponibile, se trece automat la instrucțiunea DATA următoare dacă aceasta există.

Instrucțiunea READ poate fi plasată oriunde în corpul programului.

Datele introduse prin DATA și neexploatate prin READ sînt ignorate.

3.6. Instrucțiunea RESTORE

Instrucțiunea RESTORE permite refolosirea prin READ a datelor introduse prin DATA. Acest lucru este necesar cînd într-un program este necesară folosirea în mod repetat a acelorași date.

Formatul general al instrucțiunii :

RESTORE <nr. linie>

<nr. linie> indică numărul de linie al instrucțiunii DATA pe blocul de date al căreia se dorește reinițializarea pointerului.

Dacă nr. linie nu este indicat, reinițializarea pointerului se face pe blocul de date al instrucțiunii DATA cu cel mai mic număr de linie.

Exemple :

```
50 RESTORE 1000
100 RESTORE
1000 DATA 'A—B', 5,7, .21
```

3.7. Instrucțiunea INPUT sau LISTEN

Instrucțiunea INPUT oferă utilizatorului posibilitatea introducerii dinamice a datelor necesare executării unui program. Utilizarea datelor prin DATA READ RESTORE este statică deoarece în momentul executării programului acestea sînt înghețate putînd fi modificate numai prin corectarea unei instrucțiuni DATA.

Folosind instrucțiunea INPUT utilizatorul poate introduce datele necesare execuției programului la cererea expresă a acestuia direct pe terminalul de conversație. În loc de cuvîntul cheie INPUT se poate folosi și LISTEN.

Formatul general al instrucțiunii INPUT :

INPUT <const șir>, <var.>, <var.>

unde :

<const.șir> reprezintă orice conatantă șir admisă în limbaj
<var.> reprezintă orice variabilă simplă sau indexată, numerică sau șir, admisă în limbaj

Exemplu :

```
10 INPUT "RAZA CERCULUI: ", R
```

Utilizarea <const.șir> în instrucțiunea INPUT mărește siguranța introducerii corecte a datelor necesare în fiecare moment. La execuția liniei program 10 se editează pe linie nouă mesajul :

RAZA CERCULUI :

după care se lansează efectiv cererea de introducere a valorii necesare care se va atribui variabilei R.

Introducerea datelor se consideră terminată în momentul tastării caracterului retur de car.

Dacă numărul datelor introduse este mai mic decât numărul variabilelor din lista instrucțiunii INPUT se relansează pe linie nouă cererea de introducere a datelor suplimentare necesare prin imprimarea caracterului '?'.
 Dacă numărul datelor introduse este mai mare decât numărul variabilelor din lista instrucțiunii INPUT datele introduse în exces sînt neglijate cu afișarea mesajului :

◆EXTRA LOST

Remarcă : la utilizarea instrucțiunii INPUT datele vor fi separate între ele la introducerea prin caracterul virgulă.

3.8. Instrucțiunea LINE INPUT

Prin convenția fixată la instrucțiunea INPUT caracterul ',' (virgulă) dispus pe o linie de date cerută de execuția unei instrucțiuni INPUT are semnificația de separator între date. Dacă la execuția liniei program :

```
10 INPUT A$
```

se introduce următoarea linie de date :

```
INTERPRETOR BASIC, SISTEM DE CALCUL
```

Variabilei A\$ i se atribuie ca valoare constantă șir 'INTERPRETOR BASIC' restul caracterelor de pe linia de date fiind neglijate.

Instrucțiunea LINE INPUT permite atribuirea întregului text al liniei de date introduse (cuprins între ? și caracterul retur de car) ca valoare a unei variabile șir. Dacă se înlocuiește linia 10 anterior descrisă cu linia :

```
10 LINE INPUT A$
```

prin introducerea liniei de date :

```
INTERPRETOR BASIC, SISTEM DE CALCUL
```

variabilei A\$ i se atribuie ca valoare constantă șir

```
'INTERPRETOR BASIC, SISTEM DE CALCUL'.
```

În contextul instrucțiunii LINE INPUT caracterul ',' (virgulă) își pierde semnificația de separator, întreaga linie de date (textul cuprins între ? și retur de car) fiind privită ca constantă șir.

Formatul general al instrucțiunii :

```
LINE INPUT [<mesaj>], <var.șir>
```

unde :

<mesaj> reprezintă orice constantă șir încadrată între ghilimele (''); și reprezintă mesajul de cerere a liniei.
 <var.șir> reprezintă o variabilă simplă sau indexată de tip șir de caractere admisă de limbaj.

La execuția instrucțiunii LINE INPUT linia de date introdusă are promterul :

? dacă mesaj lipsește altfel apare promterul < mesaj > ?.

3.9. Instrucțiunea PRINT

Instrucțiunea PRINT este utilizată pentru editarea anumitor date rezultate intermediare sau finale ale unei execuții la terminalul utilizatorului. Formatul general al instrucțiunilor este :

PRINT <lista>

sau

PRINT AT <expr1>, <expr2> ; <lista>

unde :

<lista> reprezintă o listă de elemente admise de limbaj separate între ele prin caracterele ',' (virgula) sau ';' (punct virgulă).
 <expr1>, <expr2> reprezintă coordonatele x,y a ecranului începând de unde se dorește scrierea. În cazul când ieșirea nu este pe ecran această parte a instrucțiunii este inefectivă. Valoarea x se trunchează la multiplu de 8.

Elementele unei liste PRINT pot fi :

- + expresii numerice sau șir care se pot reduce la constante sau variabile ;
- TAB (expr. num.) unde expr. num. poate fi orice expresie numerică admisă de limbaj ;
- + SPC (expr. num.)

O succesiune de două caractere separator semnifică absența elementului de tipărit. Lista de elemente se poate termina prin separator.

Prin execuția operatorului PRINT se construiește un text care se transmite perifericului. Acest text este rezultatul prelucrării succesive a elementelor de tipărit și a separatorilor.

Prin editarea expresiilor șir se construiește o secvență constituită din toate caracterele valori expresiei șir.

Exemplu :

```
10 PRINT "BASIC-PRAE" + "CLUJ-NAPOCA"
```

va extrage pe terminal

```
BASIC-PRAE CLUJ-NAPOCA
```

Prin editarea unei expresii aritmetice se construiește o succesiune de simboluri care constituie reprezentarea zecimală a valorii expresiei. Convențiile de editare în formatul implicit sînt :

- reprezentarea zecimală a unui număr este precedată de spațiu (nr. pozitiv) sau semnul minus (număr negativ), și urmată de blank.
- precizia reprezentării numerelor este de 11 cifre semnificative.
- Orice număr care poate fi reprezentat ca întreg va fi extras ca număr zecimal întreg (fără punct).

Numerele a căror valoare absolută este cuprinsă între 1/100 și 1 000 000 și se pot reprezenta exact sub forma unui număr zecimal, se reprezintă cu format zecimal. Zerourile nesemnificative nu se reprezintă. Restul numerelor se reprezintă în format zecimal cu exponent sub forma : semn mantisa E semn exponent spațiu cu mantisa cuprinsă între 1 și 10 avînd cel mult 11 cifre semnificative, iar exponentul cuprins între 0 și 38.

De exemplu :

- valoarea 625 se va extrage sub forma : 625
- valoarea 0.012345 se va extrage sub forma .012345
- valoarea 0.00123456 se va extrage sub forma :
1.23456E-03

Linile care vor fi extrase se împart în zone a 14 caractere (5 cimpuri pentru un terminal cu 80 caractere, respectiv 8 cimpuri pentru linia cu 132 caractere).

Separatorul ';' produce generarea în textul de tipărit a unui șir vid. Textele generate din două valori de elemente separate prin ';' se succed nemijlocit.

Separatorul ',' produce generarea în textul de ieșire după valoarea elementului care-l precede a unuia sau a mai multor caractere blank (spațiu) pînă la umplerea zonei respective.

Existența unui separator (',' sau ';') după ultimul element al listei instrucțiunii PRINT suprimă operația RETUR DE CAR și salt la linie nouă. În acest fel instrucțiunea PRINT imediat următoare va edita datele pe aceeași linie terminal.

Exemplu :

```
10 R=2. : PRINT "RAZA=";R;
15 PI=4*ATN(1)
20 PRINT "ARIA CERCULUI="; PI*R**2
RUN
RAZA=2 ARIA CERCULUI=12.566370614
```

Instrucțiunea PRINT fără listă extrage pe terminal tot șirul creat anterior, iar dacă nu există scrie o linie vidă.

Pentru un control mai eficient al formatului de editare, limbajul BASIC PRAE pune la dispoziția utilizatorilor funcția de tabulare TAB.

Funcția TAB acceptă ca parametru orice expresie numerică. La execuție se evaluează valoarea expresiei și dacă este necesar se realizează conversia real-intreg pentru a obține o valoare întreagă care va reprezenta numărul coloanei curente în care se va scrie în continuare. Deoarece numărul coloanei curente are o evoluție strict progresivă rezultă că funcția TAB este inefectivă pentru argumente negative, nule sau mai mici decât numărul coloanei curente. Dacă valoarea transmisă ca parametru este mai mare decât lungimea unei linii terminal, funcția TAB provoacă extragerea liniei curente și poziționarea la începutul liniei terminal imediat următoare. Instrucțiunea PRINT poate provoca extragerea unei sau mai multor linii terminal.

Funcția SPC acceptă ca parametru orice expresie numerică. Funcția generează altele spații cât este valoarea expresiei trunchiată la întreg.

Instrucțiunea PRINT poate fi folosită în mod imediat și poate să apară în orice poziție pe o linie multiplă.

3.10. Instrucțiunea PRINT USING

Dacă într-un program se dorește editarea datelor altfel decât cu formatul implicit, limbajul BASIC-PRAE pune la dispoziția utilizatorului instrucțiunea PRINT USING.

Formatul general al instrucțiunii este :

```
PRINT USING <format>, <listă>
PRINT AT <e1>, <e2> : USING <format>, <listă>
```

unde :

<format>	reprezintă o expresie șir ce cuprinde formatul de editare
<listă>	reprezintă o listă de elemente separate între ele prin virgulă sau punct-virgulă. Din punct de vedere sintactic lista este identică cu lista de la instrucțiunea PRINT.
<e1>, <e2>	sînt coordonatele x,y ale ecranului începînd de unde se dorește tipărirea.

Instrucțiunea PRINT USING asigură editarea datelor, rezultatelor la terminalul utilizatorului sub forma specificată la format.

În timpul execuției instrucțiunii, lista este parcursă în paralel cu șirul format. La terminarea tratării listei caracterele rămase netratate în șir dacă există, vor fi extrase așa cum apar. La terminarea șirului format elementele rămase în listă sînt editate cu formatul dat.

Exemplu :

```
PRINT USING "##.# ESTE SOLUȚIA", 12.34 ; 12.34E-2
12.34.12 ESTE SOLUȚIA
```

Separatorii dintre elementele listei nu își păstrează semnificația la instrucțiunea PRINT și funcția TAB nu mai poate figura în listă.

Șirul format este interpretat caracter cu caracter și poate conține orice caracter admis de limbaj. Pentru definirea formatului anumite caractere au semnificație specială (prin convenție). Caracterele cu semnificație specială sînt : (diez), (virgulă), * (asterisc), . (punct),

\$ (dolar), ^ (circumflex), — (minus), ' (apostrof), și în anumite forme literele : L, R, C, E. Orice alt caracter diferit de cele înșirate va fi editat așa cum apare în format :

Exemplu :

```
PRINT USING "ALFABETA/DELTA",X
ALFABETA/DELTA      O
```

3.10.1. Editarea elementelor numerice

PRINT USING editează elementele numerice cu formă formatului și ir specificat. Convențiile pentru format sînt :

— caracterul (dier) plasat în format specifică un cîmp numeric. Numărul caracterelor apărute consecutiv specifică numărul de cîmpuri ocupate de numără cîmpuri la dreapta. Cîmpurile neocupate se umplu cu zerouri.

Exemplu :

```
PRINT USING "#####",1234 ; 1.E6
1234 1000000
PRINT USING "#####",12
12
```

Dacă numărul nu începe în cîmpul specificat numărul va fi extras cu format implicit

Exemplu :

```
PRINT USING "##",135
%135
```

Numerele vor fi rotunjite dacă este cazul.

Exemplu :

```
PRINT USING "##.##",12.4 ; 12.5
12 13
```

Dacă se dorește editarea unui număr cu punct zecimal, se introduce caracterul (punct) în șirul format. Numărul caracterelor dispuse la stînga punctului zecimal în șirul format indică numărul de cifre zecimale a părții întregi a numărului ; cele dispuse la dreapta punctului zecimal dau numărul de cifre a părții zecimale a numărului.

Exemplu :

```
PRINT USING "##.##.##",1.1 ; -1.1.6
1.1 -1.2
```

Pentru editarea numerelor negative trebuie să fie introdus în șirul format un caracter pentru semn :

Exemplu :

```
PRINT USING "##.##.##.##",-1 ; -2
-1.00 -2.00
```

Dacă se dorește editarea semnului la dreapta numărului, atunci șirul caracterelor se termină cu caracterul — (minus) :

Exemplu :

```
PRINT USING "##.##.##.##-",1.0 ; -1.0
1.0 1.0-
```

Dacă în șirul format două caractere * (asterisc) preced formatul numeric, atunci cîmpul este completat la stînga de asteriscuri :

Exemplu :

```
PRINT USING "SOLUȚIA :X=**##",1
SOLUȚIA X=***1
```

Numerele negative pot fi extrase cu acest format, sau se folosește — la sfârșit de format ;

Exemplu :

```
PRINT USING "'*##",-1
*-1
PRINT USING "'*##-", -1
**1-
```

Dacă formatul numeric este precedat de două caractere \$ (dolar), la editare numărul va fi precedat de semnul \$ (dolar). Numerele negative pot fi extrase cu acest format:

Exemplu :

```
PRINT USING "PRET=##$,12
PRET=$12
PRINT USING "PRET=##$", -1
PRET=-$1
PRINT USING "PRET=##$-", -1
PRET= $1-
```

Plasarea caracterului, (virgulă) oriunde în stînga punctului zecimal între caracterele în cadrul formatului produce separarea prin virgulă în grupe de 3 (trei) cifre din stînga punctului zecimal a numărului care trebuie editat.

Exemplu :

```
PRINT USING "##,#####$,#,###",1.E6,12345-6
1,000,000 $ 12,346
```

Formatul exponențial este specificat prin caractere (circumflex) după formatul numeric. Formatul exponențial nu admite caracterele *, \$ și —. Dacă șirul format conține mai puține caractere decît 4, ele vor apare la editare așa cum sînt specificate la format. În cazul cînd un element din listă nu încapă în cîmpul specificat și este editat în mod implicit se abandonează tratarea șirului format în continuare și toate elementele vor fi extrase cu formatul implicit. Primul cîmp caracter din format este rezervat pentru semn.

Exemplu :

```
PRINT USING "## ^ ^ ^ ^", 12.56E20
1E+20
PRINT USING
"##### ^ ^ ^ ^ ^", 1E7 ; 13.2 ; 1E8
1.0E+07 13.2 ^ ^ ^ 1.0E+08
```

Formatul exponențial nu este compatibil cu descriptorii '\$' și '*'.

3.10.2. Editarea șirurilor prin PRINT USING

Formatul de editare șir furnizează informații privind : numărul de caractere alfanumerice de extras, cadrarea șirurilor (stînga, dreapta).

Descrierea formatului șir începe totdeauna cu caracterul (apostrof) și se poate continua cu un șir de caractere identice (caracterul repetat poate fi 'C', 'E', 'L', 'R'). Lungimea cîmpului șir indicat de un format șir este dat de numărul de repetiții din format mărit cu unu.

În cazul în care lungimea șirului de extras este mai mare decît lungimea cîmpului șir descris de format, la editare se vor extrage începînd de la stînga atîtea caractere cîte indică lungimea cîmpului. Dacă lungimea șirului este mai mică decît lungimea șirului format atunci șirul va fi extras în întregime cadrat conform descriptorului și completat cu blancuri. În situația în care formatul conține numai caracterul apostrof ('), se va edita primul caracter din șir :

Exemplu :

```
PRINT USING " ' ", "ABCD"
A
```

Formatul șir cu indicația cadraj la stînga este constituit din caracterul aspostrof (') urmat de o serie de caractere "L". În acest caz șirul se editează începînd cu primul caracter din stînga, de lungime egală cu numărul caracterelor "L" plus 1.

Exemplu :

```
PRINT USING " 'LLLL", "ABC"
ABC
PRINT USING " 'LLLL", "LMNOPRS"
LMNOP
```

Formatul șir cu indicația cadraj la dreapta este format din caracterul apostrof ('), urmat de o serie de caractere "R". În acest caz șirul se editează cadrat la dreapta completat eventual la stînga cu blankuri.

Dacă lungimea șirului depășește lungimea cîmpului șirul va fi extras cadrat la stînga

Exemplu :

```
PRINT USING " 'RRRR", "ABC"
ABC
PRINT USING " 'RRRR", "ABCDE"
ABCDE
PRINT USING " 'RRRR", "BASIC-PROE CLUJ-NAPOCA"
BASIC
```

Formatul centrat constă dintr-un caracter apostrof (') urmat de o serie de caractere "C". Cu acest format se editează un șir pe cîmpul format cadrat central și completat cu blankuri în stînga și dreapta.

Dacă lungimea șirului este mai mare decît lungimea cîmpului format șirul va fi cadrat la stînga.

Exemplu :

```
PRINT USING " 'CCCCCC", "ABC"
ABC
PRINT USING " 'CCCCCC", "ABCD"
ABCD
PRINT USING " 'CCCCCC", "ABCDEFGHIJKLMNO"
ABCDEFGHIH
```

Formatul extins constă dintr-un caracter apostrof (') urmat de o serie de caractere "E". Cu acest format se editează un șir pe cîmpul format, cadrat la stînga.

Dacă lungimea șirului depășește cîmpul specificat cîmpul va fi extins și tot șirul va fi extras :

Exemplu :

```
PRINT USING " 'EEE****", "A"
A ****
PRINT USING " 'EEE****", "ABCD"
ABCD****
PRINT USING " 'EEE****", "ABCDEFGH"
ABCDEFGH****
```

3.11. Instrucțiunea GOTO

Instrucțiunea GOTO servește la modificarea ordinii secvențiale de execuție a instrucțiunilor program, (salt necondiționat).

Formatul general al instrucțiunii :

```
GOTO <nr. linie>
```

unde nr. linie este orice număr de linie admis (23767).

La execuția acestei instrucțiuni controlul este transferat primei instrucțiuni a liniei indicate. Din formatul specificat este evident că nu se poate da controlul altei instrucțiuni decât la prima de pe o linie program :

Exemplu :

```
50 GOTO 200
200 X=X+1 : PRINT X
```

În exemplul dat controlul nu poate fi transferat instrucțiunii PRINT.

Dacă prima instrucțiune a liniei specificate este neexecutabilă (DATA, REM) controlul va fi transferat primei instrucțiuni care urmează.

Exemplu :

```
10 GOTO 50
50 DATA 1,2,3
60 X=A 2+B 2
```

În acest caz, controlul va fi transferat instrucțiunii $X=A+2+B$. Încercarea de transfer a controlului la o linie inexistentă se soldează cu oprirea execuției și afișarea unei erori.

Instrucțiunea GOTO poate avea orice poziție pe o linie multiplă.

3.12. Instrucțiunea ON...GOTO

Instrucțiunea ON...GOTO servește la realizarea transferului selectiv conform valorii unei expresii. Formatul general al instrucțiunii este :

```
ON <expr. num.> GO TO <nr. linie>, <nr. linie>, ...
```

La execuția instrucțiunii se evaluează expresia numerică iar partea întreagă a valorii calculate se folosește ca index în lista numerelor de linie pentru a determina linia căreia i se predă transferul. La execuția instrucțiunii :

```
ON X GOTO 100,200,300
```

controlul este transferat liniei 100 dacă X are valoarea 1, liniei 200 dacă valoarea liniei X este 2 liniei 300 dacă X are valoarea 3. În cazul în care partea întreagă este negativă, nulă, sau mai mare decât numărul elementelor din listă, controlul trece la instrucțiunea ce urmează în secvență.

Instrucțiunea ON...GOTO poate avea orice poziție pe linia multiplă.

3.13. Instrucțiunea IF

Instrucțiunea IF servește pentru realizarea unui transfer al controlului sau execuția unei instrucțiuni în funcție de valoarea logică a unei condiții. Formatul general al instrucțiunii :

```
IF <cond.> THEN <nr. linie>, <instr.> [ELSE <nr.linie>, <instr.>]
```

unde :

<cond.>	poate fi orice expresie logică de relație sau aritmetică.
<nr. linie>	poate fi orice număr de linie admis
<instr.>	poate fi orice instrucțiune alta decât : FOR, NEXT, FNEND, END, DIM, DEF, IF, REM, DATA.

Dacă valoarea logică a condiției este adevărată atunci se execută partea THEN (sau GOTO) a instrucțiunii, altfel se execută partea ELSE a instrucțiunii :

```
IF A>B THEN PRINT "ADEVĂRAT" ELSE PRINT "FALS"
```

3.14. Cieluri

Prin ciclul se înțelege un set de instrucțiuni a căror execuție se repetă pînă la îndeplinirea unei condiții. Fiecare ciclu posedă elemente caracteristice :

- variabilă de ciclu ;
- valoarea inițială ;
- test final.

Exemplu :

```
100 I=1
110 IF I 10 THEN 140
120 PRINT I, I 2, I 3
130 I=I+1 : GO TO 110
140 PRINT "SFIRSIT CICLU"
```

Fiecare ciclu are patru părți :

- inițializarea ciclului prin care se inițializează variabila ciclului. (linia 100) ;
- condiția de terminare a execuției ciclului (linia 110) ;
- corpul ciclului (linia 120).
- modificarea valorii variabilei cu pasul ciclului.

Limbajul BASIC-PRAE prin intermediul instrucțiunilor FOR și NEXT permite declararea și constituirea comodă și simplă a ciclurilor.

3.15. Instrucțiunea FOR ... TO

Instrucțiunea FOR definește începutul ciclului, formatul general al instrucțiunii FOR :

FOR <var.num> = <expr.num1> TO <expr.num2> STEP <expr.num3>

unde :

- <var. num.> poate fi orice variabilă simplă și reprezintă variabila de ciclu.
- <expr. num1> poate fi orice expresie numerică, valoarea ei reprezentînd valoarea inițială a variabilei ciclului.
- <expr. num2> poate fi orice expresie numerică, valoarea ei reprezentînd valoarea finală a variabilei ciclului.
- <expr. num3> poate fi orice expresie numerică, valoarea ei reprezentînd valoarea pasul cu care se modifică variabila ciclului.

În cazul în care indicarea pasului lipsește, pasul de ciclare se consideră implicit 1.

Variabila de control a ciclului poate fi modificată în interiorul ciclului.

Expresile care apar pe instrucțiunea FOR sînt evaluate o singură dată la execuția instrucțiunii FOR.

3.16. Instrucțiunea NEXT

Instrucțiunea NEXT definește sfîrșitul ciclului. Formatul general al instrucțiunii este :

NEXT <var.num.>

unde <var.num.> este variabila cu același nume ca și pe instrucțiunea FOR corespunzătoare. La execuția instrucțiunii NEXT se incrementează valoarea variabilei ciclului cu valoarea pasului și se face testul final al ciclului. Ciclurile FOR ... NEXT pot fi îmbricate adică corpul unui ciclu poate să conțină alt ciclu.

3.17. Subrutine

Limbajul BASIC-PRAE permite declararea unui set de instrucțiuni ca o subrutină prin intermediul a două instrucțiuni :

- Instrucțiunea GOSUB de apel a subrutinei.

— instrucțiunea RETURN, de retur din subrutina la instrucțiunea imediat următoare apelului.

Intr-un program pot fi definite mai multe subrutine. Acestea pot fi plasate oriunde în corpul programului.

3.17.1. Instrucțiunea GOSUB, instrucțiunea RETURN

Instrucțiunea GOSUB provoacă lansarea execuției unei secvențe de instrucțiuni declarate ca subrutină a programului, cu reținerea instrucțiunii care urmează în secvența instrucțiuni de apel.

Formatul general :

GOSUB <nr. linie>

În BASIC-PRAE nu există posibilitatea declarării explicite a unei subrutine, mai precis a începutului subrutinei. Controlul poate fi transferat oricărei instrucțiuni. Odată controlul transferat unei subrutine, acesta se execută pînă la întîlnirea instrucțiunii RETURN. Instrucțiunea RETURN asigură marcarea sfîrșitului subrutinei și revenirea în program la instrucțiunea care urmează după instrucțiunea GOSUB de apel.

Formatul general :

RETURN

Transferul controlului dintr-o subrutină spre exteriorul acesteia este permis, dar este mai indicată ieșirea prin instrucțiunea RETURN.

Ca și ciclurile program, subrutinele pot fi imbricate :

Exemplu :

```
10 REM IMBRICARI SUBRUTINE
20 GOSUB 100
30 STOP
100 PRINT "SINT IN SUBRUTINA 1"
110 GOSUB 200
120 RETURN
200 PRINT "SINT IN SUBRUTINA 2"
210 RETURN
```

O subrutină poate apela orice subrutină chiar și pe ea însăși. Profunzimea imbricării depinde numai de spațiul de memorie pus la dispoziția utilizatorului.

O subrutină poate conține mai multe instrucțiuni de revenire (RETURN).

3.17.2. Instrucțiunea ON ... GOSUB

Instrucțiunea ON ... GOSUB permite transferul controlului la una din subrutinele indicate.

Formatul general al instrucțiunii :

ON <expr. num.> GOSUB <nr. linie> , <nr. linie> ...

La execuția instrucțiunii se evaluează expresia numerică, partea întreagă a valorii calculate este folosită ca index în lista numerelor de linie. Controlul va fi transferat subrutinei indicată de index. Execuția instrucțiunii RETURN provoacă returul în program la instrucțiunea care urmează după ON ... GOSUB.

Instrucțiunea ON ... GOSUB poate servi la apelarea unei subrutine prin unul din punctele sale de intrare.

Exemplu :

```
50 ON X GOSUB 100,200,300
100 REM INCEPUTUL SUBRUTINEI
200 REM AL DOILEA PUNCT DE INTRARE
300 REM AL TREILEA PUNCT DE INTRARE
500 RETURN
```


Controlul este transferat liniei 100 dacă X este egal cu 1, liniei 200 dacă X are valoarea 2, liniei 300 dacă X=3, și instrucțiunii care urmează după ON ... GOSUB dacă X=0 sau X=4.

3.18. Funcții definite utilizator

Practica programării arată că deseori într-un program apare necesitatea repetării, în diverse puncte, a unei expresii matematice sau șir, sau a unui set de instrucțiuni, ceea ce duce la mărirea spațiului de memorie ocupat de program.

Pentru rezolvarea acestei dificultăți BASIC-PRAE pune la dispoziție posibilitatea definirii funcțiilor care se pot apela ca și funcțiile standard.

Funcțiile utilizator se definesc prin instrucțiunea, DEFFN. Funcția utilizator poate fi:

— funcție simplă definită a cărei definiție este data printr-o instrucțiune DEF FN.

— funcție multilinie a cărei definiție este reprezentată de un set de instrucțiuni cuprins între instrucțiunile DEFFN și FNEND.

3.18.1. Funcția simplu-definită

Formatul general al instrucțiunii DEF FN pentru definirea unei funcții simplu definite este:

```
DEF FN <fct> (<par. form.>, ...) = <expr.>
```

unde:

<fct> poate fi orice nume de funcție admis inclusiv tipul (\$).
 <par. formal> poate fi orice nume de variabilă simplă sau șir.
 <expr.> poate fi orice expresie de același tip cu tipul funcției.

Exemplu:

```
DEF FNA (X) = X 2+X-1
```

Expresia unei funcții poate conține un apel la orice funcție standard sau utilizator definită în prealabil:

Exemplu:

```
DEF FN B (X, Y) = Y 2-INT (FNA (X))
```

De reținut că tipul expresiei de definiție (numerică sau șir) trebuie să coincidă cu tipul declarat al funcției (numerică sau șir). De exemplu instrucțiunea:

```
DEF FNC $(X $, Y) = LEN (X $) + Y
```

nu este admisă.

Orice variabilă care apare în expresia funcției și nu apare în lista variabilelor formale va fi considerată ca fiind o variabilă a programului în care este definită funcția. La apelul unei funcții definite se face o verificare strictă în privința coincidenței între numărul și tipul parametrilor actuali și numărul și tipul parametrilor formali.

3.18.2. Funcția multilinie

Formatul definiției unei funcții multilinie este următorul:

```
DEFFN <fct.> (<par. form> ...)
```

Instrucțiuni care definesc corpul definiției

```
FNEND <val. funcției>
```

Instrucțiunea DEFFN pentru funcția multilinie se deosebește de cea pentru funcția simplă definită prin absența semnului '='.

Valoarea produsă la apelul unei funcții multilinie este valoarea expresiei de pe instrucțiunea FNEND.

În corpul definiției unei funcții nu au sens instrucțiunile : DATA, DEF, DIM, END.

Definiția unei funcții multilinie poate să apară oriunde în program, cu condiția ca execuția ei trebuie să precedă primul apel al funcției.

Din corpul unei funcții multilinie se poate ieși și cu instrucțiunea FNRETURN, care are sintaxa :

FNRETURN <expresie rezultat>

Variabilele ce apar în corpul definiției și nu sînt parametri formali ai funcției sînt considerate ca fiind variabile ale programului apelant.

Exemplu : (definiția funcției factorial)

```
10 DEF FNF (N)
20 IF N=1 THEN FNRETURN 1
30 FNEND FNF (N-1)*N
40 INPUT N : PRINT N;" " : "=" ; FNF(N)
```

Funcția multilinie poate fi de tip numeric sau de tip șir.

Parametrii formali pot avea orice tip. La apelul unei funcții definite se face o verificare strictă privind numărul și tipul parametrilor de apel (actuali) și numărul și tipul parametrilor formali.

Instrucțiunile din corpul definiției unei funcții multilinie pot fi introduse în orice ordine dar la introducerea trebuie să încadrate între DEFFN și FNEND care stabilesc cimpul de acțiune al parametrilor formali.

3.19. Instrucțiunea RANDOMIZE

Formatul instrucțiunii :

RANDOMIZE

Instrucțiunea RANDOMIZE poate fi plasată oriunde în program dar ea trebuie să fie totuși la începutul programului, practica recomandă utilizarea instrucțiunii RANDOMIZE după punerea la punct a programului.

3.20. Instrucțiunile STOP și END

Instrucțiunile STOP și END sînt utilizate la oprirea execuției programului. Formatul general al instrucțiunii END :

END

Instrucțiunea END poate fi dispusă și pe linie multiplă.

Instrucțiunea STOP suspendă temporar execuția unui program care poate fi oricînd reluată începînd cu instrucțiunea următoare printr-o comandă CONTINUE sau începînd cu o instrucțiune careare specificată prin GOTO <nr. linie> în mod imediat.

Formatul general :

STOP

Instrucțiunea STOP poate să apară în orice punct al programului și pe orice poziție a unei instrucțiuni multiple. La execuția instrucțiunii programul este oprit cu mesajul :

!_*BREAK LINE N

N fiind numărul de linie pe care se află STOP. Execuția programului putînd fi reluată în orice moment.

Instrucțiunea STOP oferă un mijloc foarte eficient de punere la punct a programelor cînd este utilizată în combinație cu modul imediat de lucru prin care se pot testa elementele programului.

3.21. Funcția INP

BASIC-PRAE pune la dispoziția utilizatorului funcția :

INP (<arg>)

pentru citirea unui port. Argumentul funcției este numărul portului iar caracterul citit este atribuit valorii A în exemplu :

A=INP (0)

3.22. Instrucțiunea OUT

BASIC-PRAE pune la dispoziția utilizatorului instrucțiunea OUT pentru a scrie un caracter pe un port. Formatul general al instrucțiunii este :

OUT <nr. canal>, <expr>.

Instrucțiunea dă valoarea <expr.> canalului specificat de <nr. canal>.

3.23. Funcția PEEK

BASIC-PRAE permite utilizatorului accesul la memoria fizică. Sintaxa funcției este :

PEEK (<adresa>)

unde :

<adresa> este o valoare care reprezintă adresa zecimală a locației de memorie care urmează să fie citită.

Exemplu :

B=PEEK (A)

conținutul adresei A este atribuit variabilei B.

3.24. Instrucțiunea POKE

BASIC-PRAE permite utilizatorului scrierea în memoria fizică. Sintaxa instrucțiunii este :

POKE <adresa>,<valoare>

Prin instrucțiunea POKE valoarea specificată de expresia <valoare> este atribuită locației de adresă dată de <adresa>.

3.25. Instrucțiunea BEEP

Instrucțiunea BEEP servește la producerea unui sunet cu ajutorul generatorului sonor.

Sintaxa instrucțiunii este :

BEEP <durata>, <frecvența>

unde <durata> este o valoare cuprinsă între 0 și 255 și stabilește durata sunetului ; <frecvența> este o valoare cuprinsă între 0 și 32767 și stabilește frecvența sunetului.

3.26. Instrucțiunea CALL

BASIC-PRAE permite utilizatorului apelarea unor subprograme scrise în limbaj de asamblare (cod mașină).

Formatul instrucțiunii este :

CALL <adresă>, <arg. 1>, <arg. 2>, ...

Instrucțiunea predă controlul adresei specificate. Fiecare argument se reprezintă pe doi octeți și schema de transmitere a lor este următoarea :

Răgistrii : HL adresa primului argument de pe STACK.
 BC conține numărul argumentelor aflate pe STACK
 SP conține adresa vârfului STACK-ului unde se află argumentele astfel :
 ARG.N ← SP
 ...
 ARG.2
 ARG.1 ← HL
 ADRESA DE RETUR

3.27. Funcția FRE

Funcția FRE are rolul de a furniza utilizatorului memoria liberă. Dacă parametrul este o variabilă, se dă memoria disponibilă pentru variabile iar dacă se dă o variabilă și se furnizează zona liberă și.

Exemplu :

FRE(X) furnizează zona liberă variabile.
 FRE(X\$) furnizează zona liberă și.

4. MODUL IMEDIAT

Pentru rezolvarea unor probleme simple reduse la calcularea unor expresii folosind BASIC-PRAE nu se impune scrierea unui program complet și apoi lansarea execuției acestuia.

Majoritatea instrucțiunilor limbajului BASIC-PRAE pot fi utilizate ca niște comenzi "ON LINE" și executate imediat după introducere.

Apar astfel două moduri de lucru care se pot folosi separat sau combinate dacă se utilizează sub sistemul BASIC-PRAE :

- modul imediat (direct) ;
- modul program (indirect).

Instrucțiunea folosită în mod imediat, executându-se după introducere, rezultă că nu poate fi folosită pe o linie multiplă ; de aici apare echivalența dintre instrucțiunea imediată și linia imediată.

Deosebirea dintre linia program și linia imediată constă în faptul că linia imediată nu posedă număr de linie.

Exemplu : linia :

10 PRINT "BASIC-PRAE"

este o linie program, în timp ce linia :

PRINT "BASIC-PRAE"

este o linie imediată.

Linile care încep cu un număr de linie sînt memorate într-o formă internă specială și executate ulterior la o comandă RUN.

Modul de lucru imediat este deosebit de util în două situații tipice :

- punerea la punct a programelor ;

— efectuarea unor calcule care nu ar justifica scrierea unui program.

Folosirea anumitor instrucțiuni în mod imediat nu are sens. Din această categorie fac parte instrucțiunile DATA, DEF, FNEND.

Posibilitatea marcării într-un program a punctelor de întrerupere, prin utilizarea instrucțiunii STOP, pune la dispoziția utilizatorilor un mijloc foarte eficace de punere la punct a programelor prin combinarea judicioasă a celor două moduri de lucru. În plus execuția unui program întrerupt prin STOP poate fi oricând reluată printr-o comandă CONTINUE sau printr-o instrucțiune GO TO în mod imediat.

5. COMENZI BASIC-PRAE

Introducerea unui program în memorie nu este suficientă pentru rezolvarea problemei pentru care acesta a fost proiectat.

Practica programării arată că sînt foarte rare situațiile în care un program scris este absolut corect. Apare astfel problema punerii la punct a unui program introdus. Există situații în care punerea la punct a unui program necesită listarea acestuia sau ștergerea și corectarea unor linii. Odată pus la punct pentru o utilizare ulterioară apare problema salvării programului pe un suport de unde să poată fi regăsit în orice moment. Salvările succesive ale programelor pot duce la lipsa spațiului pe suport extern pentru salvări. Apare astfel problema eliminării unor programe care nu mai sînt utilizate.

Pentru rezolvarea dificultăților mai sus enumerate BASIC-PRAE pune la dispoziția utilizatorilor un set de instrucțiuni speciale numite comenzi. Din punct de vedere al tratării o comandă poate fi privită ca o instrucțiune imediată, ele pot să apară și în program.

Linia de comandă este identică cu linia imediată fiind lipsită de număr de linie; deoarece se execută imediat după introducerea ei poate fi și linie multiplă. Între comenzi și instrucțiuni există însă o deosebire esențială constînd în aceea că în timp ce o instrucțiune operează asupra entităților dintr-un program, comanda operează asupra programului privit ca o entitate de sine stătătoare.

În funcție de tipul de operații executate asupra programului, comenzile se împart în trei categorii:

a) comenzi de prelucrare a programului curent în memoria pusă la dispoziția utilizatorului (comenzi de editare):

1. DELETE
2. LIST
3. NEW
4. RENUMBER
5. EDIT
6. AUTO

b) comenzi de execuție și asistare a execuției unui program:

8. CONTINUE
9. TRACE
10. RUN

c) comenzi de salvare (încărcare a programelor în) din bibliotecile de programe:

12. AMERGE
13. LENGTH
14. LOAD, ALOAD
15. PRECISION
16. SAVE, ASAVE
17. KILL

Formatul general al unei linii de comandă este:

<cuvînt cheie> <argumente>

Fiecare dintre paragrafele care urmează conțin informații complete privind comanda pe care o tratează.

5.1. Comenzi de editare

În categoria comenzilor de editare intră comenzile prin care se realizează ștergerea unui set de instrucțiuni, listarea parțială sau totală.

Aceste comenzi nu influențează execuția unui program și pregătesc spațiul de memorie al utilizatorului și servesc la corectarea programului.

5.1.1. Comanda DELETE

Comanda DELETE servește la eliminarea dintr-un program a unui set de instrucțiuni conținute din punctul de vedere al numărului de linie. În fapt, comanda realizează eliminarea unui set de linii program

Formatul general al comenzii:

DELETE N1—N2

unde N1, N2 poate fi orice număr de linie admis de limbaj, Se impune însă respectarea strictă a condiției:

$N1 \leq N2$

Comanda realizează eliminarea din program a liniilor având numărul de linie cuprins în intervalul închis N1, N2.

Dacă $N1 = N2$ comanda realizează eliminarea liniei cu numărul N1 (=N2) dacă aceasta există.

5.1.2. Comanda LIST

Comanda LIST servește la extragerea pe terminalul utilizatorului a programului sau a unei părți din programul curent dispus în memoria internă.

Formatul general al comenzii:

LIST {N-/-N/N1-N2}

unde N, N1, N2 poate fi orice număr de linie admis de limbaj.
În formatul:

LIST N-

comanda realizează extragerea la terminalul utilizatorului a tuturor liniilor programului având numărul de linie mai mare sau egal cu N-. În formatul:

LIST -N

comanda realizează extragerea la terminalul utilizatorului a tuturor liniilor programului având numărul de linie mai mic sau egal cu N. În formatul:

LIST N1-N2

comanda realizează extragerea la terminalul utilizatorului a liniilor programului având numărul de linie cuprins în intervalul închis N1, N2.

De reținut că în acest format se impune condiția $N1 \leq N2$. Dacă $N1 = N2$ se editează linia cu numărul $N1 (=N2)$ dacă aceasta există.

Listarea programului la un alt terminal nu poate fi realizată prin comanda LIST ci numai prin comanda LLIST.

Pentru a lista toate variabilele programului cu valorile corespunzătoare, utilizatorul are la dispoziție comanda LVAP

Sintaxa comenzii este:

LVAR

Dacă utilizatorul dorește să scrie (afișeze) pe imprimantă, atunci BASIC-PRAE îi pune la dispoziție următoarele instrucțiuni și comenzi echivalente cu comenzile din paranteză :

LLIST (LIST) ; LLVAR (LVAR) ; LNULL (NULL) ;
LPRINT (PRINT) ; LPRINT USING (PRINT USING) ;
LTRACE (TRACE) ; LWIDTH (WIDTH) ; LPOS (POS)

Pentru a fixa lungimea rândului de editare, BASIC-PRAE pune la dispoziția utilizatorului comanda WIDTH având formatul :

WIDTH <arg.>

unde <arg.> este lungimea liniei.

5.1.3. Comanda TRACE

Comanda TRACE cuplecă modul de trasare a programului pentru a se putea urmări urma programului. Sintaxa comenzii este :

TRACE <expresie>

Dacă valoarea <expresie> -1 este diferită de 0 va fi trasat (imprimit) ; adică se vor afișa numerele de linie ale liniilor executate. Dacă <expresie> are valoarea 0 programul va fi executat fără facilitatea TRACE.

5.1.4. Comanda EDIT

Comanda EDIT servește la cotețarea unei linii BASIC. Comanda are formatul general :

EDIT <nr. linie>

Unde :

<nr. linie> este numărul de linie al liniei ce urmează să fie corectată.

La această comandă mai există un set de comenzi speciale ce constă din cifre și litere, care nu sînt afișate la tastare. Numerele pot fi în intervalul 1,255.

Lista comenzilor este următoarea :

A	încarcă din nou bufferul EDIT din memorie.
ND	șterge N caractere consecutive.
E	termină editarea liniei și o înlocuiește.
NFX	caută al N-lea caracter X din linie și păstrează pointerul înaintea caracterului.
H	șterge tot ce este la dreapta pointerului și intră în modul de inserare.
I	intră în modul de inserare și inserează toate caracterele ce se introduc pînă cînd se tastează CR (return) sau ESC (SHIFT-E).
NKX	șterge caracterele de la pointer pînă la al N-lea caracter X (acesta nu va fi șters).
L	listează linia
Q	abandonează editarea fără a înlocui linia.
NR	înlocuirea a N caractere următoare cu N caractere introduse.
X	duce pointerul la sfîrșitul liniei și se intră în modul de inserare.
ELANK	duce pointerul la dreapta.
RUBOUT	duce pointerul la stînga.
CR	sfîrșitul editării liniei
ESC	sfîrșit mod de inserare.

5.1.5. Comanda NEW

Comanda NEW realizează eliminarea din memorie a programului curent pentru a face posibilă introducerea unui nou program.

Formatul general al comenzii :

NEW

Exemplu :

NEW

Prin această comandă se elimină din memorie orice informație legată de programul curent dacă acesta a existat, pregătindu-se astfel introducerea unui nou program.

5.1.6. Comanda RENUMBER

Comanda RENUMBER servește la renumerotarea liniilor BASIC. Formatul general al comenzii :

RENUMBER [*i*, <număr *i*>, <număr *p*>, <număr *b*>]

Unde :

- <număr *i*> reprezintă numărul de linie de la care se începe renumerotarea.]
- <număr *p*> reprezintă mărimea pasului dintre numerele de linie atribuite la două linii consecutive.
- <număr *b*> reprezintă numărul de linie minim ce se renumerotează.

5.1.7. Comanda NULL

Sintaxa comenzii este :

NULL <expr.>, <cod>

Unde <expr> este numărul caracterelor ce vor fi adăugate caracterelor CR și LF la orice cerere de I/E de pe terminal; <cod> este codul ASCII al caracterului. Pentru terminale lente este indicată execuția comenzii la începerea sesiunii de lucru :

NULL 3,255

Comanda NULL 0 restabilește configurația de bază.

5.1.8. Comanda AUTO

Cu comanda AUTO numerele de linie ale întregului program BASIC sînt generate automat. Formatul general al comenzii este :

AUTO [*nr. lin.*] [, <pas>]]

Unde :

- <nr. lin.> reprezintă numărul de linie atribuit primei linii din program. Valoarea implicită este 10.
- <pas> reprezintă incrementul dintre două numere de linie consecutive. Valoarea implicită este 10.

5.1.9. Comanda CLEAR

Pentru a putea șterge toate variabilele, BASIC-PRAE dispune de comanda CLEAR. Formatul general al comenzii este :

CLEAR <număr>

Dacă argumentul număr este prezent, șalocă pentru zona de șiruri lungimea număr.

5.2. Comenzi de execuție

În categoria comenzilor de execuție intră comenzile de lansare a programului, de pregătirea relansării și de continuare a unui program întrerupt. Aceste comenzi afectează direct execuția unui program.

5.2.1. Comanda CONTINUE

Comanda CONTINUE permite reluarea execuției programului din punctul de întrerupere realizat prin instrucțiunea STOP.

Formatul general al comenzii :

CONTINUE

Împreună cu instrucțiunea STOP și cu modul de lucru imediat, constituie mijloace foarte eficiente de punere la punct a programului.

Comanda CONTINUE nu poate înlocui comanda RUN și nu are sens decât în punctele de întrerupere ale unui program lansat prin RUN.

5.2.2. Comanda RUN

Comanda RUN realizează lansarea în execuție a programului curent în memorie.

Formatul general al comenzii :

RUN

În cazul unei comenzi RUN se lansează în execuție programul curent dispus în memoria internă a utilizatorului de la instrucțiunea cu cel mai mic număr de linie.

Comanda RUN poate fi lansată și dintr-un program dacă ea are număr de linie :

<nr. linie> RUN

5.3. Comenzi de bibliotecă

În categoria comenzilor de bibliotecă intră comenzile de salvare a programelor din memoria internă pe un suport extern, cele de încărcare a programelor dispuse pe un suport extern în memoria internă și cele care asigură gestiunea programelor pe suportul extern.

5.3.1. Comanda LOAD, ALOAD

Comanda LOAD servește la încărcarea unui program de pe un suport extern indicat, în memoria internă.

Formatul general al comenzilor :

LOAD <nume fișier> <adresă început>

respectiv

ALOAD₁<nume fișier>

unde nume fișier este un șir de caractere cuprinse între apostroafe iar adresa început este adresa la care se începe încărcarea.

Programele scrise în cod ASCII se pot încărcă prin comanda ALOAD. Fiecare linie începe cu un număr de linie și se termină cu caracterul CR (retur de car). Sfârșitul se detectează fie prin codul CTRL/Z din text fie prin marcajul EOF.

Înainte de a încărcă programul specificat, comanda ALOAD șterge programul vechi din memorie. În timpul căutării programului se listează pe consolă, nu nele tuturor programelor întâlnite. Un program salvat cu SAVE nu poate fi încărcat cu comanda ALOAD. Programul salvat cu ASAVE poate fi încărcat cu ALOAD.

5.3.2. Comanda PRECISION

Precizia de calcul a componentei BASIC-FRAE este stabilită la 11 cifre semnificative. Totuși, dacă utilizatorul dorește ca rezultatele să fie tipărite cu o altă precizie mai mică decât 11, poate folosi comanda PRECISION.

Formatul general al comenzii este :

PRECISION <număr>

Unde :

<număr> este un număr mai mic sau egal cu 11 și reprezintă precizia de tipărire a tuturor numerelor. (Interior se lucrează în continuare cu 11 cifre semnificative).

5.3.3. Comanda SAVE, A SAVE

Comanda SAVE produce salvarea blocului de memorie curent, dispus în memoria internă pe un suport extern.

Formatul general al comenzii :

SAVE <nume fișier> <expr. 1>, <expr. 2>

Unde :

<nume fișier> este un șir de caractere cuprins între apostroafe

<expr. 1> este adresa de la care se salvează

<expr. 2> este adresa pînă la care se salvează

La execuția comenzii SAVE se creează pe suportul indicat un fișier cu numele conținut în specificație.

Exemplu :

SAVE „D“, 3000, 6000

Această comandă SAVE salvează programul din memorie într-un fișier cu numele D.

Comanda :

ASAVE <nume fișier>

convertește programul curent aflat în memorie în cod ASCII pentru a permite salvarea programului și îl salvează apoi pe suportul extern-

5.3.4. Comanda KILL

Prin comanda KILL se recreează zona alocată pentru liste și matrici. Formatul general :

KILL <nume matrice>, <nume matrice> ...

Unde :

<nume matrice> este numele unui masiv sau liste,

5.3.5. Comanda AMERGE

Comanda AMERGE servește pentru încărcarea și interclasarea unui program de pe caseta cu programul aflat în memorie.

Sintaxa comenzii este :

AMERGE <nume fișier>

unde nume fișier este un șir de caractere cuprins între apostroafe.

În timpul căutării programului specificat se listează numele tuturor fișierelor întâlnite pe suport, astfel obținându-se repertoriul programelor existente pe casetă.

PORTABILITATEA PROGRAMELOR DE CALCULE TEHNICO-ȘTIINȚIFICE

Dr. ing. V. Sima
I.T.C.I.

În contextul diversificării echipamentelor de calcul și a creșterii continue a costurilor de elaborare a programelor, asigurarea portabilității reprezintă o cerință majoră. Lucrarea tratează câteva aspecte ale problemei portabilității rutinelor de calcule tehnico-științifice. Sunt comentate și exemplificate unele soluții adoptate de autor în implementarea pe minicalculatoare a unor pachete de programe științifice, soluții care pot fi folosite și în realizarea sau adaptarea altor programe de acest gen.

1. PORTABILITATEA PROGRAMELOR ȘTIINȚIFICE

Ultimul deceniu a înregistrat, pe lângă o creștere substanțială a volumului codurilor destinate calculelor tehnico-științifice, și importante progrese calitative în acest domeniu. Pentru exemplificare, să considerăm biblioteca de subprograme Harwell (AERE Harwell, Anglia), care se bucură de prestigiu internațional. Astfel, dacă versiunea în simplă precizie din anul 1972 a acestei biblioteci conținea aproximativ 40 000 de linii sursă, fără comentarii, neîndentate, greu de urmărit și din care circa 15% erau scrise în limbajul de asamblare pentru IBM 360, în prezent biblioteca este extinsă semnificativ și conține coduri echivalente Fortran pentru rutiunile scrise în asamblor și versiuni portabile pentru majoritatea componentelor sale.

În ultimii ani au fost concepute și realizate o serie de pachete de programe de calcule tehnico-științifice întrunind în mare măsură atributele de generalitate, eficiență, portabilitate, precizie, robustețe, comoditate în utilizare și adaptare. Menționăm câteva dintre aceste pachete, disponibile și în țară: EISPACK — pentru rezolvarea problemelor algebrice de valori proprii, LINPACK — pentru analiza și rezolvarea sistemelor de ecuații liniare, MINPACK — pentru minimizări de funcții, IMSL și SANDIA — ambele acoperind o gamă largă de calcule matematice și statistice. Pachetele EISPACK, LINPACK și MINPACK au fost realizate la Argonne National Laboratory, Argonne, IL, S.U.A. și sunt distribuite, alături de IMSL și de alte produse, de International Mathematical and Statistical Libraries (IMSL) Inc., Houston, TX, S.U.A. iar pachetul SANDIA a fost elaborat și este distribuit de Sandia National Laboratories, Albuquerque, NM, S.U.A.

Problema portabilității a fost tratată în mod diferit de laboratorii de programe. Astfel, în versiunea originală EISPACK [1] constantele dependente de mașina de calcul erau nedefinite, urmînd a fi precizate la fiecare implementare. În versiunea EISPACK integrată în pachetul SANDIA, valorile acestor parametri sînt comunicate prin intermediul unei rutine speciale. Pachetul LINPACK [2] este complet portabil. În schimb, IMSL Inc. dispune și furnizează versiuni distincte ale bibliotecii IMSL pe diferite mașini din următoarele familii de calculatoare [3]: IBM 360/370, Xerox Sigma, Data General Eclipse, Digital Equipment (seriile 11, 10/20 și VAX), Hewlett-Packard 3000 (seriile II și III), Univac 1100, Honeywell 6000, Burroughs 6700/7700 și CDC 6000/7000 și Cyber 70/170.

Lucrarea prezintă unele soluții utilizate de autor în realizarea unei versiuni portabile a bibliotecii de subprograme IMSL. Aceste soluții au fost deja folosite, dar pot fi folosite și în viitor, și pentru elaborarea sau adaptarea altor pachete de programe de calcule tehnico-științifice. De asemenea, mostrele de coduri date în lucrare permit cunoașterea stilului în care sînt scrise rutiunile IMSL și pot constitui modele de programare în Fortran demne de urmat. În plus, subprogramele originale listate în Anexele 2 și 3 pot fi direct utilizate în alte programe.

Biblioteca IMSL cuprinde peste 500 de subprograme în simplă precizie și aproximativ 350 de subprograme în dublă precizie, scrise în limbajul Fortran IV standard și grupate pe capitole notate cu literele alfabetului. Adaptarea bibliotecii s-a făcut pe un minicalculator Coral 4011, pornind de la o variantă Felix (compilată cu opțiunea de „dublă lungime“ DBL), furnizată de Oficiul de calcul al IRNE Pitești, variantă având la bază versiunea în simplă precizie pe calculatorul CDC Cyber 170/720. Dată fiind amploarea lucrării, trecerea pe minicalculator s-a făcut în câteva etape, rezultând mai multe biblioteci și pachete de programe. Dintre acestea menționăm bibliotecile: EIGAN (care implementează subrutinele din capitolul E, destinate analizei și rezolvării problemelor algebrice de valori și vectori proprii), TSPACK (analiza și predicția seriilor de timp — capitolul F) și VEMA (calculul cu matrice și vectori — capitolul V), și colecțiile de biblioteci: STAT (calculul statistice diverse — capitolele A, B, C, N, O, R, S, și parțial M și U) și MATH (calculul matematic diverse — capitolele D, E, G, I, L, V, Z, și parțial M și U). La realizarea versiunii în simplă precizie și-a adus contribuția și dr. ing. Th. D. Popescu (I.C.I.), care a adaptat și implementat subprogramele capitolelor B, C, D, F, I, R, S, și parțial M. De asemenea, pentru obținerea fișierelor în format exploatabil pe minicalculator au fost folosite programele de conversie elaborate de ing. Fl. Harțescu (I.C.I.).

Principalele dificultăți ivite în elaborarea variantei portabile a bibliotecii IMSL s-au datorat următoarelor elemente:

- prezența constantelor în cod;
- modul de memorare a cuvintelor întregi;
- calculul cu variabile complexe;
- calculul cu precizie extinsă.

Aceste probleme sînt tratate în secțiunile următoare ale lucrării. Menționăm că ultimele două aspecte, legate de realizarea versiunii în dublă precizie a bibliotecii pornind de la versiunea în simplă precizie, au implicat modificări însemnate în codul sursă. Astfel, lipsa unui compilator cu opțiune pentru calculul cu numere complexe în dublă precizie a necesitat codificarea în Fortran a unor operații aritmetice de bază. De asemenea, anumite calcule intermediare, care reclamau o precizie deosebită, au fost efectuate apelînd rutine Fortran speciale, cu precizie extinsă (aproape cvadruplă).

2. TRATAREA CONSTANTELOR DIN COD

Foarte multe programe de calcule tehnico-științifice necesită utilizarea unor *parametri dependenți de mașina de calcul* concretă, cum ar fi: cel mai mic (mare) număr reprezentabil, baza reprezentării în virgulă mobilă, precizia relativă a mașinii etc. Prezența explicită în cod a unor constante reprezentînd valorile acestor parametri reduce portabilitatea, făcînd necesară efectuarea de modificări la fiecare nouă implementare pe o altă mașină cu parametri diferiți. În biblioteca IMSL, acești parametri apar de regulă în instrucțiuni de inițializare DATA. Există însă și cazuri în care se utilizează variabile a căror valoare este fixată prin instrucțiuni de atribuire. Pentru depistarea acestor cazuri au trebuit studiate toate subprogramele, întrucît în general testele obișnuite în execuție nu detectează inadecvauța valorilor respective.

Soluția implementată pe minicalculator, care conferă maximum de portabilitate, a fost cea utilizată și în biblioteca SANDIA, și anume: parametrii dependenți de mașină sînt comunicați fiecărei rutine care îi reclamă utilizînd subprogramele funcție IIMACH, R1MACH sau DIMACH, adaptate după PORT Mathematical Subroutine Library (Bell Laboratories, S.U.A.). IIMACH, R1MACH și DIMACH furnizează valorile unor constante întregi, reale și, respectiv, dublă precizie, pentru următoarele familii de calculatoare: Burroughs 1700, 5700 și 6700/7700, CDC 6000/7000, Cray 1, Data General Eclipse, Harris 220, Honeywell 600/6000, IBM 360/370, Xerox Sigma 5/7/9, PDP 10 (cu procesor KA sau KI), PDP 11 și Univac 1100. Particularizarea acestor subprograme pentru o anumită mașină se face simplu prin înlocuirea cu blanc a caracterului C din prima coloană a liniilor de comandă rîu corespunzătoare mașinii respective. IIMACH, R1MACH și DIMACH au un singur parametru, I, a cărui valoare precizează care anume constantă este cerută. Tabelele 1 și 2 prezintă semnificația valorilor funcțiilor IIMACH (I) și respectiv R1MACH (I) (sau DIMACH (I)).

Semnificația valorilor funcției IIMACH (I)

Tabel 1

I	IIMACH (I)	
1	unitatea de intrare standard	
2	unitatea de ieșire standard	
3	unitatea de perforare standard	
4	unitatea standard pentru mesaje de croare	
5	numărul de biți pe unitatea de memorare (cuvinți) de tip întreg	
6	numărul de caractere pe cuvânt întreg	
7	a, baza reprezentării numerelor întregi	
8	s, numărul de cifre în baza a	
9	$a^s - 1$, cel mai mare număr întreg reprezentabil	
10	b, baza reprezentării numerelor în virgulă mobilă	
11	t, numărul de cifre în baza b	
12	e_{min} , cel mai mic exponent	} pentru reprezentarea în simplă precizie
13	e_{max} , cel mai mare exponent	
14	t, numărul de cifre în baza b	} pentru reprezentarea în dublă precizie
15	e_{min} , cel mai mic exponent	
16	e_{max} , cel mai mare exponent	

Semnificația valorilor funcțiilor R1MACH (I) și D1MACH (I)

Tabel 2

I	R1MACH (I), D1MACH (I)
1	$b^{e_{min}-1}$, cel mai mic număr pozitiv reprezentabil
2	$b^{e_{max}}(1-b^{-t})$, cel mai mare număr reprezentabil
3	b^{-t} , cea mai mică spațiere relativă
4	b^{1-t} , cea mai mare spațiere relativă (precizia relativă)
5	$\log_{10}(b)$

Parametri dependenți de mașină apar în zeci de rutine din IMSL. În aceste rutine au fost suprimate instrucțiunile de inițializare sau de atribuire explicită și au fost incluse apeluri la subprogramele IIMACH, R1MACH sau D1MACH. Dificultăți mai mari am întâmpinat, de pildă, la adaptarea rutinelor destinate evaluării unor funcții speciale, matematice sau statistice (capitolul M), unde au trebuit identificate constante corespunzătoare valorilor în (R1MACH (I)), pentru I=1, 2, 3, sau alte constante de acest gen.

Vom da acum câteva detalii despre modificările făcute în cadrul bibliotecii EIGAN [4]. În rutinele EBALAC și EBALAF este apelată funcția IIMACH pentru a obține baza reprezentării în virgulă mobilă b. În câteva cazuri a fost posibil să se elimine complet parametrii dependenți de mașină. De exemplu, s-a suprimat parametrul RDELP (sau REPS), care denotă precizia relativă a mașinii, din codurile rutinelor IECUSS, EIFFIC și EQRTS, prin înlocuirea instrucțiunilor de forma

```
IF (TEST .LE. RDELP*TEST1) GO TO ...
```

cu instrucțiuni

```
IF (TEST1+TEST .EQ. TEST) GO TO ...
```

Trebuie remarcat că cele două tipuri de instrucțiuni nu sînt întotdeauna echivalente. Astfel, pe mașinile avînd aritmetică cu *rotunjită exactă* în simplă precizie cîturăm $1.+ALPHA$ în E. 1., pentru $ALPHA \geq EPS \triangleq 1/2R1MACH(4)$ și $1.+ALPHA = 1.$, pentru $ALPHA < EPS$. Instrucțiunile de al doilea tip sînt folosite și în pachetul LINPACK [2], ceea ce a permis ca acest pachet să nu conțină nici o constantă dependentă de mașină. Modificări similare au fost efectuate și în alte subrutine din EIGAN (anume ELRH2C, EQRH3F, EQRT1S și EQRT3S). Totuși, în aceste rutine, ca și în subrutinele EBNDV, EQRH1F și EQRT2F, variabila RDELP a fost reținută (dar furnizată de funcțiile R1MACH, respectiv D1MACH), căci apare ca numitor în unele situații rare care altfel ar implica împărțiri prin zero. De asemenea, în subrutinele de interes general (EIGBS, EIGCC, EIGCH, EIGRF, EIGRS, EIGZC și EIGZF), parametrul RDELP este utilizat în evaluarea indicelui de performanță.

În afara parametrilor dependenți de mașină, în coduri pot apare unele *constante corespunzătoare algoritmului* implementat. Se recomandă ca la elaborarea programelor, constantele în virgulă mobilă să fie definite prin instrucțiuni de inițializare sau, eventual, prin instrucțiuni de atribuire grupate în prima secțiune a codului, după declarațiile de dimensiune și tip. Acest lucru înlesnește obținerea unui cod portabil și permite elaborarea comodă a versiunii în dublă precizie. Trebuie evitată utilizarea explicită a unor astfel de constante în instrucțiuni de calcul. Aceste recomandări au fost avute în vedere și în cazul bibliotecii originale IMSL. Totuși, în multe subprograme apar constante reale utilizate explicit. La realizarea variantei portabile a bibliotecii, cel puțin în acele rutine pentru care trebuia elaborată și versiunea în dublă precizie am înlocuit constantele din cod cu variabile inițializate prin DATA, completând corespunzător declarațiile de tip. Acest lucru a redus riscul ca în versiunea dublă precizie să rămână constante în simplă precizie, asigurându-se portabilitate și precizie sporite. (Menționăm că unele compilatoare nu fac conversia de tip în expresii mixte, deci este uneori posibilă o pierdere substanțială de precizie.)

În codurile originale, constantele inițializate prin DATA aveau maximum 14 zecimale (corespunzător versiunii în simplă precizie pentru calculatoarele CDC). Acolo unde a fost necesar și cu puțință, la elaborarea versiunii în dublă precizie am extins reprezentarea la 18 zecimale. Astfel, pentru unele funcții speciale am utilizat tabelele disponibile (de exemplu, pentru funcția Erfc [5]). De asemenea, extinderea reprezentării s-a putut face ușor în cazul fracțiilor periodice. În alte cazuri, am preferat să calculăm efectiv valoarea constantei respective (de pildă, $(1 + 17.1^{1/3})/8.$), soluție folosită și în pachetul LINPACK [2].

O problemă colaterală care a apărut în faza inițială de testare în execuție a constituit-o apariția unor erori fatale de „violare memorie” pentru subrutine cu declarații de tipul

```
SUBROUTINE XX (TAU, ...)
REAL TAU (2, 1)
```

tabloul TAU putând avea mai multe coloane și fiind dimensionat corect în programul apelant. Eșecul se datora opțiunii implicite de *vectorizare* a tablourilor a compilatorului FOR, care în exemplul prezentat tratează TAU ca un tablou monodimensional cu 2 elemente. Am considerat secvența de mai sus portabilă, introducând în bibliotecă modulul obiect generat fără vectorizare (opțiunea /NOVA).

3. UTILIZAREA TABLOURILOR ÎNTREGI

Probleme de portabilitate pot apare și datorită variației lungimii în biți a cuvintelor întregi și a caracterelor pentru diferite mașini. În tabelul 3 se prezintă valorile funcției IIMACH (I), pentru I=5, 6, 10 și 11. IIMACH (5) și IIMACH (6) reprezintă numărul de biți și, respectiv, numărul de caractere pe cuvint întreg. Valorile pentru IIMACH (10) și IIMACH (11) definesc baza b a reprezentării în virgulă mobilă și, respectiv, numărul de cifre ale mantisei în baza b, pentru reprezentarea în simplă precizie. Cifrele din coloana "Cod" a tabelului indică numărul de biți pe cuvint real. Un cuvint dublă precizie ocupă un spațiu de memorie dublu.

Valorile funcției IIMACH (I), I=5, 6, 10, 11

Tabel 3

Nr crt.	Familia de calculatoare	Cod	IIMACH (I)			
			I=5	I=6	I=10	I=11
1	Burroughs 1700	H36	36	4	2	24
2	Burroughs 5700/6700/7700	H48	48	6	8	13
3	CDC 6000/7000	H60	60	10	2	48
4	Cray 1	H64	64	8	2	48
5	Data General Eclipse S/200	H32	16	2	16	6
6	Harris 220	H32	24	3	2	23
7	Honeywell 600/6000	H36	36	6	2	27
8	IBM 360/370	H32	32	4	16	6
9	Xerox Sigma 5/7/9	H32	32	4	16	6
10	PDP-10	H36	36	5	2	27
11	PDP-11	H32	16	2	2	24
12	Univac 1100	H36	36	6	2	27

Una din problemele de portabilitate, determinată de *inegalitatea cuvintelor reali întreg*, apare atunci când un parametru formal declarat de tip întreg într-o rutină este tratat ca real (sau dublă precizie) în (sub)programul apelant, fiind totuși necesară utilizarea valorilor întregi comunicate de rutină. Acesta este cazul subrutinei LINV3F din IMSL. Versiunea originală a acestei rutine dă rezultate eronate pe calculatoarele din familiei PDP-11 sau pe minicalculatoarele românești. În Anexa 1 este listată în întregime versiunea în dublă precizie elaborată. Instrucțiunile fără numerotare în dreapta sînt cele adăugate pentru portabilitate. Listingul include și comentariile, cu toate că sînt în limba engleză, căci acestea pot servi ca model de documentație minimală pentru un subprogram de calcul științific.

Subrutina LINV3F operează asupra unei matrice A de ordin N, memorată într-un tablou de formă A(NA, N), și eventual, asupra vectorului B de dimensiune N. În funcție de opțiunea IJOB, se calculează: inversa matricei A (IJOB=1), soluția ecuației $Ax=B$ (IJOB=2), soluția ecuației și inversa (IJOB=3), și determinantul (IJOB=4). În toate aceste cazuri se determină întâi o factorizare triunghiulară a unei matrice obținută prin permutarea liniilor lui A, deci $PA=LU$, unde P este matricea de permutare, L este o matrice inferior triunghiulară cu elemente diagonale (necunoscute) egale cu 1, iar U este o matrice superior triunghiulară. Această factorizare se obține utilizînd subrutina LUDATF. Pentru rezolvarea sistemului $Ax=B$ este apelată în continuare subrutina LUELMF. LUDATF necesită doi vectori de lucru de dimensiune N, unul întreg, IPVT, în care se înregistrează informații referitoare la permutările efectuate (care definesc matricea P), și un vector real, EQUIL, al cărui conținut nu prezintă importanță în acest context. Informațiile din vectorul IPVT sînt utilizate atât de LUELMF (pentru IJOB=2, sau 3), cît și direct de LINV3F, pentru a obține în final inversa $A^{-1}=U^{-1}L^{-1}P$ (pentru IJOB=1, sau 3). Spre a nu extinde lista de parametri ai subrutinei originale LINV3F, a fost însă prevăzut un singur tablou de lucru real WKAREA, de dimensiune 2N, care este folosit atât pentru IPVT, cît și pentru EQUIL. Pe calculatoarele cu cod H32 și cuvinte întregi pe 16 biți, această soluție este uzuală, întrucît un element al vectorului WKAREA conține două (sau patru) elemente ale vectorului IPVT în versiunea în simplă (dublă) precizie. Aparent, pentru versiunea în simplă precizie situația ar putea fi rezolvată simplu înlocuind în LUDATF și LUELMF declarațiile

```

      INTEGER      IPVT(1)
cu              REAL      IPVT(1)

```

dar, aceasta nu ar corespunde decu necesității și ar implica efectuarea de modificări similare în toate (sub)programele care le apelează. De asemenea, sîm șobse rvăm că o declarație de tip INTEGER*4 ar fi rezolvat problema în simplă precizie pentru mașinile pe 16 biți, dar nu ar fi fost operantă pe alte mașini, și ar fi implicat modificări în alte rutine. În sfîrșit, o altă rezolvare posibilă ar fi fost adăugarea unui tablou întreg în lista de parametri ai rutinei LINV3F, cu actualizarea corespunzătoare a documentației și a tuturor subprogramelor care apelează LINV3F.

Soluția portabilă împleneată nu afectează interfața subrutinei LINV3F și nu necesită nici o modificare în alte rutine. Se detectează automat cazul mașinilor cu lungime a cuvîntului întreg mai mică decît a cuvîntului real, comparînd IIMACH(5) cu $MAX \frac{A}{B} \max$ (IIMACH(10), IIMACH(11)). Conform Tabelului 3, dacă IIMACH(5) > MAX sîntem în situația standard, cu lungimi egale (STAN= .TRUE., în listing), exceptînd calculatorul Harris 220, pentru care oricum soluția IMSL nu este posibilă. În caz contrar (STAN= .FALSE.), se ține seama că WKAREA(J/2), J par (sau WKAREA(J/4), J multiplu de 4), conține informații cu privire la permutările liniilor J-1 și J (respectiv J-3, J-2, J-1 și J), pentru versiunea în simplă (dublă) precizie. Soluția este tehnic ceva mai complicată datorită faptului că permutările se aplică în ordine inversă. Menționăm că modificările aduse nu afectează practic performanțele codului.

Ambele versiuni ale subrutinei LINV3F modificate au fost testate și au funcționat corect. De asemenea, au fost simulate pe minicalculator și alte mașini. Spre exemplu, pentru calculatoarele din familiile FELIX C-253 sau IBM 360/370, s-a testat funcționarea versiunii în simplă

precizie prin înlocuirea declarațiilor INTEGER IPVT(1) din rutinele LUDATF și LUELMF cu REAL IPVT(1), și prin folosirea unui cod ad-hoc pentru IIMACH :

```
INTEGER FUNCTION IIMACH(I)
```

```
IF(IEQ.5) IIMACH=32
```

```
IF(IEQ.10) IIMACH=16
```

```
IF(IEQ.11) IIMACH=6
```

```
RETURN
```

```
END
```

Alte probleme de portabilitate au apărut la unele rutine utilitare, de afișare a rezultatelor. Vom ilustra acest lucru și vom indica o soluție posibilă portabilă în cazul cel mai simplu : afișarea unui vector folosind rutina USWFV. Primele declarații în codul original sînt :

```
SUBROUTINE USWFV(ITITLE, NC, A, N, INC, IOPT)
```

```
INTEGER ITITLE(1), INC, IOPT, N, NC, NCA, NCMAX, NW
```

```
REAL A(1)
```

```
DATA NCMAX/20/
```

Subrutina permite afișarea elementelor $1, INC+1, 2*INC+1, \dots$, ale vectorului A, de dimensiune N, cu un format selectabil prin opțiunea IOPT, conform tabelului 4.

Tabel 4

Opțiuni pentru formatul de afișare (IOPT)

Opțiuni pentru 129 coloane	Opțiuni pentru 80 coloane	Format
1	2	F18.5
3	4	E15.6
5	6	E25.ISIG

ISIG din Tabelul 4 (pentru IOPT= 5, sau 6) are o astfel de valoare încît să furnizeze o reprezentare cu precizie aproape completă a elementelor vectorului A. Valorile lui ISIG sînt precizate în Tabelul 5, conform documentației [3].

Tabel 5

Numărul de cifre zecimale afișate (ISIG)

ISIG	Precizie	Cod
7	simplă	H32
16	dublă	H32
9	simplă	H36
11	simplă	H48
14	simplă	H60

Rutina permite opțional afișarea unui titlu avînd $NC \leq NC_{MAX} = 20$ caractere. Dacă $NC \leq 0$, acest titlu nu este afișat. Titlul este transmis prin tabloul întreg ITITLE, dar parametrul efectiv poate fi însă o constantă Hollerith, ca în exemplul de apel de mai jos :

```
CALL USWFV (1CHTEST USWFV,10,A,15,1,6)
```

De remarcat că aici $INC=1$. O valoare $INC \neq 1$ poate fi folosită, de exemplu, pentru afișarea elementelor unei linii dintr-o matrice ($INC > 1$), sau pentru afișarea elementelor unui vector în ordine inversă ($INC = -1$).

Apar astfel două probleme de portabilitate : *comunicarea și afișarea titlului și tipărirea valorilor la precizia mașinii* (IOPT=5, 6). Versiunea originală a rutinei USWFV nu este portabilă. Pentru afișarea titlului s-a utilizat formatul 2A10, iar ISIG a fost fixat la 14 (valori pentru CDC). Pentru mașinile cu cod H32 trebuie însă folosit ISIG=7, în simplă precizie, iar formatul pentru titlu poate fi 1CA2 (pentru minicalculatoare pe 16 biți), sau 5A4 (pentru FELIX C-256). Soluția portabilă realizată implementează pentru cele două probleme menționate formatele variabile IFM și IFS, respectiv, inițializate și actualizate corespunzător în timpul execuției, utilizând informațiile I1MACH(5) și I1MACH(6). Datorită limitărilor de spațiu, nu este posibil să listăm întreaga rutină. Totuși, considerăm instructiv să prezentăm structura codului modificat, insistând asupra modificărilor efectuate.

C URMĂTOARELE 20 DE LINII SÎNT ADĂUGATE PENTRU PORTABILITATE

INTEGER IFM(6),IFS(7),NUM(5),NUS(3),I1MACH,NBIT,NC2,NC5

DATA IFM/2H(1, 2HX,, 2H 4, 2HA , 2H5 , 1H)/

DATA IFS/2H(4, 2HX,, 2H5E, 2H25, 2H. , 2H7 , 1H)/

DATA NUM/2H 2, 2H 4, 2H 6, 2H 5, 2H10/

DATA NUS/2H 9, 2H11, 2H14/

C PREGĂTIRE FORMAT E25.ISIG

IF (IOPT .LT. 5) GO TO 1

NBIT=I1MACH(5)

IF (NBIT .LT. 36) GO TO 1

NBIT=NUS(NBIT/12-2)

IFS(6)=NBIT

1 NCHAR=I1MACH(6)

C PREGĂTIRE FORMAT AFIȘARE TITLU

IF (NCHAR .EQ. 5 .OR. NC .LE. 0) GO TO 2

NC2=NCHAR/2

NC5=5-NC2

IF (NCHAR .NE. 6) NC5=NC5+1

IFM(3)=NUM(NC5)

IFM(5)=NUM(NC2)

2 CONTINUE

C PREGĂTIRE ȘI AFIȘARE TITLU

NCA=MIN0(NC,NCMAX)

NW=(NCA-1)/NCHAR+1

CALL UGETIO (1,NIN,NOUT)

C URMĂTOAREA LINIE — FORMAT PORTABIL

IF (NC .GT. 0) WRITE(NOUT, IFM) (ITITLE(I), I=1,NW)

.

.

.

C AFIȘARE VECTOR

GO TO (30,30,35,35,40,40), IOPT

.

.

.

C URMĂTOAREA LINIE — FORMAT PORTABIL

40 WRITE(NOUT,IFS) (A(I),I=1,N,INC)

.

.

.

Subrutina elaborată implică un număr minim de operații suplimentare față de versiunea originală, dar operează pe toate calculatoarele din Tabelul 3, exceptând Cray 1 și Harris 220. Testarea completă a funcționalității s-a făcut prin simularea pe minicalculator a altor mașini. De exemplu, pentru a simula FELIX C-253, am înlocuit declarația INTEGER ITITLE(1) cu REAL ITITLE(1) și am utilizat rutina ad-hoc IIMACH listată mai jos

```

      INTEGER FUNCTION IIMACH(I)
      IF (I .EQ. 5) IIMACH=32
      IF (I .EQ. 6) IIMACH=4
      RETURN
      END
  
```

4. CALCULUL CU NUMERE COMPLEXE

Elaborarea versiunii în dublă precizie a bibliotecii IMSL a implicat o serie de modificări substanțiale în coduri. Dintre modificările generale, efectuate în toate subprogramele, menționăm: schimbarea declarațiilor de tip REAL în DOUBLE PRECISION; adaptarea corespunzătoare, inclusiv completarea valorilor numerice din instrucțiunile DATA; schimbarea numelor funcțiilor intrinseci (ABS, SQRT, SIGN etc). Amploarea acestor acțiuni nu trebuie subestimată; de pildă, există și rutine în care lista variabilelor inițializate prin DATA se întinde pe două pagini de listing. După efectuarea modificărilor indicate, codurile au fost compilate cu opțiunea /LI: 2, pentru a putea analiza listele de variabile locale, subprogramele apelate și funcțiile intrinseci referite și pentru a stabili corectitudinea formală a textului sursă. Menționăm că există și cazuri în care dintr-un subprogram în dublă precizie este apelat unul în simplă precizie, de exemplu pentru a obține o anumită distribuție de probabilitate. Aceste cazuri au trebuit tratate cu atenție, pentru a se respecta cerințele referitoare la tipul variabilelor.

În afara modificărilor generale, într-o serie de rutine au fost necesare modificări speciale pentru a permite efectuarea anumitor operații cu numere complexe. Majoritatea compilatoarelor Fortran, și în particular și cele disponibile pe minicalculatoarele românești, nu pot opera cu numere complexe având părțile reală și imaginară reprezentate în dublă precizie. Excluzând din prima versiune a bibliotecii în dublă precizie circa 10 subprograme, care ar fi trebuit practicat rescrise complet, din analiza codurilor originale a rezultat că este suficientă realizarea a trei operații de bază, și anume:

- calculul valorii absolute a unui număr complex dublă precizie;
- calculul rădăcinii patrate a unui număr complex dublă precizie;
- împărțirea a două numere complexe dublă precizie.

Aceste trei operații au fost implementate în trei subprograme originale: DCABS, DCSQRT și, respectiv, DCDIV. DCABS este un subprogram de tip funcție, iar celelalte două sînt subrutine. Toate au la bază algoritmi ALGOL prezentați în [6] și au fost scrise în Fortran IV, pentru portabilitate. Dat fiind faptul că aceste subprograme pot fi utilizate și în alte programe de calcule tehnico-științifice, am considerat oportun să le listăm în Anexa 2 (fără comentarii, din motive de spațiu). Pentru apelare comodă, listele de parametri conțin separat părțile reale și imaginare ale numerelor complexe care intervin. (Soluția alternativă, de grupare a părților reale și imaginare în vectori cu cîte 2 componente, ar fi necesitat modificări mai ample în rutinele apelante, fără a aduce un spor de eficiență.) Considerînd numerele complexe $x=(XR, XI)$, $y=(YR, YI)$ și $z=(ZR, ZI)$, DCABS, DCSQRT și DCDIV calculează $DCABS=|x|$, $y=x^{1/2}$ și, respectiv, $z=x/y$. De remarcat că în toate cele trei subprograme nu se lucrează direct cu parametrii formali, ci cu variabile locale, pentru eficiență mai mare — prin evitarea adresărilor indirecte — și pentru a permite suprascriserea rezultatelor peste date. De asemenea, aceste subprograme pot fi ușor transcrise într-un limbaj mașină, dacă o aplicație specială cere viteză de execuție mai mare și cod mai compact decît rezultă în Fortran.

Menționăm că DCABS, DCSQRT și DCDIV au fost folosite și în versiunea în dublă precizie a pachetului EIGAN [4], pentru determinarea vectorilor proprii ai matricelor generale, utilizînd algoritmul QR. Cu același scop, aceste subprograme au fost incluse și în biblioteca matematică generală MIMAT [7], disponibilă pe microcalculatoarele M18 și M118. Întrucît compilatoarele Fortran cu care sînt înzestrate microcalculatoarele românești nu posedă opțiunea de calcul cu numere complexe, nici măcar în simplă precizie am implementat și versiunile cores-

punzătoare în simplă precizie ale rutinelor prezentate mai sus, anume: CABS, CSQRT și CDIV. Testele efectuate au confirmat corectitudinea și eficiența tuturor implementărilor.

În final, pentru a ilustra posibilitățile oferite de utilizarea rutinelor DCABS, DCSQRT și DCDIV indicăm câteva modificările mai ample efectuate pentru elaborarea variantei în dublă precizie a subrutinei MDGCI, destinată determinării inversei unei funcții generale de distribuție a probabilității, fiind date ordcratele densității. Subrutina folosește interpolare liniară sau neliniară pe o rețea de puncte echidistante sau neechidistante. Este necesară rezolvarea unei ecuații algebrice de ordin 1, 2, 3, sau 4. Calculul cu numere complexe intervine la rezolvarea ecuațiilor cubice sau cvartice. Vom considera cazul ecuațiilor cvartice. În versiunea în simplă precizie, structura codului, incluzind doar una din secvențele modificate, este următoarea:

```

REAL      EB,CC,D,FOURTH,HALF,PRED,QRED,SIXTH,THIRD,
1         THREE,TWNT7,ZER
COMPLEX   AL,BT,Z
DATA      FOURTH/.25E0/,HALF/.5E0/,THREE/3.0E0/,ZER/0.0E0/,
1         SIXTH/.16666666666667E0/,
2         THIRD/.33333333333333E0/,
3         TWNT7/.03703703703704E0/

```

```

Z=CMPLX(-QRED*HALF,ZER)
Z=Z+CSQRT(CMPLX(PRED*PRED*PRED*TWNT7+QRED*QRED*FOURTH,ZER))
Z=CEXP(CMPLX(THIRD,ZER)*CLOG(Z))
Z=Z-CMPLX(PRED,ZER)/(THREE*Z)+CMPLX(CC*SIXTH,ZER)
AL=CSQRT(Z+Z+CMPLX(FOURTH*BB*BB-CC,ZER))
BT=(BB*Z-CMPLX(D,ZER))/(AL+AL)

```

Funcțiile CMPLX, CSQRT și CEXP sînt tratate de compilatorul Fortran. De remarcat că este necesar să se lucreze cu numere complexe datorită posibilității ca partea reală a numărului complex din care se extrage radicalul prima cară să fie negativă.

Codul în dublă precizie elaborat, echivalent matematic cu cel prezentat mai sus, este listat în continuare.

```

DOUBLE PRECISION EB,CC,D,FOURTH,HALF,PRED,QRED,SIXTH,THIRD,
1         THREE,TWNT7,ZER
DOUBLE PRECISION ALI,ALR,BTI,BTR,ZI,ZR,ZZI,ZZR
C         Z=(ZR,ZI), AL=(ALR,ALI), BT=(BTR,BTI)
DATA      FOURTH/.25D0/,HALF/.5D0/,THREE/3.0D0/,ZER/0.0D0/,
1         SIXTH/.1666666666666667D0/,
2         THIRD/.3333333333333333D0/,
3         TWNT7/.0370370370370370D0/

```

```

ZR=-QRED*HALF
CALL DCSQRT (PRED*PRED*PRED*TWNT7+QRED*QRED*FOURTH,ZER,ZZR,
1ZZI)
ZR=ZR+ZZR
ZZR=DEXP(THIRD)
ZR=ZZR*ZR
ZI=ZZR*ZZI
CALL DCDIV (PRED,ZER,THREE*ZR,THREE*ZI,ZZR,ZZI)
ZR=ZR-ZZR+CC*SIXTH
ZI=ZI-ZZI
CALL DCSQRT (ZR+ZR+FOURTH*BB*BB-CC,ZI+ZI,ALR,ALI)
CALL DCDIV (BB*ZR-D,BB*ZI,ALR+ALR,ALI+ALI,BTR,BTI)

```

5. CALCULUL CU PRECIZIE EXTINSĂ

Convenim să numim aritmetică sau reprezentare cu *lungime simplă* (LS) aritmetica și respectiv reprezentarea în virgulă mobilă — simplă sau dublă precizie — disponibilă pe calculatorul folosit. Anumite calcule matematice, cu n ar fi: determinarea produsului scalar a doi vectori cu dimensiune mare, sau evaluarea unor funcții speciale, impun utilizarea unei precizii extinse față de cea oferită de aritmetica LS. În acest scop, au fost elaborate tehnici care permit descrierea și realizarea reprezentării și aritmeticii cu *lungime multiplă* în funcție de reprezentarea și aritmetica LS. De exemplu, un număr cu *lungime dublă* (LD) este exprimat prin suma a două numere LS, unul dintre ele fiind aproape neglijabil în precizia LS. În cele ce urmează vom considera doar calcule în aritmetica LD și vom prezenta soluții portabile pentru efectuarea unor operații aritmetice cu precizie extinsă. Pentru aceasta, sînt însă necesare unele elemente și precizări preliminare.

Preliminarul. Să notăm cu R mulțimea numerelor în virgulă mobilă (LS) reprezentabile în calculator și să presupunem că R este de forma

$$R = \{x \mid x = mb^e, |m| < M, -D < e < E\}, \quad (1)$$

unde m, b, e, M, D , și E sînt numere întregi a căror semnificație rezultă. Neluarea în considerare a depășirilor aritmetice echivalează cu a considera D și E infinite. Fie $x, y \in R$ și $*$ o operație aritmetică binară pe R (de exemplu $+$, $-$, \cdot , $/$). Notăm $fl(x*y)$ rezultatul obținut cu calculatorul prin aplicarea operației $*$ asupra numerelor x și y (utilizînd aritmetica LS).

Spunem că operația în virgulă mobilă corespunzătoare lui $*$ (pe scurt, operația $*$) este *corectă* dacă pentru $\forall x, y \in R$, $fl(x*y)$ este fie cel mai mare element din R mai mic decît sau egal cu $x*y$, fie cel mai mic element din R mai mare decît sau egal cu $x*y$. Deci, $fl(x*y) \in \{a, b\}$, unde $a, b \in R$ și $[a, b]$ este cel mai mic interval cu extremități în R astfel încît $x*y \in [a, b]$. Dacă $x*y \in R$, $fl(x*y)$ este rezultatul corect. Operația $*$ este *optimală* (sau *cu rotunjire corectă*) dacă pentru $\forall x, y \in R$, $fl(x*y)$ este elementul din R cel mai apropiat de $x*y$.

Adunarea în virgulă mobilă este *cu trunchiere corectă* dacă $fl(x+y) = fl(y+x)$ (comutativitate) și, pentru $\forall x, y \in R$, cu $|x| \geq |y|$, $fl(x+y)$ este cel mai mare element din R mai mic decît sau egal cu $x+y$, dacă $y \geq 0$, sau cel mai mic element din R mai mare decît sau egal cu $x+y$, dacă $y < 0$. Așadar, dacă $x+y \in R$, rezultatul este trunchiat în direcția lui $-y$. Scăderea este *cu trunchiere corectă* dacă pentru $\forall x, y \in R$, avem $fl(x-y) = fl(x+y')$, unde $y' = -y$ și adunarea este *cu trunchiere corectă*. Adunarea și scăderea sînt *supercorecte* dacă pentru $\forall x, y \in R$, $fl(x \pm y)$ este obținut fie prin rotunjire corectă, fie prin trunchiere corectă.

Notăm $z = fl(x*y)$ (deci $z \in R$) și fie zz corecția necesară astfel încît să fie satisfăcută *exact* relația

$$z + zz = x*y.$$

În anumite condiții [8], se poate arăta că $zz \in R$ și că zz este (aproape) neglijabil cu precizia mașinii în raport cu $x*y$ (deci cu $z+zz$).

Spunem că perechea (z, zz) , $z \in R$, $zz \in R$, este un *număr aproape LD*, dacă

$$|zz| \leq |z+zz| C \cdot 2^{-t} \quad (2)$$

unde C este o constantă nu mult mai mare decît 1, iar t este numărul de cifre binare ale mantisei în reprezentarea LS. Dacă în (2), $C = 1/(1+2^{-t})$, perechea (z, zz) este un *număr LD*.

Operații exacte. Algoritmii pentru calculul cu precizie extinsă au la bază adunarea exactă și înmulțirea exactă a două numere LS. De aceea, prezentăm în continuare proceduri pentru realizarea acestor operații și condiții suficiente pentru valabilitatea rezultatelor.

Procedura cea mai simplă pentru *adunarea exactă* a numerelor x și y , $x, y \in R$, cu $|x| \geq |y|$, este următoarea

$$z = fl(x+y)$$

$$w = fl(z-x), \quad zz = fl(y-w). \quad (3)$$

Dacă $|x| < |y|$, rolurile lui x și y se inversează. Seturi de condiții suficiente pentru ca z și zz obținute să fie astfel încît $z+zz = x+y$ sînt, de exemplu, a) sau b) indicate mai jos:

a) În (1) $b=2$ sau 3, M este multiplu de b ; adunarea este optimală sau supercorectă, iar scăderea este corectă.

b) În (1) b și M sînt oarecare ; adunarea este cu trunchiere corectă, iar scăderea este corectă în condițiile enunțate, $z-x \in R$, $y-w \in R$, deci w și zz sînt exacte ($w=z-x$, $zz=y-w$). De asemenea, dacă într-un sistem binar, adunarea este optimală, atunci (z, zz) este număr LD, deci zz satisface (2) cu $C=1/(1+2^{-t})$. Condițiile nu sînt însă suficiente dacă la calculul lui zz apar depășiri inferioare, ceea ce este posibil în sistemele în virgulă mobilă cu normalizare. Dacă adunarea și scăderea sînt optimale, dar b și M sînt oarecare, nu se poate garanta că $w=z-x$. În acest caz, soluția se poate obține cu o procedură mai complicată, introducînd un termen de corecție pentru w :

$$\begin{aligned} z &= fl(x+y), \\ w &= fl(z-x), \quad z1 = fl(y-w), \\ v &= fl(z-w), \quad z2 = fl(v-x), \\ zz &= fl(z1-z2). \end{aligned}$$

Relațiile (3) se pot scrie echivalent

$$w = fl(x-z), \quad zz = fl(w-y)$$

astfel încît codul Fortran corespunzător procedurii simple pentru $|x| \geq |y|$, este

$$\begin{aligned} Z &= X + Y \\ ZZ &= X - Z + Y \end{aligned}$$

unde variabila w nu trebuie explicitată, ținînd seama de ordinea de evaluare a expresiilor aritmetice în Fortran.

În cazul *înmulțirii exacte* vom presupune, pentru simplitate, următorul set de condiții suficiente :

c) În (1), $b=2$, $M=2^t$, $D=E=\infty$, deci R este de forma

$$R = R(t) = \{x \mid x = m2^e, \mid m \mid < 2^t\}; \quad (4)$$

adunarea și scăderea sînt optimale, iar înmulțirea este corectă. Procedura pentru înmulțire exactă conține două etape. În prima etapă numerele x și y sînt descompuse fiecare în cîte două numere cu „lungime jumătate“

$$x = hx + tx, \quad y = hy + ty,$$

iar în a doua etapă se efectuează produsul exact utilizînd hx , tx , hy , ty . Secvența calculelor este următoarea :

$$\begin{aligned} p &= fl(x \cdot c), \quad q = fl(x \cdot p), \quad hx = fl(q + p), \quad tx = fl(x - hx) (= x - hx) \\ p &= fl(y \cdot c), \quad q = fl(y \cdot p), \quad hy = fl(q + p), \quad ty = fl(y - hy) (= y - hy) \\ p &= fl(hx \cdot hy), \quad q = fl(hx \cdot ty + tx \cdot hy) \end{aligned} \quad (5.a)$$

$$z = fl(p + q), \quad zz = fl(p - z + q + tx \cdot ty) \quad (5.b)$$

unde $c = 2^{t-[t/2]} + 1$, $[.]$ denotînd partea întregă. Justificarea procedurii se bazează pe faptul că, prin alegerea lui c , $hx \cdot hy \in R(t)$, $hx \cdot ty$, $tx \cdot hy \in R(t-1)$ (și au același exponent) și $tx \cdot ty \in R(t-2)$. Ca atare, cu ipotezele din c) referitoare la operații, în (5.a) p și q sînt exacte, deci $z + z1 = p + q$, cu $z1 = fl(p - z + q) \in R(t-1)$. De asemenea, în (5.b) $z1 + tx \cdot ty \in R$, astfel încît rezultă $z + zz = x \cdot y$.

Se poate arăta că procedura furnizează un număr (z, zz) aproape LD, căci $|zz| < < |x \cdot y| \tau 2^{-t}/(1 + \tau 2^{-t})$, unde $\tau = 2$ pentru t par și $\tau = 3$, în caz contrar. Rezultatul (z, zz) poate fi transformat într-un număr LD efectuînd în final o adunare exactă a lui z cu zz :

$$w = z, \quad z = fl(w + zz), \quad zz = fl(w - z + zz).$$

Aritmetica LD. Procedurile prezentate mai sus pot fi aplicate direct pentru a obține algoritmi de calcul în aritmetica LD. Pentru simplitate, presupunem îndeplinite condițiile c).

Pentru a calcula suma a două numere (aproape) LD se poate aplica

Procedura AD : $(z,zz) = (x,xx) + (y,yy)$ (pentru $|x| \geq |y|$)

- 1) Se adună exact x cu y , obținând (r,rr) astfel încât $r+rr=x+y$.
- 2) Se corectează $rr : s = fl(rr+yy+xx)$, astfel încât $r+s$ aproximează $(x,xx) + (y,yy)$.
- 3) Se adună exact r cu s , obținând (z,zz) , astfel încât $z+zz=r+s$.

Pentru scădere se procedează similar. Pentru a calcula produsul a două numere (aproape) LD se poate utiliza

Procedura MUL : $(z,zz) = (x,xx) \cdot (y,yy)$

- 1) Se înmulțesc exact x și y , obținând (r,rr) aproape LD, cu $r+rr=x \cdot y$.
- 2) Se corectează $rr : s = fl(x \cdot yy + xx \cdot y + rr)$, astfel că $r+s$ aproximează $(x,xx) \cdot (y,yy)$.
- 3) Se adună exact r cu s , obținând (z,zz) , astfel încât $z+zz=r+s$.

Analog se pot descrie procedurile pentru operațiile de împărțire și rădăcină patrată, implicând de asemenea înmulțiri și adunări exacte. În [8] sunt prezentate codurile ALGOL corespunzătoare operațiilor $+$, $-$, \cdot , $/$, și $\sqrt{\quad}$. Esențial pentru însemnătatea acestor proceduri este faptul că analiza erorii furnizează rezultate de forma

$$|E^*| \leq (|x+xx| |y+yy|) C^* 2^{-2t} \quad (6)$$

unde E^* denotă eroarea în efectuarea operației $*$, iar C^* este o constantă nu mult mai mare decât 1. (De exemplu, $C \leq 9 + \tau$, cu τ definit anterior). Formula (6) arată că eroarea este comparabilă cu cea care ar fi produsă lucrând efectiv cu un sistem LD, cu reprezentare $R(2t)$ și aritmetică proprie, deși de fapt calculele se fac cu aritmetica LS. Tehnica este eficace atunci când sistemul LS adoptat este sistemul cu cea mai mare precizie disponibil pe măsura de calcul. Pentru mașinile cu opțiune de dublă precizie se pot obține rezultate cu precizie cvadruplă. De notat că efectuarea operațiilor nu necesită acumulator cu lungime dublă și că implementarea algoritmilor se poate face eficient în limbaje de înalt nivel, asigurându-se astfel portabilitatea.

Aplicații și exemple. Tehnicile prezentate au fost aplicate la obținerea versiunilor în dublă precizie ale unor subprograme din biblioteca IMSL. După cum rezultă din documentația [3], în IMSL sunt prevăzute trei subrutine pentru calculul cu precizie extinsă, anume VXADD, VXMUL și VXSTO, care sunt apelate de aproximativ 70 de subprograme. Întrucât versiunea originală CDC se livrează numai în simplă precizie, nu am avut la dispoziție codurile sursă pentru aceste subrutine. Codurile originale elaborate sunt prezentate (fără comentarii) în Anexa 3. Listele de parametri din VXADD, VXMUL și VXSTO sunt respectiv (A,ACC), (A,B,ACC) și (ACC,D), unde A, B și D sunt variabile în dublă precizie, iar ACC este un tablou cu două elemente în dublă precizie. ACC are rol de acumulator conținând reprezentarea LD a rezultatului unei operații efectuate. Astfel, dacă (z,zz) este un număr (aproape) LD, ACC(1) conține z , iar ACC(2) conține zz . Subrutina VXADD adună numărul A la acumulator. VXMUL adună produsul numerelor A și B la conținutul acumulatorului și furnizează rezultatul tot în acumulator. În sfârșit, VXSTO transformă numărul LD din acumulator într-un număr LS, obținut în D.

Subrutina VXADD realizează operația $(z,zz) := (x,C) + (z,zz)$ prin particularizarea pentru $xx=C$ a procedurii AD, prezentată mai sus. Sunt tratate adecvat ambele posibilități: $|x| \geq |y|$ și $|x| < |y|$.

Subrutina VXMUL realizează operația $(z,zz) := (x,0) \cdot (y,0) + (z,zz)$, parcurgând următoarele etape:

- i) Se efectuează înmulțirea exactă a lui x cu y , rezultând (v,vv) aproape LD.
- ii) Se transformă (v,vv) într-un număr LD (u,uu) , utilizând adunarea exactă.
- iii) Se calculează suma $(u,uu) + (z,zz)$, utilizând procedura AD.

Formația de erori și obținerea cu particularizarea procedurii MUL pentru $xx=yy=0$. Etapa i) a fost implementată folosind procedura AI CCL *null2* din [8]. Constanta $c = 2^{1-|l|/2} + 1$, necesară pentru descrierea metodei, a fost inițializată prin instrucțiunea DATA. Această soluție nu este portabilă, însă un procedeu puțin eficient, întrucât se recurge volumul codului și nu este necesară evaluarea constantelor la fiecare apel al rutinei, apeluri care înțelegerea se fac în cod. (Evident, codul ar fi putut fi complicat astfel încât evaluarea lui c să se facă numai la primul apel într-un program.) De altfel, chiar algoritmul care stă la baza rutinei VXMUL nu pare satisfacerea unei condiții care nu sunt general valabile.

Subrutina VXMUL dezvoltată aiciă de cea din IMSL. Așa cum rezultă din documentația [3], versiunea IMSL apelează de 4 ori rutina VXADD, în timp ce în versiunea listată în Anexa

3 acest lucru este evitat. Ca atare, estimăm o economie de timp de calcul unitate centrală echivalentă cu 12 operații de adunare și 4 apeluri de rutină.

Subrutina VXSTO realizează operația simplă $d=z+zz$, (z,zz) fiind numărul (aproape) (LD memorat în acumulator).

Pentru *testarea rutinelor* de calcul cu precizie extinsă s-au evaluat sumele

$$\sum_{i=1}^{1000} |a_i|, \quad \sum_{i=1}^{1000} a_i b_i$$

unde $a_i=vx_i$, $b_i=vy_i$, și x_i și y_i fiind numere generate aleator în intervalul (0,1), iar $v=ds$, unde d se modifică după fiecare 100 de eșantioane prin program, s fiind un factor de scală cu valoare fixată la o rulare. În diferite rulări am modificat inițializarea șirurilor aleatoare, iar s a avut valorile 1, 10, 100, 1 000, 1.E5, 1.E15, 1.E-10 etc. Valori pentru s prea mari, de exemplu $s=1.E20$ (sau prea mici, ca $s=1.E-15$), au produs depășiri aritmetice supracare (inferioare) explicabile. Problemele de test prezentate au fost rezolvate în simplă precizie cu trei programe diferite:

TEST1 — program în simplă precizie pură;

TEST2 — program cu calcule intermediare în dublă precizie, folosind funcția DBLE a compilatorului Fortran;

TEST — program apelind rutinele VXADD, VMUL și VXSTO scrise în simplă precizie. Nu am utilizat un program complet în dublă precizie pentru ca datele de intrare să poată fi identice. Din analiza comparativă a rezultatelor obținute au reieșit următoarele concluzii

○ În toate cazurile subrutina VXADD s-a comportat perfect, dovedindu-se echivalență calculelor în dublă precizie. Într-adevăr, valorile afișate de programele TEST și TEST2 aveau 14—17 cifre zecimale semnificative identice, cu 7—9 cifre exacte în plus față de valorile afișate de TEST1. De notat că la testele preliminare VXADD a furnizat aparent doar 1—2 cifre exacte în plus față de TEST1. De fapt, valorile maror erau greșite, datorită prezenței în TEST2 a instrucțiunii TEMP=TEMP+DABS(A(I)), în loc de TEMP=TEMP+DABS(DBLE(A(I))).

○ În toate cazurile subrutina VMUL s-a comportat foarte bine. Valorile afișate de programele TEST și TEST2 aveau 13—15 cifre zecimale identice, cu 6—8 cifre exacte în plus față de cele afișate de TEST1.

○ VXADD și VMUL necesită aproximativ de 3,5 și, respectiv, de 4 ori mai mult timp unitate centrală decât utilizarea funcțiunii DBLE a compilatorului.

Aceste concluzii pot fi probabil extrapolate și în cazul versiunilor în dublă precizie ale rutinelor VXADD, VMUL și VXSTO. Pentru a putea aprecia riște rezultate numerice și în acest caz ar fi fost însă necesare valori maror cu precizie cvadruplă.

Ținând seama de consumul mai mare de timp implicat de utilizarea rutinelor cu precizie extinsă, în biblioteca VEMA [9], destinată operațiilor de bază cu matrice și vectori, pentru acele rutine care apelează VXADD, VMUL și/sau VXSTO am elaborat și versiuni în care aceste apeluri sint suprimate, care nu utilizează deci precizie extinsă. Această soluție permite mai mare flexibilitate, făcând posibilă adaptarea codurilor la necesitățile problemelor concrete.

Menționăm că versiunile în simplă precizie pentru VXADD, VMUL și VXSTO pot fi folosite pe acele calculatoare sau sub acele sisteme de operare care nu au acumulator pe dublu cuvânt, deci care nu tratează realmente opțiunea DBLE a compilatorului Fortran. Aceasta este, de pildă, cazul unor sisteme pe microcalculatoare.

Ca exemplu banal de utilizare a rutinelor de calcul cu precizie extinsă, prezentăm o rutină pentru evaluarea produsului scalar a doi vectori A și B, de lungime N.

```

DCUBLE PRECISION FUNCTION DSCAL (A,B,N)
DCUBLE PRECISION      A,ACC(2),B,ZERO
DATA                    ZERO/0.0D0/
ACC(1)=ZERO
ACC(2)=ZERO
DO 1 I=1,N
1 CALL VMUL (A(I),B(I),ACC)
CALL VXSTO (ACC, DSCAL)
RETURN
END

```

În încheiere, vom discuta succint câteva exemple de utilizare directă în coduri a facilităților de calcul cu precizie extinsă. Acest lucru este esențial în primul rând la evaluarea unor funcții matematice speciale. Astfel în rutina MMBSJ0 din IMSL, destinată evaluării funcției Bessel de speța I și ordin 0, se adoptă o reprezentare LD pentru constanta 2π și se iau măsuri pentru a prezerva precizia, folosind instrucțiunile de calcul de forma

$$\text{PROD} = ((\text{AX} - \text{XO1}) - \text{XO2}) * (\text{AX} + \text{XO1})$$

unde (XO1, XO2) este un număr LD. La fel, în subrutina MMBSYN, destinată calculului funcției Bessel de speța II cu ordin fracționar nenegativ și argumente pozitive, se utilizează o tehnică de sumare cu precizie extinsă, similară celei din VXADD, și se introduce o variabilă suplimentară pentru a preveni o posibilă eroare de optimizare a codului, efectuată de compilator. Un exemplu simplu pentru păstrarea preciziei apare în rutina MMDELK (integrala eliptică completă de speța II); pentru evaluarea expresiei $\tau = 1 - x^2$ se folosește următorul cod

$$Y = \text{ABS}(X)$$

$$\text{ETA} = (H - Y) + H$$

$$\text{ETA} = \text{ETA} + Y * \text{ETA}$$

unde H este inițializat prin instrucțiunea DATA H/0.5D0/.

6. CONCLUZII

Asigurarea portabilității este una dintre principalele cerințe impuse programelor de calcul tehnico-științifice. Rezolvarea eficientă a diferitelor probleme privind portabilitatea permite transferul comod și sigur al programelor de la o familie sau categorie de calculatoare la alta. Datorită restricțiilor de spațiu, în lucrare s-au făcut referiri doar la unele aspecte ale problematicei portabilității, și anume: tratarea constantelor dependente de mașină și algoritmi, soluționarea diferențelor de reprezentare (mai precis, lungime) a întregilor, calcule cu numere complexe sau cu precizie extinsă. Au fost prezentate unele soluții concrete, originale, utilizate de autor în implementări pe mini- și microcalculatoare a unor pachete de programe valoroase, făcându-se referiri și la rezultatele testelor efectuate, inclusiv prin simularea altor mașini. Aceste soluții pot fi aplicate cu ușurință și pentru realizarea sau adaptarea altor programe de calcul tehnico-științifice.

Mulțumiri. Autorul adresează mulțumiri ing. R. Popescu de la Oficiul de Calcul al IRNE Pitești, și dr. fiz. O. Petruș, de la Laboratorul de Fizică Computațională, Universitatea Al.I. Cuza, Iași, pentru furnizarea la I.C.I. a unor versiuni FELIX C-256 a pachetelor de programe IMSL și respectiv, SANDIA.

REFERINȚE BIBLIOGRAFICE

1. Smith, B.T., J.M. Boyle, J.J. Dongarra, B.S. Garbow, Y. Ikebe, V.C. Klema, C.B. Moler, **Matrix Eigensystem Routines - EISPACK Guide**, 2nd Ed., Lect. Notes in Comp. Sci., Vol. 6, Springer-Verlag, New York, 1976.
2. Dongarra, J.J., C.B. Moler, J.R. Bunch, C.W. Stewart, **LINPACK Users' Guide**, SIAM, Philadelphia, 1979.
3. * * * **IMSL LIBRARY. Reference Manual**, IMSL LIB-0008, IMSL Inc., Houston, 1980.
4. Sima, V., *Eigan*. General Description, Rep. ICI TR-05.84, 1984.
5. Cody, W.J., Rational Chebyshev approximations for the error function, **Math. Comp.**, vol. 23, 107, pag. 631-638.
6. Wilkinson, J.H., C. Reinsch, **Handbook for Automatic Computation**, Vol. II Linear Algebra, Springer-Verlag, Heidelberg, 1971.
7. Popescu, A., V. Sima, C. Nomolosanu, **Mimat V2. Completarea bibliotecii matematice pe microcalculatorul MC-18**, Studiu I.C.I., 1984.
8. Dekker, T.J., A floating-point technique for extending the available precision, **Numer. Math.**, Vol. 18, pag. 224-242, 1971.
9. Sima, V., *Vema*. General Description, Rep. ICI, TR-06.84, 1984.

ANEXA 1

LINV3F — SUBRUTINA DE INVERSARE A UNEI MATRICE GENERALE (VERSIUNEA ÎN DUBLĂ PRECIZIE)

C	MATH ROUTINE NAME — LINV3F	1
C		2
C	-----	3
C		4
C	COMPUTER — PDP/DOUBLE	5
C		6
C	LATEST REVISION — NOVEMBER 1, 1984	7
C	CENTRAL INST. FOR MANAGEMENT AND INFOR-	
	MATICS	
C		9
C	PURPOSE — IN PLACE INVERSE, EQUATION SOLUTION, AND/	
	OR	10
C	DETERMINANT EVALUATION — FULL STORAGE	
	MODE	11
C		12
C	USAGE — CALL LINV3F (A,B,IJOB,N,IA,D1,D2,WKAREA,IER)	13
C		14
C	ARGUMENTS A — INPUT/OUTPUT MATRIX OF DIMENSION N BY	
	N. SEE	15
C	PARAMETER IJOB.	16
C	B — INPUT/OUTPUT VECTOR OF LENGTH N WHEN	
	IJOB=	17
C	2 OR 3. OTHERWISE, B IS NOT USED.	18
C	ON INPUT, B CONTAINS THE RIGHT HAND	
	SIDE OF	19
C	OF THE EQUATION AX=B.	20
C	ON OUTPUT, THE SOLUTION X REPLACES B.	21
C	IJOB — INPUT OPTION PARAMETER. IJOB=I IMPLIES :	22
C	I=1, INVERT MATRIX A. A IS REPLACED BY	23
C	ITS INVERSE.	24
C	I=2, SOLVE THE EQUATION AX=B. A IS	25
C	REPLACED BY THE LU DECOMPOSITION OF A	26
C	ROWWISE PERMUTATION OF A, WHERE U IS	27
C	UPPER TRIANGULAR AND L IS LOWER	28
C	TRIANGULAR WITH UNIT DIAGONAL.	29
C	THE UNIT DIAGONAL OF L IS NOT STORED.	30
C	I=3, SOLVE AX=B AND INVERT MATRIX A.	31
C	A IS REPLACED BY ITS INVERSE.	32
C	I=4, COMPUTE THE DETERMINANT OF A.	33
C	A IS REPLACED BY THE LU DECOMPOSITION	34
C	OF A ROWWISE PERMUTATION OF A.	35
C	N — ORDER OF A. (INPUT)	36
C	IA — ROW DIMENSION OF MATRIX A EXACTLY AS	37
C	SPECIFIED IN THE DIMENSION STATEMENT IN	38
C	THE CALLING PROGRAM. (INPUT)	39
C	D1 — INPUT/OUTPUT. IF THE D1 AND D2 COMPONENTS	40
C	OF DETERMINANT(A)=D1*2**D2 ARE DESIRED,	41
C	INPUT D1.GE.O. OTHERWISE, INPUT D1.LT.O.	42
C	D2 IS NEVER INPUT.	43
C	WKAREA — WORK AREA OF LENGTH AT LEAST 2*N FOR IJOB	44
C	=1 OR IJOB=3.	45
C	WORK AREA OF LENGTH AT LEAST N FOR IJOB=	46
C	= 2 OR IJOB=4.	47

C	IER	— ERROR PARAMETER. (OUTPUT)	48
C		WARNING WITH FIX	49
C		IER=65 INDICATES THAT IJOB WAS LESS THAN	50
C		1 OR GREATER THAN 4. IJOB IS ASSUMED TO	51
C		BE 4.	52
C		TERMINAL ERROR	53
C	IER=130	INDICATES THAT MATRIX A IS	54
C		ALGORITHMICALLY SINGULAR. (SEE THE	55
C		CHAPTER PRELUDE).	56
C			57
C	PRECISION/HARDWARE	— SINGLE AND DOUBLE/H32	58
C		— DOUBLE/H30,H48,H60	59
C			60
C	REQD. MATH ROUTINES	— LUDATF,LUELMF,UERTST,UGETIO,HMACH	61
C			62
C	-----		73
C			74
C	SUBROUTINE LINV3F	(A,B,IJOB,N,IA,D1,D2,WKAREA,IER)	75
C			76
C	DOUBLE PRECISION	A(IA,1),B(1),WKAREA(1),C1,C2,D1,D2,WA,ZERO,	77
		ONE,SUM,C,WJ	
	REAL	W(2),WJP,WJP1	
	INTEGER	IWP(2),NMOD	
	LOGICAL	EVEN,STAN	
C		STAN IS TRUE FOR MACHINES WITH	
C		INTEGER WORDS LENGTH EQUAL TO	
C		REAL WORDS LENGTH	
	EQUIVALENCE	(WJ,WJP1,W(1)),(WJP,JP1,IWP(1),W(2)),	
1		(JP,IWP(2))	
	DATA	ZERO/0.0D0,ONE/1.0D0/	80
C		FIRST EXECUTABLE STATEMENT	81
C		LU DECOMPOSITION OF A	82
	CALL LUDATF	(A,A,N,IA,0,C1,C2,WKAREA,WKAREA,WA,IER)	83
	IF	(D1 .LT. ZERO .AND. IJOB .GE. 1 .AND. IJOB .LT. 4) GO TO 5	84
	D1=C1		85
	D2=C2		86
5	IF	(IER .GE. 128) GO TO 60	87
	IF	(IJOB .LE. 0 .OR. IJOB .GT. 4) GO TO 55	88
C		SOLVE AX=B	89
	IF	(IJOB .EQ. 2 .OR. IJOB .EQ. 3) CALL LUELMF (A,B,WKAREA,N,IA,B)	90
	IF	(IJOB .NE. 1 .AND. IJOB .NE. 3) GO TO 9005	91
C		MATRIX INVERSION	92
	A(N,N)=ONE/A(N,N)		93
	NM1=N-1		94
	IF	(NM1 .LT. 1) GO TO 9005	95
	DO	40 II=1,NM1	96
		L=N-II	97
		M=L+1	98
	DO	15 I=M,N	99
		SUM = ZERO	100
		DO	10 K=M,N
		SUM = SUM-A(I,K)*A (K,L)	101
10		CONTINUE	102
		WKAREA(N+I) = SUM	103
15		CONTINUE	104
	DO	20 I=M,N	105
		A(LL)=WKAREA(N+I)	106
20		CONTINUE	107
	DO	30 J=L,N	108
		SUM = ZERO	109
			110

	IF (J .EQ. L) SUM = ONE	111
	DO 25 K=M,N	112
	SUM = SUM-A(L,K)*A(K,J)	113
25	CONTINUE	114
	WKAREA(N+J) = SUM/A(L,L)	115
30	CONTINUE	116
	DO 35 J=L,N	117
	A(L,J)=WKAREA(N+J)	118
35	CONTINUE	119
40	CONTINUE	120
C	PERMUTE COLUMNS OF A INVERSE	121
	STAN = .TRUE.	
	IF(1/MACH(5) .LE. MAX0 (1/MACH(10),1/MACH(11))) STAN = .FALSE.	
	EVEN = .TRUE.	
	IF(MOD(N,2) .NE. 0) EVEN = .FALSE.	
	IF(STAN .AND. .NOT. EVEN) WJ = WKAREA((N+1)/2)	
	IF(STAN) GO TO 41	
	NMOD = MOD(N,4)	
	IF (NMOD .EQ. 0) GO TO 41	
	WJ = WKAREA((N+3)/4)	
	IF(NMOD .I.E. 2) WJP = WJP1	
	IF(.NOT. EVEN) JP = JP1	
41	CONTINUE	
	DO 50 I=1,N	122
	J = N-I+1	123
	IF (.NOT. STAN) GO TO 42	
	IF(.NOT. EVEN) WJP = WJP1	
	IF(EVEN) WJ = WKAREA(J/2)	
	JP = JP1	
	GO TO 44	
42	CONTINUE	
	IF (MOD(J,4) .NE. 0) GO TO 43	
	WJ = WKAREA(J/4)	
	GO TO 44	
43	IF(EVEN) WJP = WJP1	
44	CONTINUE	
	IF (J .EQ. JP) GO TO 47	125
	DO 45 K=1,N	126
	C = A (K,JP)	127
	A(K,JP) = A(K,J)	128
	A(K,J) = C	129
45	CONTINUE	130
47	CONTINUE	
	IF(STAN) GO TO 49	
	IF(EVEN) JP = JP1	
49	EVEN = .NOT. EVEN	
50	CONTINUE	131
	GO TO 9005	132
55	CONTINUE	133
C	WARNING WITH FIX — IJOB WAS SET	134
C	INCORRECTLY	135
	IER = 65	136
	GO TO 9000	137
C	TERMINAL ERROR — MATRIX A IS	138
C	ALGORITHMICALLY SINGULAR	139
	60. IER = 130	140
9000	CONTINUE	141
	CALL UERTST(IER,6HLINV3F)	142
9005	RETURN	143
	END	144

**DCABS, DCSQRT, DCDIV — SUBPROGRAME ÎN DUBLA PRECIZIE PENTRU
 OPERAȚII CU NUMERE COMPLEXE
 (MODUL, RĂDĂCINA PĂTRATA ȘI RAPORT)**

```

DOUBLE PRECISION FUNCTION DCABS(XR,XI)
C
C           SPECIFICATIONS FOR ARGUMENTS
DOUBLE PRECISION XR,XI
C           SPECIFICATIONS FOR LOCAL VARIABLES
DOUBLE PRECISION H,YR,YI,ONE,ZERO
DOUBLE PRECISION DABSD,SQRT
DATA           ONE,ZERO/1.0D0,0.0D0/
C           FIRST EXECUTABLE STATEMENT
  YR = DABS(XR)
  YI = DABS(XI)
  IF (YR .GE. YI) GO TO 5
  H = YR
  YR = YI
  YI = H
5  DCABS = YR
  IF (YR .EQ. ZERO .OR. YI .EQ. ZERO) GO TO 10
  DCABS = YR * DSQRT (ONE + (YI/YR)**2)
10 RETURN
  END
SUBROUTINE DCSQRT(XR,XI,YR,YI)
C
C           SPECIFICATIONS FOR ARGUMENTS
DOUBLE PRECISION XR,XI,YR,YI
C           SPECIFICATIONS FOR LOCAL VARIABLES
DOUBLE PRECISION H,ZR,ZI,TWO,ZERO
DOUBLE PRECISION DABS,DCABS,DSQRT
DATA           TWO,ZERO/2.0D0,0.0D0/
C           FIRST EXECUTABLE STATEMENT
  H = DSQRT ((DABS(XR) + DCABS(XR,XI))/TWO)
  ZR = H
  ZI = XI
  IF (ZI .NE. ZERO) ZI=ZI/(TWO*H)
  IF (XR .GE. ZERO) GO TO 10
  IF (ZI .LT. ZERO) GO TO 5
    ZR = HI
    ZI = H
  GO TO 10
5  ZR = -ZI
  ZI = -H
10 YR = ZR
  YI = ZI
  RETURN
  END
SUBROUTINE DCDIV(XR,XI,YR,YI,ZR,ZI)
C
C           SPECIFICATIONS FOR ARGUMENTS
DOUBLE PRECISION XR,XI,YR,YI,ZR,ZI
C           SPECIFICATIONS FOR LOCAL VARIABLES
DOUBLE PRECISION H,XXR,XXI,YYR,YYI,ZERO
DOUBLE PRECISION DABS
DATA           ZERO/0.0D0/
C           FIRST EXECUTABLE STATEMENT

```

```

XXR = XR
XXI = XI
YYR = YR
YYI = YI
ZI = ZERO
IF (YYR .NE. ZERO) ZR = XXR/YYR
IF (XXI .EQ. ZERO .AND. YYI .EQ. ZERO) GO TO 20
IF (DABS(YYR) .LE. DABS(YYI)) GO TO 10
  H=YYI/YYR
  YYR = H*YYI + YYR
  ZR = (XXR + H * XXI)/YYR
  ZI = (XXI - H * XXR)/YYR
  GO TO 20
C
10 CONTINUE
  H = YYR/YYI
  YYI = H * YYR + YYI
  ZR = (H * XXR + XXI)/YYI
  ZI = (H * XXI - XXR)/YYI
C
20 CONTINUE
RETURN
END

```

ANEXA 3

**VXADD, VMUL, VXSTO — SUBPROGRAME ÎN DUBLĂ PRECIZIE
PENTRU CALCULE CU PRECIZIE EXTINSĂ
(ADUNARE LA ACUMULATOR, ÎNMULȚITRE ȘI ACTUALIZARE
ACUMULATOR, CONVERSIE REPREZENTARE)**

```

SUBROUTINE VXADD (A,ACC)
C
DOUBLE PRECISION A,ACC(2)
C
DOUBLE PRECISION X,Y,YY,R,S
C
SPECIFICATIONS FOR ARGUMENTS
SPECIFICATIONS FOR LOCAL VARIABLES
FIRST EXECUTABLE STATEMENT
X=A
Y = ACC(1)
YY = ACC(2)
R = X + Y
IF (DABS(X) .LE. DABS(Y)) GO TO 10
  S = X - R + Y + YY
  GO TO 20
10 CONTINUE
  S = Y - R + X + YY
20 CONTINUE
  X = R + S
  ACC(1) = X
  ACC(2) = R - X + S
RETURN
END
SUBROUTINE VXSTO (ACC,L

```

```

C          DOUBLE PRECISION D,ACC(2)          SPECIFICATIONS FOR ARGUMENTS
C          FIRST EXECUTABLE STATEMENT
D = ACC(1) + ACC(2)
RETURN
END
SUBROUTINE VXMUL (A,B,ACC)

C          DOUBLE PRECISION          A,B,ACC(2)          SPECIFICATIONS FOR ARGUMENTS
C          DOUBLE PRECISION          DABS          SPECIFICATIONS FOR LOCAL VARIABLES
DOUBLE PRECISION          CONST,HA,TA,HB,TB,P,Q,U,UU,Z,ZZ
EQUIVALENCE          (HA,U),(TA,UU)

C          CONST IS A MACHINE DEPENDENT
C          PARAMETER WHICH MUST BE INI-
C          TIALIZED BY 2**(T-T/2)+1, WHERE
C          T IS THE NUMBER OF MANTISSA
C          DIGITS IN THE BINARY FLOATING-
C          POINT SYSTEM USED.
C          THE ALGORITHM DOES NOT
C          DIRECTLY WORK FOR NON-BINARY
C          SYSTEMS.
DATA          CONST/268435457.0D0/
C          FIRST EXECUTABLE STATEMENT

P = A * CONST
HA = A - P + P
TA = A - HA
P = B * CONST
HB = B - P + P
TB = B - HB
P = HA * HB
Q = HA * TB + TA * HB
Z = P + Q
ZZ = P - Z + Q + TA * TB
U = Z + ZZ
UU = Z - U + ZZ
Z = ACC(1)
ZZ = ACC(2)
P = Z + U
IF (DABS(Z) .LE. DABS(U)) GO TO 10
Q = Z - P + U + UU + ZZ
GO TO 20
10 CONTINUE
Q = U - P + Z + ZZ + UU
20 CONTINUE
Z = P + Q
ACC(1)=Z
ACC(2) = P - Z + Q
RETURN
END

```

PACHET DE PROGRAME PENTRU MODELAREA ȘI PREDICȚIA SERIILOR DE TIMP

Dr. ing. Th. D. Popescu
I.T.C.I.

1. INTRODUCERE

Construcția modelelor dinamice stohastice ale seriilor de timp se practică într-o varietate de domenii : tehnică, economie, biologie, ecologie, astronomic, statistică aplicată etc. Aceste modele pot fi utilizate pentru :

- investigarea fenomenului ce a generat seria de timp ;
- predicția optimală a valorilor seriei ;
- determinarea relațiilor dintre două sau mai multe serii de timp ;
- stabilirea unor strategii de conducere optimală prin minimizarea efectului perturbațiilor asupra variabilelor dependente.

Problemele ce apar în acest domeniu sînt suficient de complexe pentru a justifica utilizarea tehnicii de calcul. Pachetele de programe specifice acestui domeniu, raportate în literatură, CAPTAIN (Shellswell, 1972), TIMSAC/74 (Akaike, Arahata, Ozaki, 1975), X-11-ARIMA (Dagum, 1979), ARIMAF (Goel, Rocco, 1982), PACK System (Reilly, 1984), AUTOB & J (Popescu, 1985a) etc. implementează diferite tehnici de modelare și predicție a seriilor de timp.

Pachetul de programe AUTOB & J care constituie obiectul acestei lucrări este destinat modelării și predicției seriilor de timp scalare și asigură suportul software pentru implementarea metodologiei Box-Jenkins de construcție a modelelor stohastice ARIMA și predicție optimală a valorilor seriei pe baza acestor modele. Metodologia de analiză Box-Jenkins reprezintă una dintre cele mai frecvent utilizate și mai precise tehnici de predicție pe termen scurt, fapt ilustrat și de atenția deosebită de care se bucură în ultimul timp în literatura de specialitate.

Pachetul de programe este complet interactiv și utilizarea sa necesită un minim de cunoștințe de informatică din partea utilizatorului. Produsul poate fi utilizat pentru :

- analiza și modelarea unor procese din industrie (chimie, metalurgie, energetică), transporturi, agricultură, ecologie, biologie, medicină, economie, asistență socială, etc.
- predicția evoluției unor parametri tehnologici : presiuni, temperaturi, concentrații, etc. din cadrul proceselor industriale sau a unor indicatori economici, vânzărilor, cererii de servicii publice etc.

Utilizarea produsului AUTOB & J prezintă următoarele avantaje :

- permite modelarea și predicția interactivă a seriilor de timp lăsînd mai mult timp utilizatorului pentru interpretarea rezultatelor ;
- reprezintă un instrument eficient de predicție, implementînd una dintre cele mai precise și mai larg utilizate tehnici de predicție pe termen scurt ;
- permite instruirea rapidă a celor interesați în domeniul modelării și predicției seriilor de timp utilizînd metodologia Box și Jenkins.

Pachetul de programe este implementat pe un minicalculator CORAL 4011 sub sistemul de operare RSX11M (facilități standard), este scris în Fortran/77 și utilizează biblioteca de subprograme TSPACK (Popescu, 1933) ce conține module software pentru modelarea și predicția seriilor de timp, abordarea Box-Jenkins.

După o scurtă trecere în revistă a metodologiei Box-Jenkins de modelare și predicție a seriilor de timp, în lucrare se prezintă structura și funcțiunile pachetului și procedura practică de utilizare a acestuia. În final este prezentată în detaliu o aplicație a produsului privind modelarea și predicția unei serii de timp reprezentînd concentrația în cadrul unui proces chimic-

2. METODOLOGIA BOX — JENKINS DE MODELARE ȘI PREDICȚIE A SERIILOR DE TIMP

Fie o serie de timp nesezonieră $\{X_t\}$. Conform metodologiei de modelare Box-Jenkins (1976), evoluția sa poate fi exprimată în termenii unei combinații a valorilor trecute ale seriei și a unei secvențe de tip zgomot alb, o serie de valori necorelate identic distribuite. În cadrul metodologiei se presupune că există un întreg (d), astfel încît staționaritatea seriei

$$w_t = (1-B)^d X_t \tag{1}$$

este asigurată, unde $B^d X_t = X_{t-d}$. Prin urmare valorile seriei la momentul de timp t pot fi exprimate în felul următor :

$$w_t = \frac{\Theta_q(B)^q}{\Phi_p(B)^p} a_t \tag{2}$$

unde a_t este o secvență de tip zgomot alb și :

$$\Theta_q(B)^q = 1 - \theta_1 B - \dots - \theta_q B^q$$

$$\Phi_p(B)^p = 1 - \varphi_1 B - \dots - \varphi_p B^p$$

Combinînd cele două ecuații de mai sus se obține forma generală a modelului Box-Jenkins pentru serii de timp nesezoniere :

$$\Phi_p(B)^p (1-B)^d X_t = \theta_0 + \Theta_q(B)^q a_t \tag{3}$$

unde θ_0 este o constantă diferită de zero.

În notația Box-Jenkins, ecuația (3) reprezintă un model autoregresiv și de medie alunecătoare integrat (ARIMA), (p, d, q), unde p este gradul părții autoregresive a modelului (AR), d este ordinul de diferențiere a seriei, iar q este gradul părții de medie alunecătoare (MA) a modelului. Condiția ca toate rădăcinile ecuațiilor polinomiale în B : $\Phi(B)=0$ și $\Theta(B)=0$, să se găsească în afara cercului unitate asigură staționaritatea seriei w_t și respectiv inversabilitatea modelului, garantînd că modelul astfel specificat este unic reprezentabil.

În cadrul abordării Box-Jenkins se parcurg de regulă următoarele etape : identificarea (specificarea) modelului, estimarea parametrilor modelului, verificarea modelului și predicția valorilor seriei, reprezentate în Fig. 1.

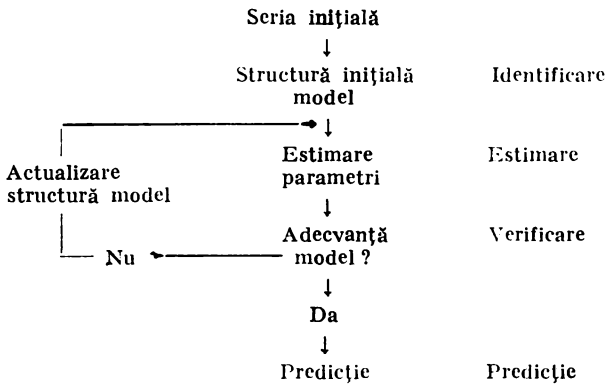


Fig. 1 Procesul de modelare și predicție

În etapa de identificare a modelului se pornește de la un set de valori ale parametrilor p, d, q ce se obține utilizând o procedură ce va fi prezentată mai jos. În aceste condiții se determină un prim set de valori ale coeficienților modelului $\varphi_1, \varphi_2, \dots, \varphi_p$ și $\theta_1, \theta_2, \dots, \theta_q$. Modelul astfel obținut este verificat pentru a se stabili gradul său de adecvare în raport cu datele seriei. Dacă în urma etapei de verificare se impune o nouă structură a modelului, ciclul de estimare a parametrilor modelului și verificarea acestuia se repetă pentru noul set de parametri p, d, q . Procedura continuă pînă la obținerea modelului reprezentativ al seriei, pe baza căruia se pot determina valorile de predicție ale seriei.

Principalele instrumente de calcul ce se utilizează în etapa de identificare a modelului seriei sînt funcțiile de autocorelație și de autocorelație parțială. Fie seria staționară w_t de medie μ . Valoarea funcției de autocorelație de ordin k , în acest caz, reprezintă corelația dintre valorile w_t și w_{t-k} :

$$\rho_k = E \{ (w_t - \mu) (w_{t-k} - \mu) \} / E \{ (w_t - \mu)^2 \} \quad \text{cu } \rho_{-k} = \rho_k$$

Valoarea funcției de autocorelație parțială de ordin k dintre w_t și w_{t-k} , φ_{kk} , reprezintă soluția sistemului:

$$\rho_j = \sum_{i=1}^k \varphi_{ki} \rho_{j-i}, \quad j=1, 2, \dots, k$$

În cazul în care $(1 - \varphi_1 B - \dots - \varphi_k B^k) w_t = a_t$, $\varphi_{kk} = \varphi_k$ și $\varphi_{k+j}, k+j=0$, pentru toate valorile $j \geq 1$.

Revenim acum la procedura de identificare a modelului:

(i) Dacă seria inițială $\{X_t\}$ nu este staționară, adică $d \neq 0$ în ecuația (3), funcția de autocorelație a seriei nu va descrește rapid pentru valori mari ale întîrzierii k , impunîndu-se în această situație diferențierea seriei în scopul obținerii staționarității acesteia.

(ii) Presupunem că în urma aplicării de un număr suficient de ori a operatorului de diferențiere $1 - B$, seriei $\{X_t\}$, am obținut o serie staționară $\{w_t\}$. În acest caz:

(a) Presupunînd $q=0$, deci un model autoregresiv de ordin p al seriei, ARIMA($p, d, 0$), valorile funcției de autocorelație a seriei vor descrește — pentru toate valorile k — conform ecuației:

$$\rho_k = \sum_{i=1}^p \varphi_i \rho_{k-i}$$

sau, cu alte cuvinte, ele se vor atenua sub forma unei unde exponențiale și/sau sinusoidale, în timp ce valorile funcției de autocorelație parțială φ_{kk} se vor anula pentru toate valorile $k > p$.

(b) Dacă însă $p=0$, deci se presupune un model de medie alunecătoare al seriei, ARIMA($0, d, q$), structura funcției de autocorelație este de forma $\rho_k=0$ pentru toate valorile $k > q$, în timp ce funcția de autocorelație parțială se atenuază sub forma unei combinații de unde exponențiale și sinusoidale.

(c) În cazul în care $p \neq 0$ și $q \neq 0$, deci seria admite un model ARIMA(p, d, q), structura funcției de autocorelație este descrisă de următoarea ecuație cu diferențe:

$$\rho_k = \sum_{i=1}^p \varphi_i \rho_{k-i}$$

pentru toate valorile întîrzierii $k > q$, ca atenuîndu-se sub forma unei combinații de unde exponențiale și sinusoidale.

Deoarece în situațiile practice nu cunoaștem structurile reale ale funcțiilor de autocorelație și de autocorelație parțială se impune estimarea acestora pe baza eșantioanelor seriei ce se analizează. Fie, de exemplu eșantioanele w_1, w_2, \dots , aparținînd seriei globale $\{w_t\}$. În acest caz estimarea valorilor funcției ρ_k se poate obține pe baza următoarei relații:

$$\hat{r}_k = 1/N \sum_{t=k+1}^N (w_t - \bar{w})(w_{t-k} - \bar{w}) / 1/N \sum_{t=1}^N (w_t - \bar{w})^2$$

unde N reprezintă numărul eșantioanelor seriei, iar \bar{w} este media eșantioanelor seriei.

Valoarea estimată a funcției de autocorelație parțială φ_{kk} , va fi în acest caz $\hat{\varphi}_{kk}$ și se obține prin rezolvarea următorului sistem de ecuații :

$$r_j = \sum_{i=1}^k \varphi_{ki} r_{j-1} \quad j=1, \dots, k$$

În cadrul analizei se va presupune că funcțiile de autocorelație și de autocorelație parțială ale eșantioanelor disponibile ale seriei sînt identice cu cele ale seriei globale. Acest lucru este valabil pentru un număr suficient de mare de eșantioane, situație în care specificarea modelului seriei poate fi făcută corect.

Valorile estimate ale coeficienților modelului seriei (ecuația 3) reprezintă rezultatul mini-mizării sumei pătratelor reziduurilor $\sum a_i^2$ utilizînd o tehnică ce va fi descrisă în continuare. Această tehnică necesită un prim set de valori ale parametrilor modelului pentru inițializarea procedurii iterative de calcul, ce se determină în etapa de identificare a modelului pe baza valorilor funcției de autocorelație a seriei.

În ceea ce privește etapa de validare a modelului, Box și Jenkins (1973) propun mai multe teste ce urmează a se efectua asupra reziduurilor modelului, generate în etapa de estimare a parametrilor modelului. Dacă notăm aceste reziduuri și funcția lor de autocorelație cu \hat{a}_t și respectiv cu $r_k(\hat{a})$ și presupunem că seria reziduurilor reale $\{a_t\}$ este o secvență de tip zgomet alb, atunci $r_k(\hat{a})$ va avea valoarea medie nulă și deviația standard egală cu aproximativ $1/\sqrt{N}$. Eșantioanele funcției de autocorelație a reziduurilor seriei reprezintă abateri de la comportarea tipică a unei secvențe de tip zgomet alb a reziduurilor și pot sugera chiar o alternativă de specificare a modelului seriei. Un test general, dar nu suficient de puternic, de punere în evidență a caracterului de zgomet alb a secvenței reziduurilor, în acest caz, constă în compararea valorii Q și a distribuției χ^2 pentru $(M-p-q)$ grade de libertate, unde :

$$Q = N \sum_{k=1}^M r_k^2(\hat{a})$$

și $M \geq 20$. În cadrul acestui test este posibilă utilizarea periodogramei cumulative a reziduurilor seriei (Box-Jenkins, 1976).

Presupunem că am obținut un model acceptabil pentru seria X_1, \dots, X_n și urmăm să determinăm valorilor de predicție ale seriei X_{n+m} unde $m=1, 2, 3, \dots$ etc. În plus se va presupune că dacă ne situăm la momentul (n) , predicția optimă — în termenii erorilor de predicție minime — a valorii X_{n+m} reprezintă valoarea așteptată condițională la momentul (n) . Totuși, valorile condiționale X_n, X_{n-1} etc. reprezintă valorile reale ale seriei, cunoscute la momentul (n) , în timp ce valorile a_n, a_{n-1} etc. reprezintă reziduurile generate de modelul seriei ; valorile a_{n+1}, a_{n+2} etc. sînt egale cu zero în timp ce X_{n+1}, X_{n+2} etc. reprezintă valorile de predicție ale seriei determinate la momentul (n) . Metoda de predicție, dacă se cunoaște un model al seriei de forma ecuației (3) reprezintă, prin urmare, un proces iterativ în care m ia valorile 1, 2, 3 etc., iar $(n+m)$ se substituie variabilei (t) în ecuația modelului seriei.

Principiile referitoare la modelarea și predicția seriilor de timp sezoniere sînt legate de cele prezentate în cadrul acestei secțiuni, dar apar unele probleme legate de lipsa de informație asupra comportării teoretice a funcțiilor de autocorelație și de autocorelație parțială.

3. STRUCTURA ȘI FUNCȚIUNILE PACHETULUI AUTOB&J

3.1. Structura pachetului

Pachetul de programe AUTOB&J este orientat pe comenzi. Comenzile sînt interactive și solicită utilizatorului răspunsuri la întrebările formulate. Produsul este prietenos în raport cu utilizatorii (user friendly), complet interactiv, include o opțiune "help" și verificarea răspunsurilor utilizatorului. De asemenea sînt incluse facilități de salvare a fișierelor — pot fi memorate modelul seriei, valorile reziduurilor, valorile de predicție ale seriei etc. — la solicitarea utiliza-

torului. Aceste fișiere pot fi utilizate ca fișiere de date de intrare pentru acest pachet sau pentru alte pachete de programe. Structura sistemului de programe este reprezentată în Fig. 2.

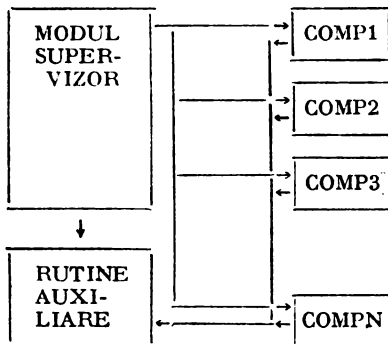


Fig. 2 Structura sistemului AUTOB&J

COMP1, . . . , COMPn reprezintă module program aplicative, corespunzătoare diferitelor funcțiuni ale pachetului, ce operează sub controlul unui modul supervisor al sistemului. Acesta din urmă lansează în execuție modulele de program aplicative, specificate de utilizator prin comenzi, verifică disponibilitatea datelor necesare și corectitudinea răspunsurilor utilizatorului la întrebările formulate de program.

Fiecare program de aplicație rezolvă un număr de probleme dintr-o anumită clasă, funcție de opțiunile utilizatorului și de datele furnizate. Programele permit un dialog întrebare-răspuns, adecvat unui utilizator începător sau sporadic.

3.2. Manipularea datelor

Pachetul AUTOB&J este orientat pe fișiere. Fiecare din seturile de date ce se analizează sau rezultatele furnizate de programele de aplicație : modelul seriei, valorile de predicție etc. reprezintă fișiere pe disc ce pot fi regăsite prin specificarea numelor acestora.

Programul supervisor utilizează următorii vectori de date pentru :

- memorarea datelor originale ale seriei ORDAT ;
- memorarea datelor ce se analizează în mod curent ANDAT ;
- memorarea reziduurilor ce rezultă în urma determinării modelului seriei RESID ;
- memorarea valorilor rezultate în urma predicției FOREC.

Comenzile utilizate pentru : încărcarea datelor seriei de pe disc în vectorul de date ORDAT, scrierea datelor din vectorul de date ANDAT într-un fișier de date pe disc, deschiderea unui fișier de date, închiderea unui fișier de date precum și alte comenzi specifice manipulării datelor vor fi prezentate în cadrul unei alte secțiuni.

3.3. Funcțiunile pachetului

Pachetul AUTOB&J implementează toate etapele de calcul specifice metodologiei Box-Jenkins de analiză și predicție a seriilor de timp și anume :

- transformarea datelor seriei, calculul diferențelor și diferențelor sezoniere ale eșantionnelor seriei în scopul obținerii staționarității acesteia ;
- calculul funcțiilor de autocorelație și de autocorelație parțială ale seriei în scopul identificării modelului ;
- estimarea preliminară a parametrilor părții de autoregresie (AR) a modelului stohastic ARIMA ;
- estimarea preliminară a parametrilor părții de medie alunecătoare (MA) a modelului stohastic ARIMA ;
- estimarea de verosimilitate maximă a parametrilor modelului stohastic ARIMA ;
- calculul valorilor de predicție ale seriei și al limitelor de probabilitate ale acestora.

Aceste etape de calcul sînt acoperite de cele aproximativ 26 de comenzi ale pachetului ce se specifică de c tre utilizator.

Modulul supervisor al pachetului verifică numai primele 3 caractere ale comenzii, restul de caractere fiind utilizate pentru comentariu. Comenzile sînt interactive și solicită utilizatorului, prin întrebări, datele necesare prelucrărilor specifice. De asemenea se validează toate răspunsurile utilizatorului.

Prezentăm în continuare comenzile disponibile în cadrul pachetului, funcțiunile acestora, algoritmi utilizați, precum și exemple simple de utilizare a acestor comenzi.

Comanda HELP

Funcția : Afișează pe terminal informații generale asupra funcțiunilor pachetului, modului de organizare a datelor și comenzilor disponibile în cadrul pachetului.

Comanda INPUT

Funcția : Realizează citirea datelor seriei de timp dintr-un fișier de date de pe disc a cărui nume este specificat de c tre utilizator și încărcarea acestora în vectorii de lucru ORDAT și ANDAT. Fișierul de date de pe disc reprezintă o coloană de numere reale sau întregi.

Exemplu :

```
AB&J>INPUT
DATELE CITITE DIN FIȘIERUL ? ALFA
  120 DATE CITITE DIN FIȘIERUL : ALFA
    1.73406    7.45670    2.00543    1.98765 ....
```

Comanda OUTPUT

Funcția : Realizează scrierea datelor conținute în vectorul de date ANDAT într-un fișier pe disc al cărui nume este specificat de utilizator.

Exemplu :

```
AB&J>OUTPUT
FIȘIERUL ÎN CARE SE SCRIB DATELE ? BETA
  200 DATE SCRIB ÎN FIȘIERUL : BETA
```

Comanda OPEN

Funcția : Realizează deschiderea unui fișier pe disc.

Exemplu :

```
AB&J>OPEN
NUMELE FIȘIERULUI ? ALFA
```

Comanda CLOSE

Funcția : Realizează închiderea unui fișier pe disc.

Exemplu :

```
AB&J>CLOSE
```

Comanda FILE

Funcția : Permite specificarea stării dorite (on, off) a ieșirii pe disc.

Exemplu :

```
AB&J>FILE
SPECIFICAȚI STAREA DORITĂ A IEȘIRII PE DISC (ON, OFF) ? ON
```

Comanda TERM

Funcția : Permite specificarea stării dorite (on, off) a ieșirii pe terminal.

Exemplu :

```
AB&J>TERM
SPECIFICAȚI STAREA DORITĂ A IEȘIRII PE TERMINAL (ON, OFF) ? ON
```

Comanda PRINT

Funcția : Realizează afișarea valorilor datelor seriei conținute în vectorul ANDAT.

Exemplu :

```
AB&J>PRINT
      12 VALORI
          17.00000   16.60000   16.30000   16.10000
          17.10000   16.90000   16.80000   17.40000
          17.10000   17.00000   16.70000   17.40000
```

Comanda PLOT

Funcția : Realizează reprezentarea grafică a datelor seriei ce se analizează.

Exemplu :

```
AB&J>PLOT
2.56277      5
2.25120      4 6
1.93962      3 7
1.62805      2 8
1.316481      90123456
1.00491      78901234567
.69333      8901234567890
.38176      123456789
          ^-----^-----^-----^-----^
          10      20      30      40      50
```

Comanda PWIDTH

Fucția : Realizează modificarea lățimii reprezentării grafice a datelor seriei ; valoarea implicită este 66.

Exemplu :

AB&J>PWIDTH
LĂȚIMEA REPREZENTĂRII ? 50

Comanda PLENGTH

Funcția : Realizează modificarea lungimii reprezentării grafice a datelor seriei ; valoarea implicită este 19.

Exemplu :

AB&J>PLENGTH
LUNGIMEA REPREZENTĂRII ? 10

Comanda ACF

Funcția : Realizează calculul funcției de autocorelație a seriei ce se analizează.

Metoda : Pentru seria ce se analizează $W_i, i=1, \dots, LW$, valoarea medie (AMEAN), dispersia (VAR) și funcția de autocorelație a seriei $AC_j, j=1, \dots, K$ se determină cu următoarele relații :

$$AMEAN = 1/LW \sum_{i=1}^{LW} W_i$$

$$VAR = 1/LW \sum_{i=1}^{LW} (W_i - AMEAN)^2$$

$$ACV_j = 1/LW \sum_{i=1}^{LW-j} (W_i - AMEAN)(W_{i+j} - AMEAN) \quad j=1, \dots, K$$

$$AC_j = ACV_j / VAR \quad j=1, \dots, K$$

Exemplu :

AB&J>ACF
NUMĂRUL TERMENILOR ? 10
VALOAREA MEDIE=0.00204

DISPERSIA=0.13551

-1.0 -0.8 -0.6 -0.4 -0.2 0.0 0.2 0.4 0.6 0.8 1.0
+-----+-----+-----+-----+-----+-----+-----+-----+

1	-0.411	XXXXXXXXXXXX	
2	0.016		X
3	-0.063		XXX
4	-0.013		X
5	-0.072		XXX
6	-0.011		X
7	0.139		XXXX
8	-0.066		XXX
9	0.038		XX
10	0.017		X

Comanda PACF

Funcția : Realizează calculul funcției de autocorelație parțială a seriei.

Metoda : Calculul funcției de autocorelație parțială a seriei PACV_j, j=1, ..., L utilizează valorile funcției de autocorelație determinate în cadrul comenzii ACF, astfel :

$$PACV_j = \begin{cases} AC_1 & j=1 \\ AC_j - \sum_{i=1}^{j-1} (AC_{j-i}) (\vartheta_{j-i,1}) & j=2, \dots, L \\ \dots & \dots \\ 1 - \sum_{i=1}^{j-1} (AC_i) (\vartheta_{j-i,1}) & \dots \end{cases}$$

unde

$$\vartheta_{jk} = \begin{cases} PACV_j & \text{dacă } k=j \\ \vartheta_{j-1,k} - (PACV_j) (\vartheta_{j-1,j-k}) & k=1, 2, \dots, j-1 \end{cases}$$

Exemplu :

```

AB&J>PACF
NUMĂRUL TERMENILOR ? 10
VALOAREA MEDIE=0.00204                DISPERSIA=0.13551
-1.0 -0.8 -0.6 -0.4 -0.2 0.0 0.2 0.4 0.6 0.8 1.0
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 -0.411                                xxxxxxxxxxxxxx
2 -0.184                                xxxxxxxx
3 -0.163                                xxxxxx
4 -0.138                                xxxxx
5 -0.198                                xxxxxx
6 -0.207                                xxxxxx
7 -0.001                                 x
8 -0.046                                 xx
9 -0.015                                 x
10 0.037                                 xx
    
```

Comanda TRANS

Funcția : Realizează transformarea datelor seriei conținute în vectorul ANDAT ; în urma operației în vectorul ANDAT se vor găsi valorile rezultate.

Metoda : Se pot efectua următoarele tipuri de transformări :

(i) $W_i = \ln(W_i + SHIFT)$, i=1, 2, ..., LW. Constanta SHIFT se determină prin calcul astfel încât $\min(W_i + SHIFT) = 1$, i=1, ..., LW.

(ii) $W_i = W_i^{IP}$, i=1, ..., LW.

Exemplu :

```

AB&J>TRANS
AVERTISMENT : ACEASTĂ FUNCȚIE VA DISTRUGE CONȚINUTUL VECTORULUI DATELOR SERIEI
EXPONENTUL TRANSFORMĂRII (SAU 0 ÎN CAZUL LOGARITMĂRII) ? 2
REALIZEAZĂ URMĂTOAREA TRANSFORMARE : DATA(1)=DATA(1)**2
    
```

Comanda DIFF

Funcția : Realizează calculul diferențelor sezoniere și nesezoniere ale datelor seriei conținute în vectorul ANDAT ; în urma operației în vectorul ANDAT se vor găsi valorile rezultate.

Metoda : Pentru seria analizată $W_i, i=1, \dots, LW$, diferențele $Z_i = \nabla^{ID1} \nabla^{ID2} W_i, i=1, \dots, LZ$ se determină cu relațiile

$$\nabla W_i \equiv W_{i+1} - W_i$$

$$\nabla_{IS} W_i \equiv W_{i+IS} - W_i$$

$$\nabla_{IS}^{ID2} W_i \equiv \nabla (\nabla_{IS}^{ID2-1} W_i)$$

$$\nabla^{ID1} W_i \equiv \nabla (\nabla^{ID1-1} W_i)$$

unde

ID1 — ordinul diferenței nesezoniere

ID2 — ordinul diferenței sezoniere

IS — valoarea perioadei componente sezoniere

Prin definiție $\nabla^0 W_i \equiv W_i, \nabla_{IS}^0 W_i \equiv W_i$.

Exemplu :

AB&J>DIFF

AVERTISMENT : ACEASTĂ FUNCȚIE VA DISTRUGE CONȚINUTUL VECTORULUI DATELOR SERIEI

ORDINUL DIFERENȚEI NESEZONIERE ? 1

ORDINUL DIFERENȚEI SEZONIERE ? 0

S-A DETERMINAT DIFERENȚA NESEZONIERĂ DE ORDIN 1

Comanda ARIMA

Funcția : Realizează calculul estimațiilor de verosimilitate maximă ale parametrilor modelului ARIMA și reziduurilor modelului.

Metoda : Determinarea estimațiilor de verosimilitate maximă ale parametrilor modelului ARIMA (p, d, q) :

$$\nabla^d X_t = \sum_{i=1}^p \varphi_i \nabla^d X_{t-1} + a_t - \sum_{i=1}^q \theta_i a_{t-1}$$

sau, într-o formă echivalentă :

$$X_t = \theta_0 + \sum_{i=1}^{p+d} \psi_i X_{t-1} + a_t - \sum_{i=1}^q \theta_i a_{t-1}$$

se poate face cu sau fără specificarea estimațiilor preliminare ale parametrilor φ_i și θ_i .

Funcție de valoarea parametrului d seria inițială X_t este transformată astfel :

$$W_t = \begin{cases} X_t - \bar{X}, & \text{dacă } d=0 \\ \nabla^d X_t, & \text{dacă } d>0 \end{cases}$$

unde \bar{X} reprezintă valoarea medie a seriei.

În cazul nespacificării de către utilizator a valorilor estimațiilor preliminare ale parametrilor modelului ARIMA, acestea se determină astfel :

a) Calculul parametrilor părții AR a modelului

Într-o primă etapă se determină valorile funcției de autocovarianță a seriei $\{W_t\}$: ACV_i , $i=1, \dots, p+q+1$, parametrilor părții AR a modelului φ_i obținându-se ca soluție a următorului sistem linear de ecuații:

$$Ax=b$$

unde

$$\begin{aligned} A_{ij} &= ACV_{|q+i-j|+1} & i, j &= 1, \dots, p \\ x_j &= \varphi_j & j &= 1, \dots, p \\ b_j &= ACV_{q+j+1} & j &= 1, \dots, p \end{aligned}$$

Valoarea preliminară a parametrului θ_0 se determină cu relația:

$$\theta_0 = \bar{X} \left(1 - \sum_{i=1}^q \varphi_i \right)$$

b) Calculul parametrilor părții MA a modelului

Evaluarea parametrilor θ_i ai modelului ARIMA al seriei se face cu relația:

$$\theta_i = -x_i/x_0, \quad i=1, \dots, q$$

unde x_1 și x_0 reprezintă rădăcinile următorului sistem neliniar:

$$f_i(x_0, \dots, x_q) = \sum_{j=0}^{q-i} x_j x_{i+j} - C'_i = 0, \quad i=0, \dots, q$$

Calculul rădăcinilor sistemului se realizează cu o metodă propusă de Brown (1969, 1971).

Valorile C'_i reprezintă covarianțele modificate ale seriei și se determină conform relațiilor:

$$C'_i = \begin{cases} \sum_{j=0}^p \sum_{k=0}^p \varphi_j \varphi_k ACV_{|i+j-k|+1}, & p > 0 \\ ACV_{i+1}, & p = 0 \end{cases}$$

unde $\varphi_0 = -1$. Dispersia reziduurilor modelului este x_0^2 .

Dispunând de valorile preliminare ale parametrilor modelului ARIMA al seriei, estimațiile de verosimilitate maximă ale parametrilor se obțin prin minimizarea unei funcții obiectiv de forma:

$$S \equiv \sum_{t=1}^N a_t^2$$

cu $a_t = 0$ pentru $t=1, \dots, \max(p, q)$, utilizând algoritmul modificat al pantei de coborîre maximă (Luenberger, 1973). Convergența algoritmului este presupusă în cazul îndeplinirii unuia din următoarele criterii:

- (i) numărul de digiți semnificativi fixat ai funcției obiectiv nu se modifică după un număr fixat de iterații;
- (ii) modificarea parametrilor modelului de la o iterație la alta este nesemnificativă.

Exemplu:

AB&J > ARIMA

SPECIFICAȚI STRUCTURA MODELULUI ? 0 0 2

DORIȚI SĂ INTRODUCEȚI ESTIMAȚIILE PRELIMINARE ALE PARAMETRI-
LOR ? N

PARAMETRI INIȚIALI MA :

— .53388 .19593

ESTIMAȚIILE DE VEROSIMILITATE MAXIMĂ ALE PARAMETRIILOR
MODELULUI ARIMA (0, 0, 2) PENTRU 120 DE VALORI ALE SERIEI
DUPĂ DIFERENȚIERE

CONSTANTA DE MEDIE ALUNECĂTOARE = — .13741

DISPERSIA ZGOMOTULUI ALB = .99076

PARAMETRI MA :

— .65530 .30809

Comanda AGAIN

Funcția : Realizează calculul estimațiilor de verosimilitate maximă ale parametrilor modelului ARIMA și reziduurilor modelului utilizând ca valori inițiale ale parametrilor estimațiile obținute cu comanda ARIMA.

Metoda : Se utilizează procedura descrisă în cadrul comenzii ARIMA, pentru valori specificate ale estimațiilor preliminare ale parametrilor modelului i seriei.

Exemplu :

```
AB&J > AGAIN
```

```
PARAMETRI INIȚIALI AR :  
.90194
```

```
PARAMETRI INIȚIALI MA :  
.55885
```

```
ESTIMAȚIILE DE VEROSIMILITATE MAXIAMĂ ALE PARAMETRILOR  
MODELULUI ARIMA (1, 0, 1) PENTRU 197 DE VALORI ALE SERIEI  
DUPĂ DIFERENȚIERE
```

```
CONSTANTA DE MEDIE ALUNECĂTOARE = 1.63077  
DISPERSIA ZGOMOTULUI ALB = .09786
```

```
PARAMETRI AR :  
.90442
```

```
PARAMETRI MA :  
.56341
```

Comanda SMOD

Funcția : Realizează salvarea modelului curent ARIMA ($p; q; d; \varphi_i, i=1, \dots, p; \theta_i, i=1, \dots, q; \sigma^2$) într-un fișier pe disc al cărui nume este specificat de utilizator. σ^2 reprezintă dispersia reziduurilor.

Exemplu :

```
AB&J > SMOD
```

```
NUMELE FIȘIERULUI? MOD1  
MODELUL SALVAT ÎN FIȘIERUL : MOD1
```

Comanda RMOD

Funcția : Realizează încărcarea în memorie a modelului ARIMA ($p; q; d; \varphi_i, i=1, \dots, p; \theta_i, i=1, \dots, q; \sigma^2$) dintr-un fișier pe disc al cărui nume este specificat de utilizator.

Exemplu :

```
AB&J > RMOD
```

```
NUMELE FIȘIERULUI? MOD1  
MODELUL ÎNCĂRCAT DIN FIȘIERUL : MOD1
```

Comanda SWAP

Funcția : Realizează interschimbarea datelor conținute în vectorii de lucru ANDAT și RESID în scopul analizei reziduurilor în cadrul etapei de validare a modelului.

Exemplu :

AB&J > SWAP

Comanda ORIG

Funcția : Transferă datele originale din vectorul ORDAT în vectorul ANDAT, în cazul în care prin utilizarea unor comenzi de tipul TRANS, DIFF etc. datele din vectorul ANDAT au fost modificate și se dorește a se lucra cu datele seriei originale.

Exemplu :

AB&J > ORIG

Comanda FOREC

Funcția : Realizează calculul valorilor de predicție și al limitelor de probabilitate al acestora, utilizând ultimul model determinat al seriei.

Metoda : Calculul valorilor de predicție și al limitelor de probabilitate al acestora se realizează utilizând modelul ARIMA al seriei care poate fi exprimat în termenii unei sume ponderate a valorilor curente și trecute ale zgomotului a_t :

$$X_t = \sum_{j=0}^{\infty} \psi_j a_{t-j}$$

unde $\psi_0=1$, iar ponderile ψ se obțin din ecuația :

$$\Phi(B)(1 + \psi_1 B + \psi_2 B^2 + \dots) = \Theta(B)$$

Procedura de predicție este cea prezentată în secțiunea 2 a acestei lucrări, unde a_n, a_{n-1}, \dots reprezintă reziduurile generate de model, iar a_{n+1}, a_{n+2}, \dots etc. au valori nule ; X_{n+1}, X_{n+2}, \dots reprezintă valorile de predicție ale seriei. Predicția se realizează la momentul $t=n$.

Limitele de probabilitate $1-\epsilon, X_{n+m}(-)$ și $X_{n+m}(+)$ pentru X_{n+m} sînt date de :

$$X_{n+m}(\pm) = \hat{X}_{n+m} \pm u_{\epsilon/2} \left\{ 1 + \sum_{j=1}^{m-1} \psi_j^2 \right\}^{1/2} s_a$$

unde s_a^2 reprezintă valoarea estimată a dispersiei σ_a^2 , iar $u_{\epsilon/2}$ este un coeficient referitor la abaterea de la distribuția normală a zgomotului a_t .

Exemplu :

AB&J > FOREC

EPSILON ? 0,05

NUMĂR VALORI PREDICȚIE ? 8

MOMENT PREDICȚIE	PONDERI PSI	VALOARE PREDICȚIE	EROARE STD. 5.00 %
1	1.51587	7.27307	.72662
2	1.69071	6.94729	1.31954
3	1.64252	6.59124	1.80288
4	1.46333	6.24930	2.16213
5	1.22095	5.94715	2.40943
6	.96233	5.69674	2.56757
7	.71746	5.50060	2.66108
8	.50329	5.35531	2.71167

Comanda UPDATE

Funcția : Realizează actualizarea valorilor de predicție ale seriei după obținerea unor noi observații. În cadrul acestei comenzi se parcurg de regulă următoarele etape :

(a) Citirea datelor seriei și încărcarea acestora în vectorii ANDAT și ORDAT (ca în cazul comenzii INPUT) ;

(b) Încărcarea în memorie a modelului ARIMA al seriei pentru care se realizează predicția (ca în cazul comenzii RMOD) ;

(c) Specificarea noii observații care se adaugă la datele seriei inițiale ;

(d) Calculul valorilor de predicție și al limitelor de probabilitate al acestora (ca în cazul comenzii FOREC) :

(e) Salvarea într-un fișier pe disc a seriei actualizate.

Unele din aceste etape pot fi evitate dacă în locul introducerii numelui fișierului ce so solicită de către program se apasă tasta <CR>.

Metoda : Procedurile utilizate în cadrul acestei comenzi sînt cele prezentate în cadrul comenzilor INPUT, RMOD, FOREC, înlănțuirea lor fiind realizată de secvența de calcul specifică comenzii UPDATE.

Exemplu :

AB&J > UPDATE

DATELE CITITE DIN FIȘIERUL : SER

132 DATE CITITE DIN FIȘIERUL : SER

3.7300 3.67000 3.77000 3.8300

NUME FIȘIER MODEL ? MODX

MODELUL ÎNCĂRCAT DIN FIȘIERUL : MODX

ESTIMAȚIILE DE VEROSIMILITATE MAXIMĂ ALE PARAMETRIILOR
MODELULUI ARIMA (2, 0, 0) PENTRU 132 DE VALORI ALE SERIEI
DUPĂ DIFERENȚIERE

CONSTANTA DE MEDIE ALUNECĂTOARE = .47593

DISPERSIA ZGOMOTULUI ALB = .13744

PARAMETRI AR :

1.51587 —.60716

VALOAREA NOII OBSERVAȚII ? 7.35

EPSILON ? .5

NUMĂR VALORI PREDICȚIE ? 8

MOMENT PREDICȚIE	PONDERI PSI	VALOARE PREDICȚIE	EROARE STD. 5.00%
1	1.51587	7.06391	.72400
2	1.69071	6.72130	1.31478
3	1.64252	6.37566	1.79639
4	1.46333	6.05972	2.15434
5	1.22095	5.79066	2.40075
6	.96233	5.57463	2.55832
7	.71746	5.41051	2.65149
8	.50329	5.29289	2.70190

FIȘIERUL ÎN CARE SE SCRIU DATELE ? BETA

133 DATE SCRISE ÎN FIȘIERUL BETA

Comanda STAT

Funcția : Realizează afișarea numărului de date conținute în vectorii ORDAT, ANDAT, RESID, FOREC.

Exemplu :

```
AB&J > STAT
```

```
197 DATE ÎN VECTORUL DATELOR ORIGINALE ORDAT.
197 DATE ÎN VECTORUL DATELOR CE SE ANALIZEAZĂ ANDAT.
75 Date ÎN VECTORUL REZIDUURILOR RESID.
0 DATE ÎN VECTORUL VALORILOR DE PREDICȚIE FOREC.
```

Comanda CONC

Funcția : Realizează concatenarea valorilor de predicție la secvența datelor seriei analizate.

Exemplu :

```
AB&J > CONC
```

Comanda EXIT

Funcția : Închide fișierul de ieșire pe disc, dacă acesta este deschis și încheie sesiunea de lucru cu pachetul AUTOB&J.

Exemplu :

```
AB&J > EXIT
```

3.4. Mesaje de eroare

Practic este imposibil a se specifica acțiunea ce se impune din partea utilizatorului în cazul apariției unui mesaj de eroare, datorită numărului mare și varietății mesajelor de eroare ce pot apare în timpul unei sesiuni de lucru. Totuși, în general, acestea pot fi grupate în următoarele categorii :

(a) Mesaje de eroare ce apar în cadrul rutinelor de scriere/citire pe disc : eroare deschidere/închidere fișier, scriere/citire date etc.

(b) Mesaje de eroare datorate specificării de către utilizator a unor valori necorespunzătoare pentru parametrii de calcul ;

(c) Mesaje de eroare datorate unor probleme de natură numerică ce apar în procedurile de calcul incluse în pachet, de exp. lipsa convergenței parametrilor modelului după un număr specificat de iterații etc.

Pentru mesajele de eroare aparținând primelor două categorii operația ce a generat apariția erorii este reluată. În cazul mesajelor din ultima categorie, utilizatorul urmează să-și reevalueze strategia aleasă.

3.5. Restricții

Pachetul AUTOB&J este proiectat pentru :

(a) Numărul maxim de date al seriei = 500.

(b) Numărul maxim de valori ale funcțiilor de autocorelație și de autocorelație parțială = 50.

- (c) Ordinul maxim al părții AR și al părții MA a modelului stohastic ARIMA = 20.
 - (d) Numărul maxim de valori de predicție ale seriei = 100.
 - (e) Numărul maxim de iterații pentru estimarea de verosimilitate maximă a parametrilor modelului stohastic ARIMA = 25.
 - (f) Numărul de digiți semnificativi ai funcției obiectiv utilizate în cadrul algoritmului de estimare prin metoda verosimilității maxime a parametrilor modelului ARIMA = 6.
- În cazul în care aceste restricții nu sunt respectate, utilizatorul va fi avertizat printr-un mesaj adecvat și sesiunea de lucru continuă.

4. PROCEDURA PRACTICĂ DE MODELARE ȘI PREDICȚIE

În scopul facilitării înțelegerii modului de utilizare a pachetului de programe în cadrul unei probleme practice de modelare și predicție, în Fig. 3 se prezintă o diagramă ce ilustrează modul în care programele individuale, specificate prin comenzi, interacționează unul cu celălalt pentru atingerea obiectivului final. În cadrul diagramei, programele incluse în aria mărginită de liniile întrerupte se utilizează în mod repetat, pînă cînd datele ce rezultă sînt aduse într-o formă adecvată prelucrării următoare.

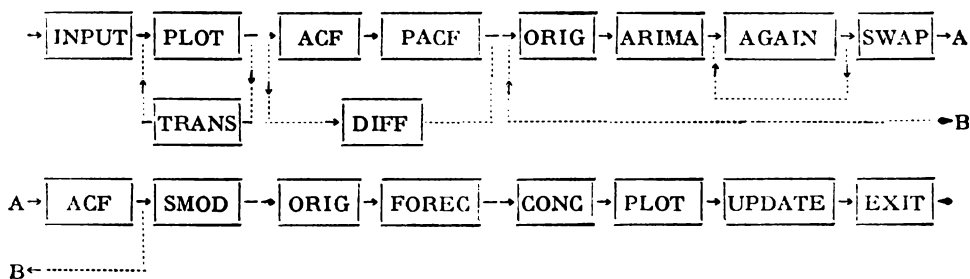


Fig. 3 Procedura practică de modelare și predicție a seriilor de timp utilizând pachetul AUTOB&J

La începutul unei sesiuni de lucru și după terminarea etapelor de calcul specifice unei comenzi, pe terminalul utilizator se va afișa mesajul "AB&J >", urmînd ca utilizatorul să specifice comanda dorită, conform strategiei de modelare și predicție alese. Comenzile sînt interactiv active și adresează utilizatorului întrebările specifice etapelor de calcul din cadrul comenzii-specificate.

Prezentăm în continuare strategia de modelare și predicție, așa cum apare în cadrul diagramei reprezentate în Fig. 3.

(i) Datele seriei sînt citite din fișierul specificat de utilizator (comanda INPUT) și sînt reprezentate grafic (comanda PLOT). În cadrul etapelor următoare de modelare se pot utiliza datele originale ale seriei sau datele obținute în urma unor transformări (comanda TRANS).

(ii) Se determină și se analizează funcțiile de autocorelație și de autocorelație parțială (comenzile ACF și PACF) în scopul detectării unor eventuale nestaționarități ale seriei sub forma unor tendințe stohastice. În cazul prezenței nestaționarităților de acest tip, seria va fi diferențiată de d ori ($d > 1$) pînă cînd se va observa că funcția de autocorelație a seriei se atenuază rapid (se presupune că nestaționaritatea se datorează celor d rădăcini egale cu unitatea ale polinomului Φ al modelului seriei). Fluctuațiile sezoniere, de tipul ciclurilor în date, pot fi eliminate în mod similar prin calculul diferențelor seriei cu o perioadă egală cu perioada fluctuațiilor (comanda DIFF).

Dacă în continuare se dorește efectuarea analizei cu datele originale ale seriei, deoarece comanda DIFF distruge aceste date din vectorul de date ANDAT, se va utiliza comanda ORIG ce are drept scop reincărcarea datelor originale ale seriei în vectorul de date ANDAT.

În cadrul etapei de analiză a funcțiilor de autocorelație și de autocorelație parțială pe lângă valoarea parametrului d se vor determina de asemenea și ordinele părților AR și MA ale modelului stohastic ARIMA, p și q ; este posibilă chiar estimarea preliminară a parametrilor modelului.

(iii) În urma utilizării comenzii ARIMA se determină estimățiile de verosimilitate maximă ale parametrilor modelului ARIMA specificat în etapa anterioară (p, d, q) și valorile reziduurilor modelului. Dacă parametrii modelului seriei nu converg în mai puțin de 25 de iterații ale procedurii de estimare, se poate utiliza, în continuare, comanda AGAIN pentru reluarea procedurii de estimare a parametrilor modelului, cu valorile curente ale acestora, drept valori inițiale.

(iv) În scopul validării modelului seriei se va utiliza, într-o primă etapă, comanda SWAP care realizează încărcarea în vectorul de date ANDAT a seriei reziduurilor. În cazul în care rezultatele analizei reziduurilor (se poate utiliza și comanda ACF) nu denotă abateri ale acestora de la comportarea unei serii de tip zgomot alb, modelul ARIMA determinat este reprezentativ pentru seria analizată și poate fi salvat într-un fișier pe disc, utilizând comanda SMOD. În caz contrar, procedura de analiză și modelare se va relua de la etapa de staționarizare a seriei și specificare a modelului.

(v) Pentru realizarea predicției valorilor seriei se utilizează într-o primă etapă, comanda ORIG în scopul încărcării în vectorul de date ANDAT a datelor originale ale seriei, după care calculul valorilor de predicție și al limitelor de probabilitate al acestora se efectuează cu comanda FOREC. Valorile de predicție pot fi concatenate cu datele seriei (comanda CONC) și reprezentate grafic (comanda PLOT).

(vi) În cazul în care se dorește actualizarea valorilor de predicție ale seriei, în urma obținerii unei noi observații, se va utiliza comanda UPDATE.

Deși toate programe de calcul incluse în pachet sînt eficiente din punct de vedere al calculelor efectuate, procedura completă de modelare și predicție necesită un oarecare timp de lucru; aceasta în primul rînd datorită timpului de luare a unor decizii, necesar operatorului în primele etape ale analizei preliminare de specificare a modelului seriei, aceasta reprezentînd o artă și fiind necesară o anumită experiență pentru realizarea sa în mod eficient. Timpul total necesar rezolvării unei probleme de modelare și predicție se va reduce pe măsură ce utilizatorul va căștiga experiență în recunoașterea unor forme tipice în cadrul reprezentărilor funcțiilor de autocorelație și de autocorelație parțială, care nu apar întotdeauna așa de clar cum sînt prezentate de Box și Jenkins (1976).

De asemenea, informația a priori de care se dispune, legată de natura fizică a procesului ce se analizează, poate contribui în mare măsură la reducerea timpului de analiză, în special în cadrul etapelor de specificare a modelului seriei, conducînd chiar la omiterea unor etape din cadrul analizei.

Cu toate că în cadrul acestei lucrări s-a căutat să se realizeze o prezentare netehnică a funcțiunilor pachetului, utilizarea unor comenzi necesită cunoașterea în detaliu a metodologiei de modelare și predicție.

5. APLICAȚIE

Pachetul de programe AUTOB&J a fost utilizat pentru modelarea și predicția unui număr mare de serii de timp reale. Cîteva din aceste aplicații sînt prezentate într-o lucrare elaborată de autor (Popescu, 1985b) și constau în modelarea și predicția unor serii de timp din procese industriale, modelarea unei populații biologice, modelarea și predicția transportului aerian internațional de călători.

În cele ce urmează se prezintă în detaliu rezultatele modelării și predicției unei serii de timp reprezentînd concentrația într-un proces chimic, măsurată la intervale de două ore, una din seriile din colecția de serii de timp utilizate ca exemple de test de către Box și Jenkins (1976). Seria conține 197 eșantioane. Aceeași serie este analizată și în lucrarea elaborată de Terțico, Stoica, Popescu (1985) utilizînd analiza canonică de corelație.

Exemplul ce urmează ilustrează implementarea diferitelor etape, specifice metodologiei Box-Jenkins, în cadrul pachetului de programe AUTOB&J.

AUTOB&J VERSIUNEA 1.00

AB&J > INPUT

DATELE CITITE DIN FIȘIERUL ? BJS

197 DATE CITITE DIN FIȘIERUL : BJS

17.00000 16.60000 16.30000 16.10000 . . .

AB&J > PRINT

197 VALORI

17.00000	16.6000	16.30000	16.10000
17.10000	16.90000	16.80000	17.40000
17.10000	17.00000	16.70000	17.40000
.....
.....
.....
17.40000	17.00000	18.00000	18.20000
17.60000	17.80000	17.70000	17.20000
17.40000			

AB&J > ACF

NUMĂRUL TERMENILOR ? 20

VALOAREA MEDIE = 17.06294

DISPERSIA = 0.15837

-1.0 -0.8 -0.6 -0.4 -0.2 0.0 0.2 0.4 0.6 0.8 1.0
 +.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+

1	0.572	XXXXXXXXXXXXXXXXXX
2	0.496	XXXXXXXXXXXXXXXXXX
3	0.400	XXXXXXXXXXXXXX
4	0.356	XXXXXXXXXXXXXX
5	0.327	XXXXXXXXXXXXXX
6	0.353	XXXXXXXXXXXXXX
7	0.390	XXXXXXXXXXXXXX
8	0.321	XXXXXXXXXXXXXX
9	0.303	XXXXXXXXXXXXXX
10	0.253	XXXXXXXXXX
11	0.188	XXXXXXX
12	0.164	XXXXXX
13	0.192	XXXXXX
14	0.235	XXXXXXX
15	0.138	XXXX
16	0.178	XXXXX
17	0.194	XXXXXX
18	0.201	XXXXXX
19	0.140	XXXXX
20	0.182	XXXXXX

AB&J > PACF

NUMĂRUL TERMENILOR ? 20

VALOAREA MEDIE = 17.06294

DISPERSIA = 0.15837

	-1.0	-0.8	-0.6	-0.4	-0.2	0.0	0.2	0.4	0.6	0.8	1.0
	+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+										
1	0.572						XXXXXXXXXXXXXX				
2	0.251						XXXXXXX				
3	0.070						XXX				
4	0.067						XXX				
5	0.065						XXX				
6	0.129						XXXX				
7	0.150						XXXXX				
8	-0.032					XX					
9	0.009					X					
10	-0.020					X					
11	-0.069					XXX					
12	-0.020					X					
13	0.054					XX					
14	0.091					XXX					
15	-0.126					XXXX					
16	0.048					XX					
17	0.097					XXX					
18	0.069					XXX					
19	-0.071					XXX					
20	0.051					XX					

AB&J > DIFF

AVERTISMENT : ACEASTĂ FUNCȚIE VA DISTRUGE CONȚINUTUL VECTORULUI DATELOR SERIEI

ORDINUL DIFERENȚEI NESEZONIERE ? 1

ORDINUL DIFERENȚEI SEZONIERE ? 0

S-A DETERMINAT DIFERENȚA NESEZONIERĂ DE ORDIN 1

AB& > ACF

NUMĂRUL TERMENILOR ? 20

VALOAREA MEDIE = 0.00204

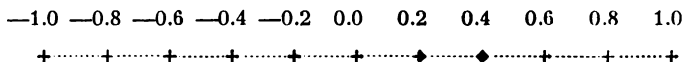
Dispersia = 0.13551

	-1.0	-0.8	-0.6	-0.4	-0.2	0.0	0.2	0.4	0.6	0.8	1.0
	+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+										
1	0.411					XXXXXXXXXXXXXX					
2	0.016					X					
3	-0.063					XXX					
4	-0.013					X					
5	-0.072					XXX					
6	-0.011					X					
7	0.139					XXXX					
8	-0.066					XXX					
9	0.038					XX					
10	0.017					X					
11	-0.045					XX					
12	-0.059					XX					
13	-0.017					X					
14	0.165					XXXXX					
15	-0.174					XXXXX					
16	0.035					XX					
17	0.012					X					
18	0.082					XXX					
19	-0.121					XXX					
20	0.153					XXXXX					

AB&J > PACF

NUMĂRUL TERMENILOR ? 20
VALOAREA MEDIE = 0.00204

DISPERSIA = 0.13551



1	-0.411	XXXXXXXXXXXX
2	-0.184	XXXXXX
3	-0.163	XXXXX
4	-0.138	XXXX
5	-0.198	XXXXXX
6	-0.207	XXXXXX
7	-0.001	X
8	-0.046	XX
9	-0.015	X
10	0.037	XX
11	-0.007	X
12	-0.070	XXX
13	-0.106	XXXX
14	0.106	XXXX
15	-0.087	XXX
16	-0.128	XXXX
17	-0.097	XXX
18	0.045	XX
19	-0.072	XXX
20	0.094	XXX

AB&J > ORIG

AB&J > ARIMA

SPECIFICAȚI STRUCTURA MODELULUI ? 0 1 1

DORIȚI SĂ INTRODUCEȚI ESTIMAȚIILE INIȚIALE ALE PARAMETRILOR ? N

PARAMETRI INIȚIALI MA :

0.52418

ESTIMAȚIILE DE VEROSIMILITATE MAXIMĂ ALE PARAMETRILOR
MODELULUI ARIMA (0, 1, 1) PENTRU 196 de VALORI ALE SERIEI
DUPĂ DIFERENȚIERE

CONSTANTA DE MEDIE ALUNECĂTOARE = 0.00204
DISPERSIA ZGOMOTULUI ALB = 0.09940

PARAMETRI MA :

0.69146

AB&J > ARIMA

SPECIFICAȚI STRUCTURA MODELULUI ? 1 0 1

DORIȚI SĂ INTRODUCEȚI ESTIMAȚIILE INIȚIALE ALE PARAMETRILOR ? Y

ESTIMAȚII INIȚIALE PENTRU MODELUL AR ? 2

ESTIMAȚII INIȚIALE PENTRU MODELUL MA ? 2

PARAMETRI INIȚIALI AR :

0.20000

PARAMETRI INIȚIALI MA :

0.20000

AVERTISMENT : DUPĂ 25 ITERAȚII PARAMETRII MODELULUI NU CONVERG
 ESTIMAȚIILE DE VEROSIMILITATE MAXIMĂ ALE PARAMETRIILOR
 MODELULUI ARIMA (1, 0, 1) PENTRU 197 DE VALORI ALE SERIEI
 DUPĂ DIFERENȚIERE

CONSTANTA DE MEDIE ALUNECĂTOARE = 1.69613

DISPERSIA ZGOMOTULUI ALB = 0.09741

PARAMETRI AR :

0.90060

PARAMETRI MA :

0.55382

AB&J > AGAIN

PARAMETRI INIȚIALI AR :

0.90060

PARAMETRI INIȚIALI MA :

0.55382

ESTIMAȚIILE DE VEROSIMILITATE MAXIMĂ ALE PARAMETRIILOR
 MODELULUI ARIMA (1, 0, 1) PENTRU 197 DE VALORI ALE SERIEI
 DUPĂ DIFERENȚIERE

CONSTANTA DE MEDIE ALUNECĂTOARE = 1.65346

DISPERSIA ZGOMOTULUI ALB = 0.09731

PARAMETRI AR :

0.90311

PARAMETRI MA :

0.55899

AB&J > SMOD

NUMELE FIȘIERULUI ? MOD

MODELUL SALVAT ÎN FIȘIERUL : MOD

AB&J > ORIG

AB&J > FOREC

EPSILON ? 0.05

NUMĂR VALORI PREDICȚIE ? 8

MOMENT PREDICȚIE	PONDERI PSI	VALOARE PREDICȚIE	EROARE STD. 5.00%
1	0.34408	17.36903	0.59685
2	0.31074	17.33937	0.63119
3	0.28063	17.31258	0.65788
4	0.25343	17.28839	0.67886
5	0.22887	17.26654	0.69551
6	0.20670	17.24681	0.70880
7	0.18667	17.22900	0.71945
8	0.16858	17.21290	0.72803

AB&J > UPDATE

DATELE CITITE DIN FIȘIERUL ? BJS

197 DATE CITITE DIN FIȘIERUL : BJS

17.00000

16.60000

16.30000

16.10000

NUME FIȘIER MODEL ? MOD

MODELUL ÎNCĂRCAT DIN FIȘIERUL : MOD

ESTIMAȚIILE DE VEROSIMILITATE MAXIMĂ ALE PARAMETRILOR
MODELULUI ARIMA (1, 0, 1) PENTRU 197-DE VALORI ALE SERIEI
DUPĂ DIFERENȚIERE

CONSTANTA DE MEDIE ALUNECĂTOARE = 1.65346

DISPERSIA ZGOMOTULUI ALB = 0.09731

PARAMETRI AR :

0.90311

PARAMETRI MA :

0.55899

VALOAREA NOII OBSREVAȚII ? 17.42

EPSILON ? 0.05

NUMĂR VALORI PREDICȚIE ? 10

MOMENT PREDICȚIE	PONDERI PSI	VALOARE PREDICȚIE	EROARE STD. 5.00 %
1	0.34408	17.35690	0.59538
2	0.31074	17.32842	0.62964
3	0.28063	17.30269	0.65626
4	0.25343	17.27946	0.67719
5	0.22887	17.25848	0.69380
6	0.20670	17.23953	0.70705
7	0.18667	17.22241	0.71768
8	0.16858	17.20696	0.72624
9	0.15224	17.19300	0.73314
10	0.13749	17.18040	0.73872

FIȘIERUL ÎN CARE SE SCRIBU DATELE ? UPD

198 DATE SCRIBU ÎN FIȘIERUL UPD

AB&J > EXIT

6. CONCLUZII

Pachetul AUTOB&J reprezintă un sistem de programe complet interactiv pentru modelarea și predicția seriilor de timp, proiectat pentru implementarea metodologiei Box-Jenkins, ce reprezintă una dintre cele mai larg utilizate și precise tehnici de predicție pe termen scurt.

Produsul este ușor de utilizat în practică, robust, portabil și necesită resurse de calcul modeste.

Pachetul oferă soluții pentru rezolvarea unor probleme de modelare și predicție ce apar în industrie, transporturi, agricultură, geologie, meteorologie, demografie, biologie și medicină, ecologie, economie etc.

BIBLIOGRAFIE

- Akaike, H., E. Arakata și T. Ozaki* (1975). **A Computer Time Series and Control Program Package (1), (2)**, Computer Science Monographs, 5, 6, The Institute of Statistical Mathematics, Tokyo.
- Box, G.E.P. și G.M. Jenkins* (1976). **Time Series Analysis : Forecasting and Control**, Revised Edition, Holden Day, San Francisco.
- Brown, K.M. și J.E. Dennis, Jr.* (1971). **On the second order convergence of Brown's derivative-free method for solving simultaneous nonlinear equations**, Yale University Department of Computer Science, Technical Report.
- Dağum, E.B.* (1979). **The X-11-ARIMA Seasonal Adjustment Method**, Outline of the Methodology, Statistics Canada.
- Goel, P.K. și A.G. Rocco* (1982). **ARIMA Modelling and Forecasting : An Interactive Program Based on IMSL Subroutine Package – II**, Technical Report 82–12, Department of Statistics, Purdue University.
- Popescu, Th. D.* (1983). **TSPACK – Subroutine Package for Time Series Analysis and Forecasting**, Report TSA&F/1, Institutul Central pentru Conducere și Informatică, București.
- Popescu, Th. D.* (1984). **Analiza și predicția seriilor de timp. Aspecte metodologice și aplicații**, Raport de Cercetare TSA&F/2, Institutul Central pentru Conducere și Informatică București.
- Popescu, Th. D.* (1985a). **AUTOB&J – A Computer Aided Procedure for Time Series Analysis and Forecasting**, 3rd IFAC Symposium on Computer Aided Design in Control and Engineering Systems, Technical University of Denmark, Lyngby.
- Popescu, Th. D.* (1985b). **Experiences with a Computer Aided Procedure for Time Series Analysis and Forecasting Using Box-Jenkins Philosophy**, 2nd International Symposium on System Analysis and Simulation, 26–31 August 1985, Berlin.
- Luenberger, D.G.* (1973). **Introduction to Linear and Nonlinear Programming**, Addison-Wesley, Reading, Massachusetts.
- Reilly, D.P.* (1984). **An Overview of the PACK System and AUTOBJ/AUTOBOX**, Automatic Forecasting Systems Inc., Hatboro.
- Shellswell, S.H.* (1972). **A Computer Aided Procedure for Time Series Analysis and Identification of Noisy Processes (CAPTAIN)**, CUED/B Control/TR 25, Cambridge.
- Terlișco, M., P. Stoica și Th. Popescu* (1985). **Modelarea și predicția seriilor de timp**, Editura Academiei, București.

ACTIVITATEA SECȚIEI DE AUTOMATICĂ A CNIT

În cadrul Secției de Automatică a CNIT a fost constituită comisia „Tehnologii, echipamente și metode de asigurare a calității mijloacelor de automatizare”. În cadrul acestei comisii, au fost cooptați specialiști din domeniile proiectării și utilizării sistemelor de testare automată, a controlului calității și fiabilității echipamentelor de automatizare și pentru informatică industrială, precum și specialiști în domeniul proiectării tehnologice și asigurării calității acestor produse. Membrii comisiei provin din unitățile cele mai reprezentative din cercetare, proiectare, producție și din institutele de învățământ superior din București, Cluj-Napoca, Timișoara, Craiova.

Conducerea comisiei este asigurată de către tov. profesor dr. ing. Marius Hăngănuț — șef filială I.P.A. Cluj-Napoca — președinte, și dr. ing. M. Vlăduțiu — Institutul Politehnic Timișoara; dr. ing. I. Hohan — Institutul Politehnic București vicepreședinti, și ing. D. Nasui — I.I.R.U.C. București — secretar.

Obiectivele acestei comisii constau în promovarea și intensificarea în cadrul tuturor întreprinderilor a metodelor moderne de asigurare și control a calității, fiabilității, mentenabilității și disponibilității echipamentelor de automatizare, precum și în introducerea tehnologiilor de testare automată.

Comisia își desfășoară activitatea conform unui plan anual care cuprinde diverse manifestări ca: mese rotunde, dezbateri, conferințe în întreprinderi, simpozioane.

Din planul comisiei pe anul 1985 se menționează :

- Creșterea productivității și competitivității producției în industria textilă prin automatizare;
- Calitatea producției prin utilizarea aparatului analitice automate — analizoare de gaze;
- Tehnologii de proiectare și testare asistată de calculator;
- Proiectarea pentru testabilitate a echipamentelor de automatizare;
- Metode moderne de analiză sistemică a fiabilității, mentenabilității și disponibilității echipamentelor de automatizare;
- Problematika generării automate a testelor, stadiul actual și perspective;
- Proiectarea și realizarea standurilor de testare — rolul acestora în creșterea calității.

În atenția comisiei stă și activitatea de coordonare a publicării lucrărilor elaborate în domeniu. Evenimentul major al comisiei îl constituie „Simpozionul de tehnologii și echipamente de testare automată” organizat anual la Cluj-Napoca și care în acest an se află la a patra ediție.

Toți cei care sînt interesați în activitatea comisiei pot obține informații suplimentare scriind pe adresa :

IPA, Filiala Cluj-Napoca
pt. comisia CNIT
3400 Cluj-Napoca
Str. Republicii nr. 109
Telefon 16155/951. Telex 031.391.

Ciclul
„Proiectarea asistată
de calculator“
Sisteme automate

**TEHNICI DE FACTORIZARE ÎN ALGORITMI
DE CONDUCERE ADAPTIVĂ**

Dr. ing. V. Sima
I.T.C.I.

Numeroși algoritmi propuși recent pentru conducerea adaptivă a sistemelor multivariabile discrete utilizează metoda recursivă a celor mai mici patrate (CMMPR). Exemple tipice sînt algoritmi CMMPR modificate sau CMMPR ponderate. Programarea directă a relației de actualizare a matricei de covarianță care apare în acești algoritmi implică posibile dificultăți numerice: instabilitate, pierderea pozitivității. Lucrarea tratează tehnicile de factorizare aplicabile matricei de covarianță sau inversei acesteia, care permit evitarea dificultăților menționate. Sînt prezentate și analizate noi variante ale algoritmilor de factorizare; sînt date detalii de implementare și recomandări pentru utilizarea unor rutine matematice performante, recent elaborate, pentru rezolvarea problemelor de calcul asociate. De asemenea, sînt discutate unele rezultate numerice obținute și sînt listate subprograme FORTRAN originale pentru actualizarea factorizărilor.

1. Introducere

Numeroase procese fizice necesită sisteme de reglare capabile să lucreze în diferite puncte de funcționare. Datorită neliniarității, un singur model liniar nu poate descrie corespunzător un proces în toate punctele de funcționare. În consecință, un sistem de reglare cu parametri fixați, determinați pe baza unui anumit model, se poate dovedi necadecvat conducerii procesului în ansamblu. Soluții pentru problema modificării parametrilor ar fi: utilizarea mai multor legi de reglare și a unei strategii de comutare corespunzătoare, sau folosirea unui sistem de reglare cu coeficienți funcționali dependenți de parametrii modelului, precalculați și memorați pentru fiecare punct de funcționare. Nici una din aceste soluții nu asigură o funcționare optimă dacă numărul de parametri este mare, sau dacă se impun reglării cerințe de calitate severe. În astfel de cazuri devine promițătoare adoptarea unui sistem de reglare adaptiv,

capabil să-și ajusteze parametrii, pe baza informațiilor colectate din proces, pentru a compensa modificările lente dar semnificative ale caracteristicilor procesului la trecerea de la un punct de funcționare la altul.

În ultimii ani eforturile de cercetare în domeniul sistemelor adaptive s-au intensificat considerabil. Principalele rezultate teoretice obținute sînt stabilitatea globală a unor legi de conducere adaptivă a sistemelor monovariabile continue deterministe [1], [2] și a unor algoritmi pentru sisteme mono- și multivariabile discrete deterministe [3]—[6], cît și convergența globală cu probabilitate 1 a unor algoritmi de conducere, predicție și filtrare adaptivă pentru sisteme mono- și multivariabile discrete stocastice [7]—[14]. Unele rezultate originale au fost prezentate de autor în [15—18] și se aplică unor algoritmi de conducere adaptivă a sistemelor de fază minimă monovariabile discrete, deterministe sau stocastice. Principalele instrumente matematice folosite în analiza legilor și algoritmilor adaptivi sînt teoria stabilității Liapunov, teoria hiperstabilității și, mai general, teoria pasivității. În cazul discret stocastic se utilizează din ce în ce mai mult funcții Liapunov stocastice și rezultate ale teoriei martingalelor.

Pînă în prezent s-a manifestat tendința de a propune diverse soluții de conducere/predicție adaptivă și de a demonstra stabilitatea, respectiv convergența. Au existat și unele încercări de unificare, prin adoptarea unor algoritmi suficient de generali, reprezentînd în fapt clase de algoritmi [12]. De asemenea, abordările folosite permiteau generalizări, astfel încît multe rezultate stabilite pentru anumite categorii de procese au putut fi ușor extinse la alte categorii (de exemplu, de la sisteme monovariabile la multivariabile, de la sisteme de fază minimă la cele de fază neminimă etc). Totuși, deocamdată, nu există o teorie unitară, suficient de elaborată și generală, ca în cazul sistemelor liniare multivariabile.

Au fost studiate cu precădere două clase de algoritmi de conducere adaptivă: algoritmi utilizînd iterația aproximației stocastice și algoritmi utilizînd iterația celor mai mici patrate (CMMPR), în diverse variante. Există o bogată experiență numerică în simulare, privind comportarea acestor algoritmi. De asemenea, unii algoritmi au fost deja adoptați pentru conducerea cu calculator a unor procese tehnologice [29]. În aplicații, sînt preferați algoritmi de tip CMMPR, datorită vitezei lor de convergență superioare celei corespunzătoare algoritmilor de tip aproximație stocastică.

Implementarea algoritmilor CMMPR pe calculatoare de proces implică luarea în considerare a unor aspecte practice, cum ar fi: viteza de convergență, necesarul de memorie, dificultăți numerice datorate preciziei limitate a calculelor. Ultimul aspect este cel mai important, deoarece programarea directă a algoritmilor poate conduce la calcule prost condiționate și la instabilitate numerică, interzicînd utilizarea on-line a codurilor respective. Adoptarea tehnicilor de factorizare a matricei de covarianță sau a inversei acesteia (matricea de informație), prezentate în această lucrare, evită dificultățile numerice asociate implementării directe a algoritmilor. Astfel de tehnici au fost folosite în domeniul estimării secvențiale a parametrilor [19—23] și în problemele de optimizare neliniară utilizînd metode cvasi-Newton [24—26].

Orice algoritm de conducere adaptivă parcurge la fiecare tact două etape de calcul: actualizarea estimăției parametrilor $\theta(t)$ și determinarea mărimii de comandă $u(t)$, prelucrând estimăția $\theta(t)$ și informația curentă disponibilă în proces $\Phi(t)$. Lucrarea tratează etapa fundamentală de actualizare a estimăției, utilizând tehnici de factorizare fie a matricei de covarianță, fie a matricei de informație. Sînt discutate în detaliu, din punctul de vedere al implementării, atît factorizarea de tip rădăcină patrată [19], [22], cît și factorizarea de tip U—D [20, 21]. Principalul instrument matematic adoptat pentru actualizarea factorizării îl constituie rotațiile plane, standard [27], sau modificate [28]. Acestea oferă baza pentru o tratare unitară a diferiților algoritmi de factorizare și asigură în final proprietățile numerice bune ale acestora. Sînt obținute pe o nouă cale unele rezultate cunoscute; alte rezultate sînt originale. Sînt discutate detalii de implementare și se fac recomandări pentru utilizarea unor rutine matematice performante, recent elaborate, pentru rezolvarea problemelor implicate.

Lucrarea este organizată astfel. În secțiunea 2 sînt prezentați cîțiva algoritmi de conducere adaptivă tipici. În secțiunea 3 se tratează problema actualizării estimăției și se obțin relațiile de bază care vor fi folosite în continuare. În secțiunile 4 și 5 se prezintă tehnicile de factorizare a matricei de informație și, respectiv, de covarianță. În secțiunea 6 se discută unele rezultate numerice obținute. Secțiunea 7 conține concluziile lucrării. În Anexa 1 se deduce un algoritm de actualizare a factorizării unei matrice simetrice pozitiv definite după adunarea (scăderea) unei diade simetrice. În Anexele 2 și 3 sînt listate coduri tipice, elaborate de autor, pentru actualizarea factorizărilor.

2. Algoritmi tipici pentru conducere adaptivă

Înainte de a prezenta cîțiva algoritmi de conducere adaptivă vom indica temeiurile care stau la baza formulării lor.

Considerăm un sistem multivariabil descris de următorul model intrare/ieșire (de tip ARMA) *

$$A(q^{-1})y(t) = q^{-d}[B(q^{-1})u(t) + [C(q^{-1})]w(t)] \quad (1)$$

unde $y(t) \in \mathbb{R}^m$ este mărimea de ieșire a procesului care trebuie condus, $u(t) \in \mathbb{R}^r$ este mărimea de intrare (comanda), $w(t) \in \mathbb{R}^m$ este o mărime perturbatoare, q^{-d} reprezintă o întîrziere pură de d tacte, $A(q^{-1})$ este un polinom scalar în operatorul de întîrziere cu un tact q^{-1} , iar $[B(q^{-1})]$ și $[C(q^{-1})]$ sînt matrice ale căror elemente sînt polinoamele scalare $B_{ij}(q^{-1})$ și $C_{ij}(q^{-1})$, respectiv:

$$A(q^{-1}) = 1 + a_1q^{-1} + \dots + a_nq^{-n} \quad (2.a)$$

$$B_{ij}(q^{-1}) = (B_{ij})_0 + (B_{ij})_1q^{-1} + \dots + (B_{ij})_s q^{-s} \quad (2.b)$$

$$C_{ij}(q^{-1}) = (C_{ij})_0 + (C_{ij})_1q^{-1} + \dots + (C_{ij})_r q^{-r}, \quad (C_{ij})_0 = \begin{cases} 1, & i=j \\ 0, & i \neq j \end{cases} \quad (2.c)$$

* Rezultatele pot fi obținute și pentru anumite modele de stare [13, 14].

Coeficienții polinoamelor sînt necunoscuți și numai semnalele $\{y(t)\}$ și $\{u(t)\}$ sînt disponibile. Se presupune că procesul îndeplinește anumite condiții de stabilitate inversă și pasivitate [13, 14] cerute pentru convergența algoritmilor de conducere adaptivă. De asemenea, șirul $\{w(t)\}$ este un proces stocastic (de tip zgomot alb) cu medie nulă, satisfăcînd anumite condiții de independență condițională și de mărginire.

Obiectivul conducerii constă în determinarea comenzii $\{u(t)\}$, utilizînd o lege de reglare cu reacție, care să stabilizeze sistemul și să asigure că $\{y(t)\}$ urmărește cu dispersie condițională minimă o ieșire dorită, mărginită, $\{y^*(t)\}$.

Modelul (1) se poate rescrie în forma de predicție liniară [9], [13]

$$y(t) = \hat{y}^*(t) + v(t) \quad (3)$$

unde

$$\hat{y}^*(t) = q^{-d} \{ [G(q^{-1})] \hat{y}^*(t) + [\alpha(q^{-1})] y(t) + [\beta(q^{-1})] u(t) \} \triangleq \theta^T x(t-d) \quad (4.a)$$

$$x(t)^T \triangleq [y(t)^T, \dots, y(t-n_1)^T, u(t)^T, \dots, u(t-n_2)^T, \hat{y}^*(t)^T, \dots, \dots, \hat{y}^*(t-n_3)^T] \quad (4.b)$$

$$v(t) \triangleq [F(q^{-1})] w(t) = \sum_{i=0}^{d-1} F_1 w(t-i), \quad F_1 \in \mathbb{R}^{m \times m}, \quad F_0 = I_m \quad (4.c)$$

În relațiile (4), indicele superior T denotă transpunerea, I_m reprezintă matricea unitate de ordin m , $\hat{y}^*(t+d)$ este predicția liniară optimală cu d pași înainte a lui $y(t)$, determinată utilizînd informațiile pînă la momentul t . Matricele polinomiale $[G(q^{-1})]$, $[\alpha(q^{-1})]$, $[\beta(q^{-1})]$ și $[F(q^{-1})]$ depind de coeficienții polinoamelor $A(\cdot)$, $B_{1j}(\cdot)$ și $C_{1j}(\cdot)$; n_1 , n_2 și n_3 sînt gradele maxime (finite) ale polinoamelor din $[G]$, $[\alpha]$ și respectiv $[\beta]$; $\theta \in \mathbb{R}^{p \times m}$ este matricea parametrilor definită în (4.a), iar $p = m(n_1 + n_3 + 2) + r(n_2 + 1)$.

În ipoteza parametrilor cunoscuți (deci θ cunoscut), obiectivul conducerii adaptive este atins dacă se alege $u(t)$ ca soluție a sistemului liniar implicit definit

$$\theta^T x(t) = y^*(t+d) \quad (5)$$

În cazul parametrilor necunoscuți, matricea θ se înlocuiește cu o estimatie a sa $\hat{\theta}(t)$, actualizată de un algoritm recursiv, iar $\hat{y}^*(t)$ se înlocuiește cu o mărime calculabilă, funcție de informația disponibilă $\Phi(t)$.

Din formularea generală de mai sus se pot obține ușor prin particularizare formulările valabile în cazul sistemelor monovariabile sau deterministe

$$(w(t) = 0, v(t) = 0).$$

În continuare, vom considera cîțiva algoritmi tipici de conducere adaptivă CMMPR.

Algoritmul CMMPR1, pentru sisteme monovariabile deterministe [15, 16], [18]:

$$\theta(t) = \theta(t-1) + \frac{a(t) P(t-2) \Phi(t-d)}{\lambda(t) + a(t) \Phi(t-d)^T P(t-2) \Phi(t-d)} [y(t) - \theta(t-1)^T \Phi(t-d)] \quad (6.a)$$

$$P(t-1) = \left[P(t-2) - \frac{a(t) P(t-2) \Phi(t-d) \Phi(t-d)^T P(t-2)}{\lambda(t) + a(t) \Phi(t-d)^T P(t-2) \Phi(t-d)} \right] / \lambda(t) \quad (6.b)$$

$$\theta(t)^T \Phi(t) = y^*(t+d) \quad (6.c)$$

$$\Phi(t)^T = [y(t), \dots, y(t-n+1), u(t), \dots, u(t-m-d+1)] \quad (6.d)$$

unde $P(\cdot) \in \mathbb{R}^{p \times p}$ este matricea de covarianță, $\lambda(t) \in (0, 1]$ este un factor de ponderare permițând eliminarea („uitarea“) informațiilor vechi, iar $a(t)$ este un număr pozitiv ales astfel încât să se evite împărțirea la zero în (6.c) la determinarea lui $u(t)$. Alegeri posibile pentru $\lambda(t)$, asigurând mărginirea urmei matricei $P(t)^{-1}$ sînt indicate în [6], [16]. Un algoritm similar, pentru cazul $d=t$, se găsește în [3].

Algoritmul multirecursiv CMMPR2; CMMPR modificate pentru sisteme monovariabile stocastice [15], [17, 18]:

$$\theta(t) = \theta(t-d) + \frac{P(t-2d) \Phi(t-d)}{1 + \Phi(t-d)^T P(t-2d) \Phi(t-d)} [y(t) - \theta(t-d)^T \Phi(t-d)] \quad (7.a)$$

$$P'(t-d) = P(t-2d) - \frac{P(t-2d) \Phi(t-d) \Phi(t-d)^T P(t-2d)}{1 + \Phi(t-d)^T P(t-2d) \Phi(t-d)},$$

$$P(-d) = \dots = P(-1) = \varepsilon I_p, \quad \varepsilon \text{ mic} \quad (7.b)$$

$$r(t-d) = r(t-2d) [1 + \Phi(t-d)^T P(t-2d) \Phi(t-d)],$$

$$r(-d) = \dots = r(-1) = \rho > 0 \quad (7.c)$$

$$P(t-d) = P'(t-d), \text{ dac\u0103 } r(t-d) \text{ tr}[P'(t-d)] < K, \quad K > 0 \quad (7.d)$$

$$P(t-d) = \frac{K}{r(t-d) \text{ tr}[P'(t-d)]} P'(t-d), \text{ astfel} \quad (7.e)$$

$$\theta(t)^T \Phi(t) = y^*(t+d) \quad (7.f)$$

$$\Phi(t)^T = [y(t), \dots, y(t-n_1), u(t), \dots, u(t-n_2), \bar{y}(t), \dots, \bar{y}(t-n_3)] \quad (7.g)$$

$$y(t) = \theta(t)^T \Phi(t-d), \quad \bar{y}(\tau) = 0, \quad \tau \leq d-1. \quad (7.h)$$

Mai sus, $\text{tr}[A]$ denot\u0103 urma matricei A . Algoritmul generalizeaz\u0103 pe acela propus \u00een [11] pentru $d=1$.

Algoritmul CMMPR3; CMMPR ponderate pentru sisteme multivariabile stocastice [13-15]:

$$\theta(t) = \theta(t-1) + \frac{a(t)P(t-2)\Phi(t-d)}{1+a(t)\Phi(t-d)^T P(t-2)\Phi(t-d)} [y(t) - \theta(t-1)^T \Phi(t-d)]^T \quad (8.a)$$

$$P(t-1) = P(t-2) - \frac{a(t)P(t-2)\Phi(t-d)\Phi(t-d)^T P(t-2)}{1+a(t)\Phi(t-d)^T P(t-2)\Phi(t-d)}, \quad P(-1) = \varepsilon I_p \quad (8.b)$$

$$\theta(t)^T \Phi(t) = y^*(t+d) \quad (8.c)$$

$$\Phi(t)^T = [y(t)^T, \dots, y(t-n_1)^T, u(t)^T, \dots, u(t-n_2)^T, \bar{y}(t)^T, \dots, \dots, \bar{y}(t-n_3)^T] \quad (8.d)$$

$$\bar{y}(t) = \theta(t)^T \Phi(t-d) \quad (8.e)$$

iar $a(t)$ este un coeficient de ponderare ales în funcție de măsura de stabilitate $M(t)$, definită de

$$M(t) = \Phi(t-d)^T P(t-2)\Phi(t-d), \quad (9)$$

astfel încît să se asigure convergența algoritmului.

Menționăm că pentru algoritmi prezentați succint mai sus există demonstrații matematice de stabilitate, respectiv convergență cu probabilitate [1]. Algoritmi similari sînt folosiți și la regulatoarele autoacordabile [29, 30].

3. Actualizarea estimăției parametrilor

Etapa de calcul esențială a algoritmilor de conducere adaptivă o constituie actualizarea estimăției parametrilor. Examinînd algoritmi prezentați în secțiunea precedentă, rezultă că ecuațiile recursive de bază sînt următoarele :

$$\theta(t) = \theta(t-1) + \frac{a(t)P(t-2)\Phi(t-d)}{\lambda(t) + a(t)\Phi(t-d)^T P(t-2)\Phi(t-d)} [y(t) - \theta(t-1)^T \Phi(t-d)]^T \quad (10)$$

$$P(t-1) = \left[P(t-2) - \frac{a(t)P(t-2)\Phi(t-d)\Phi(t-d)^T P(t-2)}{\lambda(t) + a(t)\Phi(t-d)^T P(t-2)\Phi(t-d)} \right] / \lambda(t) \quad (11)$$

unde $\theta(\cdot) \in \mathbb{R}^{p \times m}$, $P(\cdot) \in \mathbb{R}^{p \times p}$, $\Phi(\cdot) \in \mathbb{R}^p$, $y(\cdot) \in \mathbb{R}^m$, $a(\cdot)$, $\lambda(t) \in \mathbb{R}$ au fost definite anterior. Ecuațiile de mai sus sînt inițializate cu $\theta_0 = \theta(0)$ și $P_{-1} = P(-1) > 0$. De notat că și în cazul algoritmului CMMPR2, putem considera că intervin relațiile (10) și (11).

Menționăm că ecuații similare cu (10) și (11) apar în problemele de estimare secvențială discretă [21, 22]. Din aplicațiile teoriei estimării parametrice, este bine cunoscut faptul că implementarea directă pe calculatoare a recurenței (11), utilizînd de exemplu produse exterioare, este numeric instabilă. Pot

apare covarianțe negative [23] și chiar matrice de „covarianță“ negativ definite. Acest lucru se poate obține și pentru exemple de mici dimensiuni, cum este cel dat în continuare [20], [15]:

Exemplul 1: Fie $a(t) \equiv 1$, $\lambda(t) \equiv 1$, $d=1$ și $\Phi(t-1)^T = [1 \ \varepsilon]$, $\Phi(t)^T = [1, 1]$, $P(t-2) = \sigma^2 I_2$, $\sigma = 1/\varepsilon$, unde ε este ales astfel încât $0 < \varepsilon < 1$ și $f_1(1+\varepsilon^2) = 1$. Notăția $f_1(x)$ denotă rezultatul obținut evaluând expresia x pe un calculator cu aritmetică în virgulă mobilă. Pentru un calculator cu precizia relativă de 6 cifre zecimale exacte, putem presupune $\varepsilon = 10^{-4}$. Este ușor de verificat (vezi [15]) că dacă $P(\cdot)$ se actualizează direct utilizând relațiile (corespunzătoare filtrului Kalman clasic)

$$K(t-1) = P(t-2) \Phi(t-d), \quad b(t) = a(t) / [\lambda(t) + a(t) \Phi(t-d)^T K(t-1)] \quad (12. a)$$

$$P(t-1) = [P(t-2) - b(t) K(t-1) K(t-1)^T] / \lambda(t) \quad (12. b)$$

se obține

$$f_1[P(t)] = \frac{1}{1-2\varepsilon} \begin{vmatrix} -1 & 1 \\ 1 & 0 \end{vmatrix}.$$

Așadar, la a doua iterație matricea $P(\cdot)$ calculată devine negativ definită.

Problemele numerice sînt probabil mai importante pentru implementarea pe calculator a algoritmilor de conducere adaptivă. Acești algoritmi trebuie să funcționeze on-line pe perioade de timp practic infinite. Mai mult, pentru conducerea adaptivă în aplicațiile industriale se utilizează frecvent microcalculatoare cu lungime de cuvînt mică, deci cu precizie de reprezentare a numerelor reale redusă.

Soluții numerice bune se obțin prin factorizarea matricei P , ori a inversei acesteia și prin actualizarea factorilor respectivi. În acest scop, au fost propuse tehnici de factorizare de tip rădăcină patrată (Cholesky) [19], sau de tip $U-D$ [20], [23].

În continuare, vom deduce cîteva relații utilizate în secțiunile viitoare. Pentru a simplifica notația, argumentele vor fi omise de aici încolo, iar cantitățile actualizate vor fi marcate prin simbolul „ $\hat{\cdot}$ ”. Relațiile (10) și (11) se rescriu astfel

$$\hat{\theta} = \theta + \frac{aP\Phi}{\lambda + a\Phi^T P \Phi} (y - \theta^T \Phi)^T \quad (13)$$

$$\hat{P} = \left(P - \frac{aP\Phi\Phi^T P}{\lambda + a\Phi^T P \Phi} \right) / \lambda. \quad (14)$$

Utilizînd (14), din (13) rezultă

$$\hat{\theta} = \left(I_p - \frac{aP\Phi\Phi^T}{\lambda + a\Phi^T P \Phi} \right) \theta + \frac{aP\Phi}{\lambda + a\Phi^T P \Phi} y^T = \lambda \hat{P} P^{-1} \theta + a \hat{P} \Phi y^T. \quad (15)$$

Notînd $Q = P^{-1}$, matricea de informație, (15) și (14) implică, respectiv

$$\hat{Q} \hat{\theta} = \lambda Q \theta + a \Phi y^T \quad (16)$$

$$\hat{Q} = \lambda Q + a \Phi \Phi^T. \quad (17)$$

Relațiile (16) și (17) sînt relațiile de bază care vor fi utilizate în secțiunea următoare. Relații similare cu (17) apar de asemenea în metodele cvasi-Newton din problemele de optimizare neliniară [24—26].

4. Tehnici de factorizare a matricei de informație

Vom prezenta mai întii factorizarea de tip rădăcină pătrată (Cholesky) a matricei de informație și apoi factorizarea de tip „U—D”.

Factorizarea tip rădăcină patrată

Considerăm factorizarea $\mathbf{Q} = \mathbf{R}^T \mathbf{R}$, unde \mathbf{R} este o matrice superior triunghiulară, și notăm

$$\mathbf{z} = \mathbf{R}\theta. \quad (18)$$

Fie \mathbf{G} o matrice ortogonală de ordin $p+1$ (adică $\mathbf{G}\mathbf{G}^T = \mathbf{G}^T\mathbf{G} = \mathbf{I}_{p+1}$), astfel ca

$$\mathbf{G} \begin{bmatrix} \lambda^{1/2} \mathbf{R} \\ \mathbf{a}'/s \Phi^T \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{R}} \\ 0 \end{bmatrix} \quad (19)$$

unde $\hat{\mathbf{R}}$ este o matrice superior triunghiulară. Observăm că $\hat{\mathbf{R}}$ este un factor al lui $\hat{\mathbf{Q}}$, deoarece cu (17)

$$\hat{\mathbf{R}}^T \hat{\mathbf{R}} = [\lambda^{1/2} \mathbf{R}^T \quad \mathbf{a}'/s \Phi] \begin{bmatrix} \lambda^{1/2} \mathbf{R} \\ \mathbf{a}'/s \Phi^T \end{bmatrix} = \lambda \mathbf{R}^T \mathbf{R} + \mathbf{a} \Phi \Phi^T = \hat{\mathbf{Q}}.$$

Ecuțiile (16) și (17) se pot rescrie astfel

$$\hat{\mathbf{R}}^T \hat{\mathbf{z}} = \lambda \mathbf{R}^T \mathbf{z} + \mathbf{a} \Phi \mathbf{y}^T \quad (20)$$

$$\hat{\mathbf{R}}^T \hat{\mathbf{R}} = \lambda \mathbf{R}^T \mathbf{R} + \mathbf{a} \Phi \Phi^T. \quad (21)$$

Notînd

$$\begin{bmatrix} \hat{\mathbf{z}} \\ \hat{\rho}^T \end{bmatrix} = \mathbf{G} \begin{bmatrix} \lambda^{1/2} \mathbf{z} \\ \mathbf{a}'/s \mathbf{y}^T \end{bmatrix} \quad (22)$$

din (19)—(22) avem,

$$\hat{\mathbf{R}}^T \hat{\mathbf{z}} = [\lambda^{1/2} \mathbf{R}^T \quad \mathbf{a}'/s \Phi] \begin{bmatrix} \lambda^{1/2} \mathbf{z} \\ \mathbf{a}'/s \mathbf{y}^T \end{bmatrix} = \begin{bmatrix} \lambda^{1/2} \mathbf{R} \\ \mathbf{a}'/s \Phi^T \end{bmatrix}^T \mathbf{G}^T \mathbf{G} \begin{bmatrix} \lambda^{1/2} \mathbf{z} \\ \mathbf{a}'/s \mathbf{y}^T \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{R}} \\ 0 \end{bmatrix}^T \begin{bmatrix} \hat{\mathbf{z}} \\ \hat{\rho}^T \end{bmatrix} = \hat{\mathbf{R}}^T \hat{\mathbf{z}}.$$

deci $\hat{\mathbf{z}} = \hat{\mathbf{z}}$, deoarece $\hat{\mathbf{R}}$ este nesingulară.

Astfel, procedura pentru actualizarea lui θ cuprinde următoarele etape :

- 1) Determină G și \hat{R} astfel încât (19) este satisfăcută.
- 2) Calculează \hat{z} aplicind G asupra matricei $[\lambda^{1/2}z^T \ a^{1/2}y]^T$.
- 3) Rezolvă sistemele $\hat{R}\hat{\theta}=\hat{z}$.

Matricea ortogonală G poate fi determinată ca produs a p rotații plane [27], $G=G_p G_{p-1} \dots G_1$, unde G_i este o rotație în planul $(i, p+1)$ dată de

$$G_i = \begin{bmatrix} \downarrow i & & & \downarrow p+1 \\ 1 & 0 & 0 & 0 \\ 0 & c_i & 0 & s_i \\ 0 & 0 & 1 & 0 \\ 0 & -s_i & 0 & c_i \\ \leftarrow i & & & \leftarrow p+1 \end{bmatrix}, \quad c_i^2 + s_i^2 = 1. \quad (23)$$

Pentru $\lambda=a=1$, procedura de mai sus se reduce la cea descrisă în documentația pachetului LINPACK [27]. Subrutina SCHUD din acest pachet implementează primii doi pași, iar subrutina STRSL poate fi folosită pentru a rezolva sistemele liniare de la pasul 3. SCHUD poate fi direct utilizată și în cazul $\lambda \neq 1$, $a \neq 1$ dacă, înainte de a o apela, matricele R și z sînt multiplicare prin $\lambda^{1/2}$, și vectorii Φ și y sînt multiplicați prin $a^{1/2}$. Totuși, aceste operații de preprelucrare implică $1/2 p^2 + (m+1,5)p + m$ multiplicări suplimentare. Efortul de calcul poate fi redus, prin modificarea și rafinarea procedurii precedente, după cu m se indică în algoritmul următor.

Algoritmul 1: Actualizează θ utilizind factorizarea tip rădăcină patrată a lui P^{-1} .

Intrare : R, z ($R^T R = P^{-1}$, $z = R\theta$).

Ieșire : $\hat{R}, \hat{z}, \hat{\theta}$ (suprascrind R, z , și θ , respectiv).

1) $\Phi := a^{1/2} \Phi$; $y := a^{1/2} y$.

2) Pentru $i=1, 2, \dots, p$

1) $r_{11} := \lambda^{1/2} r_{11}$.

2) Determină $g_i = \begin{bmatrix} c_i & s_i \\ -s_i & c_i \end{bmatrix}$, astfel încit $\begin{bmatrix} r_{11} \\ 0 \end{bmatrix} := g_i \begin{bmatrix} r_{11} \\ \Phi_{11} \end{bmatrix}$.

3) Pentru $i < p$

$$\begin{bmatrix} r_{1, i+1} \dots r_{1p} \\ \Phi_{i+1} \dots \Phi_p \end{bmatrix} := \begin{bmatrix} \lambda^{1/2} c_i & s_i \\ -\lambda^{1/2} s_i & c_i \end{bmatrix} \begin{bmatrix} r_{1, i+1} \dots r_{1p} \\ \Phi_{i+1} \dots \Phi_p \end{bmatrix}$$

4)

$$\begin{bmatrix} z_{11} \dots z_{1m} \\ y_1 \dots y_m \end{bmatrix} := \begin{bmatrix} \lambda^{1/2} c_i & s_i \\ -\lambda^{1/2} s_i & c_i \end{bmatrix} \begin{bmatrix} z_{11} \dots z_{1m} \\ y_1 \dots y_m \end{bmatrix}$$

3) $\theta = R^{-1} z$.

Observații

1) La sfârșitul algoritmului 1, R , z , și θ conțin cantitățile actualizate \widehat{R} , \widehat{z} , și respectiv $\widehat{\theta}$. Rotațiile g_i sînt determinate utilizînd matricele $[\lambda^{1/2}R^T \ a^{1/2}\Phi]^T$. La pașii 2.3 și 2.4 se aplică

$$g_i^\lambda = \begin{bmatrix} \lambda^{1/2}c_i & s_i \\ -\lambda^{1/2}s_i & c_i \end{bmatrix};$$

rezultatul este identic cu cel care ar fi obținut utilizînd direct SCHUD pentru $\lambda^{1/2}R$, $a^{1/2}\Phi^T$, $\lambda^{1/2}z$, și $a^{1/2}y^T$. Soluția propusă aici necesită doar $4p+m$ multiplicări suplimentare pentru $\lambda \neq 1$ și $a \neq 1$. Pentru a calcula rotațiile plane g_i^λ se poate utiliza subrutina SROTG din pachetul BLAS [28]. Subrutina SROT, de asemenea din BLAS, trebuie modificată pentru a aplica g_i^λ .

2) Costul total al algoritmului este de $(\frac{1}{2}m+2)p^2 + (4,5m+7)p + m$ multiplicări și $p+2$ rădăcini patrute. În cazul monovariabil numărul este $2,5p^2 + 11,5p$.

3) Algoritmul aplică rotațiile pe linii. Pentru a opera pe coloane, se pot adopta factorizările $Q=UU^T$, sau $Q=LL^T$, unde U și L sînt matrice superior și, respectiv, inferior triunghiulare. Alternativ, se poate recurge la tehnica folosită în LINPACK, dar acest lucru ar necesita $4p$ locații suplimentare de memorie pentru a memora c_i , s_i , $\lambda^{1/2}c_i$, și $\lambda^{1/2}s_i$, pentru $i=1, \dots, p$.

4) Măsura de stabilitate $M(t)$, definită de relația (9) și utilizată în algoritmul CMMPR3 poate fi calculată eficient astfel $M=\Phi^T Q^{-1} \Phi = \|R^{-1} \Phi\|$. Deci, pentru calculul ei sînt necesare numai $\frac{1}{2}p^2 + 1,5p$ multiplicări.

În Anexa 2 este listată subrutina SQRINF care actualizează factorizarea UU^T a matricei de informație, pentru sisteme monovariabile, folosind tehnica prezentată mai sus. SQRINF apelează subrutina SROTG din BLAS pentru determinarea rotațiilor.

Factorizarea LDL^T

Luînd în considerare observația 3, să considerăm acum factorizarea $Q=LDL^T$, unde L este o matrice inferior triunghiulară, iar D este o matrice diagonală. Fie G o matrice ortogonală, astfel încît

$$G \begin{bmatrix} \lambda^{1/2}D^{1/2}L^T \\ a^{1/2}\Phi^T \end{bmatrix} = \begin{bmatrix} \widehat{D}^{1/2}\widehat{L}^T \\ 0 \end{bmatrix} \quad (24)$$

unde \widehat{D} este diagonală și \widehat{L} este inferior triunghiulară. Din (24) și (17) se obține ușor că $\widehat{Q}=\widehat{L}\widehat{D}\widehat{L}^T$. Definind $z=L^T\theta$ și

$$\begin{bmatrix} \widehat{D}^{1/2}\widehat{z} \\ \widehat{\rho}^T \end{bmatrix} = G \begin{bmatrix} \lambda^{1/2}D^{1/2}z \\ a^{1/2}y^T \end{bmatrix} \quad (25)$$

din (16), (24) și (25) avem

$$\widehat{L}\widehat{D}\widehat{z} = \lambda LDz + a\Phi y^T = [\lambda^{1/2}LD^{1/2} \quad a^{1/2}\Phi] G^T G \begin{bmatrix} \lambda^{1/2}D^{1/2}z \\ a^{1/2}y^T \end{bmatrix} = \widehat{L}\widehat{D}\widehat{z} \quad (26)$$

deci $\hat{z} = \bar{z}$. Astfel, procedura de actualizare a lui θ cuprinde următorii pași:

- 1) Determină G , \hat{D} și \hat{L} , astfel încât să fie satisfăcută relația (24).
- 2) Calculează \hat{z} aplicând G asupra matricii $[\lambda^{1/2}z^T D^{1/2} \quad a^{1/2}y]^T$.
- 3) Rezolvă $\hat{L}^T \hat{\theta} = \hat{z}$.

Matricea G poate fi obținută ca produs a p rotații plane, $G = G_p G_{p-1} \dots G_1$. Vom utiliza rotații plane modificate, evitând astfel rădăcinile ptrate. Fie c_j și s_j cosinusul și, respectiv, sinusul care determină rotația G_j . Atunci putem scrie (vezi Anexa 1 pentru detalii)

$$\begin{bmatrix} c_j & s_j \\ -s_j & c_j \end{bmatrix} \begin{bmatrix} \lambda^{1/2} d_j^{1/2} & 0 \\ 0 & (a^j)^{1/2} \end{bmatrix} = \begin{bmatrix} \hat{d}_j^{1/2} & 0 \\ 0 & (a^{j+1})^{1/2} \end{bmatrix} H_j \quad (27)$$

unde $a^1 = a$, matricea H_j , de ordin 2, are în general două elemente egale cu 1, iar indicele superior j denotă valorile variabilelor la a j -a etapă de calcul. Se poate utiliza subrutina SROTMG din pachetul BLAS [28], pentru a calcula H_j , \hat{d}_j și a^{j+1} . La pasul 2 al procedurii avem

$$\begin{bmatrix} c_j & s_j \\ -s_j & c_j \end{bmatrix} \begin{bmatrix} \lambda^{1/2} d_j^{1/2} & 0 \\ 0 & (a^j)^{1/2} \end{bmatrix} \begin{bmatrix} z_{j1} \dots z_{jm} \\ y_1^j \dots y_m^j \end{bmatrix} = \begin{bmatrix} \hat{d}_j^{1/2} & 0 \\ 0 & (a^{j+1})^{1/2} \end{bmatrix} H_j \begin{bmatrix} z_{j1} \dots z_{jm} \\ y_1^j \dots y_m^j \end{bmatrix}$$

asa încât în final obținem

$$\hat{z} = H_p' \dots H_1' \begin{bmatrix} z \\ y^T \end{bmatrix},$$

unde

$$H_j' = \begin{bmatrix} I & 0 & 0 & 0 \\ 0 & h_{11} & 0 & h_{12} \\ 0 & 0 & I & 0 \\ 0 & h_{21} & 0 & h_{22} \end{bmatrix} \begin{matrix} \leftarrow j \\ \\ \leftarrow p+1 \end{matrix}$$

și $H_j = (h_{ik})$. Astfel, rezultă următorul algoritm:

Algoritmul 2: Actualizează θ utilizând factorizarea LDL^T a lui P^{-1} .

Intrare: D, L, z ($LDL^T = P^{-1}$, $z = L^T \theta$).

Ieșire: $\hat{D}, \hat{L}, \hat{z}, \hat{\theta}$ (suprascrisind D, L, z , și respectiv θ).

1) Pentru $j=1, 2, \dots, p$

1) $d_j := \lambda d_j$.

2) Utilizează SROTMG cu intrările d_j, a, l_{jj} , și Φ_j pentru a determina

H_j , și pentru a actualiza d_j, a , și l_{jj} .

3) Pentru $j < p$

$$\begin{bmatrix} 1_{j+1, j} \dots 1_{pj} \\ \Phi_{j+1} \dots \Phi_p \end{bmatrix} := H_j \begin{bmatrix} 1_{j+1, j} \dots 1_{pj} \\ \Phi_{j+1} \dots \Phi_p \end{bmatrix}.$$

4)

$$\begin{bmatrix} z_{j1} \dots z_{jm} \\ y_1 \dots y_m \end{bmatrix} := H_j \begin{bmatrix} z_{j1} \dots z_{jm} \\ y_1 \dots y_m \end{bmatrix}.$$

2) $\theta = L^{-T}z$.

Observații

5) La sfârșitul algoritmului 2, D , L , z , și θ conțin cantitățile actualizate \widehat{D} , \widehat{L} , \widehat{z} , și respectiv, $\widehat{\theta}$. Pentru a aplica H_j la pașii 1.3 și 1.4 se poate utiliza subrutina SROTM din pachetul BLAS.

6) Algoritmul necesită în general $(\frac{1}{2}m+1)p^2 + (2,5m+10)p$ multiplicări. Numărul de operații poate fi de două ori mai mare în cel mai nefavorabil caz, adică atunci când matricele H_j nu au nici un element egal cu 1, pentru orice j . Acest caz este extrem de rar (vezi Anexa 1).

7) Algoritmul aplică rotațiile plane modificate, pe coloane asupra lui L și pe linii asupra lui z .

8) Măsura de stabilitate $M = D^T P D$ poate fi calculată simplu ca $\|L^{-T}\|_{D^{-1}}^2$.

5. Tehnici de factorizare a matricei de covarianță

Vom discuta aici mai în detaliu tehnica de factorizare $U-D$, deoarece tehnica rădăcinii pătrate este bine cunoscută [19], [31, 32]. În plus, algoritmul de tip rădăcină pătrată se poate obține prin particularizarea algoritmului de actualizare a factorizării $U-D$.

Tehnica de factorizare $U-D$

Considerăm factorizarea $P = UDU^T$, unde U este o matrice superior triunghiulară, iar D este matrice diagonală, și notăm

$$f = U^T \Phi, \quad v = Df, \quad K = Uv, \quad b = a/(\lambda + a\Phi^T P \Phi). \quad (28)$$

Ecuția (14) poate fi rescrisă [20], [23]

$$\widehat{P} = \widehat{U} \widehat{D} \widehat{U}^T = U [D - b (DU^T \Phi) (DU^T \Phi)^T] U^T / \lambda = U (D - bvv^T) U^T / \lambda. \quad (29)$$

Fie U și D factorii $U-D$ ai matricei $D - bvv^T$, adică $\overline{U} \overline{D} \overline{U}^T = D - bvv^T$. Atunci, din (29) putem identifica

$$\widehat{U} = U \overline{U}, \quad \widehat{D} = \overline{D} / \lambda$$

deoarece U și \overline{U} sînt matrice superior triunghiulare. Deci, actualizarea factorizării $U-D$ a lui P este reductibilă la un caz special de actualizare a factorizării

unei matrice pozitiv definite după o modificare „de rang 1” (adică, după adăugarea, respectiv scăderea, unei diade simetrice), discutată în Anexa 1. Nu vom utiliza aici rotații plane modificate, deoarece prezența semnului „-” poate produce dificultăți numerice, după cum se va arăta la exemplul 2 de mai jos. Factorii \bar{U} și \bar{D} pot fi determinați particularizând algoritmul din Anexa II din [23], ale cărui relații de bază sînt regăsite pe o cale nouă în Anexa 1. Astfel, din (A9) obținem, pentru $j=p, p-1, \dots, 1$

$$\bar{d}_j = d_j - b_j (v_j)^2, \quad b^{j-1} = b^j d_j / \bar{d}_j, \quad \bar{u}_{ij} = -b_j v_j v_i / \bar{d}_j \quad (i < j), \quad \bar{u}_{jj} = 1 \quad (30)$$

și $v_i^{j-1} = v_i^j = v_i, \quad i \leq j, \quad b^p = b$, deoarece $D = b v v^T = I_p D I_p^T - b v v^T$. Din nefericire, algoritmul bazat pe aceste relații este însoțit de dificultăți numerice. Din primele două egalități (30) obținem

$$1/b^{j-1} = \bar{d}_j / (b^j d_j) = 1/b^j - v_j^2/d_j$$

astfel încît, notînd $a^j = 1/b^j$, rezultă următoarele relații recursive stabile

$$a^j = a^{j-1} + v_j^2/d_j, \quad \bar{d}_j = d_j a^{j-1} / a^j, \quad \bar{u}_{ij} = -v_j v_i / (a^{j-1} d_j) \quad (31)$$

și putem alege $a^0 = 1/b - \sum_{j=1}^p v_j^2/d_j$, astfel încît $a^p = 1/b$. A fost astfel regăsită soluția propusă în [26].

Ținînd acum seama de structura matricei \bar{U} , produsul $U\bar{U}$ poate fi evaluat în numai o (p^2) multiplicări. Într-adevăr, să observăm că \bar{U} rezultă în forma

$$\bar{U} = [\bar{U}_1 \bar{U}_2 \dots \bar{U}_p], \quad \text{unde } \bar{U}_j = [s^j v_1 s^j v_2 \dots s^j v_{j-1} \quad 1 \quad 0 \dots 0]^T \text{ și } s^j = -v_j / (a^{j-1} d_j).$$

Atunci,

$$\hat{U}_j = U \bar{U}_j = s^j U [v_1 v_2 \dots v_{j-1} \quad 1/s^j \quad 0 \dots 0]^T = U_j + s^j K_{j-1} \quad (32)$$

unde am notat $K_{j-1} = \sum_{i=1}^{j-1} v_i U_i$

Astfel, se obține următorul algoritm care constituie o generalizare a algoritmului propus în [23].

Algoritmul 3: Actualizează θ utilizînd factorizarea UDU^T a lui P .

Intrare : D, U, θ ($UDU^T = P$).

Ieșire : $\hat{D}, \hat{U}, \hat{\theta}, K$ ($\hat{U}\hat{D}\hat{U}^T = \hat{P}$).

1) Calculează $f = U^T \Phi$; $v = Df$; $a^0 = \lambda/a$.

2) Pentru $j=1, 2, \dots, p$

1) Pune $\hat{u}_{jj} = 1$; $k_j = v_j$; $s^j = -f_j/a^{j-1}$.

2) Actualizează $a^j = a^{j-1} + v_j f_j$;

$$\hat{d}_j = d_j a^{j-1} / (\lambda a^j).$$

3) Dacă $j > 1$, atunci pentru $i=1, \dots, j-1$

$$\hat{u}_{ij} = u_{ij} + s^j k_i;$$

$$k_i := k_i + v_j u_{ij}.$$

$$3) \hat{\theta} = \theta + K (y - \theta^T \Phi)^T / a^p.$$

Observații

9) La sfârșitul algoritmului 3 obținem $a^p = 1/b = (\lambda + a\Phi^T P \Phi) / a$ și $K = UDU\Phi^T = P\Phi$ este vectorul de amplificare Kalman normalizat. În fond, a^0 a fost ales astfel încît $a^p = 1/b$. Algoritmul poate fi ușor modificat pentru ca \hat{D} , \hat{U} , și $\hat{\theta}$ să suprascrîie D , U , și respectiv θ .

10) Algoritmul necesită numai $1,5p^2 + (2m + 4,5)p$ multiplicări.

11) Algoritmul 3 are bune proprietăți numerice; detalii despre etapa de actualizare a factorizării pot fi găsite în [23].

În Anexa 3 este listată subrutina UDCOV, pentru actualizarea factorizării $U-D$ a matricei de covarianță. Rutina este programată pentru a necesita un minim de locații de memorie pentru date. UDCOV apelează subprogramul RFVPS pentru calculul produselor scalare; RFVPS a fost obținut prin adaptarea funcției SDOT din BLAS și utilizează cicluri desfășurate pentru mărirea eficienței calculului.

Să considerăm acum relația (29) rescrisă astfel

$$\hat{P} = UDU^T / a - bKK^T. \quad (33)$$

Problema de actualizare se reduce la actualizarea factorizării $U-D$ după o modificare de rang 1. Se poate aplica direct procedura A1 dată în Anexa 1, utilizînd subrutinele SROTGMG și SROTM din pachetul BLAS. Totuși, această soluție nu este atrăgătoare din punct de vedere numeric, după cum se poate constata analizînd exemplul de mai jos, care constituie de fapt adaptarea exemplului 1 din secțiunea 3.

Exemplul 2: Fie $U = \text{diag}(u_1, u_2)$, $D = \text{diag}(d_1, d_2)$, $\Phi = [1 \ e]^T$, unde $d_1 u_1^2 = \sigma^2$, $i=1, 2$, $\sigma = 1/\epsilon$, și $f(1 + \epsilon^2) = 1$. Călcule elementare arată că, dacă se utilizează procedura A1, la a doua apelare a subrutinei SROTGMG, datorită erorilor de rotunjire, matricea calculată H_1 are elementele $h_1' = -d_1 u_1$, $h_2' = u_1 \epsilon^2$, astfel încît $\tau' = 1 + h_1' h_2' = 1 - d_1 u_1^2 \epsilon^2 = 0$. Calculele nu mai pot fi deci continuate pentru nici o pereche de valori u_i , d_i satisfăcînd $d_1 u_1^2 = \sigma^2$ (vezi relația (A8) din Anexa 1). Deci, această tehnică nu poate fi în general recomandată. Toți ceilalți algoritmi discutați în articol au rezolvat cu succes această problemă-test.

Factorizarea tip rădăcină patrată

Punînd $S = UD^{1/2}$ în demonstrațiile de mai sus se poate obține ușor următorul algoritm de tip rădăcină pătrată:

Algoritmul 4: Actualizează θ utilizînd factorizarea SS^T a lui P .

Intrare: S , θ ($SS^T = P$).

Ieșire: \hat{S} , $\hat{\theta}$, K ($\hat{S}\hat{S}^T = \hat{P}$).

1) Calculează $f = S^T \Phi$; $a^0 = \lambda/a$.

2) Pentru $j=1, 2, \dots, p$

1) Pune $k_j = f_j s_{jj}$; $s^j = -f_j/a^{j-1}$.

2) Actualizează $a^j = a^{j-1} + f_j^2$;

$d^j = (a^{j-1}/(\lambda a^j))^{1/2}$;

$\hat{s}_{jj} = d^j s_{jj}$.

3) Dacă $j > 1$, atunci pentru $i=1, \dots, j-1$

$\hat{s}_{ij} = d^j (s_{ij} + s^j k_i)$;

$k_i := k_i + f_j s_{ij}$.

3) $\hat{\theta} = \theta + K (y - \theta^T \Phi)^T / a^p$.

Observații

12) Observația 9 de mai sus se poate transcrie aici.

13) Algoritmii necesită $2p^2 + (2n+5)p$ multiplicații și p rădăcini pătrate.

6. Rezultate experimentale

O parte din rezultatele teoretice prezentate în lucrare au fost aplicate în cadrul pachetului de programe ADCON [33, 34], implementat pe minicalcutoarele românești și destinat conducerii adaptive a proceselor tehnologice. ADCON include și alți algoritmi adaptivi [30]; deocamdată este tratat numai cazul sistemelor monovariabile, dar numărul buclelor de reglare adaptivă este practic nelimitat.

Algoritmii de factorizare analizați în lucrare au fost programați și testați de autor, obținându-se rezultate bune în toate cazurile studiate. Nu s-a putut stabili experimental superioritatea din punct de vedere numeric a unui algoritm asupra altora. De remarcat, totuși, că algoritmul 2 este de preferat algoritmului 3, întrucât în implementare este evitată apariția depășirilor aritmetice.

Am studiat intensiv prin simulare pe un minicalculator CORAL 4011 comportarea algoritmilor prezentați în secțiunea 2. Pentru actualizarea factorizării a fost folosită subrutina UDCOV, listată în Anexa 3. Condițiile de simulare au fost alese astfel încât să fie apropiate celor întâlnite în mediu industrial. Zgomotul $\{w(t)\}$ a fost realizat ca semnal pseudoaleator cu distribuție normală, cu media \bar{w} și abaterea standard σ . Pentru calculul mărimii de referință cu d pași înainte ($y^*(t+d)$) am folosit un model etalon. Introducerea acestuia a avut ca efect și atenuarea variațiilor mărimilor de comandă și de ieșire la modificarea bruscă a intrării de referință $r(t)$. Semnalul $\{r(t)\}$ a fost ales de tip undă pătrată dreptunghiulară cu valoarea minimă 0 și valoarea maximă 1. În cazul algoritmului CMMPR1 eroarea de urmărire $y(t) - y^*(t)$ a fost

filtrată printr-un filtru specificat (vezi [4, 5]). Au fost efectuate numeroase teste pentru a studia influența condițiilor inițiale sau a unor parametri definitorii ai algoritmilor. În toate experimentele au fost calculate statistici elementare ale erorii de urmărire, atât în etapa tranzitorie de adaptare inițială, cât și în etapa finală. Analiza acestor statistici a confirmat proprietățile bune de urmărire ale algoritmilor prezentați.

În continuare ilustrăm o parte din rezultatele obținute pentru sistemul

$$(1 - 2,1q^{-1} + 1,53q^{-2} - 0,365q^{-3}) y(t) = q^{-2} (1 + 0,4q^{-1}) u(t) + \\ + (1 - 1,2q^{-1} + 0,48q^{-2} - 0,64q^{-3}) w(t).$$

S-a utilizat modelul etalon definit de :

$$(1 - 1,9q^{-1} + 1,23q^{-2} - 0,265q^{-3}) y^*(t) = q^{-2} (0,28 + 0,22q^{-1}) r(t).$$

În fig. 1, 2 și 3 sînt prezentate traiectoriile ieșirii sistemului și modelului etalon rezultate folosind algoritmii CMMPR1, CMMPR2 și respectiv, CMMPR3. În cazul algoritmului CMMPR1 s-a utilizat filtrul de eroare

$$1 - 1,2q^{-1} + 0,48q^{-2} - 0,64q^{-3}.$$

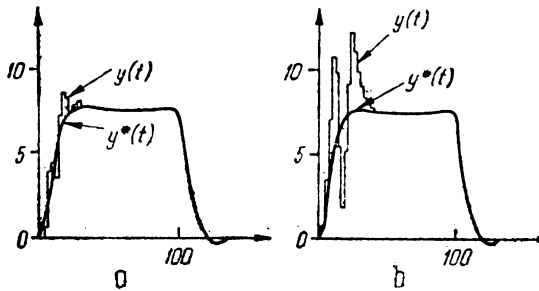


Fig. 1. Simulare utilizînd algoritmul CMMPR1.

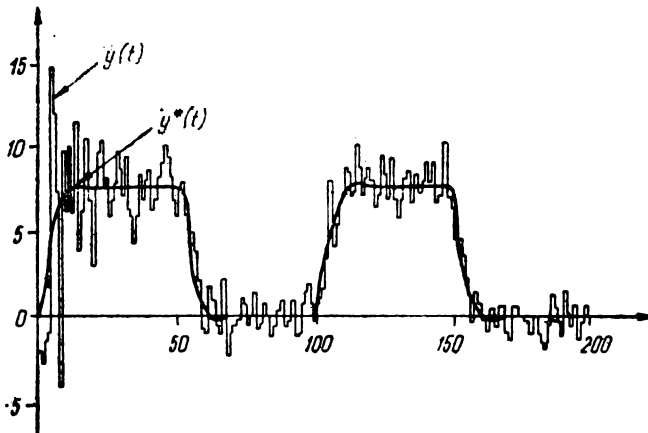


Fig. 2. Simulare utilizînd algoritmul CMMPR2.

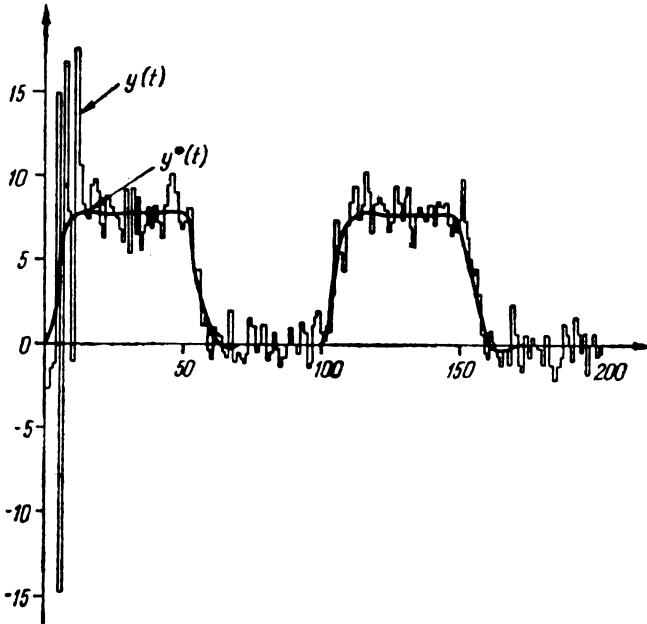


Fig. 3. Simulare utilizând algoritmul CMMPR3.

În cazul algoritmilor CMMPR2 și CMMPR3 rezultatele au fost obținute în ipoteza $\bar{w}=0$, $\sigma=0,5$. În toate cazurile s-au adoptat valorile $\lambda \equiv 1$ și $P(-1) = -10I$. În fig. 1, a valorile inițiale ale parametrilor regulatorului adaptiv au fost alese apropiate de valorile lor optimale.

7. Concluzii

În lucrare sînt analizate tehnici de factorizare eficiente și atractive din punct de vedere numeric, aplicabile algoritmilor recursivi de tipul celor mai mici pătrate înfrînți în domeniul conducerii adaptive a sistemelor multivariabile deterministe sau stocastice. Sînt discutate într-o concepție unitară variante de tip rădăcină pătrată și de tip U—D pentru actualizarea factorizării matricei de informație sau a matricei de covarianță. Instrumentul matematic de bază utilizat îl constituie rotațiile plane, standard sau modificate. Sînt prezentate detalii de implementare și sînt date recomandări de utilizare a unor rutine matematice performante, recent elaborate, pentru rezolvarea problemelor de calcul asociate. Codurile de program listate în Anexe pot fi folosite și în probleme de estimare recursivă sau în unele probleme de minimizare. Din pricina limitării spațiului, nu s-au furnizat decît două subrutine de actualizare a factorizărilor.

Din punctul de vedere al efortului de calcul, algoritmi utilizînd matricea de informație sînt mai costisitori, în cazul multivariabil, decît ceilalți algoritmi.

Ei sînt însă de preferat în problemele de estimare secvențială cu estimări rare și măsurări frecvente, în care actualizarea parametrilor θ se face mai rar decît actualizarea matricei de informație.

O parte dintre algoritmi discutați în lucrare au fost implementați de autor în cadrul pachetului de programe ADCON, destinat conducerii adaptive a proceselor tehnologice. Rezultatele numerice obținute în simulare sînt încurajatoare. Cercetările vor continua cu studiul unor probleme rău condiționate și cu investigarea unor aspecte de analiză numerică asociate. De asemenea, este necesară testarea comportării algoritmilor prezentați, și în primul rînd a celor originali, în aplicațiile industriale.

BIBLIOGRAFIE

1. A. Feuer, A. S. Morse, Adaptive control of a single-input single-output linear systems, IEEE Trans. Autom. Control, vol. AC-23, p. 557—569, 1978.
2. A. S. Morse, Global stability of parameter-adaptive control systems, IEEE Trans. Autom. Control, vol. AC-25, p. 433—439, 1980.
3. G. C. Goodwin, P. J. Ramadge, P. E. Caines, Discrete-time multivariable adaptive control, IEEE Trans. Autom. Control, vol. AC-25, p. 419—456, 1980.
4. I. D. Landau, L. Lozano, Unification and evaluation of discrete time explicit model reference adaptive control designs, Automatica, vol. 17, p. 593—611, 1981.
5. L. Lozano, I. D. Landau, Redesign of explicit and implicit discrete time model reference adaptive control systems, Int. J. Control, vol. 33, p. 247—268, 1981.
6. L. Lozano, Adaptive control with forgetting factor, In Prepr. of the 8-th IFAC World Congress, Kyoto, Japan, vol. VII, p. 83—88, 1981.
7. G. C. Goodwin, K. S. Sin, K. K. Saluja, Stochastic adaptive control and prediction — The general delay-colored noise case, IEEE Trans. Autom. Control, vol. AC-25, p. 946—950, 1980.
8. G. C. Goodwin, P. J. Ramadge, P. E. Caines, A globally convergent adaptive predictor, Automatica, vol. 17, p. 135—140, 1981.
9. G. C. Goodwin, P. J. Ramadge, P. E. Caines, Discrete time stochastic adaptive control, SIAM J. Control Optimiz., vol. 19, p. 829—853, 1981.
10. K. S. Sin, G. C. Goodwin, R. R. Bitmead, An adaptive d-step ahead predictor based on least squares, IEEE Trans. Autom. Control, Vol. AC-25, p. 1161—1165, 1980.
11. K. S. Sin, G. C. Goodwin, Stochastic adaptive control using a modified least squares algorithm, Techn. Rep. EE7909, Sec. Rev., Dep. Electr. Engng., Univ. of Newcastle, Australia, 1981.
12. J. B. Moore, G. Ledwich, Multivariable adaptive parameter and state estimators with convergence analysis, J. Austr. Math. Soc., vol. 21, Ser. B, p. 176—197, 1979.
13. J. B. Moore, R. Kumar, Convergence of weighted minimum variance N-step ahead prediction/control schemes, Proc. of the 19-th IEEE Conf. Decision and Control, Albuquerque, NM, p. 968—973, 1980.
14. R. Kumar, J. B. Moore, Convergence of adaptive minimum variance algorithms via weighting coefficient selection, IEEE Trans. Autom. Control, Vol. AC-27, p. 146—153, 1982.
15. V. Sima, Contribuții la realizarea sistemelor adaptive cu model etalon, Teză de doctorat, IPB, 1982.
16. V. Sima, A sufficient condition for convergence of recursive least squares adaptive control algorithms, Prepr. 27 Intern. Wiss. Kolloquium, Techn. Hoch. Ilmenau, GDR, oct., 1982.
17. V. Sima, On recursive least squares adaptive control algorithms, Proc. of the 1-st IASTED Int. Symp. and Course „Measurement and Control”, Tunis, Tunisia, sept. 1—3, 1982.
18. V. Sima, Rezultate noi la conducerea adaptivă, Prepr. A III-a Sesiune „Modelarea, simularea, identificarea și optimizarea proceselor tehnologice”, Univ. Galați, 29—30 oct., 1982.

19. V. Peterka, A square-root filter for a real time multivariable regression, *Kybernetika*, vol. 11, p. 53, 1975.
20. G. J. Bierman, Measurement updating using the U-D factorization, *Automatica*, vol. 12, p. 375-382, 1976.
21. G. J. Bierman, *Factorization Methods for Discrete Sequential Estimation*, Academic Press, New York, 1977.
22. G. C. Goodwin, R. L. Payne, *Dynamic System Identification : Experiment Design and Data Analysis*, Academic Press, New York, 1977.
23. C. L. Thornton, G. J. Bierman, Filtering and error analysis via the UDU^T covariance factorization, *IEEE Trans. Autom. Control*, vol. AC-23, p. 901-907, 1978.
24. P. E. Gill, G. H. Golub, W. Murray, M. A. Saunders, Methods for modifying matrix factorizations, *Math. Comput.*, vol. 28, 126, p. 505-535, 1974.
25. R. Fletcher, M. J. D. Powell, On the modification of LDL^T factorization, *Math. Comput.*, vol. 28, p. 1 067-1 087, 1974.
26. P. E. Gill, W. Murray, Modification of matrix factorizations after a rank-one change, In D. Jacobs (Ed.), *The State of the Art in Numerical Analysis*, Academic Press, New York, p. 55-83, 1977.
27. J. J. Dongarra, C. B. Moler, J. R. Bunch G. W. Stewart, *LINPACK. Users' Guide*, SIAM, Philadelphia, 1979.
28. C. L. Lawson, R. J. Hanson, D. R. Kincaid, F. T. Krogh, Basic Linear Algebra Subprograms for Fortran Usage, *ACM Trans. Math. Software*, vol. 5, p. 308-323, 1979.
29. K. J. Aström, V. Borisson, L. Ljung, B. Wittenmark, Theory and applications of self-tuning regulators, *Automatica*, vol. 13, p. 457-476, 1977.
30. Th. Popescu, Regulator autoacordabil pentru conducerea proceselor industriale, *AMC*, vol. 35, 1983.
31. D. V. Clarke, Some implementation considerations of self-tuning controllers, In F. Archetti, M. Cugliani (Eds.), *Numerical Techniques for Stochastic Systems*, North Holland Publ. Co., 1980.
32. D. W. Clarke, P. J. Gawthrop, Implementation and application of microprocessors-based self-tuners, *Automatica*, vol. 17, p. 233-244, 1981.
33. V. Sima, Th. D. Popescu, ADCON - Adaptive control package, In G. Ferrate, E. A. Puente (Eds.), *Software for Computer Control 1982, Proc. of 3-rd IFAC/IFIP Symp.*, Madrid, 5-8 oct., 1982.
34. Th. D. Popescu, V. Sima, Experiments with an adaptive control package, *Prepr. Applied Control and Identification*, Copenhagen, 28 iunie, 1 iulie, 1983.

Actualizarea factorizării U—D după o modificare de rang 1

Să considerăm factorizarea U—D a unei matrice pozitiv definite $P \in \mathbb{R}^{p \times p}$, adică $P = UDU^T$, unde $U = (u_{ij})$ este o matrice superior triunghiulară, iar $D = \text{diag}(d_1, \dots, d_p)$ este o matrice diagonală pozitiv definită. Dorim să calculăm, stabil și eficient, factorizarea U—D a matricei

$$\hat{P} = P + axx^T \quad (A1)$$

unde $x \in \mathbb{R}^p$ este un vector dat, iar $a \in \mathbb{R}$ este un scalar dat, precump inițial a fi pozitiv. Scriind (A1) astfel

$$\hat{P} = UDU^T + axx^T = [UD^{1/2} \quad a^{1/2}x] G^T G \begin{bmatrix} D^{1/2}U^T \\ a^{1/2}x^T \end{bmatrix} \quad (A2)$$

și alegând matricea ortogonală G așa încît

$$G \begin{bmatrix} D^{1/2}U^T \\ a^{1/2}x^T \end{bmatrix} = \begin{bmatrix} \hat{D}^{1/2}\hat{U}^T \\ 0 \end{bmatrix} \quad (A3)$$

unde \hat{U} este matrice superior triunghiulară, iar \hat{D} este matrice diagonală pozitiv definită, obținem $\hat{P} = \hat{U}\hat{D}\hat{U}^T$. Matricea G poate fi determinată ca produs a p rotații plane, $G = G_1 G_2 \dots G_p$, rotația G_j , $j = p, p-1, \dots, 1$, fiind aplicată în planul $(j, j+1)$. G_j este definită de cosinusul $c_j = d_j^{1/2} u_{j,j} / r$ și sinusul $s_j = (a)^{1/2} x_j^j / r$, unde $r = \pm (u_{j,j}^2 + (x_j^j)^2)^{1/2}$, iar indicii superiori j denotă valoarea variabilelor la a j -etapă de calcul ($a^j = a$, $x^j = x$). Pentru a evita operațiile rădăcină pătrată, se pot utiliza rotații plane modificate, ca în [2]. Dacă de keză a matricei este de a refactoriza produsul $g_j \text{diag}[d_j^{1/2} (a)^{1/2}]$ astfel

$$\begin{bmatrix} c_j & s_j \\ -s_j & c_j \end{bmatrix} \begin{bmatrix} d_j^{1/2} & 0 \\ 0 & (a)^{1/2} \end{bmatrix} = \begin{bmatrix} \hat{d}_j^{1/2} & 0 \\ 0 & (a^{j-1})^{1/2} \end{bmatrix} H_j \quad (A4)$$

unde matricea H_j de tip 2×2 are în general două elemente egale cu 1. Atunci,

$$\begin{bmatrix} c_j & s_j \\ -s_j & c_j \end{bmatrix} \begin{bmatrix} d_j^{1/2} & 0 \\ 0 & (a)^{1/2} \end{bmatrix} \begin{bmatrix} u_{1j} \dots u_{jj} \\ x_1^j \dots x_j^j \end{bmatrix} = \begin{bmatrix} \hat{d}_j^{1/2} & 0 \\ 0 & (a^{j-1})^{1/2} \end{bmatrix} H_j \begin{bmatrix} \hat{u}_{1j} \dots \hat{u}_{jj} \\ \hat{x}_1^j \dots \hat{x}_j^j \end{bmatrix} \quad (A5)$$

$$\begin{bmatrix} \hat{d}_j^{1/2} & 0 \\ 0 & (a^{j-1})^{1/2} \end{bmatrix} \begin{bmatrix} \hat{u}_{1j} \dots \hat{u}_{j-1,j} & \hat{u}_{jj} \\ \hat{x}_1^{j-1} \dots \hat{x}_{j-1}^{j-1} & 0 \end{bmatrix}$$

Vom descrie pe scurt calculul lui H_j . Pentru simplitate, indicii pentru c_j și s_j vor fi omiși. Presupunând că $|s| < |c|$, putem scrie

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} d^{1/2} & 0 \\ 0 & (a)^{1/2} \end{bmatrix} = \begin{bmatrix} d^{1/2}c & 0 \\ 0 & (a)^{1/2}c \end{bmatrix} \begin{bmatrix} 1 & t(a/d)^{1/2} \\ -t(d/a)^{1/2} & 1 \end{bmatrix} = \\ = \begin{bmatrix} \hat{d}^{1/2} & 0 \\ 0 & (a^{j-1})^{1/2} \end{bmatrix} \begin{bmatrix} 1 & ax_j^j / (d u_{jj}) \\ -x_j^j / u_{jj} & 1 \end{bmatrix} \quad (A6)$$

unde $\hat{d}_j = d_j c^2$, $a^{j-1} = a^j c^2$, iar $t = s/c = (a^j/d_j)^{1/2} x_j^j / u_{jj}$. Notînd $h_1 = a^j x_j^j / (d_j u_{jj})$ și $h_2 = x_j^j / u_{jj}$, obținem $1/c^2 = 1 + t^2 = 1 + h_1 h_2$. Deci,

$$\tau \triangleq 1 + h_1 h_2, \hat{d}_j = d_j / \tau, a^{j-1} = a^j / \tau, H_j = \begin{bmatrix} 1 & h_1 \\ -h_2 & 1 \end{bmatrix} \tag{A7}$$

Din motive de stabilitate numerică, pentru $|s| \geq |c|$, se utilizează alte relații care se deduc similar. Notînd $h'_1 = d_j u_{jj} / (a^j x_j^j)$, $h'_2 = u_{jj} / x_j^j$, avem

$$\tau' \triangleq 1 + h'_1 h'_2, \hat{d}_j = a^j / \tau', a^{j-1} = d_j / \tau', H_j = \begin{bmatrix} h'_1 & 1 \\ -1 & h'_2 \end{bmatrix} \tag{A8}$$

Pentru a evita depășirile aritmetice superioare și inferioare, \hat{d}_j și a^{j-1} sînt menținute între anumite limite prestabilite, prin scalarea problemei, dacă este necesar. În aceste rare cazuri, în care se face scalare, matricea H_j poate avea mai mult de două elemente diferite de 1.

Întrucît în calculele efective nu apar rădăcini pătrate, tehnica se poate aplica și în cazul $a < 0$.

Pentru determinarea rotațiilor plane modificate descrise mai sus se poate utiliza subrutina SROTMG din pachetul BLAS [28]. Actualizarea factorizării U-D este realizată așadar de următoarea procedură.

Procedura A1: Actualizează factorizarea U-D după adăugarea unei diade.

1) Pentru $j = p, p-1, \dots, 1$

- 1) Utilizează SROTMG cu datele d_j, a, u_{jj} și x_j pentru a determina H_j și pentru a actualiza d_j, a și u_{jj} .
- 2) Pentru $j > 1$

$$\begin{bmatrix} u_{1j} & \dots & u_{j-1, j} \\ x_1 & \dots & x_{j-1} \end{bmatrix} := H_j \begin{bmatrix} u_{1j} & \dots & u_{j-1, j} \\ x_1 & \dots & x_{j-1} \end{bmatrix}$$

Observații

A1) Unele variabile sînt suprascrise în procedură. În final, U și D conțin cantitățile actualizate \hat{U} și, respectiv, \hat{D} .

A2) Procedura necesită $p^2 \gg 10p$ multiplicări, dacă nu este necesară nici o scalare.

A3) Tehnica poate fi ușor adaptată factorizării LDL^T.

Încheiem această secțiune prin deducerea pe o cale nouă a algoritmului din Anexa II din [23]. În acest caz impunem ca U și \hat{U} să fie superior triunghiulare unitate. Să considerăm cazul $|s| < |c|$, deci H_j este definit ca în (A7). Atunci avem $\hat{u}_{jj} = u_{jj} + a^j (x_j^j)^2 / (d_j \hat{u}_{jj}) = 1 + t^2$.

Pentru a obține $u_{jj} = 1$, coloana a j -a din matricea \hat{U} , rezultată din (A5), trebuie împărțită prin $1 + t^2 (= d_j / \hat{d}_j)$, deci \hat{d}_j trebuie înmulțit cu $(1 + t^2)^2$, pentru a conserva factorizarea.

După calcule elementare, se obține :

$$\hat{d}_j = d_j + a^j (x_j^j)^2 \tag{A9.a}$$

$$a^{j-1} = a^j d_j / \hat{d}_j \tag{A9.b}$$

$$\hat{u}_{jj} = (d_j u_{jj} + \sum_{k=1}^{j-1} x_k^j)^2 / \hat{d}_j = u_{jj} + a^j \sum_{k=1}^{j-1} x_k^j / \hat{d}_j \tag{A9.c}$$

$$x_k^{j-1} = x_k^j = u_{kj} x_j^k \tag{A9.d}$$

Acestea sînt tocmai relațiile utilizate în algoritmul din [23]. Pentru actualizarea lui u_{ij} se folosește una sau alta dintre cele două variante de mai sus în funcție de valoarea raportului d_j / \hat{d}_j . Relațiile prezentate se obțin și în cazul $|s| \geq |c|$.

Actualizarea factorizării $U*U'$ a matricei de informație

```

C
C
C SQRINF ACTUALIZEAZA FACTORIZAREA U*U' A MATRICEI DE INFORMATIE,
C UNDE U ESTE SUPERIOR TRIUNGIULARA, DUPA ADAUGAREA UNEI DIADE:
C (ALAMBD)*U*U' + A*(PHI)*(PHI)'.
C SIMULTAN, SQRINF ACTUALIZEAZA CORESPUNZATOR PERECHEA (Z,Y),
C UNDE Z ESTE UN VECTOR, IAR Y ESTE UN SCALAR.
C
C MOD DE UTILIZARE
C CALL SQRINF (IP,U,PHI,Z,Y,A,ALAMBD)
C
C DESCRIEREA PARAMETRILOR
C IP - ORDINUL MATRICEI U SI DIMENSIUNEA VECTORILOR PHI, Z.
C U - VECTOR DAT CU IP*(IP+1)/2 ELEMENTE CONTININD PARTEA
C TRIUNGIULARA SUPERIOARA A MATRICEI U, PE COLOANE.
C ESTE ACTUALIZAT IN RUTINA.
C PHI - VECTOR DAT CU IP ELEMENTE. ESTE ALTERAT IN RUTINA.
C Z - VECTOR DAT CU IP ELEMENTE. ESTE ACTUALIZAT IN RUTINA.
C Y - SCALAR DAT. ESTE ACTUALIZAT IN RUTINA.
C A - SCALAR DAT.
C ALAMBD - SCALAR DAT.
C
C SUBPROGRAME UTILIZATE
C SROTG
C
C METODA
C SE DETERMINA ROTATII GIVENS PENTRU A ANULA ELEMENTELE ULTIMEI
C LINII DIN MATRICEA:
C
C [ SQRT(ALAMBD) * U' ]
C [ SQRT(A) * (PHI)' ]
C
C IN ORDINEA J = IP,IP-1,...,1. ROTATIILE SE APLICA ACESTEI MATRICI
C SI VECTORULUI
C
C [ SQRT(ALAMBD) * Z ]
C [ SQRT(A) * Y ].
C
C SUBROUTINE SQRINF (IP,U,PHI,Z,Y,A,ALAMBD)
C
C DIMENSION U(1),PHI(1),Z(1)
C
C REAL C,C1,S,S1,SQRTA,SQRTLA,TEMP
C INTEGER I,J,JI,JJ,JMI,K,L
C LOGICAL FORGET
C
C SQRTLA = SQRT (ALAMBD)
C SQRTA = SQRT (A)
C FORGET = .TRUE.
C IF (ALAMBD .EQ. 1.0) FORGET = .FALSE.
C
C IF (A .EQ. 1.0) GO TO 10
C Y = Y * SQRTA
C DO 5 J = 1,IP
C PHI(J) = PHI(J) * SQRTA
C 5 CONTINUE
C
C 10 CONTINUE
C JJ = IP * (IP + 1) / 2
C
C 00 40 L = 1,IP
C J = IP - L + 1
C JI = JJ - J
C IF (PHI(J) .EQ. 0.0) GO TO 30

```

```
IF (F0RGET) U(JJ) = U(JJ) * SQRTLA
CALL SROTG (U(JJ),PHI(J),C,S)
C
C1 = C
S1 = S
IF (.NOT. F0RGET) GO TO 15
  C1 = C1 * SQRTLA
  S1 = S1 * SQRTLA
15 CONTINUE
JMI = J - 1
IF (JMI .EQ. 0) GO TO 25
C
  K = JI
  DO 20 I = 1,JMI
    K = K + 1
    TEMP = U(K)
    U(K) = C1 * TEMP + S * PHI(I)
    PHI(I) = C * PHI(I) - S1 * TEMP
20 CONTINUE
C
25 CONTINUE
TEMP = Z(J)
Z(J) = C1 * TEMP + S * Y
Y = C * Y - S1 * TEMP
C
30 CONTINUE
JJ = JI
40 CONTINUE
C
RETURN
END
```

Actualizarea factorizării U—D a matricei de covarianță

```

C      SUBROUTINA UDCOV
C
C UDCOV ACTUALIZEAZA FACTORIZAREA U*D*UT A MATRICEI DE COVARIANTA,
C UNDE U ESTE SUPERIOR TRIUNghiULARA UNITATE, IAR D ESTE DIAGONALA,
C SI DETERMINA VECTORUL DE AMPLIFICARE KALMAN NORMALIZAT.
C
C MOD DE UTILIZARE
C CALL UDCOV (IP,U,PHI,A,ALAMBD,AK,AP)
C
C DESCRIEREA PARAMETRILOR
C IP   - ORDINUL MATRICILOR U, D SI DIMENSIUNEA VECTORILOR PHI, AK.
C J    - VECTOR DAT CU IP*(IP+1)/2 ELEMENTE CONTININD PARTEA
C      TRIUNghiULARA SUPERIDARA STRICTA A MATRICEI U, PE COLOANE.
C      IN LOCATIILE CORESPUNZATOARE ELEMENTELOR DIAGONALE UNITATE
C      NEMEMORATE ALE MATRICEI U, SE GASESC ELEMENTELE DIAGONALE
C      ALE MATRICEI D. ESTE ACTUALIZAT IN RUTINA.
C PHI  - VECTOR DAT CU IP ELEMENTE.
C A    - SCALAR DAT.
C ALAMBD - SCALAR DAT.
C AK   - VECTOR REZULTAT CU IP ELEMENTE CONTININD AMPLIFICAREA
C      KALMAN NORMALIZATA.
C AP   - SCALAR REZULTAT CONTININD INVERSUL FACTORULUI DE NORMALIZARE.
C      (AMPLIFICAREA KALMAN ESTE AK / AP).
C
C SUBPROGRAME UTILIZATE
C RFVPS
C
C METJDA
C SE UTILIZEAZA ALGORITMUL DE ACTUALIZARE A FACTORIZARII U-D A MATRICEI
C DE COVARIANTA DESCRIS IN [1].
C
C REFERINTE BIBLIOGRAFICE
C [1] THOMPSON, C.L., BIERMAN, G.J., FILTERING AND ERROR ANALYSIS
C      T
C      VIA THE UDU COVARIANCE FACTORIZATION. IEEE TRANS. AUT.
C      CONTR., AC-23, 901-907, 1978.
C
C      SUBROUTINE UDCOV (IP,U,PHI,A,ALAMBD,AK,AP)
C
C      DIMENSION U(1),PHI(1),AK(1)
C
C      REAL S,F,AJ,AJMI,AKJ,TEMP
C      INTEGER I,J,K,KD,JMI
C      LOGICAL FORGET
C
C      FORGET = .TRUE.
C      IF (ALAMBD .EQ. 1.0) FORGET = .FALSE.
C
C      K = 0
C      KD = 0
C      AJ = ALAMBD/A
C
C      DO 20 J = 1,IP
C         K = K + 1
C         KD = KD + J
C         JMI = J - 1
C         F = PHI(J)
C         IF (JMI .GT. 0) F = F + RFVPS (JMI,J(K),PHI)
C         AKJ = U(KD) * F
C         AK(J) = AKJ
C         AJMI = AJ
C         AJ = AJ + AKJ * F
C         S = AJ
C         IF (FORGET) S = S * ALAMBD

```

```

      U(KD) = U(KD) * AJM1 / S
      IF (JM1 .EQ. 0) GO TO 20
C
      S = - F / AJM1
      DO 10 I = 1, JM1
          TEMP = U(K)
          U(K) = TEMP + S * AK(I)
          AK(I) = AK(I) + AKJ * TEMP
          K = K + 1
      10  CONTINUE
C
      20 CONTINUE
C
      AP = AJ
C
      RETURN
      END
C
C
C
C
C      FUNCTIA RFVPS
C
C      RFVPS CALCULEAZA PRODUSUL SCALAR A DOI VECTORI.
C
C      MOD DE UTILIZARE
C      X = RFVPS (N, SX, SY)
C
C      DESCRIEREA PARAMETRILOR
C      N      - DIMENSIUNEA VECTORILOR SX SI SY.
C      SX     - VECTOR DAT CU N ELEMENTE.
C      SY     - VECTOR DAT CU N ELEMENTE.
C      RFVPS  - SUMA REZULTATA A PRODUSELOR SX(I)*SY(I) , I = 1,...,N.
C
C      METODA
C      SE UTILIZEAZA CICLURI DESFASURATE, PENTRU MARIREA EFICIENTEI.
C      ESTE ADAPTAT SUBPROGRAMUL SDOT DIN PACHETUL BLAS (1).
C
C      REFERINTE BIBLIOGRAFICE
C      (1) LA#SON, C.L., R.J. HANSON, D.R. KINCAID, F.T. KROGH, BASIC
C      LINEAR ALGEBRA SUBPROGRAMS FOR FORTRAN USAGE. ACM TRANS.
C      MATH. SOFTWARE, 5, 308-323, 1979.
C
C      REAL FUNCTION RFVPS(N, SX, SY)
C
C      INTEGER          N
C      REAL             SX(1), SY(1)
C
C      INTEGER          I, M, MP1
C
C      RFVPS = 0.0E0
C      IF (N .LE. 0) RETURN
C
      15  M = N - (N/5) * 5
          IF (M .EQ. 0) GO TO 25
          DO 20 I = 1, M
              RFVPS = RFVPS + SX(I) * SY(I)
          20 CONTINUE
          IF (N .LT. 5) RETURN
      25  MP1 = M + 1
          DO 30 I = MP1, N, 5
              RFVPS = RFVPS + SX(I) * SY(I) + SX(I+1) * SY(I+1) + SX(I+2) * SY(I+2) + SX(I
              + 3) * SY(I+3) + SX(I+4) * SY(I+4)
          30 CONTINUE
          RETURN
      END

```

METODE DE ESTIMARE RECURSIVĂ A PARAMETRILOR SISTEMELOR

Dr. ing. Th. D. Popescu
I.T.C.I.

1. Introducere

În multe aplicații se urmărește estimarea parametrilor sistemelor în mod *recursiv*, pe măsura evoluției acestora. Acest mod de abordare prezintă interes, în special pentru actualizarea continuă a modelului sistemului, când caracteristicile acestuia și/sau ale mediului se modifică în timp. Tehnicile de estimare recursivă au largi aplicații în modelarea și predicția seriilor de timp, estimarea stării, conducerea adaptivă a proceselor, recunoașterea formelor. Acestea utilizează datele pe măsura culegerii lor, fără a necesita întreg setul de date intrare-ieșire din proces și pot fi ușor adaptate pentru estimarea în timp real a parametrilor sistemelor.

Metodele de estimare recursivă permit o formulare secvențială a problemei de estimare și necesită un volum de memorie și de calcul redus față de metodele de estimare nerecursivă, totuși din punct de vedere al preciziei sînt inferioare acestora din urmă. Ele asigură o viteză de convergență convenabilă, în măsura în care se cunoaște a priori domeniul de variație al parametrilor procesului, se inițializează corect anumite matrici de ponderare și se aleg corespunzător valorile inițiale ale parametrilor modelului.

În cadrul estimării recursive a parametrilor sistemelor se pot utiliza următoarele clase de metode :

- metode utilizînd un model de referință ;
- metode de aproximare stohastică ;
- metode recursive.

Metodele utilizînd un model de referință (Landau, 1974) se folosesc, aproape exclusiv, în cazul sistemelor deterministe, iar în analiza acestora se folosește în principal metoda Liapunov (Parks, 1966 ; Narendra și Kudva, 1974) și teoria hiperstabilității (Landau, 1974). Multe din metodele de estimare, utilizînd un model de referință, au bune proprietăți în cazul determinist, dar nu se comportă întotdeauna satisfăcător în prezența zgomotului ; sînt totuși adecvate estimării parametrilor variabili în timp.

Metodele de aproximare stohastică (Saridis-Stein, 1968 ; Saridis, 1974 ; Sakrison, 1967), spre deosebire de schemele de estimare deterministe utilizînd un model de referință, au fost dezvoltate în special pentru estimarea parametrilor în prezența zgomotului.

Viteza de convergență a algoritmilor de aproximare stohastică poate fi, totuși, destul de redusă în cazul unei alegeri neinspirate a secvenței factorilor de amplificare.

Metodele recursive reprezintă acele metode de estimare care utilizează construcții de tipul celor mai mici pătrate recursive (Aström-Eykhoff, 1971; Eykhoff, 1974; Furth, 1973; Gertler-Banyasz, 1974; Hastings James-Sage 1974; Söderström și colab., 1974, 1978; Young, 1968, 1970, 1974, 1976 etc.). Analiza acestor metode se poate face utilizând tehnica propusă de Ljung (1977a, 1977b), care face apel la o ecuație diferențială ordinară asociată algoritmilor de estimare recursivă (se utilizează apoi metodele Liapunov), sau tehnica propusă de Hannan (1976), care reprezintă o aplicare sofisticată a teoriei probabilităților, analiza reducându-se la studiul unei funcții deterministe de tip Liapunov. Metodele recursive, în general, au bune proprietăți de convergență și se comportă bine în prezența zgomotului.

În cadrul acestei lucrări sînt analizate și comparate principalele metode de estimare recursivă tratate în literatură, accentul fiind pus în special pe metodele din ultima categorie. Aceste metode sînt prezentate în secțiunea 3. Metodele prezintă elemente comune și sînt tratate într-un cadru unitar; sînt prezentate de asemenea și unele modificări ce pot fi aduse algoritmilor recursivi avînd drept scop: obținerea versiunilor în timp real, îmbunătățirea vitezei de convergență. În secțiunea 4 sînt prezentate instrumentele de analiză a algoritmilor recursivi și se face o sinteză a rezultatelor acestei analize. Aspectele legate de implementarea acestor algoritmi constituie obiectul secțiunii 5 a acestei lucrări.

Metodele de estimare recursivă ce se prezintă au constituit obiectul unui număr mare de simulări. Rezultatele numerice obținute în estimarea parametrilor unor sisteme funcționînd în buclă deschisă și în estimarea parametrilor unor regulatoare autoacordabile, în cadrul unor sisteme funcționînd în buclă închisă, sînt prezentate în cadrul altor lucrări (Popescu, 1980, 1981).

2. Preliminarii

În cadrul acestei secțiuni sînt prezentate metode de estimare recursivă frecvent utilizate în practică și care au elemente comune:

- metoda celor mai mici pătrate recursivă — RCMMP;
- metoda celor mai mici pătrate generalizată recursivă — RCMMPG;
- metoda variabilelor instrumentale recursivă — RVI;
- metoda celor mai mici pătrate extinsă recursivă (metoda matricei extinse, metoda verosimilității maxime aproximative, metoda Panuska) — RCMMPPE;
- metoda verosimilității maxime recursivă — RVM.

Multe din aceste metode se regăsesc în literatură combinate (ex. RVI și RCMMPPE — Young 1974), sau sub alte denumiri. Metoda RCMMP este bine cunoscută, fiind tratată de mulți autori (ex. Aström și Eykhoff, 1971). Primele lucrări asupra metodei RVI au fost elaborate de Mayne (1967), Wong

și Polak (1967) și Young (1938). Metoda RCMMPG a fost sugerată de Hastings James și Sage (1969) și se bazează pe versiunea off-line a metodei (Clarke, 1967). Metoda RCMMPG a fost propusă de Panuska (1968, 1939) și Young (1968), iar prezentarea metodei RVM este făcută de Furth (1973) și Söderström (1973). Alte referințe asupra acestor metode sînt date de Söderström, Ljung și Gustavsson (1974), Tertîșco și Stoica (1980).

Așa cum se va arăta în continuare, toți acești algoritmi au aceeași structură. Pentru tratarea lor într-un cadru general vor fi făcute citeva ipoteze și convenții generale.

Fie $u(1), \dots, u(N)$ semnalul de intrare și $y(1), \dots, y(N)$ semnalul de ieșire pentru care se determină, cu metodele amintite, un model avînd următoarea structură generală :

$$\widehat{A}(q^{-1})y(t) = \widehat{B}(q^{-1})z(t) + \widehat{H}(q^{-1})\varepsilon(t) \quad (2.1)$$

unde $\varepsilon(t)$ reprezintă reziduul, iar $\widehat{A}(q^{-1})$ și $\widehat{B}(q^{-1})$ sînt polinoame în operatorul de întîrziere q^{-1} ($q^{-1}x(t) = x(t-1)$) :

$$\widehat{A}(q^{-1}) = 1 + \widehat{a}_1 q^{-1} + \dots + \widehat{a}_{n_a} q^{-n_a}$$

$$\widehat{B}(q^{-1}) = b_1 + \widehat{b}_2 q^{-1} + \dots + \widehat{b}_{n_b} q^{-n_b}$$

Filtrul $\widehat{H}(q^{-1})$ are forme diferite, funcție de metoda utilizată, astfel :

Pentru RCMMP : $\widehat{H}(q^{-1}) = 1$ (2.2)

Pentru RIV, $\widehat{H}(q^{-1})$ nu este specificat ; din motive formale se va utiliza tot modelul (2.2), reziduul $\varepsilon(t)$ fiind arbitrar.

Pentru RCMMPG : $\widehat{H}(q^{-1}) = 1/\widehat{C}(q^{-1}) = 1/(1 + \widehat{c}_1 q^{-1} + \dots + \widehat{c}_{n_c} q^{-n_c})$ (2.3)

iar pentru RCMMPG și RVM : $\widehat{H}(q^{-1}) = \widehat{C}(q^{-1}) = 1 + \widehat{c}_1 q^{-1} + \dots + \widehat{c}_{n_c} q^{-n_c}$ (2.4)

Unele dintre metode ca, de exemplu, RCMMPG (Talmon și van den Boom, 1973) și RVM pot fi utilizate, de asemenea, și pentru alte structuri ale modelului.

Includerea în structura modelului (2.1) a mai multor semnale de intrare și a timpului mort nu va ridica, în general, probleme deosebite. Din motive de simplificare a notațiilor va fi tratat numai cazul sistemelor cu o singură intrare, fără timp mort.

Deoarece sîntem interesați în special în analiza proprietăților de consistență ale algoritmilor se va presupune că, pentru fiecare din metode, sistemul are o formă corespunzătoare structurii modelului utilizat ; prin urmare se va presupune că sistemul este de forma :

$$A(q^{-1})y(t) = B(q^{-1})u(t) + H(q^{-1})e(t) \quad (2.5)$$

unde $e(t)$ este zgomot alb, iar polinoamele $A(q^{-1})$ și $B(q^{-1})$ sînt date de relațiile :

$$A(q^{-1}) = 1 + a_1 q^{-1} + \dots + a_{n_a} q^{-n_a}$$

$$B(q^{-1}) = b_1 + b_2 q^{-1} + \dots + b_{n_b} q^{-n_b}$$

Pentru anumite interpretări, caracteristice metodei verosimilității maxime, se presupune de asemenea că $e(t)$ este gaussian, dar această ipoteză nu este esențială pentru analiza de convergență.

Filtrul $H(q^{-1})$ se presupune a avea următoarele structuri:

$$\text{Pentru RCMMP : } H(q^{-1})=1 \quad (2.6)$$

$$\text{Pentru RCMMPG : } H(q^{-1})=1/C(q^{-1})=1/(1+c_1q^{-1}+\dots+c_{n_c}q^{-n_c}) \quad (2.7)$$

$$\text{iar pentru RCMMPG și RVM : } H(q^{-1})=C(q^{-1})=1+c_1q^{-1}+\dots+c_{n_c}q^{-n_c} \quad (2.8)$$

Pentru cele mai multe din variantele metodei RVI nu este necesar a se specifica structura filtrului $H(q^{-1})$. Se presupune că ordinul sistemului este cunoscut a priori.

În general se presupune că descrierea sistemului (2.5) este astfel încît perechea de polinoame $A(z)$, $B(z)$ (z fiind o variabilă complexă arbitrară, înlocuind q^{-1}) nu are nici un factor comun: aceeași ipoteză se face și pentru polinoamele $A(z)$ și $C(z)$ în cazul metodelor RCMMPG și RVM. Această ipoteză implică în special controlabilitatea procesului. De asemenea, se presupune că sistemul este asimptotic stabil. Pentru sistemul operînd în buclă deschisă, aceasta înseamnă că polinomul $A(z)$ are toate zerourile în afara cercului unitate; aceeași ipoteză se face și pentru polinomul $C(z)$, ceea ce reprezintă o restricție minoră (teorema de factorizare spectrală, Aström 1970) și este necesară pentru a se asigura inversarea modelului zgomotului (RCMMPG și RVM), deci dispersia finită a reziduurilor.

Semnalul de intrare poate fi determinat în mai multe moduri, rezultatele ce se prezintă fiind valabile pentru o intrare de tip excitație persistentă (Aström și Bohlin, 1965), de exemplu zgomot alb filtrat, situație ce apare destul de frecvent. În special, analiza acoperă cazul funcționării în buclă închisă a sistemului, în care regulatorul este variabil în timp și se determină utilizînd modelul curent al procesului (cazul sistemelor adaptive).

3. Metode de estimare recursivă a parametrilor

3.1. Prezentare unitară

Metodele menționate în cadrul secțiunii precedente pot fi descrise în mod unitar de următorul algoritm:

$$\hat{\theta}(t) = \hat{\theta}(t-1) + K(t) \varepsilon(t) \quad (3.1)$$

$$K(t) = \frac{P(t-1)z(t)}{1 + \varphi(t)^T P(t-1)z(t)} \quad (3.2)$$

$$P(t) = P(t-1) - \frac{P(t-1)z(t)\varphi(t)^T P(t-1)}{1 + \varphi(t)^T P(t-1)z(t)} \quad (3.3)$$

unde $\hat{\theta}$ reprezintă vectorul parametrilor estimați, utilizînd datele $u(1)$, $y(1)$, ..., $u(t)$, $y(t)$, fiind de forma:

Pentru RCMMP și RVI :

$$\hat{\theta} = [\hat{a}_1, \dots, \hat{a}_{n_a}, \hat{b}_1, \dots, \hat{b}_{n_b}]^T \quad (3.4a)$$

și pentru RCMMPG, RCMMPPE, RVM :

$$\hat{\theta} = [\hat{a}_1, \dots, \hat{a}_{n_a}, \hat{b}_1, \dots, \hat{b}_{n_b}, \hat{c}_1, \dots, \hat{c}_{n_c}]^T$$

(parametrii c_i au semnificații diferite pentru RCMMPG și respectiv RCMMPPE și RVM) ;

$\varepsilon(t)$ reprezintă o estimare a erorii de predicție pe un pas.

Parametrii sistemului (2.5) pot fi grupați într-un vector θ_0 al valorilor reale ale parametrilor, de forma :

Pentru RCMMP și RVI

$$\theta_0 = [a_1, \dots, a_{n_a}, b_1, \dots, b_{n_b}]^T \quad (3.4b)$$

și pentru RCMMPG, RCMMPPE, RVM

$$\theta_0 = [a_1, \dots, a_{n_a}, b_1, \dots, b_{n_b}, c_1, \dots, c_{n_c}]^T$$

cu aceeași observație, ca în cazul vectorului $\hat{\theta}$, pentru parametrii c_i .

Metodele în discuție vor fi obținute, sub forma unor cazuri particulare ale relațiilor (3.1)—(3.3), prin specificarea variabilelor $\varphi(t)$ și $z(t)$.

Metoda RCMMP. Algoritmul se obține din relațiile (3.1)—(3.3) cu următoarea alegere a vectorilor $\varphi(t)$, $z(t)$:

$$\varphi(t) = [-y(t-1), \dots, -y(t-n_a), u(t-1), \dots, u(t-n_b)]^T \quad (3.5)$$

$$z(t) = \varphi(t) \quad (3.6)$$

și cu eroarea de predicție calculată astfel :

$$\varepsilon(t) = y(t) - \varphi^T(t) \hat{\theta}(t-1) \quad (3.7)$$

Metoda RVI. Algoritmul de calcul se obține prin alegerea vectorului $\varphi(t)$ și a erorii de predicție conform relațiilor (3.5) și respectiv (3.7), iar a vectorului z , al variabilelor instrumentale, astfel încît :

$$(i) \lim_{N \rightarrow \infty} \frac{1}{N} \sum_{t=1}^N z(t) \varphi^T(t) = R, \quad R \text{ matrice nesingulară} \quad (3.8a)$$

$$(ii) \lim_{N \rightarrow \infty} \frac{1}{N} \sum_{t=1}^N z(t) H(q^{-1}) e(t) = 0 \quad (3.8b)$$

Alegerea elementelor vectorului $z(t)$ se poate face în mai multe moduri. O posibilitate de alegere este următoarea :

$$z(t) = [-x(t-1), \dots, -x(t-n_a); u(t-1), \dots, u(t-n_b)]^T \quad (3.9)$$

unde $x(t)$ se obține prin filtrarea semnalului de intrare $u(t)$, conform relației :

$$x(t) = \frac{\bar{B}(q^{-1})}{\bar{A}(q^{-1})} u(t) \quad \text{cu} \quad (3.10)$$

$$\bar{A}(q^{-1}) = 1 + \bar{a}_1 q^{-1} + \dots + \bar{a}_{n_a} q^{-n_a}$$

$$\bar{B}(q^{-1}) = \bar{b}_1 q^{-1} + \dots + \bar{b}_{n_b} q^{-n_b}$$

relativ prime (Finigan și Rowe, 1974 ; Söderstrom, 1974a). În special, în cazul semnalelor de intrare de tip excitație persistentă, este indicată alegerea $\bar{A}(q^{-1}) \equiv A(q^{-1})$, $\bar{B}(q^{-1}) \equiv B(q^{-1})$. În acest caz $x(t)$ devine egal cu partea deterministă a semnalului de ieșire. Această alegere este optimală (Wong și Polak, 1967 ; Young, 1976), totuși nu este posibilă deoarece sînt necesare valorile reale ale parametrilor modelului.

În locul variabilei $x(t)$ se propune (Wong și Polak, 1967 ; Young, 1968 ; Rowe, 1970) utilizarea variabilei :

$$\hat{x}(t) = z(t)^T \hat{\theta}(t) \quad (3.11)$$

și modificarea în consecință a vectorului $z(t)$.

S-au sugerat, de asemenea, următoarele modificări ale relației (3.11) :

(i) trecerea vectorului parametrilor estimați printr-un element de întîrziere :

$$\hat{x}(t) = z(t)^T \hat{\theta}(t-\tau) \quad (\tau \text{ întreg mic pozitiv}) \quad (3.12)$$

(ii) trecerea vectorului parametrilor estimați printr-un filtru trece jos :

$$\hat{x}(t) = z(t)^T \tilde{\theta}(t) \quad (3.13)$$

$$\tilde{\theta}(t) = (1-\gamma) \tilde{\theta}(t-1) + \gamma \hat{\theta}(t) \quad (\gamma \text{ ușor subunitar})$$

Aceste modificări nu influențează proprietățile de convergență ale metodei, discutate în această lucrare.

Frecvent se utilizează $\bar{A}(q^{-1}) \equiv 1$, $\bar{B}(q^{-1}) \equiv q^{-n_b}$, vectorul variabilelor instrumentale conținînd numai valori ale mărimii de intrare :

$$z(t) = [u(t-1), \dots, u(t-n_a-n_b)]^T \quad (3.14)$$

Așa cum s-a menționat anterior metoda RVI nu determină un model al zgomotului. Din acest motiv s-a propus (Young, 1974) combinarea metodei RVI cu RCMMPPE. Metoda sugerată determină, în prima etapă, utilizînd RVI, parametrii procesului $\hat{A}(q^{-1})$, $\hat{B}(q^{-1})$; în cea de-a doua etapă se calculează pe baza valorilor parametrilor estimați, reziduurile modelului, care se utilizează ca date de intrare pentru metoda RCMMPPE. Analiza proprietăților de convergență a metodei combinate se rezumă la analiza separată a proprietăților de convergență ale celor două metode.

Ulterior Young (1976) a propus o nouă schemă de estimare pentru metoda RVI, care permite și calculul modelului zgomotului. În acest caz, estimarea dinamicii zgomotului este întreprinsă cu estimarea dinamicii procesului (para-

metrii $\hat{A}(q^{-1})$, $\hat{B}(q^{-1})$). Metoda propusă este asemănătoare cu metoda RVM, dar nu este echivalentă cu aceasta.

În literatură sînt prezentate și alte metode de identificare recursive similare metodei RVI. Una dintre ele o reprezintă metoda principiului informației apriorice (tally principle) (Peterka-Haluszkova, 1970).

Metoda RCMMPG. Varianta recursivă a metodei a fost sugerată de Hastings-James-Sage și a fost inspirată de algoritmul off-line al lui Clarke (1967). Algoritmul include două estimatoare de tipul RCMMP cuplate prin faza de filtrare. Unul din estimatoare este utilizat pentru estimarea coeficienților \hat{A} și \hat{B} , în timp ce celălalt se utilizează pentru estimarea coeficienților $\hat{C}(q^{-1})$. Detalii asupra acestei metode sînt date de Hastings-James și Sage (1969). Algoritmii similari au fost sugerați ulterior de Sen și Sinha (1975) și Gertler și Bányász (1974).

Metoda RCMMPPE. În acest caz

$$z(t) = \varphi(t) = [-y(t-1) \dots -y(t-n_a), u(t-1), \dots, u(t-n_b), \varepsilon(t-1), \dots, \varepsilon(t-n_c)]^T \quad (3.15)$$

$$\text{cu} \quad \varepsilon(t) = y(t) - \varphi(t)^T \hat{\theta}(t-1) \quad (3.16)$$

Algoritmul este adesea utilizat pentru modelarea seriilor de timp (modele fără parametri b_1), iar uneori pentru modelarea proceselor fără parametri a_1 . Pot fi utilizate și alte structuri de modele, ca de exemplu cea propusă de Talmon și van den Boom (1973):

$$\hat{A}(q^{-1})y(t) = \hat{B}(q^{-1})u(t) + \frac{\hat{C}(q^{-1})}{\hat{D}(q^{-1})}\varepsilon(t) \quad (3.17)$$

Metoda RVM. Metoda off-line a verosimilității maxime a fost dezvoltată de Aström și Bohlin (1965), pentru modele descrise de relațiile (2.1) și (2.4).

Metoda VM off-line corespunde minimizării funcției $\sum_{t=1}^N \varepsilon^2(t)$, unde $\varepsilon(t)$ este definit de (2.1). O versiune recursivă (aproximativă) a acestei metode este dată de Söderstrom (1973). Algoritmi similari sînt prezentați de Furht (1973) și de Gertler și Bányász (1974). Metoda RCMMPPE și cea propusă de Young (1976) pot fi interpretate ca variante aproximative ale metodei RVM.

Metoda RVM este descrisă de ecuațiile (3.1)–(3.3) cu

$$z(t) = \varphi(t) = \left[-\frac{1}{\hat{C}(q^{-1})}y(t-1), \dots, -\frac{1}{\hat{C}(q^{-1})}y(t-n_a); \frac{1}{\hat{C}(q^{-1})}u(t-1), \dots, \dots, \frac{1}{\hat{C}(q^{-1})}u(t-n_b); \frac{1}{\hat{C}(q^{-1})}\varepsilon(t-1), \dots, \frac{1}{\hat{C}(q^{-1})}\varepsilon(t-n_c) \right]^T \quad (3.18)$$

Eroarea de predicție $\varepsilon(t)$ se determină cu relația:

$$\hat{C}(q^{-1})\varepsilon(t) = \hat{A}(q^{-1})y(t) - \hat{B}(q^{-1})u(t) \quad (3.19)$$

Pentru a obține eroarea de predicție exactă, ecuația (3.19) trebuie rezolvată, începând cu $t=0$, pentru fiecare nou set de măsurări. Aceasta necesită un timp de calcul considerabil, fapt pentru care, în determinarea erorii de predicție $\varepsilon(t)$, se fac aproximații convenabile. O astfel de soluție constă în determinarea erorii de predicție $\varepsilon(t)$ cu relația:

$$\begin{aligned} \hat{y}(t) = & y(t) - [-y(t-1), \dots, -y(t-n_a); u(t-1), \dots, u(t-n_b); \varepsilon(t-1), \\ & \dots, \varepsilon(t-n_c)] \hat{\theta}(t-1) \end{aligned} \quad (3.20)$$

Ecuația este iterată o singură dată pentru fiecare set de măsurări.

Pe durata efectuării calculelor polinomul $\hat{C}(z)$ poate deveni instabil, ceea ce va conduce la valori mari pentru $\varepsilon(t)$ și $\varphi(t)$, și deci la neconvergența estimărilor parametrilor modelului. Depășirea acestei dificultăți se poate face prin reducerea termenului de corecție $K(t)\varepsilon(t)$ în (3.1) astfel încât polinomul $\hat{C}(z)$, determinat de $\hat{\theta}(t)$, să fie stabil. Această modificare poate fi de asemenea avantajoasă și în cazul metodei RCMMPE.

Pentru calculul elementelor vectorilor $z(t)$ și $\varphi(t)$ în (3.18) se notează:

$$y^F(t) = \frac{1}{\hat{C}(q^{-1})} y(t)$$

rezultând

$$y^F(t) = y(t) - \sum_{i=1}^{n_c} \hat{c}_i y^F(t-i) \quad (3.21)$$

Această ecuație poate fi utilizată pentru generarea, aproximativ simplu, a întreg vectorului $\varphi(t)$. În calculul mărimii $y^F(t)$, în relația (3.21) se utilizează $\hat{\theta}(t-1)$. Variante aproximative de calcul al erorii de predicție $\varepsilon(t)$ și a vectorului $\varphi(t)$ sînt prezentate de Söderström (1973).

O comparație a metodelor RCMMPE și RVM scoate în evidență multe similități între cele două metode. Diferența constă în faptul că în cazul metodei RVM, în vectorul $\varphi(t)$, se utilizează date filtrate. Utilizarea filtrului $1/\hat{C}(q^{-1})$ derivă din relația $\varphi(t)^T = d\varepsilon(t)/d\theta$, care se dovedește foarte importantă pentru proprietățile de convergență ale algoritmului.

Metoda poate fi ușor aplicată și pentru alte structuri ale modelului. În această situație, algoritmul de bază rămîne neschimbat, dar trebuie utilizate alte relații de calcul pentru mărimile $\varepsilon(t)$ și $\varphi(t)$.

Versiuni ale metodei aproximării stochastice. Algoritmii prezentați anterior se simplifică considerabil prin substituirea matricei $P(t)$ cu un scalar $p(t)$, de exemplu c/t sau $1/\text{tr } P(t)^{-1}$, rezultînd algoritmi de forma:

$$\hat{\theta}(t) = \hat{\theta}(t-1) + p(t) z(t) \varepsilon(t) \quad (3.22)$$

În această situație se obține o reducere considerabilă a efortului de calcul, dar convergența algoritmilor rezultați va fi mai redusă decît în cazul algoritmilor originali.

Algoritmii de tipul (3.22) nu vor fi considerați explicit în cadrul acestei lucrări. Rezultatele analizei acestor algoritmi sînt similare rezultatelor obținute pentru algoritmi originali.

3.2. Modificări ale algoritmilor

Versiuni în timp real. În multe situații este important ca valorile parametrilor modelului să fie estimate în timp real. În general, în literatură, sînt prezentate două soluții de obținere a versiunilor în timp real, pentru algoritmi de estimare recursivă descriși de relațiile (3.1) — (3.3).

O primă soluție (Wieslander, 1969) constă în introducerea unui factor de ponderare λ a erorii de predicție $\varepsilon(t)$ în expresia criteriului minimizat, rezultînd în final următoarele relații de calcul pentru vectorul parametrilor estimați :

$$\hat{\theta}(t) = \hat{\theta}(t-1) + K(t) \cdot \varepsilon(t) \quad (3.23)$$

$$K(t) = \frac{\lambda P(t-1) z(t)}{\lambda + \varphi(t)^T P(t-1) z(t)} \quad (3.24)$$

$$P(t) = \left[P(t-1) - \frac{P(t-1) z(t) \varphi(t)^T P(t-1)}{\lambda + \varphi(t)^T P(t-1) z(t)} \right] / \lambda \quad (3.25)$$

Soluția propusă realizează o reducere a influenței reziduurilor „mai vechi” asupra valorii parametrilor estimați (pentru RCMMP criteriul de minimizat devine $\sum_{s=1}^t \lambda^{-s} \varepsilon(s)^2$), prin alegerea factorului de ponderare λ subunitar.

O a doua soluție (Bohlin, 1968) face uz de interpretarea de filtru Kalman, dată metodei RCMMP. Incluziunea zgomotului procesului în model conduce la următorul algoritm :

$$\hat{\theta}(t) = \hat{\theta}(t-1) + K(t) \varepsilon(t) \quad (3.26)$$

$$K(t) = \frac{P(t-1) z(t)}{1 + \varphi(t)^T P(t-1) z(t)} \quad (3.27)$$

$$P(t) = P(t-1) - \frac{P(t-1) z(t) \varphi(t)^T P(t-1)}{1 + \varphi(t)^T P(t-1) z(t)} + R1 \quad (3.28)$$

unde $R1$ este o matrice (semi)pozitiv definită.

Cele două extensii au proprietatea că $P(t)$ (deci și factorul de amplificare $K(t)$) nu tinde la zero cînd $t \rightarrow \infty$.

Îmbunătățirea vitezei de convergență. Creșterea vitezei de convergență a algoritmilor de estimare recursivă se poate obține prin utilizarea următoarei versiuni a algoritmului general (Furth, 1973 ; Talman, van den Boom, 1973) :

$$\hat{\theta}(t) = \hat{\theta}(t-1) + K(t) \varepsilon(t) \quad (3.29)$$

$$K(t) = \frac{P(t-1) z(t)}{\lambda(t) + \varphi(t)^T P(t-1) z(t)} \quad (3.30)$$

$$P(t) = \left[P(t-1) - \frac{P(t-1)z(t)\varphi(t)^T P(t-1)}{\lambda(t) + \varphi(t)^T P(t-1)z(t)} \right] / \lambda(t) \quad (3.31)$$

cu $\lambda(t)$ generat de ecuația cu diferențe de ordinul întâi:

$$\lambda(t) = \lambda_0 \lambda(t-1) + (1 - \lambda_0), \quad \lambda_0 \text{ și } \lambda(0) \text{ ușor subunitari} \quad (3.32)$$

Ecuația (3.32) indică faptul că $\lambda(t)$ tinde exponențial la 1 când t tinde la infinit.

Algoritmul propus apare ca o generalizare a algoritmului de estimare în timp real (3.23)—(3.25). Dependența de timp a factorului de ponderare $\lambda(t)$ va avea ca efect reducerea influenței primelor valori estimate ale parametrilor modelului pe măsură ce diferența dintre valorile parametrilor estimați și θ_0 se reduce. Factorul de ponderare, spre deosebire de cel din cazul versiunii în timp real a algoritmului, tinde la 1 odată cu creșterea timpului, ceea ce va conduce la obținerea unei precizii mai bune a valorilor parametrilor estimați și în consecință la îmbunătățirea convergenței.

În cazul metodei RCMMP, algoritmul de estimare realizează minimul următoarei funcții obiectiv:

$$\sum_{k=1}^t \left[\prod_{i=k}^t \lambda(i) \right] \varepsilon(k)^2 \quad (3.33)$$

Se observă clar profilul de „uitare” al datelor vechi, comparativ cu „uitarea” exponențială din cazul versiunii în timp real a algoritmului. Relația (3.32) asigură următorul coeficient de uitare:

$$\lambda(t) = \lambda_0^t \lambda(0) + (1 - \lambda_0^t) = 1 - \lambda_0^t (1 - \lambda(0))$$

Algoritmul recursiv general poate fi rescris sub o altă formă, utilă considerațiilor ce se vor face în cadrul secțiunilor viitoare.

Pentru o secvență dată $\{\lambda(t)\}$, care poate să nu verifice relația (3.32), se definește în mod recursiv următoarea secvență:

$$\gamma(t) = \frac{\gamma(t-1)}{\lambda(t) + \gamma(t-1)}; \quad \gamma(0) = 1 \quad (3.34)$$

Atunci

$$\lambda(t) = \frac{\gamma(t-1)}{\gamma(t)} (1 - \gamma(t)) \quad (3.55)$$

Se introduc următoarele mărimi:

$$P^*(t) = \frac{1}{\gamma(t)} P(t) \quad K^*(t) = \frac{1}{\gamma(t)} K(t)$$

Cu aceste notații ecuațiile (3.29)—(3.31) devin:

$$\hat{\theta}(t) = \hat{\theta}(t-1) + \gamma(t) K^*(t) \varepsilon(t) \quad (3.36)$$

$$K^*(t) = \frac{P^*(t-1)z(t)}{1 + \gamma(t)[\varphi(t)^T P^*(t-1)z(t) - 1]} \quad (3.37)$$

$$P^*(t)^{-1} = P^*(t-1)^{-1} + \gamma(t)[z(t)\varphi(t)^T - P^*(t-1)^{-1}] \quad (3.38)$$

sau

$$P^*(t) = \frac{1}{1 - \gamma(t)} P^*(t-1) - \frac{\gamma(t)}{1 - \gamma(t)} \frac{P^*(t-1)z(t)\varphi(t)^T P^*(t-1)}{1 + \gamma(t)[\varphi(t)^T P^*(t-1)z(t) - 1]}$$

Această descriere poate fi utilă, deoarece P^* și K^* nu tind la 0 când $t \rightarrow \infty$ ci către limite ce pot avea interpretare fizică.

Algoritmi de tipul celui prezentat sînt studiați de Ljung (1974). În aceeași lucrare este discutată și alegerea secvenței $\gamma(t)$, în particular fiind prezentate motivele pentru care secvențele lent descrescătoare $\{\gamma(t)\}$ (corespunzătoare factorului $\lambda(t) < 1$) sînt mai indicate decît $\gamma(t) = 1/t$.

4. Proprietăți de convergență

Analiza directă a proprietăților de convergență a algoritmilor de estimare recursivă este dificil de realizat. Unul din motive îl constituie faptul că algoritmul (3.1)—(3.3) reprezintă o ecuație cu diferențe stohastică, neliniară, variantă în timp. Mai mult, în general $\varepsilon(t)$ în (3.1) depinde implicit de toate valorile parametrilor estimați anterior $\hat{\theta}(k)$, $k < t$. Toate acestea fac relativ dificilă tratarea analitică directă.

O soluție posibilă de studiere a diferitelor metode de estimare recursivă o reprezintă fără îndoială simularea. Totuși, adesea este destul de dificil să se elaboreze o concluzie asupra proprietăților algoritmilor studiați numai prin simulare. Rezultatele simulării sînt influențate de mai mulți factori ca, de exemplu, de realizarea proceselor alcătuitoare, de alegerea exemplelor etc.

În cadrul acestei secțiuni se face o sinteză a rezultatelor analitice obținute în analiza algoritmilor recursivi, bazate pe tehnica generală de analiză a algoritmilor stochastici recursivi elaborată de Ljung (1977b).

Analiza se bazează pe asocierea algoritmului (3.1)—(3.3), unei ecuații diferențiale deterministe și invariante în timp (EDO), ale cărei proprietăți reflectă proprietățile algoritmului:

$$\frac{d\theta(\tau)}{d\tau} = R^{-1}(\tau) f(\theta(\tau)) \quad (4.1)$$

$$\frac{dR(\tau)}{d\tau} = G(\theta(\tau)) - R(\tau)$$

unde

$$f(\theta) = Ez(t; \theta) \varepsilon(t; \theta) \quad (4.2a)$$

$$G(\theta) = Ez(t; \theta) \varphi(t; \theta)^T \quad (4.2b)$$

E — operatorul de mediere.

În cadrul acestor relații $\varepsilon(t; \theta)$, $\varphi(t; \theta)$ și $z(t; \theta)$ reprezintă procesele staționare care s-ar obține, dacă secvența parametrilor estimați $\{\hat{\theta}(t)\}$ este înlocuită prin vectorul parametrilor constanți θ .

Relațiile dintre algoritmul (3.1)—(3.3), definind metodele de estimare recursivă, și ecuația diferențială (4.1) sînt următoarele (Ljung, 1977b):

— posibilele puncte de convergență ale algoritmului sînt numai punctele staționare stabile ale ecuației (4.1). Aceasta înseamnă, că dacă $\hat{\theta}(t) \rightarrow \theta^*$ cu probabilitate diferită de zero, atunci:

$$f(\theta^*) = 0 \quad (4.3)$$

și matricea

$$G^{-1}(\theta^*) \left. \frac{df}{d\theta}(\theta) \right|_{\theta=\theta^*} \quad (4.4)$$

are toate valorile proprii în semiplanul stîng;

— stabilitatea asimptotică globală a unei soluții $(\theta^*, G(\theta^*))$ a ecuației (4.1) implică $\hat{\theta}(t) \rightarrow \theta^*$ cu probabilitate 1 cînd $t \rightarrow \infty$, pentru algoritmul (3.1)—(3.3);

— traiectoriile ecuației (4.1) pot fi interpretate ca traiectorii asimptotice ale algoritmului de estimare.

În continuare, vom prezenta o sinteză a rezultatelor analizei de convergență a metodelor de estimare recursivă, analiză bazată pe abordarea ce utilizează EDO.

În general se presupune că sistemul și modelul sînt descrise de (2.5) și respectiv (2.1). Interpretarea filtrelor $H(q^{-1})$ și $\hat{H}(q^{-1})$ pentru diferite metode este dată în secțiunea 2. Din motive tehnice se consideră numai punctele pentru care $G(\theta)$ este nesingulară.

În cele ce urmează, pentru diferitele metode de estimare recursivă, se va da un răspuns următoarelor probleme:

- (i) θ_0 este soluția unică a ecuației $f(\theta) = 0$?
- (ii) θ_0 este soluția unică a ecuației $\{f(\theta) = 0, \text{ și } G^{-1}(\theta) f'(\theta) \text{ stabilă}\}$?
- (iii) $\hat{\theta}(t)$ converge către θ_0 ?

în următoarele ipoteze:

- sistemul este identificabil, în sensul că acesta poate fi regăsit în structura modelului considerat;
- ordinele polinoamelor modelului coincid cu cele ale polinoamelor respective în descrierea sistemului (ordinul modelului este egal cu ordinul sistemului real);
- sînt analizate numai punctele pentru care $G(\theta)$ este nesingulară.

Motivul separării punctelor (i) și (ii) sînt următoarele:

- ecuația $f(\theta) = 0$ determină simultan și proprietățile de unicitate ale metodelor de estimare nerecursivă corespunzătoare;
- proprietățile de unicitate ale soluției ecuației $f(\theta) = 0$ sînt esențiale pentru demonstrarea convergenței.

Rezultatele analizei convergenței sînt prezentate în tabelul 1.

Rezultate Me- toda	(i)	(ii)	(iii)
RCMMP	$\theta = \theta_0$ soluția unică a ecuației $f(\theta) = 0$ (Söderström, Ljung, Gustavsson, 1974)	$G^{-1}(\theta_0) f'(\theta_0)$ stabilă (Söderström, Ljung, Gustavsson, 1974)	$\hat{\theta}(t) \rightarrow \theta_0$ c.p.1 (Söderström, Ljung, Gustavsson, 1974)
RVI	$\theta = \theta_0$ soluția unică a ecuației $f(\theta) = 0$ (Procesul operează în buclă deschisă sau $H(q^{-1}) \equiv 1$) (Söderström, Ljung, Gustavsson, 1974)	$G^{-1}(\theta_0) f'(\theta_0)$ stabilă (Procesul operează în buclă deschisă sau $H(q^{-1}) \equiv 1$) (Söderström, Ljung, Gustavsson, 1974)	$\hat{\theta}(t) \rightarrow \theta_0$ c.p.1 pentru $z(t)$ generat de (3.9), (3.10); cazul (3.9), (3.11) rămâne deschis (Procesul operează în buclă deschisă sau $H(q^{-1}) \equiv 1$) (Söderström, Ljung, Gustavsson, 1974)
RCMMPG	$\theta = \theta_0$ soluție unică a ecuației $f_1(\theta) = 0$ ($i=1, 2$) pentru raportul semnal/zgomot foarte mare; în caz contrar există și alte soluții (Procesul operează în buclă deschisă) (Söderström, 1974b)	Sistemul linear obținut prin liniarizarea ecuației (4.1) este asimptotic stabil dacă liniarizarea este făcută în jurul punctului θ_0 (valabil și în cazul operării în buclă închisă a sistemului). Există cazuri unde de asemenea și liniarizarea în jurul altor soluții $f_1(\theta) = 0$ ($i=1, 2$) furnizează un sistem asimptotic stabil. (Söderström, Ljung, Gustavsson, 1974)	$\hat{\theta}(t) \rightarrow \theta$ c.p.1. dacă reacția nu este adaptivă și $f_1(\theta) = 0$ ($i=1, 2$) are soluție unică $\theta = \theta_0$ (Söderström, Ljung, Gustavsson, 1974)
RCMMPPE	$\theta = \theta_0$ soluție unică a ecuației $f(\theta) = 0$ dacă este îndeplinită una din condițiile: a) Sistemul este un proces ARMA $A(q^{-1})y(t) = B(q^{-1})e(t)$ (Söderström, Ljung, Gustavsson, 1974) b) Polinomul $C(q^{-1})e(t)$ (2.5), (2.8) este real	a) Pentru un proces ARMA, valorile proprii ale matricei $G^{-1}(\theta_0) f'(\theta_0)$ sînt -1 (ordin de multiplicitate n_0) și $-\frac{1}{C(\lambda_i^{-1})}$ $i=1, \dots, \dots, n_s$ unde $\{\lambda_i\}$ sînt rădăcinile ecuației $A(z) = 0$ (Prin urmare există procese ARMA pentru care această matrice este instabilă) (Holst, 1977)	$\hat{\theta} \rightarrow \theta_0$ c.p.1 dacă $\text{Re} \left[\frac{1}{C(e^{i\omega})} - \frac{1}{2} \right] > 0$ $-\pi \leq \omega \leq \pi$ (Ljung 1977 a)

Tabelul 1 (continuare)

Rezultate Metoda	(i)	(ii)	(iii)
	<p>pozitiv $\operatorname{Re} C(e^{j\omega}) > 0,$ $-\pi \leq \omega \leq \pi$ (Ljung 1977a)</p>	<p>b) Similar există procese a căror intrare este astfel încît $G^{-1}(\theta_0) f'(\theta_0)$ este instabilă (Ljung, Söderström, Gustavsson, 1975) c) Metoda propusă de Young (1976) are proprietăți de convergență similare. Astfel pentru procese ARMA valorile proprii ale matricei $G^{-1}(\theta_0) f'(\theta_0)$ sînt $-1/C(\mu_i^{-1})$ $i=1, \dots, \dots, n_s$ unde $\{\mu_i\}$ sînt rădăcinile ecuației $C(z)=0$ și $-1/C(\lambda_i^{-1})$ $i=1, \dots, \dots, n_s$ unde $\{\lambda_i\}$ sînt rădăcinile ecuației $A(z)=0$ (Holst 1977)</p>	
RVM	<p>$\theta = \theta_0$ soluție unică a ecuației $f(\theta) = 0$ dacă sistemul este un proces ARMA (Aström, Söderström, 1974)</p>	<p>$G^{-1}(\theta_0) f'(\theta_0)$ întotdeauna asimptotic stabilă (Söderström, Ljung, Gustavsson, 1974) Rezultatele valabile și pentru structuri generale ale modelului.</p>	<p>$\hat{\theta} \rightarrow \theta_0$ c.p.l. dacă reacția este neadaptivă și θ_0 este soluția unică a ecuației $f(\theta) = 0$, $\hat{\theta}(t)$ converge întotdeauna către un maxim local al funcției de verosimilitate logaritmică. (Söderström, Ljung, Gustavsson, 1974) Rezultatele valabile și pentru structuri generale ale modelului.</p>

5. Aspecte de implementare

Cu toate mărimile definite în (3.29)–(3.32) sau (3.34)–(3.38), calculele ce trebuie efectuate pentru estimarea parametrilor modelului sînt unic definite. Totuși, există mai multe variante, echivalente din punct de vedere algebric, pentru efectuarea aceluiași calcule. Diferitele variante de organizare a calcule-

lor pot avea o influență esențială asupra proprietăților numerice ale algoritmului de estimare și, în special, asupra :

- timpului de calcul necesar efectuării unei iterații, conform relațiilor de estimare recursivă ;
- necesarului de memorie calculator ;
- preciziei și stabilității numerice a calculelor (propagării erorilor) ;
- efortului de programare pentru implementarea algoritmului.

Algoritmul de estimare constă în principal din trei părți. Prima parte, calculul erorii $\varepsilon(t)$, este banală. A doua parte, calculul vectorului de amplificare $K(t)$, este cea mai interesantă din punct de vedere al calculului numeric. Partea a treia a algoritmului, calculul vectorului parametrilor estimați $\theta(t)$, este directă, odată ce a fost determinat vectorul $K(t)$.

Aspectele de implementare ce se prezintă se vor referi, în special, la metodele de estimare ce utilizează algoritmi RCMMP, RCMMPPE și RVM ($z(t) = \varphi(t)$).

5.1. Calculul vectorului de amplificare

Calculul vectorului de amplificare pentru algoritmi menționați anterior se efectuează cu relațiile :

$$P(t) = [P(t-1) - P(t-1) \varphi(t) S^{-1}(t) \varphi^T(t) P(t-1)] / \lambda(t) \quad (5.1a)$$

$$S(t) = \varphi^T(t) P(t-1) \varphi(t) + \lambda(t) \quad (5.1b)$$

$$K(t) = P(t-1) \varphi(t) S^{-1}(t) \quad (5.1c)$$

$$\text{sau } R(t) = R(t-1) + \gamma(t) [\varphi(t) \varphi^T(t) - R(t-1)] \quad (5.2a)$$

$$K(t) = \gamma(t) R^{-1}(t) \varphi(t) \quad (5.2b)$$

cînd se lucrează cu inversa matricii de covarianță a vectorului parametrilor :

$$R(t) = \gamma(t) P^{-1}(t) \quad (5.3)$$

Factorul de „uitare“ $\lambda(t)$ este legat de secvența $\gamma(t)$ prin relația (3.35)

$$\lambda(t) = \frac{\gamma(t-1)}{\gamma(t)} [1 - \gamma(t)] \quad (5.4)$$

Trecerea de la o reprezentare la alta se realizează utilizînd lema de inversare.

Expresiile de mai sus apar în toți algoritmi. Acestea nu sînt influențate de structura modelului, ci numai de modul în care $\varphi(t)$ depinde de datele de măsurare : $u(t)$, $y(t)$.

În cadrul acestei secțiuni sînt analizate cîteva variante de calcul al vectorului $K(t)$. Varianta (5.2) nu este adecvată implementării, deoarece (5.2a) conține matricea $R(t)$ a cărei inversă este necesară în relația (5.2b). Dacă (5.2) se implementează în mod direct, la fiecare pas trebuie rezolvat un sistem de ecuații liniare, ceea ce nu apare necesar în cazul implementării relației (5.1).

În continuare vor fi prezentate alte variante de calcul pentru vectorul de amplificare $K(t)$, care utilizează :

— *Tehnici de factorizare.* Acestea reprezintă alternative de calcul la utilizarea directă a ecuației de diferențe (5.1). În acest caz matricea $P(t)$ este exprimată ca un produs de factori matrice, efectuându-se actualizarea acestor factori, soluție ce conferă proprietăți numerice superioare metodei de estimare.

— *Algoritmi rapizi.* În cazul în care se urmărește numai calculul vectorului $K(t)$, poate fi evitat calculul matricei $P(t)$ sau $R(t)$. Această idee stă la baza algoritmilor rapizi, pentru care efortul de calcul și necesarul de memorie se reduc semnificativ, în special pentru structuri de model cu mulți parametri independenți.

Algoritmii de estimare recursivă sînt strîns legați de filtrul Kalman. De fapt, multe din rezultatele ce se prezintă în cadrul acestei secțiuni au fost în primul rînd obținute pentru filtrele Kalman. Deoarece (5.1), (5.2) sînt identice din punct de vedere algebric cu ecuațiile factorului de amplificare și de covarianță ale filtrului Kalman, aceste rezultate pot și vor fi aplicate în cadrul acestei secțiuni.

Utilizarea directă a relației de recurență (5.1a) nu este indicată din punct de vedere numeric, fiind sensibilă la erori de rotunjire. Acest fenomen este analizat de Bierman (1977.) Proprietățile numerice ale ecuației (5.1a) sînt afectate în special pentru $\lambda(t) \equiv 1$. În acest caz, noua matrice $P(t)$ este calculată prin scăderi succesive ale termenilor de corecție. Dacă termenii de corecție devin prea mari, datorită erorilor de rotunjire, pot apare probleme de natură numerică.

Ecuația (5.1a) poate fi rescrisă într-o formă care evident are proprietăți numerice mai bune. Astfel (5.1a) este echivalentă cu :

$$P(t) = [I - K(t)\varphi(t)^T] P(t-1) [I - \varphi(t)K(t)^T] / \lambda(t) \quad (5.5)$$

Această expresie este referită (Bierman 1977) ca „ecuația Kalman stabilizată” și evită scăderile repetate. Cu toate acestea, relația de recurență (5.5) nu garantează stabilitatea numerică, deși are proprietăți numerice mai bune decît ecuația originală (5.1a). Totuși, implementarea acesteia necesită un efort de calcul sporit.

Cînd dimensiunea matricei P este redusă ($d < 10$), adesea nu apar probleme numerice. Aceste probleme sînt în mod obișnuit asociate unor valori mari ale dimensiunii matricei $P(t)$ și/sau matricilor $P(t)$ prost condiționate, totuși aceasta nu reprezintă un motiv pentru ignorarea problemelor numerice.

Tehnici de factorizare pentru calculul vectorului de amplificare. În cazul utilizării unei tehnici de factorizare pentru calculul vectorului $K(t)$, matricea $P(t)$ din ecuația (5.1a) este descompusă într-un produs de factori matrice, fiecare din aceștia fiind actualizați la un nou moment de timp. Deoarece $P(t)$ este pozitiv definită (neglijînd eventualele efecte ale erorilor de rotunjire), factorizarea poate fi de exemplu de tip Cholesky.

În cadrul acestei subsecțiuni, pentru început, se prezintă un algoritm de tip rădăcină pătrată, care realizează următoarea factorizare a matricei $P(t)$:

$$P(t) = Q(t) Q^T(t) \quad (5.6)$$

unde $Q(t)$ este o matrice neregulară. În acest caz relația (5.1a) va fi înlocuită printr-una de calcul al matricei $Q(t)$ din $Q(t-1)$.

Este de asemenea posibil să se efectueze factorizarea (5.6) cu $Q(t)$ matrice triunghiulară. (Bierman 1977), factorizarea fiind de tip Cholesky. În acest caz

algoritmul de actualizare a matricei $Q(t)$ necesită mai multe calcule, dar este relativ simplu.

O altă metodă de factorizare frecvent utilizată este factorizarea $U-D$, care poate fi caracterizată drept o factorizare Cholesky normalizată. În acest caz :

$$P(t) = U(t) D(t) (U(t))^T \quad (5.7)$$

unde $U(t)$ este o matrice triunghiulară cu elementele de pe diagonală unitate, iar $D(t)$ este o matrice diagonală.

Utilizarea tehnicilor de factorizare, în calculul matricei $P(t)$, asigură faptul că aceasta rămâne tot timpul pozitiv definită. De asemenea, experimentele intensive în cazul filtrului Kalman au scos în evidență faptul că metodele de factorizare menționate au o bună stabilitate numerică, erorile de rotunjire neafectând în mod semnificativ soluția (Bierman, 1977 ; Thornton și Bierman, 1978).

În continuare sînt prezentați algoritmul de tip rădăcină pătrată, în variantele propuse de Potter (1963) și Bierman (1977), și algoritmul de tip factorizare $U-D$ propus de Bierman (1977).

Algoritmul rădăcină pătrată (Potter, 1963)

Algoritmul se bazează pe factorizarea (5.6), actualizarea matricii $Q(t)$ făcîndu-se astfel :

$$\begin{aligned} 1. f(t) &:= Q^T(t-1) \varphi(t) \\ 2. \pi(t) &:= \lambda(t) + f^T(t) f(t) \\ 3. \alpha(t) &:= 1/(\pi(t) + (\pi(t) \lambda(t))^{1/2}) \\ 4. \bar{K}(t) &:= Q(t-1) f(t) \\ 5. Q(t) &:= [Q(t-1) - \alpha(t) \bar{K}(t) f^T(t)] / \lambda(t)^{1/2} \end{aligned} \quad (5.8)$$

Vectorul $\bar{K}(t)$ reprezintă forma normalizată a vectorului de amplificare $K(t)$:

$$K(t) = \bar{K}(t) / \pi(t)$$

Actualizarea parametrilor modelului $\hat{\theta}(t)$ nu necesită calculul explicit al vectorului $K(t)$:

$$\hat{\theta}(t) = \hat{\theta}(t-1) + \bar{K}(t) [\varepsilon(t) / \pi(t)] \quad (5.9)$$

În cazul estimării parametrilor în timp real, conform celor menționate în cadrul secțiunii 3.2, se pot utiliza următoarele ecuații, în locul relațiilor (5.1a, 5.1b)

$$\tilde{P}(t) = P(t-1) - \frac{P(t-1) \varphi(t) \varphi^T(t) P(t-1)}{1 + \varphi(t) P(t-1) \varphi(t)} \quad (5.10a)$$

$$P(t) = \tilde{P}(t) + R_1(t) \quad (5.10b)$$

Algoritmul (5.8) poate fi utilizat, de asemenea, pentru determinarea matricii $\tilde{P}(t) = \tilde{Q}(t) \tilde{Q}^T(t)$ din $P(t-1) = Q(t-1) Q^T(t-1)$. Singura problemă

de rezolvat este de a determina matricea $Q(t)$ folosind (5.10b). Una din soluții se prezintă în continuare. Fie $R_1(t)$ factorizat astfel:

$$R_1(t) = V(t) V^T(t) \quad (5.11)$$

unde $V(t)$ este o matrice rectangulară de rang complet.

În cele mai multe dintre cazuri, $R_1(t)$ este o matrice diagonală, în care unele elemente de pe diagonală sînt egale cu 0, factorul $V(t)$ fiind ușor de determinat. În continuare se aplică transformări ortogonale matricei rectangulare $[\tilde{Q}(t) V(t)]$, problema constînd în a determina o matrice ortogonală $T(t)$ și o matrice triunghiulară $Q(t)$ astfel încît:

$$[\tilde{Q}(t) V(t)] T(t) = [Q(t) 0] \quad (5.12)$$

Atunci rezultă:

$$\begin{aligned} \tilde{P}(t) + R_1(t) &= \tilde{Q}(t) \tilde{Q}^T(t) + V(t) V^T(t) = [\tilde{Q}(t) V(t)] T(t) T^T(t) \begin{bmatrix} \tilde{Q}^T(t) \\ V^T(t) \end{bmatrix} = \\ &= [Q(t) 0] \begin{bmatrix} Q^T(t) \\ 0 \end{bmatrix} = P(t) \end{aligned}$$

așa cum se specifică prin (5.10b).

Matricile $T(t)$ și $Q(t)$ în (5.12) pot fi determinate folosind, de exemplu, factorizarea QR sau procedura de ortogonalizare Gram-Schmidt. Astfel de factorizări apar în algebra liniară numerică pentru rezolvarea problemelor de valori proprii și a celor mai mici pătrate (Stewart, 1973), programele aferente fiind incluse în pachetul LINPACK (Dongarra, Moler, Bunch și Stewart, 1979).

Algoritmul rădăcină pătrată (Bierman, 1977)

Algoritmul factorizează matricea $P(t) = Q(t) Q(t)^T$, cu $Q(t)$ matrice superior triunghiulară, incluzînd următoarele etape de calcul:

1. Se calculează: $f = Q^T(t-1) \varphi(t)$, $\beta_0 = \lambda(t)$
2. Pentru $j=1, \dots, n$ se execută pașii 3—5
3. Se calculează $\beta_j = \beta_{j-1} + f_j^2$

$$\eta = (\beta_{j-1} / \beta_j)^{1/2}$$

$$v = f_j / (\eta \cdot \beta_j) \quad (5.13)$$

$$v_j = f_j \cdot Q_{11}(t-1)$$

$$Q_{1j}(t) = Q_{1j}(t-1) \cdot \eta$$
4. Pentru $i=1, \dots, j-1$ se execută pasul 5 (pentru $j=1$ nu se execută pasul 5)
5. Se calculează: $Q_{1j}(t) = Q_{1j}(t-1) - v_i \cdot v$

$$v_i = v_i + f_j Q_{ij}(t-1)$$
6. $\bar{K} = v$

Vectorul $\bar{K}(t)$ reprezintă forma normalizată a vectorului de amplificare $K(t)$:

$$K(t) = \bar{K}(t) / \beta_n$$

Și pentru acest algoritm se poate obține o variantă adecvată estimării în timp real a parametrilor.

Algoritmul de factorizare $U-D$ (Bierman 1977)

Algoritmul utilizează factorizarea (5.7) și include următoarele etape de calcul:

1. Se calculează: $f = U^T(t-1) \varphi(t)$, $g = D(t-1)f$, $\beta_0 = \lambda(t)$

2. Pentru $j=1, \dots, n$ se execută pașii 3—5

3. Se calculează $\beta_j = \beta_{j-1} + f_j g_j$

$$D(t)_{jj} = \beta_{j-1} D(t-1)_{jj} / \beta_j \lambda(t) \quad (5.14)$$

$$v_j = g_j$$

$$\mu = -f_j / \beta_{j-1}$$

4. Pentru $i=1, \dots, j-1$ se execută pasul 5 (pentru $j=1$ nu se execută pasul 5)

5. Se calculează: $U(t)_{ij} = U(t-1)_{ij} + v_j \cdot \mu$

$$v_i = v_i + U(t-1)_{ij} v_j$$

6. $\bar{K}(t) = v$

Scalarul β_n obținut după cel de al n -lea ciclu de execuție a pașilor 3—5 reprezintă dispersia inovației:

$$\beta_n = \lambda(t) + \varphi^T(t) P(t-1) \varphi(t)$$

Ca și în cazul precedent, se obține factorul de amplificare normalizat $\bar{K}(t)$ (pasul 6 după n cicluri de execuție a pașilor 3—5).

De asemenea, și în acest caz se poate obține o variantă a algoritmului, adecvată estimării în timp real a parametrilor. Algoritmul ce se prezintă a fost propus de Thornton și Bierman (1978) și utilizează ortogonalizarea Gram-Schmidt.

Presupunem că factorizarea matricei $R_1(t)$ este disponibilă. Se construiește o matrice W ale cărei linii sînt vectorii $W_1^T \dots W_t^T$, astfel încît:

$$W = \begin{bmatrix} W_1^T \\ \vdots \\ W_n^T \end{bmatrix} = [\tilde{U}(t) \quad V(t)] \text{ și se alege } D = \begin{bmatrix} \hat{D}(t) & 0 \\ 0 & I \end{bmatrix}$$

matricea $\tilde{P}(t)$ din (5.10) fiind factorizată astfel:

$$\tilde{P}(t) = \tilde{U}(t) \tilde{D}(t) \tilde{U}^T(t)$$

Atunci factorizarea U—D a matricei P (t), conform relației (5.10b), se poate scrie :

$$P(t) = U(t) D(t) U^T(t) = \tilde{U}(t) \tilde{D}(t) \tilde{U}^T(t) + V(t) V^T(t) = W D W^T$$

Vom încerca acum să determinăm matricile U (t) și \tilde{W} astfel încît U (t) să fie matrice superior triunghiulară cu elemente unitate pe diagonală și

$$W = U(t) \tilde{W} \\ \tilde{W} D \tilde{W}^T = D_s(t)$$

Problema poate fi rezolvată aplicînd procedura de ortogonalizare Gram-Schmidt vectorilor $W_1 \dots W_n$ folosind produsul scalar

$$\langle W_i, W_j \rangle = W_i^T D W_j$$

Algoritmul final de actualizare în timp real al factorilor U—D pentru (5.10b) este următorul :

1. Se aleg $W_j^{(n)} := W_j$ $j=1 \dots n$
2. Pentru $j=n, n-1, \dots, 2$ se execută pașii 3—5
3. Se calculează

$$D(t)_{jj} := W_j^{T(n-j)} D W_j^{(n-j)}$$

4. Pentru $i=1, 2, \dots, j-1$ se execută pasul 5 (5.15)
5. Se calculează :

$$U(t)_{ij} := W_i^{T(n-j)} D W_j^{(n-j)} / D(t)_{jj}$$

$$W_i^{(n-j+1)} = W_i^{(n-j)} - U(t)_{ij} W_j^{(n-j)}$$

6. Se calculează :

$$D(t)_{ii} = W_i^{T(n-1)} D W_i^{(n-1)}$$

Complexitatea calculului

O evaluare a numărului de operații aritmetice necesare pentru actualizarea matricei P (t) în cadrul algoritmilor de factorizare prezentați este dată în tabelul 2 (Bierman, 1977).

Tabelul 2

Metoda	Numărul de operații aritmetice			
	Adunări	Înmulțiri	Împărțiri	Extragerea rădăcină pătrată
Ecuția Kalman convențională	$1,5d^2 + 3,5d$	$1,5d^2 + 4,5d$	1	0
Ecuția Kalman stabilizată	$4,5d^2 + 5,5d$	$4d^2 + 7,5d$	1	0
Algoritmul de factorizare rădăcină pătrată (Potter)	$3d^2 + 3d$	$3d^2 + 4d$	2	1
Algoritmul de factorizare rădăcină pătrată (Bierman)	$1,5d^2 + 3,5d$	$2d^2 + 5d$	2	1
Algoritmul de factorizare U—D	$1,5d^2 + 1,5d$	$1,5d^2 + 5,5d$	d	0

Din acest tabel rezultă că algoritmi de factorizare U—D necesită, aproximativ, același efort de calcul ca ecuațiile Kalman convenționale. Actualizarea matricii de covarianță reprezintă numai o etapă în cadrul algoritmului de estimare, numărul operațiilor necesare pentru calculul vectorului parametrilor fiind mult mai mare decât cel reprezentat în tabelul 2. Calculele necesită determinarea erorii de predicție $\varepsilon(t)$ și construcția vectorului $\varphi(t)$, care depind în mare măsură de structura de model utilizată.

5.2. Algoritmi rapizi pentru calculul vectorului de amplificare

Algoritmii de calcul al vectorului de amplificare $K(t)$ prezentați în secțiunea anterioară sînt valabili pentru orice structură a vectorului $\varphi(t)$. Algoritmii din cadrul acestei secțiuni utilizează o structură specifică a vectorului $\varphi(t)$, care se regăsește în cadrul celor mai multe dintre tipurile de modele utilizate.

În cele ce urmează se definește această structură și modul în care aceasta poate fi utilizată pentru calculul rapid al vectorului $K(t)$. În acest caz, efortul de calcul, pe pas de estimare, devine proporțional cu d față de cel necesar în cazul convențional, proporțional cu d^2 (d reprezintă dimensiunea vectorului parametrilor modelului).

Structura de deplasare (shift). În cele mai multe dintre cazuri $\{\varphi(t)\}$ are o structură particulară, în sensul că $\varphi(t)$ și $\varphi(t+1)$ sînt dependente. Structura ce va fi presupusă în continuare, pentru $\varphi(t)$, este următoarea :

$$\varphi(t) = \begin{bmatrix} x(t-1) \\ \vdots \\ x(t-n) \end{bmatrix} \quad (5.16)$$

unde $x(\cdot)$ reprezintă un vector coloană cu α elemente. În consecință, dimensiunea vectorului $\varphi(t)$ este $\alpha n = d$.

Caracteristica cea mai importantă a structurii vectorului $\varphi(t)$ o reprezintă faptul că $\varphi(t+1)$ și $\varphi(t)$ au un număr de elemente comune : $x(s)$; $t-n+1 \leq s \leq t-1$.

Introducînd vectorul :

$$\varphi^*(t) = \begin{bmatrix} x(t) \\ \varphi(t) \end{bmatrix} \quad (5.17)$$

aceasta poate fi exprimat astfel :

$$\varphi^*(t) = \begin{bmatrix} x(t) \\ \varphi(t) \end{bmatrix} = \begin{bmatrix} \varphi(t+1) \\ x(t-n) \end{bmatrix} \quad (5.18)$$

Se poate, de asemenea, presupune o structură ceva mai generală astfel încît :

$$\begin{bmatrix} x(t) \\ \varphi(t) \end{bmatrix} = S_F \varphi^*(t) ; \quad \begin{bmatrix} \varphi(t+1) \\ \tilde{x}(t-n) \end{bmatrix} = S_B \varphi^*(t) \quad (5.19)$$

unde S_F și S_B sînt matrici de permutare, iar $x(t)$ și $\tilde{x}(t)$ nu sînt în mod necesar aceeași. Evident (5.18) este un caz special pentru (5.19). Totuși, din considerente de simplitate a scrierii, în cele ce urmează, se va trata cazul (5.18).

Calculul rapid al vectorului de amplificare. În cadrul acestei secțiuni se prezintă algoritmul de calcul al vectorului de amplificare $K(t)$ (5.2) (Ljung, Morf și Falconer; 1978) care utilizează structura de deplasare (5.16).

Presupunem următoarea condiție inițială pentru ecuația (5.2) :

$$R(0) = \delta I \text{ cu } \delta \geq 0$$

Introducînd

$$\bar{R}(t) = \frac{1}{\gamma(t)} R(t)$$

ecuația (5.2 a) devine :

$$\bar{R}(t) = \lambda(t) \bar{R}(t-1) + \varphi(t) \varphi^T(t) \quad (5.20)$$

cu $\lambda(t)$ dat de (5.4).

În continuare se va presupune că :

$$\lambda(t) \equiv \lambda, \text{ pentru toate valorile timpului } t. \quad (5.21)$$

Prin urmare

$$\bar{R}(t) = \sum_{k=1}^t \lambda^{t-k} \varphi(k) \varphi^T(k) + \lambda^t \delta \cdot I \quad (5.22)$$

În aceste condiții vectorul de amplificare $K(t)$ reprezintă soluția ecuației :

$$\bar{R}(t) K(t) = \varphi(t) \quad (5.23)$$

cu $\bar{R}(t)$ dat de (5.20)—(5.22).

Ideea de bază a algoritmului de calcul rapid este de a utiliza structura (5.16) sau (5.18) pentru rezolvarea ecuației (5.23). Această structură permite stabilirea unei legături directe între $K(t)$ și $K(t+1)$, așa cum va rezulta în continuare.

Fie $\varphi^*(t)$ definită de (5.17), și :

$$\bar{R}^*(t) = \sum_{k=1}^t \lambda^{t-k} \varphi^*(k) [\varphi^*(k)]^T + \lambda^t \delta \cdot I \quad (5.24)$$

$\bar{R}^*(t)$ este o matrice $(n+1) \times (n+1)$.

Proprietatea (5.18) implică următoarea relație :

$$\bar{R}^*(t) = \begin{bmatrix} * & * & * \\ * & \bar{R}^*(t) & \\ * & & \end{bmatrix} = \begin{bmatrix} * & * \\ * & \bar{R}^*(t+1) & * \\ * & * & * \end{bmatrix} \quad (5.25)$$

unde prin „*“ au fost notate elementele ce nu prezintă interes în momentul de față, iar „lățimea“ acestor elemente este α . Relația (5.25) este valabilă numai dacă $x(t) = 0$ pentru $t \leq 0$. Altfel, efectul termenului $\varphi^T(1) \varphi^T(1)$ nu va fi luat în considerare în cadrul egalității a doua.

Relațiile de definiție a vectorilor $K(t)$ și $K(t+1)$

$$\bar{R}(t) K(t) = \varphi(t)$$

$$\bar{R}(t+1) K(t+1) = \varphi(t+1)$$

pot fi rescrise, utilizând relația (5.25), sub forma :

$$\bar{R}^*(t) \cdot \begin{bmatrix} 0 \\ K(t) \end{bmatrix} = \begin{bmatrix} * \\ \varphi(t) \end{bmatrix} \quad (5.26)$$

$$\bar{R}^*(t) \cdot \begin{bmatrix} K(t+1) \\ 0 \end{bmatrix} = \begin{bmatrix} \varphi(t+1) \\ * \end{bmatrix} \quad (5.27)$$

În acest caz „*“ reprezintă un vector α^*1 , a cărui expresie exactă nu este importantă în definirea vectorului $K(t)$.

În scopul determinării relației dintre $K(t+1)$ și $K(t)$ se introduce o mărime intermediară $K^*(t)$ care se definește astfel

$$\bar{R}^*(t) K^*(t) = \varphi^*(t) \quad (5.28)$$

În continuare se va urmări stabilirea unor relații, între $K(t)$, $K^*(t)$ și respectiv $K^*(t)$, $K(t+1)$.

În scopul stabilirii dependențelor menționate anterior sînt necesare instrumente cu care să se poată opera asupra elementelor „*“ în (5.26)—(5.27). Aceste instrumente sînt matricile $(n \times \alpha)$: $A(t)$ și $B(t)$ care satisfac relațiile :

$$\bar{R}^*(t) \begin{bmatrix} I \\ A(t) \end{bmatrix} = \begin{bmatrix} R^e(t) \\ 0 \end{bmatrix} \updownarrow \begin{matrix} \alpha \\ \text{linii} \end{matrix} \quad (5.29)$$

$$\bar{R}^*(t) \begin{bmatrix} B(t) \\ I \end{bmatrix} = \begin{bmatrix} 0 \\ R^r(t) \end{bmatrix} \updownarrow \begin{matrix} \alpha \\ \text{linii} \end{matrix} \quad (5.30)$$

$R^e(t)$ și $R^r(t)$ sînt matrici $\alpha \times \alpha$.

Formulele pentru actualizarea matricilor $A(t)$, $B(t)$ și a vectorului $K(t)$ (Ljung, Morf și Falconer ; 1978) sînt sintetizate în următorul algoritm de estimare rapidă :

Algoritm de calcul. Fie $\{x(t)\}$ o secvență de vectori α^*1 , astfel încît $x(t) = 0$ pentru $t \leq 0$ și fie :

$$\varphi(t) = \begin{bmatrix} x(t-1) \\ \vdots \\ x(t-n) \end{bmatrix}$$

Atunci vectorul de amplificare :

$$K(t) = \left[\sum_{k=1}^t \lambda^{t-k} \varphi(k) \varphi^T(k) + \lambda^t \delta \cdot I \right]^{-1} \varphi(t)$$

poate fi calculat recursiv astfel :

1. Se inițializează

$$A(0)=0, B(0)=0, R^e(0)=\delta \cdot I, K(1)=0$$

2. Fiind date $A(t-1)$, $B(t-1)$, $R^e(t-1)$, $K(t)$ se actualizează astfel :

$$e(t) := x(t) + A^T(t-1) \varphi(t)$$

$$A(t) := A(t-1) - K(t) e^T(t)$$

$$\beta(t) := K^T(t) \varphi(t)$$

$$\bar{e}(t) := (1 - \beta(t)) e(t)$$

$$R^e(t) := \lambda R^e(t-1) + \bar{e}(t) e^T(t)$$

$$K^*(t) = \begin{bmatrix} [R^e(t)]^{-1} \bar{e}(t) \\ K(t) + A(t) [R^e(t)]^{-1} \bar{e}(t) \end{bmatrix} \quad (5.31)$$

Se partiționează $K^*(t)$ astfel :

$$K^*(t) := \begin{bmatrix} M(t) \\ \mu(t) \end{bmatrix} \begin{matrix} n\alpha \text{ linii} \\ \alpha \text{ linii} \end{matrix}$$

$$r(t) := x(t-n) + B(t-1)^T \varphi(t+1)$$

$$B(t) := [B(t-1) - M(t) r^T(t)] [I - \mu(t) r^T(t)]^{-1}$$

$$K(t+1) := M(t) - B(t) \mu(t)$$

Complexitatea calculului. Algoritmul (5.31) necesită memorarea, la momentul $t-1$, a următoarelor elemente :

$$\varphi(t) \quad : \text{matrice } n\alpha * 1$$

$$K(t) \quad : \text{matrice } n\alpha * 1$$

$$A(t-1) \quad : \text{matrice } n\alpha * \alpha$$

$$B(t-1) \quad : \text{matrice } n\alpha * \alpha$$

$$R^e(t-1) \quad : \text{matrice simetrică } \alpha * \alpha$$

În total acestea utilizează $2n\alpha^2 + 2n\alpha + \frac{1}{2}(\alpha^2 + \alpha)$, sau $2\alpha d + 2d + \frac{1}{2}(\alpha^2 + \alpha)$, $d = 1, \alpha$, locații memorie.

Algoritmii de calcul al vectorului de amplificare, utilizând tehnici de factorizare, necesită memorarea matricei simetrice $P(t)$ (sau a matricei triunghiulare $U(t)$ și a matricei diagonale $D(t)$) împreună cu vectorul $\varphi(t)$. Aceasta înseamnă $\frac{1}{2}(n\alpha)^2 + \frac{3}{2}n\alpha$ locații memorie, deci cu un ordin de mărime (în n) mai mare.

Numărul de operații necesar în cadrul algoritmului (5.31) este de aproximativ $n\alpha^3 + 6n\alpha^2 + n\alpha + \frac{2}{3}\alpha^3 + 4\alpha^2 + 2\alpha$ înmulțiri și $n\alpha^3 + 7n\alpha^2 + 2n\alpha + 7/2\alpha^2 + 3/2\alpha$

adunări fiind cu un ordin de mărime mai mic (în n) decât cel necesar în cadrul algoritmilor de factorizare, conform tabelului 2.

Prin urmare, utilizarea algoritmului (5.31) este avantajoasă, în raport cu metodele convenționale când n este mare: algoritmul este mai rapid și necesită mai puțină memorie calculator.

Stabilitatea numerică. Proprietățile algoritmului (5.31) legate de propagarea erorilor nu sînt încă, în întregime, înțelese și investigate. Este posibil ca reducerea redundanței să conducă la degradarea stabilității numerice a algoritmului. Cele mai multe din studiile publicate indică, totuși, o comportare satisfăcătoare a algoritmului (Ljung, Morf și Falconer, 1978; Robins și Wellstead, 1979). Totuși, ocazional, se observă creșteri exagerate ale valorii elementelor matricei $R^e(t)$ când $\lambda < 1$.

Extensii ale algoritmului. Structura generală de deplasare (5.19) este tratată de Ljung, Morf și Falconer (1978). Această referință tratează de asemenea și metoda nesimetrică, a matricei instrumentale. Cazul în care $\varphi(t)$ este matrice poate fi tratat în același mod cu cazul vectorial, procedura de obținere a algoritmului fiind aceeași.

5.3. Regularizarea algoritmilor

Conform celor prezentate în secțiunea 5.1 factorul de amplificare $K(t)$ se determină cu relațiile:

$$R(t) = R(t-1) + \gamma(t) (\varphi(t) \varphi^T(t) - R(t-1)) \quad (5.32 a)$$

$$K(t) = \gamma(t) R(t-1)^{-1} \varphi(t) \quad (5.32 b)$$

Se poate întîmpla ca matricea $R(t)$, definită de (5.32 a), să devină singulară, sau aproape singulară. În general, această situație apare dacă modelul conține prea mulți parametri sau dacă semnalul de intrare nu este destul de general.

În implementarea algoritmilor de estimare recursivă în cadrul schemelor de comandă adaptivă, funcționînd un timp îndelungat, sau în cadrul aplicațiilor de supraveghere, este important să se aibă în vedere acest gen de probleme numerice, deoarece nu se poate garanta a priori că intrarea rămîne destul de generală.

În consecință, trebuie luate unele măsuri care să asigure că elementele matricei $R(t)$ rămîn mărginite, $|R^{-1}(t)| < C$. O astfel de ipoteză apare în analiza teoretică a algoritmilor (Ljung, Söderström, 1981) sub forma:

$$R(t) \geq \delta I; \quad \delta > 0 \quad (5.33)$$

O tehnică standard de asigurare a condiției (5.3) este *regularizarea*. În cadrul acestei secțiuni se discută cîteva moduri de implementare a problemei de estimare recursivă cu asigurarea condiției (5.33).

Metoda Levenberg-Marquardt. Matricea $R(t)$ în (5.32 a) este prin construcție semipozitiv definită. Deci, obținerea unei relații de tipul (5.33) se poate

realiza simplu prin adăugarea termenului δI în relația (5.32 a), rezultând următorul algoritm (regularizare Levenberg-Marquardt, Marquardt (1963)):

$$R(t) = R(t-1) + \gamma(t) (\varphi(t) \varphi^T(t) - R(t-1)) + \delta I \quad (5.34)$$

Modificarea (5.34) este conceptual simplă. Numărul pozitiv poate fi ales destul de mic ($\delta = 10^{-2} - 10^{-4}$) comparativ cu valoarea elementelor vectorului $\varphi(t)$. Prin urmare direcțiile de căutare obținute din (5.34) vor fi puțin diferite de cele algoritmului inițial.

Există totuși un dezavantaj al implementării relației (5.34). Relația recursivă de calcul a matricii $P(t) = \gamma(t) R^{-1}(t)$, se obține aplicând lema de inversare ecuației (5.32 a). Această relație va conține inversa unei matrici de dimensiune egală cu 1, rangul matricii $\varphi(t) \varphi^T(t)$, rang mult mai mic decât $d = \dim \theta$. În (5.34) această matrice este $\varphi(t) \varphi^T(t) + \delta I$ care are rang d . Deci (5.34) implică inversarea la fiecare pas a unei matrici $d \times d$.

Un posibil remediu este următorul (Ljung, Söderström, 1981). În locul adăugării unei matrici δI , $d \times d$, se adaugă un element diagonal la fiecare pas. Fie $J_d(t)$ o matrice al cărui element diagonal $t \pmod{d} + 1$ este 1, iar celelalte elemente sînt nule. Considerăm atunci algoritmul:

$$R(t) = R(t-1) + \gamma(t) (\varphi(t) \varphi^T(t) + d \delta J_d(t) - R(t-1)) \quad (5.35)$$

Acest algoritm este de fapt identic cu (5.34). După d pași se obține:

$$\sum_{t=k+1}^{k+d} d \delta J_d(t) = d \delta I$$

astfel încît se adaugă un multiplu al matricii unitate pentru a se asigura relația (5.33). Versiunea (5.35) se poate scrie:

$$R(t) = R(t-1) + \gamma(t) (\varphi^* \Lambda^* \varphi^{*T}(t) - R(t-1)) \quad (5.36 a)$$

unde: $\varphi^*(t)$ este o matrice $d \times 2$

$$\varphi^*(t) = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \varphi(t) & 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \rightarrow \text{poziția } t \pmod{d} + 1 \quad (5.36 b)$$

și

$$\Lambda^*(t) = \begin{bmatrix} 1 & 0 \\ 0 & d\delta \end{bmatrix}$$

Lema de inversare aplicată relației (5.35) conduce la următorul algoritm:

$$P(t) = [P(t-1) - P(t-1) \varphi^*(t) S^*(t)^{-1} \varphi^{*T}(t) P(t-1)] / \lambda(t) \quad (5.37 a)$$

$$S^*(t) = \varphi^*(t) P(t-1) \varphi^{*T}(t) + \lambda(t) \Lambda^*(t) \quad (5.37 b)$$

unde:

$$P(t) = \gamma(t) R^{-1}(t)$$

și $\lambda(t)$ dat de (5.4). S^* are dimensiunea 2 pentru adaptarea la modificarea Levenberg-Marquardt. În rest algoritmul pentru determinarea vectorului amplificării rămâne neschimbat.

Regularizarea algoritmului de factorizare U—D. Modificarea Levenberg-Marquardt (5.34) sau (5.37) adaugă la matricea $R(t)$ o matrice constantă pozitiv definită, fie că este necesar sau nu. Totuși, pare mai natural a se efectua o astfel de modificare numai cînd există riscul ca valorile proprii ale matricei $R(t)$ să tindă la zero.

Un astfel de algoritm necesită, în mod aparent, un efort de calcul mai mare decît regularizarea directă.

În cazul algoritmului de factorizare U—D, informația suplimentară necesară noii versiuni de regularizare se obține în mod automat.

Fie :

$$P(t) = U(t) D(t) U^T(t) \quad (5.38)$$

unde $U(t)$ este o matrice triunghiulară normalizată, iar $D(t)$ este o matrice diagonală.

Introducînd limite pentru elementele matricilor $D(t)$ și $U(t)$ putem asigura, în consecință, că matricea $P(t)$ rămîne mărginită și pozitiv semi-definită.

Din considerente practice este suficient, de fapt, să se introducă o limită numai pentru $D(t)$. Justificarea este următoarea :

$$\gamma(t) P^{-1}(t) = R(t) = \sum_{k=1}^t \beta(t, k) \varphi(k) \varphi^T(k) = \gamma(t) U^{-T}(t) D^{-1}(t) U^{-1}(t) \quad (5.39)$$

Presupunem că $\gamma(t)$ nu tinde la zero și că vectorii $\varphi(t)$ nu tind la infinit. Atunci elementele matricei $U^{-1}(t)$ sînt mărginite, ca și elementele matricei $U(t)$ (deoarece $\det U(t) = 1$). Prin urmare, este suficient să se introducă o limită numai pentru elementele lui $D(t)$. Regularizarea algoritmului U—D se obține prin urmare prin modificarea pasului 3 din (5.14), astfel :

$$3. D(t)_{jj} := \min(C, \beta_{j-1} D(t-1)_{jj} / \beta_j \lambda(t)) \quad (5.40)$$

unde C este un număr pozitiv ce mărginește elementele matricei $D(t)$. Acesta corespunde termenului $\gamma(t)/\delta$, cu δ ales conform discuției din secțiunea anterioară.

Regularizarea (5.40) este naturală și ușor de implementat în cadrul algoritmului U—D.

5.4. Teste de stabilitate și algoritmi de proiecție

În demonstrarea proprietăților algoritmilor de estimare recursivă apare destul de des ipoteza că vectorul parametrilor estimați $\theta(t)$ aparține, cel puțin înfinit de des, submulțimii predictorilor stabili. Algoritmul de actualizare a parametrilor (3.29) se poate rescrie astfel :

$$\hat{\theta}(t) = [\hat{\theta}(t-1) + K(t) \varepsilon(t)]_{D_M}$$

realizînd proiecția în submulțimea predictorilor stabili.

De asemenea, în cadrul unor situații practice este necesar ca $\hat{\theta}(t)$ să aparțină submulțimii predictorilor stabili. În consecință se impune ca algoritmi de estimare recursivă să conțină elemente de analiză a stabilității și de obținere a proiecției vectorului parametrilor în regiunea de stabilitate.

Algoritmii de proiecție nu trebuie să fie foarte sofisticăți. Cel mai frecvent facilitarea de proiecție se implementează sub forma unei reduceri succesive a termenului de corecție pentru vectorul parametrilor estimați. Aceasta poate fi realizată în felul următor :

1. Se alege un factor μ , $0 \leq \mu < 1$

2. Se calculează

$$\hat{\theta}(t) := \gamma(t) R^{-1}(t) \varphi(t) \varepsilon(t)$$

3. Se calculează

$$\hat{\theta}(t) := \hat{\theta}(t-1) + \tilde{\theta}(t)$$

4. Se testează dacă $\hat{\theta}(t) \in D_M$

Dacă da se trece la 6, în caz contrar se trece la 5

5. Se calculează

$$\tilde{\theta}(t) := \mu \tilde{\theta}(t) \text{ și se trece la 3}$$

6. Stop.

În etapa 4, testul dacă $\hat{\theta}(t) \in D_M$ constă în a examina dacă anumite polinoame au toate zerourile în interiorul cercului unitate. Aceasta este o problemă clasică în analiza sistemelor liniare discrete, existind algoritmi de efectuare a acestor teste (Kucera, 1979).

În etapa 5, factorul μ realizează reducerea modificării vectorului parametrilor. Alegerea $\mu = 0,5$ pare a fi cea mai indicată.

6. Concluzii

Scopul acestei lucrări a fost de a prezenta într-un cadru unitar unele aspecte teoretice și de implementare, specifice tehnicilor de estimare recursivă.

Algoritmii prezentați reprezintă ecuații cu diferențe neliniare și stochastice care fac ca analiza directă a convergenței să fie extrem de dificilă. Analiza teoretică a algoritmului este posibilă prin studiul soluției unei ecuații diferențiale ordinare asociate (EDO). Rezultatele analizei algoritmilor sintetizate în lucrare au fost obținute în ipoteza că structura modelului este aceeași cu cea a sistemului considerat și că matricea $G(0^*)$ este nesingulară.

În ceea ce privește aspectele de implementare a algoritmului de estimare recursivă general, a fost tratat în detaliu calculul vectorului de amplificare. Ca soluție de implementare se sugerează utilizarea algoritmului de factorizare

U—D care încorporează simplu și facilitatea de regularizare. Aceste aspecte sînt importante, în special, pentru algoritmi care se utilizează în cadrul unor sisteme de conducere adaptivă a proceselor industriale.

BIBLIOGRAFIE

1. *Aström, K. J.* (1970). **Introduction to Stochastic Control Theory**, Academic Press, New York.
2. *Aström, K. J. și T. Bohlin* (1965). **Numerical identification of linear dynamic systems from normal operating records**, IFAC Symposium on Self-Adaptive Systems, Teddington.
3. *Aström, K. J. și P. Eykhoff* (1971). **System identification — A survey**, *Automatica*, 7, 123—162.
4. *Aström, K. J. și I. Söderstrom* (1974). **Uniqueness of the maximum likelihood estimates of the parameters of an ARMA model**, *IEEE Trans. AC-19*, 769—773.
5. *Bierman, G. J.* (1977). **Factorization algorithms for discrete sequential estimation**, Academic Press, New York.
6. *Bohlin, T.* (1968). **Real time estimation of time variable process characteristics**, Technical paper TP 18.190, IBM Nordic Lab., Lidingö.
7. *Clarke, D. W.* (1967). **Generalized least squares estimation of parameters of a dynamic model**. 1st IFAC Symposium on Identification in Automatic Control Systems, Prague.
8. *Dongarra, J. J., C. B. Moler, J. R. Bunch și G. W. Stewart* (1979). **LINPACK Users' Guide**, Siam, Philadelphia.
9. *Eykhoff, P.* (1974). **System Identification — Parameter and State Estimation**, J. Wiley & Sons, London.
10. *Finigan, B. și I. H. Rowe* (1974). **Strongly consistent estimation by the introduction of strong instrumental variable**, *IEEE Trans. AC-19*, 825—830.
11. *Furth, B. P.* (1973). **New estimator for the identification of dynamic processes**, IBK Report, Institut Boris Kidrič Vinča, Belgrad.
12. *Gertler, J. și Cs Bánýász* (1974). **A recursive (on-line) maximum likelihood identification method**, *IEEE Trans. AC-19*, 816—820.
13. *Hannan, E. I.* (1976). **The convergence of some recursions**, *Ann. Statist.* 4, 1258—1270.
14. *Hasting-James, R. și M. W. Sage* (1969). **Recursive generalized least squares procedure for on-line identification of process parameters**, *IEE Proc.* 116, 2057—2062.
15. *Holst, J.* (1977). **Adaptive prediction and recursive estimation**, Report 1013, Dept. of Automatic Control, Lund Inst. of Tech., Lund.
16. *Kucera V.* (1979). **Discrete Linear Control**, John Wiley and Sons, Chichester.
17. *Landau, I. D.* (1974). **A survey of model reference adaptive techniques: Theory and applications**, *Automatica*, 10, 353—379.
18. *Ljung, L.* (1974). **Convergence of recursive stochastic algorithms**, Report 7403, Dept. of Automatic Control, Lund Inst. of Tech., Lund.
19. *Ljung L., T. Soderstrom și I. Gustavsson* (1975). **Counterexamples to general convergence of a commonly used identification method**, *IEEE Trans. AC-20*, 643—652.
20. *Ljung, L.* (1977a). **On positive real functions and the convergence of some recursive schemes**, *IEEE Trans. AC-22*, 539—551.
21. *Ljung, L.* (1977b). **Analysis of recursive stochastic algorithms**, *IEEE Trans. AC-22*, 551—575.
22. *Ljung, L., M. Morf și D. Falconer* (1978). **Fast calculation of gain matrices for recursive estimation schemes**, *Int. J. Control*, 27, 1—18.
23. *Ljung, L., și T. Söderstrom* (1981). **Theory and Practice of Recursive Identification**, în curs de apariție la MIT Press.
24. *Morquardt, D.W.* (1963). **An algorithm for least-squares estimation of non-linear parameters** *Journal SIAM*, 11, 431—441.
25. *Mayne, D.Q.* (1967). **A method for estimating discrete time transfer functions**. In *Advances in Computer Control*, Second UKAC Control Convention, University of Bristol.
26. *Narendra, K.S. și P. Kudva* (1974). **Stable adaptive schemes for system identification control — Part. II**, *IEEE Trans. on Systems, Man and Cybernetics*, SMC-4, 552—560.
27. *Panuska, V.* (1968). **A stochastic approximation method for identification of linear systems using adaptive filtering**, *Proc. JACC*.

28. *Panuska, V.* (1969). An adaptive recursive least squares identification algorithm. Proc. IEEE Symposium on Adaptive Processes, Decision and Control.
29. *Parks, P.C.* (1966). Lyapunov redesign of model reference adaptive control systems. IEEE Trans. AC-11, 362–367.
30. *Peterka V. și A. Halusková* (1970). Tally estimate of Aström model for stochastic systems, Proc. 2 nd IFAC Symposium on Identification and Process Parameter Estimation, Prague.
31. *Popescu, Th.* (1980). Metode de identificare recursive, Referat de doctorat, Fac. de Automatică, Institutul Politehnic București.
32. *Popescu, Th.* (1981). Conducerea adaptivă cu calculator a sistemelor continue, Referat de doctorat, Fac. de Automatică, Institutul Politehnic București.
33. *Potter, J.E.* (1963) New statistical formulas. Memo 40, Instrumentation Laboratory, Massachusetts Institute of Technology, Cambridge, Mass.
34. *Robins, A.J. și P.E. Wellstead* (1981). Recursive system identification using fast algorithms, Int. J. Control, 33, 455–480.
35. *Rowe, I.H.* (1970). A bootstrap method for the statistical estimation of model parameters, Int. J. Control 12, 721–738.
36. *Sakrison, D.J.* (1967). The use of stochastic approximation to solve the system identification problem, IEEE Trans. AC-12, 563–567.
37. *Sardis, G.N. și G. Stein* (1968). Stochastic approximation algorithms for linear discrete time system, IEEE Trans. AC-13, 515–523.
38. *Sardis, G.N.* (1974). Comparison of six on-line identification algorithm, Automatica, 10, 69–79.
39. *Sen, A. și N.K. Sinha* (1975). A generalized pseudoinverse algorithm for unbiased parameter estimation, Int. J. Syst. Sci 6, 1103–1109.
40. *Söderström, T.* (1973). An on-line algorithm for approximate maximum likelihood identification of linear dynamic systems, Report 7308, Dept. of Automatic Control, Lund Inst. of Tech. Lund.
41. *Söderström, T.* (1974 a). Convergence of identification method based on the instrumental variable approach, Automatica, 10, 685–688.
42. *Söderström, T.* (1974 b). Convergence properties of the generalized least squares identification method, Automatica 10, 617–626.
43. *Söderström T., L. Ljung și I. Gustavsson* (1974). A comparative study of recursive identification method, Report 7427, Dept. of Automatic Control, Lund Inst. of Tech., Lund.
44. *Söderström, T., L. Ljung și I. Gustavsson* (1978). A theoretical analysis of recursive identification methods, Automatica, 14, 231–244.
45. *Stewart, G.W.* (1973). Introduction to matrix computations, Academic Press, New York.
46. *Talmon, J.L. și A.J.W. van den Boom* (1973). On the estimation of transfer function parameters of processes and noise dynamics using a single stage estimator, Proc. 3rd IFAC Symposium on Identification and System Parameter Estimation, Haga.
47. *Tertisco, M. și P. Stoica* (1980). Identificarea și estimarea parametrilor sistemelor, Editura Academiei, București.
48. *Thornton, C.L. și G.J. Bierman* (1978). Filtering and error analysis via the UDU covariance factorization, IEEE Trans. AC-23, 901–907.
49. *Young, P.C.* (1968). The use of linear regression and related procedures for the identification of dynamic processes, Proc. 7th IEEE Symposium on Adaptive Processes, UCLA.
50. *Young, P.C.* (1970). An instrumental variable method for real time identification of a noisy process, Automatica, 6, 271–287.
51. *Young, P.C.* (1974). Recursive approaches to time series analysis, Bull. Inst. Math. Applic. 10, 209–224.
52. *Young, P.C.* (1976). Some observations on instrumental variable methods of time-series analysis, Int. J. Control, 23, 593–612.
53. *Wieslander, I.* (1969). Real time identification, Report 6908, Dept. of Automatic Control, Lund Inst. of Tech., Lund.
54. *Wong, K.Y. și E. Polak* (1967). Identification of linear discrete time system using the instrumental variable method, IEEE Trans. AC-12, 707–718.

BIMASC — PACHET DE SUBPROGRAME FORTRAN PENTRU MODELAREA, ANALIZA, PROIECTAREA ȘI SIMULAREA SISTEMELOR AUTOMATE

Dr. ing. A. Varga
I.T.C.I.

1. Introducere

În ultimii ani, pe plan mondial se observă o activitate susținută pentru dezvoltarea unor algoritmi de calcul performanți și siguri, destinați rezolvării problemelor numerice specifice teoriei sistemelor automate. Se dispune în prezent de algoritmi eficienți pentru principalele probleme de calcul ce intervin în modelarea, analiza, proiectarea și simularea sistemelor automate multi-variabile, cum sînt de exemplu analiza structurală [1], [2]; optimizarea liniar-pătratică [3]—[5]; alocarea polilor [6], [7]; simulare [8]—[12]. O sinteză comparativă amplă asupra algoritmilor de bază din domeniile de mai sus este făcută în ciclul de lucrări [13]—[15].

Datorită dezvoltărilor importante obținute în domeniul algoritmilor, s-a început organizarea unor colecții sistematice de programe ce implementează algoritmi cei mai performanți. Acest lucru a fost posibil datorită disponibilității unor pachete de programe matematice de înaltă calitate, cum sînt BLAS — pentru problemele de bază ale algebrei liniare [16], LINPACK — pentru rezolvarea sistemelor de ecuații liniare [17] și EISPACK — pentru probleme de valori și vectori proprii [18], [19]. Aceste pachete au servit ca modele de organizare, programare și documentare, pachetelor cu specific de automată.

În perioada 1981—1982, s-a elaborat în cadrul Institutului Central pentru Conducere și Informatică pachetul BIMAS destinat rezolvării problemelor numerice de bază ale analizei, proiectării și simulării sistemelor automate liniare [20], [21]. BIMAS constituie o colecție sistematică de subprograme FORTRAN portabile, elaborate pe baza standardelor de programare utilizate în LINPACK și EISPACK. De altfel, o serie de rutine din aceste pachete au fost preluate fără modificări în BIMAS.

BIMASC este un pachet nou ce extinde pachetul BIMAS cu o serie de funcții specifice analizei, modelării, proiectării și simulării sistemelor multi-variabile [22]. Algoritmii implementați în BIMASC au fost riguros selectați în vederea satisfacerii unor criterii de calitate cum sînt generalitate, siguranță, stabilitate numerică, precizie și eficiență. Majoritatea subprogramelor din BIMASC apelează subprograme din pachetele BIMAS, EISPACK, LINPACK și BLAS. BIMASC include, de asemenea, două pachete puternice pentru integrarea ecuațiilor diferențiale ordinare RK45 [11] și LSODE [12]. BIMASC

conține 47 de subprograme originale și 21 subprograme preluate din pachetele EISPACK, LINPACK, RKF45 și LSODE.

BIMAS și BIMASC au foarte multe caracteristici comune. Astfel majoritatea algoritmilor implementați utilizează transformări ortogonale în vederea asigurării unor proprietăți numerice cât mai bune și siguranță cât mai ridicată. În cazul utilizării unor transformări neortogonale, de regulă, se determină informații asupra condiționării numerice a problemelor. Ambele pachete folosesc o abordare structurală pentru rezolvarea problemelor complexe. Această abordare se reflectă printr-o înaltă modularitate a pachetelor. Majoritatea calculurilor complexe se rezolvă printr-o secvență de apeluri la mai multe subprograme. Una dintre consecințele abordării modulare este ușurința segmentării programelor. Astfel, pachetele se pot utiliza în condiții bune chiar pe minicalculatoare. Toate subprogramele din BIMAS și BIMASC sînt scrise în FORTRAN IV, în dublă precizie. Utilizarea unui FORTRAN standard, fără exploatarea unor facilități de lucru oferite de implementări particulare ale compilatoarelor, a condus la o portabilitate ridicată a pachetelor și permite implementarea lor fără dificultăți pe diferite tipuri de calculatoare.

Pachetele BIMAS și BIMASC sînt disponibile pe minicalculatoarele românești I-100, 102F și CORAL 4011, 4030, sub sistemele de operare MIX sau RSN-11M. Pachetele dispun de un set complet de programe de test. Aceste programe au fost astfel elaborate încît permit rezolvarea unei game largi de probleme concrete de analiză și proiectare a sistemelor automate.

BIMASC acoperă cu subprograme principalele probleme de calcul ce intervin într-o metodologie structurală completă de analiză și proiectare a sistemelor automate prin metode în spațiul stărilor. Funcțiile pachetului BIMASC sînt prezentate în patru secțiuni distincte dedicate problemelor de modelare (sect. 2), analiză (sect. 3), proiectare (sect. 4) și simulare (sect. 5).

2. Modelarea sistemelor liniare

Scopul modelării, în accepțiunea din această lucrare, este determinarea modelului adecvat utilizării metodelor de analiză, proiectare și simulare. Majoritatea acestor metode utilizează descrieri de stare liniare, continue sau discrete, de forma

$$\lambda x(t) = Ax(t) + Bu(t) \quad (1)$$

$$y(t) = Cx(t)$$

unde x , u și y sînt vectorul de stare n -dimensional, vectorul comenzilor m -dimensional și respectiv vectorul ieșirilor p -dimensional, iar λ este operatorul diferențial d/dt în cazul unui sistem continuu, sau operatorul de anticipare $\lambda x(t) = x(t+1)$ în cazul unui sistem discret. Sistemul (1) va fi referit de asemenea prin tripletul (A, B, C) .

O altă descriere care poate servi ca bază de plecare pentru analiză și proiectare este un model intrare-ieșire de forma

$$Y(\lambda) = G(\lambda) U(\lambda) \quad (2)$$

În care $U(\lambda)$ și $Y(\lambda)$ sînt transformatele vectorilor de intrare și respectiv ieșire, iar $G(\lambda)$ este o matrice de transfer $p \times m$. În cazul continuu se utilizează, transformata Laplace, în locul lui λ utilizîndu-se de regulă variabila complexă s , iar în cazul discret, se folosește transformata Z , iar în loc de λ se întrebuintează o variabilă complexă notată cu z . Matricea de transfer corespunzătoare modelului (1) este dată de formula:

$$G(\lambda) = C(\lambda I - A)^{-1}B \quad (3)$$

În cazul discret, elementele matricii de transfer sînt funcții raționale. În cazul continuu, elementele matricii de transfer pot fi atît funcții raționale cît și funcții raționale înmulțite cu factori de forma $e^{-\tau_{ij}s}$, în care τ_{ij} reprezintă timpul mort corespunzător canalului de la intrarea j la ieșirea i .

2.1. Transformări de similaritate

Vom spune că două sisteme (A, B, C) și (A', B', C') sînt *similare* dacă există o matrice de transformare nesingulară Q astfel încît

$$A' = Q^{-1}AQ, \quad B' = Q^{-1}B, \quad C' = CQ \quad (4)$$

Sistemele (A, B, C) și (A', B', C') legate prin (4) au aceeași matrice de transfer. Transformarea de forma (4) se numește *transformare de similaritate*.

Din punct de vedere numeric sîntem interesați să folosim numai transformări bine condiționate, în sensul că *numărul de condiționare* al matricii de transformare

$$\kappa(Q) = \|Q\| \cdot \|Q^{-1}\| \quad (5)$$

nu este prea mare (este întotdeauna supraunitar). Pentru siguranța calculelor, este de dorit ca pentru toți algoritmi care utilizează transformări de similaritate, să dispunem de estimări ale numărului de condiționare ale matricilor de transformare utilizate.

Transformări de similaritate ortogonale. O matrice nesingulară care satisface relația $Q^T Q = I$ se numește *ortogonală*. Transformările ortogonale au proprietăți numerice remarcabile. În primul rînd ele sînt perfect condiționate avînd $\kappa(Q) = 1$ și din acest motiv utilizarea lor nu modifică de regulă sensibilitatea problemelor. În al doilea rînd o serie de algoritmi bazați pe utilizarea transformărilor ortogonale sînt *numeric stabili*, în sensul că matricile calculate A' , B' și C' prin similaritate sînt exacte pentru un sistem care este foarte apropiat de sistemul (A, B, C) . Matematic acest lucru se poate scrie prin egalitățile

$$A' = Q^T (A + \delta A) Q, \quad B' = Q^T (B + \delta B), \quad C' = (C + \delta C) Q \quad (6)$$

în care δA , δB și δC satisfac condiții de forma

$$\|\delta A\| \leq \varepsilon \|A\|, \quad \|\delta B\| \leq \varepsilon \|B\|, \quad \|\delta C\| \leq \varepsilon \|C\| \quad (7)$$

În inegalitățile de mai sus, ε este un factor generic egal cu un multiplu modest al preciziei relative a mașinii.

Cîteva rutine din BIMAS se pot folosi pentru reducerea lui A la forme particulare: *forma Hessenberg*, *forma Schur reală* (FSR), sau FSR ordonată.

Transformările ortogonale rezultate cu aceste rutine, sau transformările ortogonale arbitrare se pot aplica unui sistem (A, B, C) utilizînd subrutina ORTEQ.

Multe dintre problemele de calcul ce intervin în modelarea și analiza sistemelor liniare necesită reducerea sistemului (1) prin transformări de similaritate ortogonale la forma standard de controlabilitate (FSC)

$$A' = \begin{bmatrix} A_R & X \\ 0 & A_N \end{bmatrix} \quad B' = \begin{bmatrix} B_R \\ 0 \end{bmatrix} \quad C' = [C_R \ C_N] \quad (8)$$

în care A_R , B_R și C_R au dimensiunile $k \times k$, $k \times m$ și $p \times k$, respectiv. În general, dacă nu există nici o transformare de similaritate de forma (4) care reduce sistemul (A, B, C) la forma (8) cu $0 \leq k < n$, atunci vom spune că sistemul (1) este *controlabil*. Deoarece controlabilitatea este o proprietate structurală a perechii (A, B), vom spune echivalent că perechea (A, B) este controlabilă. În cazul cînd sistemul (A, B, C) se poate pune sub forma (8) cu $k < n$, vom spune că sistemul (1) este *necontrolabil* sau echivalent, perechea (A, B) este necontrolabilă. Un sistem în FSC (8) are proprietatea că perechea (A_R , B_R) este controlabilă, iar sistemul (A_R , B_R , C_R) are aceeași matrice de transfer ca sistemul (A, B, C). A_N este partea necontrolabilă a matricii de stare.

Reducerea unui sistem la FSC se poate face utilizînd subrutina TSCO, care opțional acumulează transformările ortogonale. Algoritmii implementați în TSCO este bazat pe metoda descrisă în [2], în care calculele de rang bazate pe descompunerea valorilor singulare s-au înlocuit cu calcule de rang bazate pe descompunerea QR cu pivotare.

Forma standard de observabilitate (FSO) a unui sistem (A, B, C) se poate obține de asemenea utilizînd transformări de similaritate ortogonale. Matricile sistemului în FSO au forma

$$A' = \begin{bmatrix} A_R & 0 \\ X & A_N \end{bmatrix} \quad B' = \begin{bmatrix} B_R \\ B_N \end{bmatrix} \quad C' = [C_R \ 0] \quad (9)$$

în care A_R , B_R , și C_R au dimensiunile $k \times k$, $k \times m$ și $p \times k$, respectiv. Dacă $k = n$ vom spune că sistemul (1) este *observabil* sau echivalent perechea (C, A) este observabilă. Dacă $k < n$, sistemul este *neobservabil*. În FSO, perechea (C_R , A_R) este observabilă, iar sistemul (A_R , B_R , C_R) are aceeași matrice de transfer ca sistemul inițial. A_N este în acest caz partea neobservabilă a matricii de stare.

Reducerea sistemului (1) la FSO se poate face utilizînd TSCO pentru sistemul dual (A^T , C^T , B^T). Sistemul dual se poate construi cu subrutina DUALS.

Transformări de similaritate neortogonale. Cîteva rutine din BIMAS efectuează transformări neortogonale pentru reducerea lui A la forme particulare (Hessenberg, bloc-diagonală). Aplicarea transformărilor de similaritate arbitrare, neortogonale, se poate face în cazul general cu subrutina SIMEQ. Această rutină determină de asemenea și o estimare a numărului de condiționare a matricii de transformare.

Deseori modelele de stare minimale (controlabile și observabile) obținute din reprezentări de tip intrare-ieșire sînt slab scalate datorită unei alegeri improprii a sistemului de coordonate pentru variabilele de stare. Astfel de scalări conduc de regulă la balansări nedorite în model în raport cu gradul de

controlabilitate (GC) și gradul de observabilitate (GO) ale sistemului. Din punct de vedere calitativ, GC și GO reprezintă măsuri ale depărtării unui sistem dat de unul necontrolabil, sau neobservabil, respectiv. Balansarea unui sistem urmărește maximizarea simultană și egalizarea GC și GO ale unui sistem utilizând transformări de similaritate.

O tehnică de balansare de acest fel a fost propusă de Moore [25]. În acest caz GC și GO ale sistemului se definesc ca fiind cele mai mici valori singulare ale Grammian-ului de controlabilitate și respectiv observabilitate. Procedura de balansare asigură egalitatea și forma diagonală a matricilor Grammian ale sistemului balansat. Un sistem cu această proprietate se numește sistem *balansat intern* (SBI).

O procedură pentru calculul transformărilor de balansare pentru sisteme stabile a fost propusă de Laub [23] și este implementată în subrutina TSBALI. Modelul SBI calculat cu TSBALI se poate utiliza pentru construirea unor modele aproximative de ordin redus pentru sistem. O altă metodă de balansare, propusă de Varga [27], este implementată în subrutina TSBAL. Sistemul balansat rezultat are matricea de stare în FSR, iar matricile Grammian sînt ortogonal similare cu aceeași matrice diagonală pozitiv definită. Ambele rutine estimează numerele de condiționare ale matricilor de transformare utilizate

2.2. Construcția realizărilor de stare ale matricilor de transfer

Metodele de identificare experimentale ale proceselor determină de cele mai multe ori modele de tip intrare-ieșire, continue sau discrete, specificate prin matrici de transfer. Dacă matricea de transfer $G(\lambda)$ a unui sistem este cunoscută și are numai elemente funcții raționale, atunci putem construi mai multe realizări de stare (A, B, C) care satisfac (3) utilizînd una dintre subrutinele RENEM, RENEMC sau RENEMO. Modelul de stare construit cu fiecare dintre aceste rutine are forma

$$A = \begin{bmatrix} A_1 & 0 & \dots & 0 \\ 0 & A_2 & \dots & 0 \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \dots & A_k \end{bmatrix} \quad B = \begin{bmatrix} B_1 \\ B_2 \\ \cdot \\ \cdot \\ \cdot \\ B_k \end{bmatrix} \quad (10)$$

$$C = [C_1 \quad C_2 \quad \dots \quad C_k]$$

în care subsistemele (A_i, B_i, C_i) , $i=1, 2, \dots, k$ sînt controlabile și observabile. RENEM construiește în general modele de stare necontrolabile și neobservabile, RENEMC construiește modele controlabile, iar RENEMO construiește modele observabile. În cazul cînd sistemul este stabil, se poate face o balansare utilă la nivelul fiecărui subsistem utilizînd subrutina BALRNM. Realizări de stare particulare pentru sistemele cu o intrare și o ieșire se pot construi cu subrutinele REMC1 și REMO1.

Determinarea unui model de stare *minimal* (controlabil și observabil) pentru o matrice de transfer rațională se face de regulă în două etape. În prima

etapă se construiește o realizare neminimală de forma (10) utilizând RENEM, RENEMC sau RENEMO. În etapa a doua se calculează modelul de stare minimal utilizând subrutina REMIN. REMIN apelează TSCO pentru eliminarea succesivă a părților necontrolabile și neobservabile ale sistemului neminimal. Această procedură de calcul a realizărilor minimale este numeric stabilă. Trebuie să menționăm că în foarte multe cazuri concrete, modelul de stare construit cu RENEMC sau RENEMO rezultă minimal, etapa a doua nefiind necesară.

2.3. Evaluarea matricilor de transfer din reprezentări de stare

Pentru o reprezentare de stare (A, B, C) dată a unui sistem, sîntem cîtodeodată interesați să calculăm matricea de transfer corespunzătoare $G(\lambda)$ (3), sau a unei linii, a unei coloane sau chiar a unui singur element al acestei matrici. În acest scop se poate utiliza una dintre subrutinele TSMT sau TSMT2. TSMT implementează un algoritm propus în [28]. Acest algoritm utilizează transformări de similaritate ortogonale pentru a determina realizări de stare minimale, pe rînd pentru fiecare canal intrare-ieșire al sistemului multivariabil. Apoi, se evaluează pentru fiecare element polii, zerourile și factorul de amplificare. Calculul polilor și zerourilor are la bază determinări de valori proprii.

TSMT2 utilizează în locul transformărilor ortogonale din TSMT transformări de similaritate stabilizate. De asemenea, calculul polilor și zerourilor este înlocuit cu evaluări de polinoame caracteristice ale unor matrici Hessenberg. În general, TSMT2 este mult mai rapid decît TSMT și de regulă și mai precis. Cu toate acestea, precizia *garantată* apriori pentru TSMT este mai mare decît pentru TSMT2 și acest lucru se poate manifesta efectiv în cazul unor exemple particulare.

2.4. Discretizarea sistemelor continue

Proiectarea legilor de reglare pentru conducerea numerică directă cu calculator a proceselor continue necesită utilizarea unor modele discrete (sau eșantionate) pentru proces. De asemenea, simularea sistemelor liniare continue se poate efectua eficient prin utilizarea modelului de stare discretizat.

Considerăm sistemul liniar continuu

$$\frac{dx(t)}{dt} = Ax(t) + Bu(t) \quad (11)$$

și fie h perioada de eșantionare. Presupunem că u are valoarea constantă $u(k)$ pe fiecare interval $[kh, (k+1)h)$, $k=0, 1, \dots$ și notăm $x(k)$ valoarea lui x zentru $t=kh$. În acest caz, sistemul discretizat corespunzător sistemului (11) este dat de

$$x(k+1) = E x(k) + Fu(k) \quad (12)$$

unde

$$E = e^{Ah}, \quad F = \int_0^h e^{A^t} B dt \quad (13)$$

iar $e^{\mathbf{X}}$ este exponențiala matricii \mathbf{X} definită prin dezvoltarea în serie

$$e^{\mathbf{X}} = 1 + \frac{\mathbf{X}}{1!} + \frac{\mathbf{X}^2}{2!} + \dots$$

Dacă definim matricea

$$\mathbf{D} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ 0 & 0 \end{bmatrix}$$

atunci \mathbf{E} și \mathbf{F} din (13) se pot determina din identitatea

$$e^{\mathbf{D}h} = \begin{bmatrix} \mathbf{E} & \mathbf{F} \\ 0 & \mathbf{I} \end{bmatrix} \quad (14)$$

Această metodă de discretizare a fost propusă de Van Loan [8] și este implementată în subrutina TSCD. Exponențiala (14) se evaluează utilizând o metodă ce combină o procedură de bloc diagonalizare propusă în [24] și o procedură dezvoltată în [29] pentru evaluarea exponențialei matriciale prin aproximări raționale Padé. Această metodă propusă în [27] este prezentată și în lucrarea mai accesibilă [14].

Discretizarea matricilor de transfer a sistemelor continue se poate face utilizând subrutina TMFCD. În cazul când matricea de transfer conține elemente cu timpi morți, printr-o alegere convenabilă a perioadei de eșantionare putem elimina factorii iraționali din model. Algoritmul implementat în TMTCDD se aplică în parte fiecărui element al matricii de transfer, deci îl vom schița doar în cazul unei singure funcții de transfer. Pentru partea rațională a funcției de transfer se construiește o reprezentare de stare minimală, care se discretizează utilizând aproximări raționale Padé pentru exponențiala matricială [29]. Funcția de transfer a sistemului discretizat se calculează cu TSMT2. Dacă funcția de transfer continuă are timp mort, atunci numitorul funcției de transfer discrete se multiplică cu factorul corespunzător întârzierii datorate timpului mort.

Trebuie să facem observația că ambele metode de discretizare implementate în TSCD și TMTCDD utilizează un extrapolator de ordin zero pe post de convertor discret-continuu al semnalelor de intrare ale sistemului continuu.

3. Analiza sistemelor liniare

Analiza unui sistem are ca scop studierea unor proprietăți ale sistemului (stabilitate, controlabilitate/stabilizabilitate, observabilitate/detectabilitate), determinarea unor mărimi caracteristice (poli, zerouri, margine de stabilitate), studierea comportării dinamice a sistemului în buclă deschisă. În acest capitol ne vom referi la primele două dintre aspectele de mai sus. Studierea comportării dinamice a sistemului prin simulare va fi discutată în secț. 5.

3.1. Calculul polilor și zerourilor sistemului

Cunoașterea polilor este importantă în studierea dinamicii sistemului și a stabilității sale. Pentru sistemul (1) polii sînt valorile proprii ale matricii de stare A. Sistemul (1) este stabil dacă toate valorile proprii ale lui A au părți reale negative în cazul continuu sau au modulele mai mici decît unu în cazul discret. *Marginea de stabilitate* pentru sistemul (1) se poate defini ca partea reală maximă (în cazul continuu) sau ca modulul maxim (în cazul discret) a valorilor proprii ale matricii A. Subrutina TSTA determină polii și marginea de stabilitate ale sistemului și testează condiția de stabilitate.

Zerourile unui sistem joacă un rol important în teoria reglării sistemelor liniare [30], [31]. Pentru definirea zerourilor unui sistem considerăm descrierea de stare mai generală dată de ecuațiile

$$\lambda x(t) = Ax(t) + Bu(t) \quad (15)$$

$$y(t) = Cx(t) + Du(t)$$

în care, spre deosebire de modelul (1), apare o legătură directă intrare-ieșire în cazul cînd matricea D este nenulă.

Pentru sistemul descris de ecuațiile (15) zerourile sistemului sau zerourile invariante se definesc ca zerourile Smith ale fascicului de matrici

$$S(\lambda) = \begin{bmatrix} \lambda I - A & B \\ -C & D \end{bmatrix} \quad (16)$$

Matricea $S(\lambda)$ de dimensiune $(n+p) \times (n+m)$ se numește *matricea sistem* [32]. Zerourile Smith ale lui $S(\lambda)$ sînt zerourile (incluzînd multiplicitățile) celui mai mare divizor comun al tuturor minorilor identic nenuli de ordin $n + \min(m, p)$ ai lui $S(\lambda)$. Dacă $p=0$, zerourile sistemului se numesc *zerouri de decuplare la intrare* (z.d.i.) și sînt egale cu polii necontrolabili ai sistemului, iar cînd $m=0$, zerourile se numesc *zerouri de decuplare la ieșire* (z.d.o.) și sînt egali cu polii neobservabili ai sistemului. Cînd sistemul este minimal (controlabil și observabil), zerourile sistemului se numesc *zerouri de transmisie* și sînt egale cu zerourile McMillan [32] ale matricii de transfer a sistemului

$$G(\lambda) = C(\lambda I - A)^{-1}B + D \quad (17)$$

O metodă eficientă de calcul a zerourilor sistemului a fost propusă de Emami-Naeini și Van Dooren [33] și se bazează pe forma canonică Kronecker a matricii sistem $S(\lambda)$. Acest algoritm extrage din fascicului de matrici (16) un fascicul regulat $\lambda F - E$ ($\det(\lambda F - E) \neq 0$) ale cărui valori proprii generalizate sînt zerourile finite ale lui $S(\lambda)$. Extragerea fascicului regulat se face utilizînd în exclusivitate transformări ortogonale și este implementară în subrutina MZEROS (adaptată din [33]). Valorile proprii generalizate se calculează cu rutine din BIMAS.

3.2. Analiza controlabilității și observabilității

Pentru testarea controlabilității sistemului (1), se poate utiliza subrutina TSCO. Sistemul este controlabil dacă partea sa controlabilă în FSC (8) are ordinul n . O altă posibilitate este utilizarea subrutinei MZEROS pentru determinarea z.d.i. ale fascicului

$$S(\lambda) = [\lambda I - A \quad B]$$

Dacă $S(\lambda)$ nu are nici un z.d.i. atunci sistemul (1) este controlabil.

Pentru testarea observabilității sistemului, aceleași subrutine se utilizează pentru perechea duală (A^T, C^T). Când se utilizează MZEROS, sistemul este observabil dacă fascicul

$$S(\lambda) = \begin{bmatrix} \lambda I - A \\ -C \end{bmatrix}$$

nu are z.d.o.

Sistemul (1) este *stabilizabil* dacă polii necontrolabili (valorile proprii ale lui A_N din (8)) sînt stabili și este *detectabil* dacă polii neobservabili (valorile proprii ale lui A_N din (9)) sînt stabili. Pentru testarea stabilizabilității sau detectabilității se pot utiliza rezultatele calculate fie de TSCO sau MZEROS. Dacă se utilizează TSCO, testul se va face cu subrutine TSTA aplicat matricii A_N din (8) sau (9). Dacă se utilizează MZEROS atunci se va testa stabilitatea z.d.i. sau z.d.o. calculate.

4. Proiectarea sistemelor de reglare

Considerăm sistemul liniar descris de ecuațiile

$$\begin{aligned} \lambda x(t) &= Ax(t) + Bu(t) + Ew(t) \\ y(t) &= Cx(t) \\ y_r(t) &= C_r x(t) \\ e(t) &= r(t) - y_r(t) \end{aligned} \tag{18}$$

În care x , u și y sînt ca mai înainte, w este vectorul 1-dimensional al perturbațiilor măsurabile, iar y_r , r și e sînt vectorii q -dimensionali ai ieșirilor reglate, referințelor și erorilor de reglare, respectiv. Vom presupune că ieșirile reglate sînt măsurabile, deci matricea C_r are forma $C_r = [I \ 0] C$, unde I este o matrice unitate de ordin q .

Compensatorul este un sistem liniar care are ca intrări ieșirea măsurată y , referințele r și eventual perturbațiile măsurabile w și generează ca ieșire comanda u care se aplică procesului. Compensatorul poate fi descris de exemplu de ecuațiile:

$$\begin{aligned} \lambda z(t) &= Fz(t) + Gy(t) + Kw(t) + Lr(t) \\ u(t) &= H_1 z(t) + H_2 y(t) \end{aligned} \tag{19}$$

în care z este vectorul de stare al compensatorului. Există diverse structuri de compensator care să satisfacă aceleași cerințe de proiectare. Astfel, unele dintre matricile K , L sau H_2 din (19) pot fi identic nule. În alte situații, în expresia lui u în (19) pot apare termeni suplimentari, așa cum este în cazul acțiunii directe de la perturbații sau de la referințe.

Problema de reglare se poate formula în felul următor: să se determine matricile compensatorului (19) astfel încât: 1) sistemul în buclă închisă să fie stabil; 2) eroarea de reglare staționară să fie nulă, adică $e(t) \rightarrow 0$ pentru $t \rightarrow \infty$ (anularea erorii staționare) și 3) proprietățile de stabilitate și de anulare a erorii staționare să se mențină în prezența unor perturbații mici, dar arbitrare în parametrii sistemului sau ai compensatorului. Proprietatea 3) se mai numește *robustețe*, iar compensatorul (19) ce satisface 1), 2) și 3) se numește *compensator robust*. Problema de reglare are soluție dacă [30], [31]: 1) sistemul (A, B, C) este stabilizabil și detectabil, și 2) zerourile sistemului (A, B, C_r) nu coincid cu frecvențele care formează spectrul semnalelor exterioare (referințe și perturbații). De exemplu, în cazul cel mai frecvent în practică, al unor referințe constante și a unor perturbații de tip treaptă, condiția asupra zerourilor este ca sistemul să nu aibă zerouri nule. în cazul continuu, sau zerouri egale cu unitatea, în cazul discret. Pentru acest exemplu, compensatorul robust este o generalizare a regulatorului tradițional PI folosit curent în practica industrială.

Proiectarea unui compensator robust implică stabilizarea unui sistem extins format din sistemul (18) cuplat cu un *model intern* [30] sau *servocompensator* [31]. Modelul intern este descris de ecuația

$$\lambda x_1(t) = A_1 x_1(t) + B_1 e(t) \quad (20)$$

în care x_1 este starea modelului intern. Matricea A_1 se alege astfel încât să includă o reduplicare de q ori a dinamicii semnalelor externe. Perechea (A_1, B_1) se alege controlabilă. De exemplu modelul intern discret pentru semnale exterioare de tip treaptă are forma simplă

$$x_1(t+1) = x_1(t) + e(t)$$

Sistemul extins care trebuie stabilizat este dat de ecuația

$$\lambda x'(t) = A'x(t) + B'u(t) \quad (21)$$

în care vectorul de stare x' se poate alege

$$x' = \begin{bmatrix} x_1 \\ x \end{bmatrix} \text{ sau } x' = \begin{bmatrix} x \\ x_1 \end{bmatrix} \quad (22)$$

iar perechile (A', B') corespunzătoare sînt date de (23) sau (24), respectiv:

$$A' = \begin{bmatrix} A_1 & -B_1 C_r \\ 0 & A \end{bmatrix} \quad B' = \begin{bmatrix} 0 \\ B \end{bmatrix} \quad (23)$$

$$A' = \begin{bmatrix} A & 0 \\ -B_1 C_r & A_1 \end{bmatrix} \quad B' = \begin{bmatrix} B \\ 0 \end{bmatrix} \quad (24)$$

Procedura de proiectare generală a compensatorului robust are următoarele etape [31], [34]:

- 1) Se formează sistemul extins (21) cu matricile A' și B' date de (23) sau (24);
- 2) Se stabilizează sistemul (21) cu o reacție de stare de forma

$$u(t) = H'x'(t) = H_1x_1(t) + Hx(t) \quad (25)$$

unde $H' = [H_1 \ H]$ sau $H' = [H \ H_1]$, după cum s-a utilizat extinderea (23) sau (24), respectiv;

- 3) Se proiectează un estimator de stare pentru sistemul care are ca intrări y , u și eventual w și furnizează x_e , starea estimată a sistemului.

În BIMASC există o serie de subprograme cu ajutorul cărora proiectarea compensatorului se poate efectua. Pentru construirea matricilor A' și B' din (23) sau (24) se utilizează subrutina EXTI. Această subrutină poate genera automat matricile A_1 și B_1 pentru clasa semnalelor externe de tip treaptă sau rampă. Subprogramele pentru calculul matricii de reacție după stare sînt prezentate în § 4.1, iar proiectarea estimatorilor este considerată în § 4.2. În § 4.3. se prezintă subrutine pentru proiectarea unor regulatoare cu acțiune directă (feed-forward).

4.1. Proiectarea reguletoarelor cu reacție de la stare

În proiectarea compensatoarelor prin metoda prezentată mai înainte apare frecvent următoarea problemă de calcul: să se determine matricea de reacție de la starea H din legea de comandă liniară

$$u(t) = Hx(t) \quad (26)$$

astfel încît matricea de stare a sistemului în buclă închisă, $A + BH$, să aibă anumite proprietăți dorite. În continuare vom prezenta trei metode alternative care se pot utiliza pentru calculul matricii de reacție.

Stabilizarea sistemelor liniare. Fie S o mulțime simetrică de numere complexe definită ca

$$S = \{\lambda : \operatorname{Re}(\lambda) < a\} \quad (27)$$

pentru sisteme continue sau

$$S = \{\lambda : |\lambda| < a\} \quad (28)$$

pentru sisteme discrete. *Problema de stabilizare* cere să se determine H astfel încît toate valorile proprii ale lui $A + BH$ să aparțină lui S . Condiția necesară și suficientă pentru existența soluției acestei probleme este ca sistemul (18) să fie stabilizabil în raport cu S , adică să nu aibe z.d.i. în S . Matematic acest lucru se poate scrie sub forma:

$$\operatorname{rang} [\lambda I - A \quad B] = n \quad (29)$$

pentru toți $\lambda \in S$.

Doi algoritmi de stabilizare sînt implementați în subrutinele STAC și STAD pentru stabilizarea sistemelor continue respectiv, discrete. Ambii algoritmi separă spectrul lui A în raport cu partiționarea planului complex determinată

de S. Separarea spectrului are la bază ordonarea FSR matricii A și delimitează partea instabilă a sistemului în raport cu S. Stabilizarea sistemului se face numai pentru partea sa instabilă de regulă de ordin mult mai mic decât ordinul sistemului. Condiția de stabilizabilitate (29) se testează implicit și se determină și o estimare a gradului de controlabilitate a părții instabile a sistemului. Descrierea amănunțită a algoritmilor implementați se găsește în lucrările [35], [15].

Pentru proiectarea legii de reglare (25) cu subrutinele STAC sau STAD, se recomandă construirea extinderii (24) cu EXTI.

Alocarea polilor. Fie R o mulțime simetrică de n numere complexe în raport cu axa reală. Problema de alocare a polilor cere să se determine H din (26) astfel încât spectrul lui $A+BH$ să fie R. Dacă dorim ca anumite valori proprii „bune“ ale lui A să rămână nemodificate, atunci evident le vom include în R. Condiția necesară și suficientă pentru existența soluției problemei de alocare a polilor este ca toate valorile proprii care se modifică să aparțină părții controlabile a sistemului. Matematic acest lucru necesită satisfacerea condiției (29) de toți λ aparținând acelei părți a spectrului lui A care se modifică.

Alocarea polilor se poate efectua utilizând subrutina SALOC, care implementează un algoritm propus de Varga [2], [15]. Acest algoritm este implementat într-o manieră care permite păstrarea nealterată a polilor care aparțin mulțimii S definite prin (27) sau (23). Acest lucru este realizat separînd spectrul lui A, prin intermediul formei sale Schur, în raport cu partiționarea planului complex determinată de mulțimea S. Polii se alocă printr-o procedură iterativă. La fiecare interacție se alocă unu sau doi poli. Polii complecși se alocă în perechi complex conjugate. Matricea de reacție H rezultă ca o sumă de matrici componente, fiecare alocînd unu sau doi poli. Pentru $m > 1$, norma acestor matrici componente este minimizată. Dacă toate aceste norme sînt de ordinul lui $\|A\|/\|B\|$, atunci algoritmul efectuează calculele într-o manieră numeric stabilă [36]. Condițiile de controlabilitate pentru polii modificali se testează implicit în algoritm, iar polii necontrolabili, dar stabili, sînt automat incluși printre polii care nu se modifică. Pentru toți polii modificali, SALOC evaluează gradele de controlabilitate ale acestora. Pe lingă matricea H ce alocă polii sistemului, SALOC determină și FSR a matricii de stare $A+BH$, ca și matricea de transformare ortogonală corespunzătoare. Această facilitate se poate exploata în proiectarea estimatoarelor de stare.

Pentru proiectarea legii de reglare (25) cu subrutina SALOC, se recomandă, pentru creșterea preciziei calculare, construirea extinderii (23) cu EXTI.

Optimizarea liniar-pătratică. Matricea de reacție de la stare H se poate determina optimală astfel încît legea de comandă liniară (26) să minimizeze un indice de performanță pătratic de forma

$$\int_0^{\infty} [x(t)^T Q x(t) + u(t)^T R u(t)] dt \quad (30)$$

pentru un sistem continuu, sau

$$\sum_{t=0}^{\infty} [x(t)^T Q x(t) + u(t)^T R u(t)] \quad (31)$$

pentru un sistem discret. În mod uzual se presupune că matricile de ponderare Q și R din (30) și (31) satisfac condițiile $Q_i \geq 0$ și $R_i > 0$.

Rezolvarea problemei de optimizare liniar-pătratică presupune rezolvarea unei ecuații matriciale algebrice Riccati (EMAR) de forma

$$A^T X + XA - XBR^{-1}B^T X + Q = 0 \quad (32)$$

în cazul continuu, sau de forma

$$X = A^T X A - A^T X B (R + B^T X B)^{-1} B^T X A + Q B \quad (33)$$

în cazul discret. O condiție necesară și suficientă pentru existența și unicitatea unei soluții pozitiv definite a ecuațiilor (32) sau (33) este ca perechea (A, B) să fie stabilizabilă, iar perechea (A, \sqrt{Q}) să fie detectabilă, unde \sqrt{Q} este rădăcina pătrată matricială al lui Q .

Dacă cunoaștem soluția X a EMAR, atunci matricea de reacție optimală H rezultă cu formulele

$$H = R^{-1} B^T X \quad (34)$$

în cazul continuu, sau

$$H = -(R + B^T X B)^{-1} B^T X A \quad (35)$$

în cazul discret. Pentru evaluarea acestor formule se poate folosi subrutina OPTR.

Rezolvarea EMAR se poate face utilizând mai multe subrutine din BIMAS. Metoda iterativă Newton este implementată în subrutinele NINC și NIND (incluse de asemenea în BIMASC) pentru rezolvarea EMAR continue și respectiv discrete. Ambele rutine evaluează și matricea de reacție optimală. Metoda Newton necesită inițializarea procesului iterativ cu o matrice de reacție stabilizatoare. În acest scop, se pot utiliza fie SALOC, fie STAC, sau STAD. Algoritmul implementat este descris în [37].

Metode directe de rezolvare a EMAR sînt implementate în subrutinele EXTC și SCHV pentru sisteme continue și EXT2 și GSCHV pentru sisteme discrete cu matricea de stare inversabilă. SCHV implementează metoda propusă de Laub [3]. Pentru sisteme discrete cu matricea de stare singulară, se pot utiliza subrutinele EXT2 și GSCHV. GSCHV a fost elaborată pe baza metodei propuse de Van Dooren [5]. O prezentare a acestor metode se găsește în lucrarea [15].

Notă: subrutina OPTR și subrutinele pentru rezolvarea EMAR au fost programate de dr. ing. V. Sima.

4.2. Proiectarea estimatoarelor de stare

Implementarea legilor de comandă după stare de forma (25) sau (26) necesită cunoașterea vectorului de stare sau a unei estimații x_0 a acestui vector. Un estimator de stare de ordin „n” pentru sistemul (18) este descris de ecuațiile

$$\lambda z(t) = Fz(t) + Gy(t) + Bu(t) + Ew(t) \quad (36)$$

$$x_0(t) = z(t)$$

Din condiția ca eroarea de estimare

$$e(t) = x(t) - x_e(t) \quad (37)$$

să se anuleze pentru $t \rightarrow \infty$, rezultă următoarea expresie pentru matricea de stare a estimatorului:

$$F = A - GC \quad (38)$$

În care G trebuie astfel determinat încît F să aibă valori proprii stabile. Dacă perechea (A, C) este detectabilă, matricea G se poate calcula prin oricare dintre metodele descrise în secțiunea 3.1, utilizînd în locul perechii (A, B) , perechea duală (A^T, C^T) . Dacă se utilizează SALOC, matricea rezultată F^T este în FSR. SALOC determină de asemenea matricea de transformare ortogonală. Dacă s-a determinat în prealabil matricea de reacție H din (25) sau (26), atunci se poate obține, printr-o similaritate ortogonală aplicată sistemului $(F, [G, B, E], H)$ utilizînd ORTEQ, ca matricea F să rezulte în FSR. Această proiectare a compensatorului format din ecuațiile (25) și (36) conduce la o reducere semnificativă a numărului de operații cerut de implementarea „on-line” a estimatorului. O reducere suplimentară a numărului de operații rezultă prin bloc-diagonalizarea lui F .

Dacă se utilizează tehnica de optimizare liniar-pătratică pentru determinarea lui G , atunci matricile de ponderare care apar în EMAR asociată, au semnificații precise. Q este matricea de covarianță a zgomotului de la intrare $w(t)$, iar R este matricea de covarianță a zgomotului de măsurare. Estimatorul rezultat se numește *filtru Kalman-Bucy* în cazul continuu, sau *predictor Kalman* în cazul discret.

Deși simplu de proiectat și de implementat, estimatorul de ordin „ n ” are o anumită redundanță, deoarece determină o estimare a întregului vector de stare, cînd de fapt o parte a acestui vector se poate calcula direct din ieșirile măsurate. Astfel, pentru sistemul (18), putem utiliza un estimator de stare minimal de ordin $n-p$ ($p = \text{rang } C$) descris de ecuațiile

$$\begin{aligned} \lambda z(t) &= Fz(t) + Gy(t) + Lu(t) + Kw(t) \\ x_e(t) &= My(t) + Nz(t) \end{aligned} \quad (39)$$

O procedură eficientă de proiectare a unui estimator de stare prin alocarea polilor a fost propusă de Varga [34]. Această procedură este implementată în subrutina SAESTM. Estimatorul rezultat are matricea de stare în FSR. O reducere a lui F la forma bloc-diagonală se poate efectua printr-o transformare de similaritate aplicată sistemului $(F, [G, L, K], N)$.

4.3. Proiectarea reguletoarelor cu acțiune directă

În cazul unui sistem stabil asupra căruia acționează semnale exterioare (referințe și perturbații) constante, regulatorul cu acțiune directă de la referințe și perturbații este descris de relația

$$u(t) = H_r r(t) + H_d w(t) \quad (40)$$

H_r și H_d se determină cu formulele

$$H_r = -[G_r(s)]^\# \quad H_d = [G_r(s)]^\# G_d(s) \quad (41)$$

în care # desemnează pseudo-inversa, $G_r(\lambda)$ și $G_d(\lambda)$ sînt matrici de transfer date de

$$G_r(\lambda) = C_r(\lambda I - A)^{-1} B, \quad G_d(\lambda) = C_r(\lambda I - A)^{-1} E \quad (42)$$

În (41), $s=0$ pentru un sistem continuu și $s=1$ pentru un sistem discret.

În cazul cînd modelul sistemului este descris exact de ecuațiile (18), regulatorul cu acțiune directă (40) asigură eroare staționară de reglare nulă. Totuși de regulă acest regulator nu se poate folosi singur, deoarece asupra proceselor acționează și perturbații nemăsurabile ale căror efecte nu se pot elimina decît prin utilizarea unui regulator cu reacție.

Pentru proiectarea reguletoarelor cu acțiune directă se pot utiliza subrutinele MTFF și STFF. MTFF calculează H_r și H_d direct din matricile de transfer (42), iar STFF calculează aceste matrici pe baza modelului de stare (18).

5. Simularea sistemelor

Simularea sistemelor are un dublu scop. În primul rînd, simularea este un puternic instrument de analiză a comportării dinamice a sistemelor. Prin simulare se pot obține o serie de informații cantitative și calitative utile despre proces, cum sînt: caracterul regimului tranzitoriu (oscilant sau amortizat), valorile suprareglajelor la modificarea unor referințe, durata regimului tranzitoriu la acțiunea unor perturbații, evidențierea interconexiunilor existente în sistem. De asemenea, prin simulare se poate verifica acuratețea unor aproximații făcute la deducerea modelelor cum sînt: liniarizarea unor modele neliniare, reducerea ordinului sau discretizarea unor modele continue. În al doilea rînd, simularea este un auxiliar prețios în proiectarea sistemelor de comandă, deoarece este modalitatea prin care putem evalua performanțele obținute în urma proiectării sau putem compara performanțele diferitelor structuri de comandă în vederea fixării structurii celei mai simple de implementat, care satisface specificațiile de proiectare.

Subprogramele prezentate în acest capitol nu efectuează simulări, ci constituie mai degrabă instrumente utile ce se pot utiliza în programe de simulare a unor sisteme neliniare sau liniare. Aceste programe se vor elabora de utilizatori în funcție de scopul concret urmărit.

5.1. Integrarea numerică a ecuațiilor diferențiale

Considerăm problema integrării ecuațiilor diferențiale ordinare (EDO) de ordin întii cu condiții inițiale

$$\frac{dx(t)}{dt} = F(t, x), \quad x(t_0) = x_0, \quad t_0 \leq t \leq t_f \quad (43)$$

în care x este un vector cu n componente, iar $F(t, x)$ este o funcție vectorială pe asemenea cu n componente. Pentru integrarea numerică a sistemului (43)

am luat în considerare două clase de metode. Pentru sisteme cu dinamică uniformă (Jacobianul $J = \partial F / \partial x$ are valori proprii cu ordine de mărime apropiate) se utilizează metode Runge-Kutta-Fehlberg de ordinul 4—5. Această metodă este implementată în cadrul pachetului RKF45, una dintre implementările cele mai performante și sigure existente în momentul de față [11]. Pentru integrarea EDO „stiff”, pentru care Jacobianul lui F are o dispersie foarte mare a valorilor proprii, se utilizează metode implicite în mai mulți pași. Unul dintre pachetele cele mai puternice existente la ora actuală pe plan mondial este pachetul LSODE, care implementează două metode de integrare cu pas și ordin variabile: metode Adams de ordin pînă la 12 pentru EDO ne-„stiff” și metoda diferențelor inverse (metoda Gear) de ordin pînă la 5, pentru probleme „stiff”. Trebuie să subliniem că ambele pachete au mai multe opțiuni de control a erorilor de integrare și permit luarea în considerare cu ușurință a unor structuri particulare a unor probleme. De exemplu LSODE poate trata distinct probleme care au matricea Jacobian în formă de bandă. Aceste rutine stau la baza metodelor de simulare implementate în pachetul CASAD [23].

5.2. Simularea structurilor de comandă liniare

Considerăm sistemul liniar constant descris de ecuațiile (18). Simularea sistemului (18) în buclă deschisă se face deseori presupunînd anumite forme de semnal particulare (treaptă, rampă) pentru u , w sau r . În cazul simulării sistemului în buclă închisă, comanda se calculează sub forma:

$$u(t) = u_1(t) + u_2(t) \tag{44}$$

unde u_1 este termenul de reacție, iar u_2 este termenul de acțiune directă. Ambii termeni în (44) sînt opționali.

Termenul de reacție u_1 se poate genera în una din următoarele forme:

$$\begin{aligned} u_1(t) &= Hx_e(t) \\ u_1(t) &= Hy_r(t) \\ u_1(t) &= H_1x_1(t) \\ u_1(t) &= Hx_e(t) + H_1x_1(t) \\ u_1(t) &= Hy_r(t) + H_1x_1(t) \end{aligned} \tag{45}$$

În (45) x_e este fie starea sistemului (dacă nu se folosește estimator de stare) fie estimăția vectorului de stare generată de un estimator de ordin „ n ” (36) sau un estimator minimal (39); x_1 este vectorul de stare al modelului interu (20), în care eroarea se poate calcula în unul din următoarele moduri

$$\begin{aligned} e(t) &= r(t) - y_r(t) \\ e(t) &= y_m(t) - y_r(t) \end{aligned} \tag{46}$$

În (46), y_m este ieșirea unui model de referință descris de

$$\begin{aligned} \lambda x_m(t) &= A_m x_m(t) + B_m r(t) \\ y_m(t) &= C_m x_m(t) \end{aligned} \tag{47}$$

Termenul cu acțiune directă din (44) poate avea una dintre expresiile următoare :

$$\begin{aligned} u_2(t) &= II_r(t) \\ u_2(t) &= H_d w(t) \\ u_2(t) &= II_r(t) + H_d w(t) \end{aligned} \quad (48)$$

Ecuatiile (18), (36) sau (39), (20), (44) — (48) formează împreună un sistem extins de forma

$$\begin{aligned} \lambda v(t) &= A'v(t) + E'w(t) + E''r(t) \\ y_r(t) &= C'v(t) \end{aligned} \quad (49)$$

în care v este vectorul de stare format din vectorii de stare x , z , x_m și x_1 .

Pentru simularea sistemului (49) se pot utiliza două abordări. Prima abordare se bazează pe formarea matricilor A' , E' și E'' din (49) și discretizarea sistemului (49) pentru o perioadă de eșantionare aleasă de utilizator. Această metodă este considerată și analizată în lucrarea [14]. Deși are câteva caracteristici atractive (simularea discretă este foarte ușoară), această abordare are unele inconveniente, cum sînt: dificultăți în calculul exponențialelor matriciale pentru sisteme mari sau perioade de eșantionare mari, lipsa controlului asupra erorilor, nu permite exploatarea structurii particulare a unor probleme.

A doua abordare constă în utilizarea și adaptarea programelor de uz general pentru integrarea numerică a EDO [9], cum sînt RKF45 sau LSODE. Această abordare permite un control riguros al erorilor și exploatarea structurii particulare a unor probleme. Utilizarea acestei abordări presupune scrierea de către utilizator a rutinei de evaluare a derivatelor (membrul drept din prima ecuație (49)). În acest scop s-a elaborat rutina GSTEP, care evaluează (49) fără a forma explicit matricile A' , E' , E'' sau C' .

Simularea structurilor de comandă discrete se efectuează prin apelarea repetată a lui GSTEP. Dacă se utilizează RKF45 sau LSODE pentru simularea sistemelor continue, GSTEP stă la baza rutinei de evaluare a derivatelor. Pentru reducerea numărului de operații necesare evaluării de către GSTEP a derivatelor, matricile A , F , A_m și A_1 se pot reduce prin similaritate la forma Hessenberg înainte de primul apel al lui GSTEP. Pentru simularea sistemelor „stiff” cu LSODE, se evaluează și Jacobianul problemei — A' , sau al unei aproximații a acestei matrici. O aproximare bloc diagonală a Jacobianului este descrisă în [10]. Aceste idei stau la baza metodelor de simulare implementate în pachetul CASAD [23]

5.3. Facilități grafice

Pentru reprezentarea grafică a mai multor funcții de timp se poate utiliza subrutina AGR, care permite folosirea unui set de caractere de reprezentare specificat de utilizator. Scările de reprezentare pot fi furnizate de utilizator sau calculate automat. Lățimea reprezentării poate fi de 80 sau 132 coloane.

6. Concluzii

În această lucrare s-a prezentat un pachet, BIMASC, de subprograme FORTRAN destinat modelării, analizei, proiectării și simulării sistemelor automate multivariabile. De asemenea, s-au făcut unele referiri și la pachetul BIMAS, (care va fi prezentat într-o lucrare viitoare) destinat rezolvării unor probleme numerice de bază din domeniile de mai sus. BIMAS și BIMASC constituie o bază puternică de programe performante pentru proiectarea asistată de calculator a sistemelor automate. Pe baza acestor pachete s-a elaborat un puternic pachet Interactiv de proiectare a sistemelor automate, denumit CASAD [23]. BIMAS, BIMASC și CASAD constituie a doua generație de programe cu caracter aplicativ (după pachetul SIPAC [38]) destinate problemelor de automată, elaborate la noi în țară. Aceste pachete sînt la nivelul celor mai bune pachete existente pe plan mondial.

BIBLIOGRAFIE

1. P. Van Dooren, The generalized eigenstructure problem in linear system theory, IEEE Trans. Autom. Control, vol. AC-26, p. 111–129, 1981.
2. A. Varga, Numerically stable algorithm for standard controllability form determination, Electronics Letters, vol. 17, p. 74–75, 1981.
3. A. J. Laub, A Schur method for solving the algebraic matrix Riccati equations, IEEE Trans. Autom. Control, vol. AC-24, p. 913–921, 1979.
4. T. Pappas, A. J. Laub, N. R. Sandell, On the numerical solution of the discrete-time algebraic Riccati equation, IEEE Trans. Autom. Control, vol. AC-25, p. 631–641, 1980.
5. P. Van Dooren, A generalized eigenvalue approach for solving the Riccati equations, Rep. NA-80.02, Comp. Scie. Dept., Stanford Univ., 1980.
6. G. S. Mirmis, C. C. Paige, An algorithm for pole assignment of time-invariant multi-input linear systems, 21-st IEEE Conf. on Decision and Control, San Diego, 1982.
7. A. Varga, A Schur method for pole assignment, IEEE Trans. Autom. Control, vol. AC-26, p. 517–519, 1981.
8. C. F. Van Loan, Computing integrals involving matrix exponentials, IEEE Trans. Autom. Control, vol. AC-23, p. 365–404, 1978.
9. W. Enright, On efficient and reliable numerical solution of large linear systems of ODE's, IEEE Trans. Autom. Control, vol. AC-24, p. 905–908, 1979.
10. A. Varga, V. Sima, C. Y. Varga, On numerical simulation of linear continuous control systems, Preprints of SIMULATION'83 Symposium, Praga, iunie 1983.
11. G. E. Forsythe, M. A. Malcolm, C. B. Moler, Computer methods for mathematical computations, Prentice-Hall, Englewood Cliffs, 1977.
12. A. C. Hindmarsh, LSODE and LSODI, two new initial value ordinary differential equation solvers, ACM-Signum Newsletter, vol. 15, p. 10–11, 1980.
13. A. Varga, V. Sima, Tehnici numerice de calcul în teoria sistemelor liniare : (I) Tehnici numerice de bază ale algebrei liniare, AMC, vol. 34, p. 135–168, Editura tehnică, 1981.
14. A. Varga, Tehnici numerice de calcul la teoria sistemelor liniare : (II) Analiza sistemelor liniare, AMC, vol. 35, p. 29–61, Editura tehnică, 1983.
15. A. Varga, V. Sima, Tehnici numerice de calcul în teoria sistemelor liniare : (III) Proiectarea sistemelor automate, AMC, vol. 36, p. 5–49, Editura tehnică, 1983.
16. C. Lawson, R. Hanson, D. Kincaid, F. Krogh, Basic linear algebra subprograms for Fortran usage, ACM Trans. Math. Software, vol. 5, p. 303–323, 1979.

17. *J.J. Dongarra, J.R. Bunch, C.B. Moler, G.W. Stewart*, LINPACK User's guide, SIAM, Philadelphia, 1979.
18. *B.T. Smith, J.M. Boyle, J.J. Dongarra, B.S. Garbow, Y. Ikebe, V.C. Klema, C.B. Moler*, Matrix eigensystem routines – EISPACK guide, Lect. Notes Comp. Scie, vol. 6, Springer Verlag, Berlin, 1974.
19. *B.S. Garbow, J.M. Boyle, J.J. Dongarra, C.B. Moler*, Matrix eigensystem routines – EISPACK guide extension, Lect. Notes Comp. Scie, vol. 51, Springer Verlag, Berlin, 1977.
20. *A. Varga, V. Sima*, BIMAS – a basic mathematical package for computer-aided systems analysis and design, Preprints of VII-th IFAC World Congress, Budapest, June, 1984 (Report ICI, TR-01.83, 1983).
21. *A. Varga, V. Sima*, BIMAS – General Description, Report ICI, TR-03.82, 1982
22. *A. Varga*, BIMASC – General Description, Report ICI, TR-01.83, 1983.
23. *A. Varga*, CASAD – General Description, Report ICI, TR-12.83, 1983.
24. *C.A. Bavelly, G.W. Stewart*, An algorithm for computing reducing subspaces by block diagonalization, SIAM J. Numer. Anal., vol. 10, p. 359–367, 1979.
25. *B.C. Moore*, Principal component analysis in linear systems : controllability, observability and model reduction, IEEE Trans. Autom. Control, vol. AC-26, p. 17–32, 1981.
26. *A.J. Laub*, On computing „balancing“ transformations, Preprints of JACC Symp., San Francisco, August, 1980.
27. *A. Varga*, Contribuții la conducerea robustă cu calculator a proceselor continue, Teză de doctorat, I.P. București, 1981.
28. *A. Varga, V. Sima*, A numerically stable algorithm for transfer function matrix evaluation, Int. J. Control, vol. 33, p. 1123–1133, 1981.
29. *R.C. Ward*, Numerical computation of the matrix exponential with accuracy estimate, SIAM J. Numer. Anal., vol. 14, p. 600–610, 1977.
30. *W.M. Wonham*, Linear multivariable control. A Geometric approach, Springer Verlag, Berlin, 1979.
31. *E.J. Davison, A. Goldenberg*, The robust control of a general servomechanism problem : the servocompensator, Automatica, vol. 11, p. 461–471, 1975.
32. *H.H. Rosenbrock*, State-space and multivariable theory, Nelson, 1970.
33. *A. Emami-Naeini, P. Van Dooren*, Computation of zeros of linear multivariable systems, Automatica, vol. 18, p. 415–430, 1982.
34. *A. Varga*, Computer aided design of robust compensators by pole assignment, Preprints of SOCOCO'82 Symp. Madrid, sept. 1982.
35. *A. Varga*, On stabilization algorithms for linear time-invariant systems, Rev. Roum. Scie. Techn. – Electrotechn. et Energ., vol. 26, p. 115–124, 1981.
36. *A. Varga*, The numerical stability of an algorithm for pole assignment, Preprints of IFAC Symp. on CAD of Multivariable Technological Systems, Purdue University, West Lafayette, sept., 1982.
37. *V. Sima*, On the real Schur form in linear control system design, Rev. Roum. Scie. Techn. – Electrotech. et Energ., vol. 25, p. 625–632, 1980.
38. *Th. Popescu, V. Sima, A. Varga, C. Vasiliu*, SIPAC – Sistem de programe pentru identificarea proceselor și proiectarea asistată de calculator a sistemelor automate. Manual de utilizare. ICI, Lab. Calc. Proces, 1978.

Ciclul
„proiectarea asistată
de calculator“
Construcții de mașini

**PROIECTAREA-OPTIMIZAREA ASISTATĂ
DE CALCULATOR A PROCESELOR TEHNOLOGICE
ÎN CONSTRUCȚIA DE MAȘINI. SISTEMUL PAPT-1**

Nicolae-Valentin Ivan
Universitatea din Brașov

1. Introducere

Competitivitatea unui anumit produs, pe orice fel de piață, este tot mai mult hotărâtă de costurile legate de tehnologie. Prin urmare, acesteia trebuie să i se acorde o atenție aparte, în cadrul eforturilor generale care urmăresc sporirea în tot mai mare măsură a efectelor economice favorabile, concomitent cu asigurarea unei anumite tehnicități, astfel că, punându-se problema asigurării calității dorite a produselor la un cost cât mai scăzut, proiectarea-optimizarea proceselor tehnologice se impune cu necesitate. Optimizarea proceselor tehnologice, avându-se în vedere aspectele legate de complexitate și metodă, constituie o problemă dificilă. Soluția trebuie desprinsă din formularea completă a problemei și cunoscându-se punctul de vedere ales pentru optimizare. Desigur, formularea problemei vizează aspectul matematic al acesteia și astfel apare necesitatea modelării matematice. Formularea problemei în sens matematic necesită efectuarea unei analize atente a sistemelor tehnologice de prelucrare [12, 13, 2].

Abordarea sistemică a problemei optimizării proceselor tehnologice este necesară pentru evidențierea multitudinii de factori și aspecte în interdependențele lor, precum și pentru corelarea corespunzătoare a datelor în vederea rezolvării problemei.

Sub formă de schemă-bloc, sistemul tehnologic de prelucrare este prezentat în fig. 1 [12, 13]. Trebuie precizat faptul că fig. 1 se referă la prelucrările pe mașinile-unelte așchiitoare, domeniu vizat de prezenta lucrare.

Ceea ce trebuie remarcat în mod deosebit în fig. 1 este existența fluxului informațional, care, în cazul optimizării, capătă o importanță primordială, știut fiind faptul că informația este ingredientul esențial al conducerii [4] (optimizarea trebuie privită drept componentă a unui proces de conducere). Și astfel fiind vorba de optimizarea proceselor tehnologice, are sens să se vorbească despre sistemul informațional asociat [12, 13].

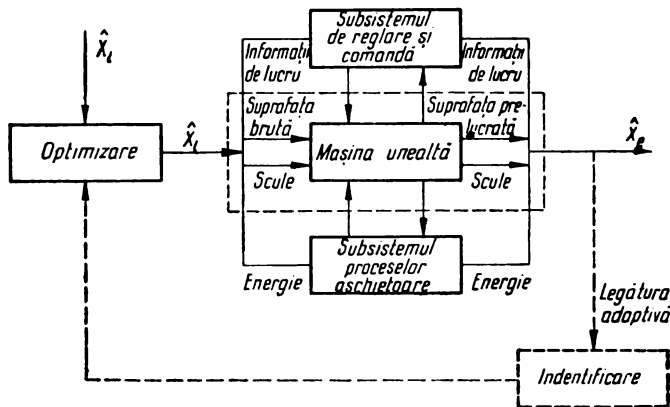


Fig. 1. Sistemul tehnologic de prelucrare.

Generalizînd [42], prin sistemul informațional asociat optimizării proceselor tehnologice trebuie să se înțeleagă ansamblul informațiilor, surselor și nivelelor consumatoare, canalelor de circulație, procedurilor și mijloacelor de tratare a informațiilor, astfel structurat încît să permită, prin intermediul unor reguli, aplicarea unor metode în vederea obținerii deciziei tehnologice optime (fig. 2 [2]).

Figurile 1, 2 evidențiază cele două moduri fundamentale de optimizare a proceselor tehnologice [35, 3, 12, 13]: optimizarea *externă* și optimizarea *internă*, diferențiate prin tehnica de rezolvare precum și prin modul în care sînt considerate informațiile.

Prezenta lucrare se înscrie pe linia optimizării externe a proceselor tehnologice de prelucrare pe mașinile-unelte așchietoare, caz în care, cu referire la fig. 2, informațiile de ieșire se referă la documentația tehnologică. Ansamblul informațiilor de intrare vizează precizia piesei, caracteristicile și mulțimea procedeelor așchietoare, echipamentele tehnologice, date contabile etc.

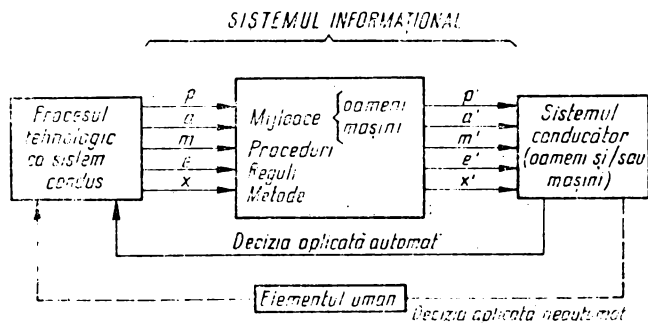


Fig. 2. Sistemul informațional asociat proiectării-optimizării proceselor tehnologice.

2. Necesitatea utilizării calculatoarelor electronice în proiectarea-optimizarea proceselor tehnologice de prelucrare prin aşchiere

Motivul principal pentru care procesele tehnologice se pretează a fi proiectate, optimizate și în general conduse cu calculatoare este faptul că informația, în asemenea activități, deține rolul esențial [44].

Implicarea calculatoarelor electronice în proiectarea-optimizarea proceselor tehnologice rezolvă două mari probleme. Prima se referă la prelucrarea, în timp util, a unui volum foarte mare de informații, iar a doua la rezolvarea unui număr relativ mare de subprobleme conexe, fiecare având mai multe grade de libertate. Avându-se în vedere optimizarea în sensul actual al noțiunii, ea nici n-ar putea fi făcută fără aportul calculatoarelor electronice care, numai ele, permit implementarea teoriilor moderne de optimizare. Tocmai de aceea optimizarea (deci și proiectarea-optimizarea) reprezintă una din funcțiile de bază ale calculatoarelor electronice.

3. Formularea problemei

Pentru formalizarea matematică a problemei proiectării-optimizării unui proces tehnologic este necesară asocierea unor variabile caracteristice precum și a relațiilor dintre ele, în scopul reflectării legităților proprii, a proprietăților care privesc destinația procesului precum și a criteriului de optimizare. Modelul matematic astfel obținut [13] va permite vehicularea după o anumită logică și o anumită tehnică a imaginilor informaționale (fig. 2) despre elementele sistemului tehnologic (fig. 1).

Generic, modelul matematic al problemei de proiectare-optimizare se poate exprima printr-un sistem de relații ce consemnează interdependențele existente între cele n variabile asociate (dependente sau nu de timp), adică :

$$\begin{cases} f(x_1, x_2, \dots, x_j, \dots, x_n) = 0, & j=1, 2, \dots, n \\ r_i(x_1, x_2, \dots, x_j, \dots, x_n) \stackrel{\Delta}{\leq} 0, & i=1, 2, \dots, m \end{cases} \quad (1)$$

în care : f este expresia analitică a criteriului de optimizare, numită funcție-obiectiv, funcție-scop, funcție-performanță sau funcție-eficiență ; r_i — cele m restricții sau constrângeri ale problemei, reflectând legitățile fizice precum și anumite conexiuni artificiale între variabile, anume create și impuse de destinația procesului (pe scurt, condițiile concrete de funcționare).

În mod corect, modelul matematic (1) reprezintă de fapt un sistem de modelare. Este preferabilă o asemenea modelare matematică, deoarece exprimarea problemei printr-un singur model matematic practic n-ar fi posibilă, datorită complexității aspectelor ce trebuie avute în vedere la proiectare-optimizare, precum și datorită interdependențelor strânse dintre ele.

Cea mai importantă problemă care trebuie rezolvată, este desigur determinarea variantei tehnologice optime. În conformitate cu ideea considerării acesteia ca o problemă de programare pseudobooleeană [1], particularizarea modelului matematic (1) duce la [1, 2, 8, 9]:

$$\begin{cases} \min CX = \min \left(\sum_{j=1}^n c_j x_j \right), j \in N = \{1, 2, \dots, n\} \\ F_i(X) \text{ rel } 0, i \in L = \{1, 2, \dots, l\} \\ x_j = 1 \text{ sau } 0 \end{cases} \quad (2)$$

unde: CX este funcția obiectiv a problemei; $F_i(X)$ — cele l condiții concrete (constrîngerile problemei); rel — oricare din operatorii relaționali: $<$, \leq , $=$, $>$, \geq ; X — vectorul coloană care descrie mulțimea celor n procedee avută inițial în vedere; C — vectorul linie ale cărui componente precizează costurile prelucrării piesei prin procedeele corespunzătoare.

Vectorul X este dat de:

$$X = [x_1 \ x_2 \ x_3 \ \dots \ x_n]' \quad (3)$$

ale cărui componente sînt variabile bivalente de cod, asociate procedeele de prelucrare ($x_j = 1$ sau 0 după cum procedeul codificat prin variabila x_j este sau nu optim).

Componentele vectorului C :

$$C = [c_1 \ c_2 \ c_3 \ \dots \ c_n] \quad (4)$$

exprimă costurile prelucrării piesei prin procedeele corespunzătoare [12, 13, 27, 8, 23].

Prin dezvoltare, partea restrictivă a modelului matematic (2) aferentă prelucrării unei porțiuni K din piesă devine [12, 13, 5, 6]:

$$\begin{cases} \sum x_j = 1, j \in N_K \\ \sum T_{ej} \cdot x_j \leq T_{PK} \\ \sum R_{ej} \cdot x_j \leq R_{PK} \end{cases} \quad (5)$$

unde: N_K este submulțimea indicilor variabilelor care codifică procedeele avute inițial în vedere, aferente prelucrării porțiunii K din piesă (aceste procedee se află într-o relație de autoexcludere); T_{ej} , R_{ej} — toleranțele, respectiv rugozitățile economice asociate procedeele codificat prin x_j ; T_{PK} , R_{PK} — toleranța respectiv rugozitatea prescrise porțiunii K a piesei.

Fiind vorba de o problemă de programare matematică în numere întregi, din categoria celor de decizii binare, după obținerea soluției [33, 34, 35] în procesul tehnologic vor trebui utilizate acele procedee pentru care $x_j = 1$.

Exprimarea analitică a funcției-obiectiv din modelul matematic (2) presupune cunoașterea regimurilor de așchiere și a adaosurilor de prelucrare, acestea fiind alte două importante subprobleme care trebuie rezolvate în cadrul

proiectării-optimizării proceselor tehnologice de prelucrare pe mașinile-unelte așchietoare.

După cum este cunoscut, determinarea regimurilor de așchiere face parte din categoria problemelor de programare matematică neliniară [14]. Prin urmare, modelul matematic (1) particularizat în acest caz este :

$$\min C(Y) \mid a_i(Y) \leq 0, Y > 0, i = \{1, 2, \dots, r\} \quad (6)$$

unde vectorul Y este :

$$Y = [v \ s]' \text{ sau } Y = [v_1 s_1 \ v_2 s_2 \ \dots \ v_q s_q]' \quad (7)$$

după cum este vorba de prelucrări cu scule singulare sau cu multiscule, și unde componentele vectorului Y reprezintă parametrii regimurilor de așchiere pentru cele q scule (viteza v de așchiere și avansul s).

$C(Y)$ respectiv $a_i(Y)$ reprezintă funcția-obiectiv (costul), respectiv cele r restricții [14, 27, 8, 11, 7, 13, 12, 30].

În conformitate cu așa-numita „metodă a planului de secțiune“ a lui Kelley [4], mulțimea programelor problemei (6) poate fi reprezentată ca intersecția unui număr suficient de mare de semispații care o conțin. Aceste semispații pot fi generate cunoscându-se un vector inițial [14, 12] :

$$Y^o = [v^o \ s^o]' \text{ sau } Y^o = [v_1^o s_1^o \ \dots \ v^o s^o]' \quad (8)$$

și asociindu-se inegalitățile :

$$\varphi_i(Y, Y^o) = a_i(Y^o) + [Y - Y^o]' \nabla a_i(Y^o) \leq 0 \quad (9)$$

pentru fiecare restricție neliniară și :

$$\psi(Y, Y^o) = C(Y^o) + [Y - Y^o]' \nabla C(Y^o) - Z \leq 0 \quad (10)$$

pentru funcția obiectiv, dacă este neliniară, în care : $\nabla a_i(Y^o)$, $\nabla C(Y^o)$ sînt gradientii funcțiilor respective pentru vectorul inițial Y^o ; Z este o nouă variabilă introdusă reclamată de metoda de rezolvare [4] și care trebuie acuma minimizată, devenind de fapt funcție-obiectiv.

Modelul matematic care trebuie rezolvat devine :

$$\min Z \mid \varphi_i(Y, Y^o) \leq 0, \psi(Y, Y^o) \leq 0, Y > 0, i = \{1, \dots, r\} \quad (11)$$

Problema (11) se rezolvă în mai multe etape. Pentru fiecare etapă modelul fiind liniar se poate aplica algoritmul SIMPLEX [4, 15], algoritmul care este cel mai corespunzător pentru domeniul programării matematice liniare [38].

În ceea ce privește adaosul de prelucrare, aspectul este și el destul de complicat, chiar dacă se are în vedere utilizarea calculatorului electronic. Mai ales din punctul de vedere al datelor inițiale, deci al băncii de date, apare complicația. Metodologia de calcul tradițională [41, 37] poate fi utilizată cu condiția algoritmizării adecvate și organizării corespunzătoare a datelor inițiale [18]. Pentru o exploatare avantajoasă a calculatorului se impune, în primul rînd, modelarea matematică a datelor inițiale experimentale [19], cunoscându-se faptul că acestea se prezintă în volum mare tabelar [41]. Adică, pentru parametri cum sînt : curbura specifică a semifabricatelor, erorile de fixare etc. trebuie stabilite formule empirice, în general de tipul :

$$y = ax^b ; y = ax + b \quad (12)$$

unde y reprezintă parametrul supus modelării, iar x — parametrul independent, în general diametrul [19, 13]. Determinarea valorilor lui a și b se poate face prin metoda celor mai mici pătrate sau alte metode [43].

După cum rezultă din cele de mai sus, subproblemele conexe proiectării-optimizării proceselor tehnologice sînt deosebit de complexe în esență, complicația matematică amplificîndu-se atunci cînd este vorba de introducerea calculatoarelor electronice (evident, în stadiul de analiză). Este firesc să fie așa, deoarece utilizarea calculatoarelor electronice nu trebuie să se rezume la aceleași metode tradiționale de prelucrare a informațiilor. Pe de altă parte, trebuie subliniat în mod deosebit că succesul unui sistem informatic depinde în foarte mare măsură de banca de date, precum și de modul în care este concepută exploatarea sistemului. Legat de acest ultim aspect, trebuie amintit și faptul că minutul-calculator costă încă destul de mult. În sfîrșit, mai trebuie adăugat că succesul unui asemenea sistem este asigurat atunci cînd utilizatorul nu este necesar să se specializeze și în domeniul calculatoarelor.

De cele de mai sus s-a ținut cont la realizarea sistemului PAPT-1, care va fi tratat în cele ce urmează. În dorința de a optimiza exploatarea calculatorului, în ceea ce privește sistemul PAPT-1, în faza de analiză s-a realizat un studiu de preoptimizare [21, 31, 32], referitor la varianta tehnologică optimă precum și la regimurile și adaosurile optime de prelucrare. Doar rezultatele acestui studiu au fost introduse în programele operative ale sistemului PAPT-1. În acest fel sistemul PAPT-1 a devenit extrem de rentabil. O anumită experiență cîștigată în domeniu face oportună sublinierea necesității studiului de preoptimizare ori de cîte ori se pornește la proiectarea unui sistem informatic de concepție-optimizare a proceselor tehnologice.

4. Sistemul PAPT-1

Creat în cadrul catedrei de T.C.M. a Universității din Brașov, sistemul PAPT-1 (Proiectarea Automată a Proceselor Tehnologice de prelucrare pe strunguri automate multiax) [12, 21, 31, 26, 17, 29, 13] a fost conceput ca un subsistem grefat pe sectorul de pregătire a fabricației și permite obținerea documentației tehnologice, sub forma planului de operații, la imprimanta calculatorului Felix C-256, fără intervenția omului în fazele de luare a deciziilor tehnologice.

Ca sistem informatic, PAPT-1 este prezentat în fig. 3, de unde rezultă cu claritate lanțurile și procedurile de tratare a informațiilor.

Sistemul este astfel conceput, încît procesul tehnologic rezultă rapid funcție doar de un minimum de informații de intrare, strict legate de particularitățile geometrice și de precizie ale piesei ale cărui proces tehnologic interesează.

Experiența cîștigată prin proiectarea și exploatarea sistemului PAPT-1 arată că atunci cînd este vorba de utilizarea calculatoarelor electronice în rezolvarea problemelor tehnologice, este avantajos să se utilizeze tipizarea proceselor tehnologice [13, 28]. În acest sens trebuie precizat că sistemul PAPT-1 are în vedere familia de piese din fig. 4 și 5. Fig. 5 se referă la sub-

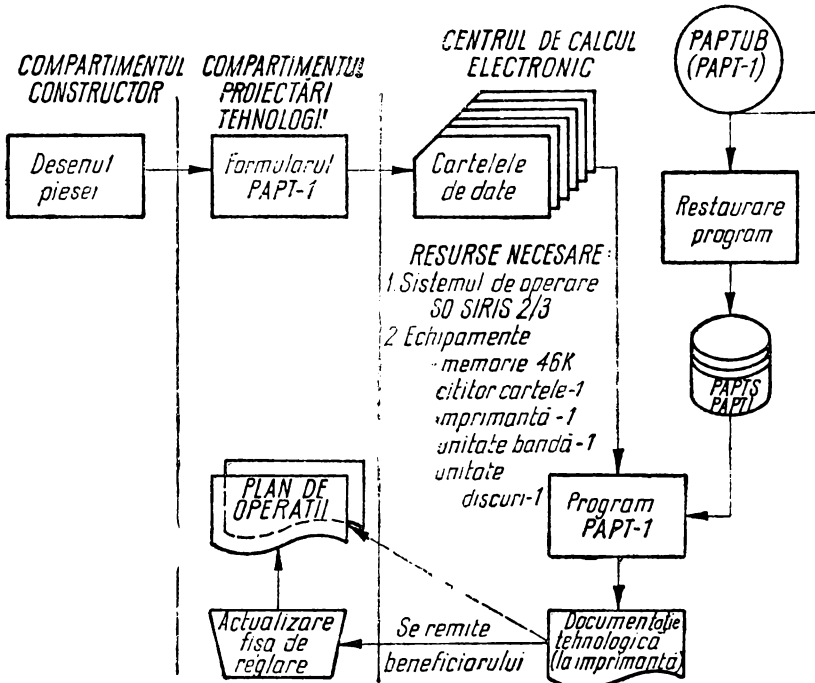


Fig. 3. Organigramma de proceduri a sistemului PAPT-1.

familia de bază cu care a început devenirea operativă a sistemului PAPT-1. Cu această ocazie trebuie menționat și faptul că aceeași experiență a demonstrat că pentru elaborarea și exploatarea unor asemenea sisteme informatice este preferabil să se lucreze cu mai multe subfamilii de piese, toate aparținând

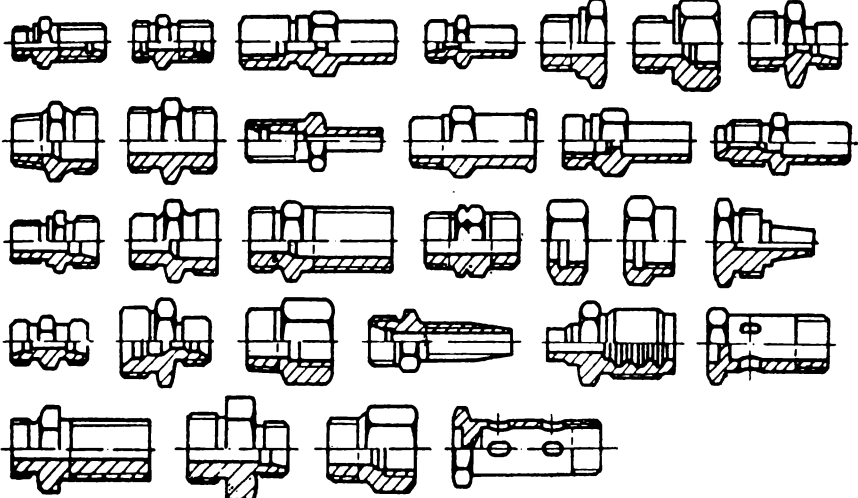


Fig. 4. Familia de piese PAPT-1.

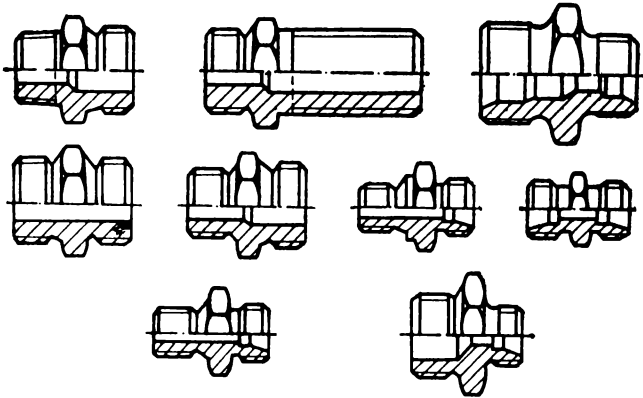


Fig. 5. Subfamilia principală de piese PAPT-1.

aceleiași familii mamă, fiecare subfamilie avînd ca reprezentativă o piesă-cod [12, 21, 31, 13].

Din organigrama de proceduri din fig. 3 rezultă modul de funcționare a sistemului.

Informațiile de intrare trebuie să plece de la serviciul tehnologic (utilizatorul) printr-un formular de intrare anume proiectat în acest sens, conform fig. 6 [21, 2]. Pe acest formular, denumit formularul PAPT-1, se vor memora, în limbaj uman, informațiile inițiale variabile (de nivel doi) care descriu geometria piesei precum și condițiile de precizie impuse acesteia. În continuare, aceste informații vor fi codificate pe cartele, obținîndu-se astfel prin perforare șase cartele de date. Urmează apoi prelucrarea acestor date de către programul PAPT-1 (de fapt un sistem de subprograme conduse de un program principal), memorat pe bandă magnetică sau chiar pe discul-sistem. În urma prelucrării celor șase cartele de date de către programul amintit, la imprimanta calculatorului va rezulta documentația tehnologică solicitată conform listei 1. Lista 1 (fig. 7) a fost obținută respectîndu-se organigrama de proceduri din figura 3 pentru piesa din fig. 8 (ca exemplu).

După cum se poate observa, documentația tehnologică din lista 1 cuprinde toate informațiile necesare reglării automatului și prelucrării piesei.

Sistemul PAPT-1 permite și obținerea schemei tehnologice necesare.

Pentru familia de piese amintită a fost concepută fișa de reglare din fig. 9 [21, 29, 13]. În această fișă se regăsesc variantele posibile de prelucrare a tuturor pieselor din familie, deci se regăsește mulțimea inițială de procedee dată de vectorul :

$$X = [C1 \ S2 \ S3 \ B4 \ S5 \ S6 \ B7 \ S8 \ S9 \ S10 \ S11 \ C12 \ C13]' \quad (13)$$

Semnificațiile variabilelor bivalente de cod ale vectorului (13) rezultă din fig. 9.

În lista 1, calculatorul din mulțimea X a procedeelelor a indicat numai cîteva ca fiind optime, listîndu-le în ordinea succesiunii tehnologice în coloana „COD”. În continuare, utilizatorul trebuie să identifice în fișa de reglare (fig. 9) schemele indicate de calculator prin variabilele de cod. Acestea vor fi eviden-

FORMULARUL PAPT-1

Nr. reper										
Denum reper										
Completat date	DATE FORTRAN									
Perforare date si rulare program	DATE FORTRAN									
C.C.F.	S 590									
Instructiuni de perforare										
Caracter										
Perforare	φ									
	cifra zero									
	1 1									

Exterior stanga

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
D6	ASD6	AID6	L6	ASL6	AIL6	RA6	ALFA6	PFS	DS	D8	ASD8	AID8	L8	ASL8	AIL8	RA8
1	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21

Exterior dreapta

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
D4	ASD4	AID4	L4	ASL4	AIL4	RA4	ALFA4	PF4	DD	D1	ASD1	AID1	L1	ASL1	AIL1	RA1
1	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21

Interior 2

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
D2	ASD2	AID2	L2	ASL2	AIL2	RA2	ALFA2	AID3	L3	ASL3	AIL3	RA3	ALFA3	60	61	62
1	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21

Interior 3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
D2	ASD2	AID2	L2	ASL2	AIL2	RA2	ALFA2	AID3	L3	ASL3	AIL3	RA3	ALFA3	60	61	62
1	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21

Alte date

1	2	3	4	5	6	7
D10	H	ALFA	MM	L	ASL	ALA
1	6	7	8	9	10	11

Observatii: Ordinea cartelelor de date este:

Nr. 1 P1
 Nr. 2 P2
 Nr. 3 P3
 Nr. 4 P4
 Nr. 5 INTREG
 Nr. 6 NR.

INTREG

1	5	10	15
1	6	11	16
1	7	12	17
1	8	13	18
1	9	14	19
1	10	15	20
1	11	16	21
1	12	17	22
1	13	18	23
1	14	19	24
1	15	20	25
1	16	21	26
1	17	22	27
1	18	23	28
1	19	24	29
1	20	25	30
1	21	26	31
1	22	27	32
1	23	28	33
1	24	29	34
1	25	30	35
1	26	31	36
1	27	32	37
1	28	33	38
1	29	34	39
1	30	35	40
1	31	36	41
1	32	37	42
1	33	38	43
1	34	39	44
1	35	40	45
1	36	41	46
1	37	42	47
1	38	43	48
1	39	44	49
1	40	45	50
1	41	46	51
1	42	47	52
1	43	48	53
1	44	49	54
1	45	50	55
1	46	51	56
1	47	52	57
1	48	53	58
1	49	54	59
1	50	55	60
1	51	56	61
1	52	57	62
1	53	58	63
1	54	59	64
1	55	60	65
1	56	61	66
1	57	62	67
1	58	63	68
1	59	64	69
1	60	65	70
1	61	66	71
1	62	67	72
1	63	68	73
1	64	69	74
1	65	70	75
1	66	71	76
1	67	72	77
1	68	73	78
1	69	74	79
1	70	75	80
1	71	76	81
1	72	77	82
1	73	78	83
1	74	79	84
1	75	80	85
1	76	81	86
1	77	82	87
1	78	83	88
1	79	84	89
1	80	85	90
1	81	86	91
1	82	87	92
1	83	88	93
1	84	89	94
1	85	90	95
1	86	91	96
1	87	92	97
1	88	93	98
1	89	94	99
1	90	95	100

NR (rep)

1	5	10	15
1	6	11	16
1	7	12	17
1	8	13	18
1	9	14	19
1	10	15	20
1	11	16	21
1	12	17	22
1	13	18	23
1	14	19	24
1	15	20	25
1	16	21	26
1	17	22	27
1	18	23	28
1	19	24	29
1	20	25	30
1	21	26	31
1	22	27	32
1	23	28	33
1	24	29	34
1	25	30	35
1	26	31	36
1	27	32	37
1	28	33	38
1	29	34	39
1	30	35	40
1	31	36	41
1	32	37	42
1	33	38	43
1	34	39	44
1	35	40	45
1	36	41	46
1	37	42	47
1	38	43	48
1	39	44	49
1	40	45	50
1	41	46	51
1	42	47	52
1	43	48	53
1	44	49	54
1	45	50	55
1	46	51	56
1	47	52	57
1	48	53	58
1	49	54	59
1	50	55	60
1	51	56	61
1	52	57	62
1	53	58	63
1	54	59	64
1	55	60	65
1	56	61	66
1	57	62	67
1	58	63	68
1	59	64	69
1	60	65	70
1	61	66	71
1	62	67	72
1	63	68	73
1	64	69	74
1	65	70	75
1	66	71	76
1	67	72	77
1	68	73	78
1	69	74	79
1	70	75	80
1	71	76	81
1	72	77	82
1	73	78	83
1	74	79	84
1	75	80	85
1	76	81	86
1	77	82	87
1	78	83	88
1	79	84	89
1	80	85	90
1	81	86	91
1	82	87	92
1	83	88	93
1	84	89	94
1	85	90	95
1	86	91	96
1	87	92	97
1	88	93	98
1	89	94	99
1	90	95	100

Fig. 6. Formularul PAPT-1.

S I S T E M U L P A P T - 1

STRUNG AUTOMAT CU 6 AXE - MODEL : 1A 240-6
 SECTIA:X73 REPER NR.43-5731-0258
 CONDITII RACIRE: ULEI RACIRE STAS 2800-61
 NR.

FAZA	SUPPORT	COD	CONTINUT	FAZA	V	N	S	L
1	TRANSV	S03	STRUNJ.TRANSV.	DEGRD.D13=28.49 L13=17.00	36.	314.	.019	5.15
1	LONGIT	C01	CENTRUIRE	D11=15.00 ALFA11=90	15.	314.	.113	31.00
2	LONGIT	S02	STRUNJ.LONG.	DEGRD.D12=28.49 L12=12.20	36.	314.	.113	31.00
3	TRANSV	S05	STRUNJ.TRANSV FINIS.PROFIL	D6=25.49 AID6=-.13 L6=12.00 ALFA6=18. D14=23.21 L16=17.00 PS=2 RA6=6.3	28.	314.	.013	3.64
4	TRANSV	S08	STRUNJ.TRANSV FINIS.PROFIL	D4=25.49 AID4=-.13 L4=12.00 ALFA4=18. D15=23.21 PD=2 RA4=6.3	28.	314.	.005	2.50
5	LONGIT	B04	BURC.PRIMA PORTIUNE	D16=15.00 L14=30.00	15.	314.	.113	31.00
6	TRANSV	S13	RETEZARE	L=30.00	36.	314.	.022	6.11

TURATIA AXE:314.(ROT/MIN)
 ROTI SCHIMB:A=37 B=47 V=22 G=62
 NUMAR TURE :274.(ROT)
 ROTI SCHIMB:N=27 M=57 K=44 L=40
 TIMP/BUC.PIEASA:54.3(SEC)

Fig. 7. Documentația tehnologică obținută prin sistemul PAPT-1.

fiate, de exemplu, printr-un chenar, iar cele rămase vor fi anulate prin barare. Cuplind lista obținută la caculator cu fișa de reglare actualizată ca mai sus (eventual copertare) se obține documentația tehnologică sub forma unui variabil plan de operații în specificul prelucrărilor pe strunguri automate multiax.

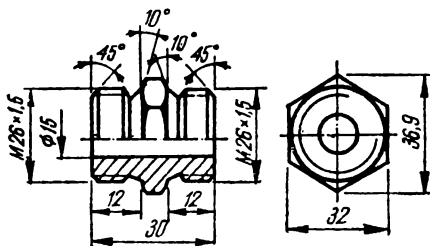


Fig. 8. Piesa al cărui proces tehnologic trebuie proiectat (ca exemplificare).

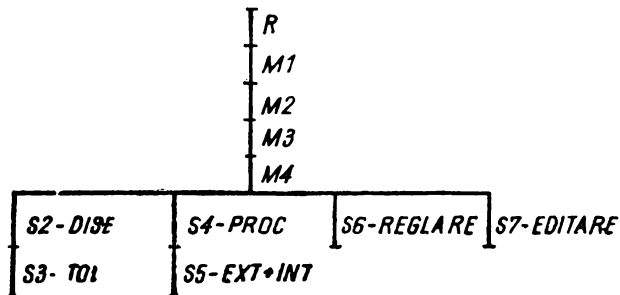


Fig. 10. Segmentarea programului.

În ceea ce privește sistemul de programe PAPT-1, acesta, principal, este prezentat în fig. 10. Pentru a se depăși dificultățile legate de memoria insuficientă a calculatorului Felix C-256, s-a recurs ca soluție de rezolvare la segmentare, limbajul de programare fiind FORTRAN [39].

Particularitățile precum și metodologia de proiectare a proceselor tehnologice în specificul prelucrărilor pe strunguri automate multiax se consideră cunoscute [40, 45].

Sistemul de programare PAPT-1 se bazează pe un program principal (segmentul rădăcină) care gestionează după necesități 7 subprograme de tip SUBROUTINE (segmente). Aceste subprograme sînt: DISE TOL, PROC, EXT, XINT, REGLARE și EDITARE. Subproblemele rezolvate de acestea, pe scurt, sînt prezentate mai jos.

Subprogramul DISE [21, 12] rezolvă problema prelucrării filetelor, stabilind tehnologia de prelucrare prin așchiere sau rulare, determinînd totodată parametrii dimensionali (cote nominale și abateri) ai suprafeței pe care ulterior se va realiza filetul [9, 10].

Subprogramul TOL calculează toleranțele optime pentru dimensiunile tehnologice intermediare în funcție de procedeul așchietor implicat și cota nominală a suprafeței ce trebuie prelucrată [21].

Subprogramul PROC [21] este acel subprogram care are ca sarcină stabilirea procedeele optime de prelucrare. De fapt subprogramul PROC rezolvă problema prin intermediul celor două subprograme EXT și XINT [21]. Subprogramul EXT stabilește procedeele optime de prelucrare pentru toate suprafețele exterioare ale piesei, determinînd totodată parametrii caracteristici ca: factorii regimului de așchiere, adaosurile de prelucrare, indicii de prioritate necesari ordonării procedeele într-o anumită succesiune tehnologică, cursele de lucru, alte calcule etc. Subprogramul XINT realizează același lucru ca și EXT, dar pentru suprafețele interioare ale piesei.

Legat de subprogramele PROC, EXT și XINT, trebuie făcută precizarea că procedeele optime de prelucrare sînt decise din mulțimea X a procedeele avute inițial în vedere în baza caracteristicilor geometrice și de precizie ale piesei (date prin formularul PAPT-1). Alte criterii care trebuie avute în vedere la selectarea optimă a procedeele au fost utilizate în faza de preoptimizare.

Subprogramul REGLARE [21] rezolvă problema punerii de acord a datelor calculate în cadrul celorlalte subprograme cu posibilitățile reale ale mașinii-

unelte implicate, stabilind și roțile de schimb pentru reglarea lanțurilor cinematice.

Subprogramul EDITARE [21, 20, 22] este subprogramul care rezolvă problema editării la imprimanta calculatorului, sub forma planului de operații, a documentației tehnologice calculate. De asemenea, tot aici se determină și norma de timp pe bucată piesă.

În principiu sistemul de programe PAPT-1 este un sistem modular și astfel acest sistem poate fi extins la noi și noi subfamilii de piese, lărgindu-se în acest fel plaja de valabilitate a întregului sistem PAPT-1.

5. Concluzii

În condițiile actuale de dezvoltare tehnologică, introducerea sistemelor informatice în proiectarea-optimizarea proceselor tehnologice se dovedește a fi mai mult decât necesară, eficiența tehnico-economică fiind argumentul [16].

În primul rând trebuie subliniat faptul că productivitatea în activitatea concepțională a proceselor tehnologice crește enorm de mult. Avîndu-se în vedere sistemul PAPT-1, trebuie arătat că durata obținerii documentației tehnologice pe cale informatică este de numai 35—60 de minute, în funcție de complexitatea piesei. Pe cale tradițională, aceeași documentație se obține în circa 25—30 de ore [21, 13, 22]. În același timp munca tehnologului s-a îmbunătățit considerabil. Dar nu numai eficiența economică în munca de proiectare crește. Întreaga activitate a tehnologului se ridică acum la un nivel calitativ superior, acesta printre altele avînd mai mult timp la dispoziție pentru asistența tehnică, atît de necesară în atelier.

Paralel cu cele de mai sus, proiectarea asistată de calculator asigură optimizarea procesului tehnologic, cu toate consecințele referitoare la costul și productivitatea uzinării pieselor. Un alt mare avantaj al sistemului PAPT-1 este acela că nu necesită un personal utilizator specializat în calculatoare.

În încheiere, trebuie menționat faptul că, prin prisma modului în care s-au pus și s-au rezolvat problemele, sistemul PAPT-1 poate fi generalizat și la prelucrările altor tipuri de piese.

BIBLIOGRAFIE

1. BJORKE, O., HAUGRUD, B., *Mathematical Methods in Planning of Machining Operations*. Annals of the C.I.R.P., XVI, 4, 399—407 (1968).
2. CARRO, CAO, G., *Considerations sur le système machine-plèce-outil à l'égard de l'optimisations des usinages par enlèvement de copeau*. Annals of the C.I.R.P., XVIII, 4, 633—652 (1970).
3. CHIRIACESCU, T.S., *Automatizarea proceselor tehnologice*. Universitatea din Brașov, 1975.
4. DRAGOMIRESCU, M., MALIȚA, M., *Programarea pătratică*. Ed. științifică, București, 1968.
5. DRĂGHICI, G., IVAN, N.V., *Über die automatische Entwerfung der technologischen Vorgänge mit Verwendung von elektronischen Rechenmaschinen*. Vorträge „COMPCONTROL '72“ — Anwendung der Elektronenrechner in der Leitung der Industrieunternehmen, III. Section, Sopron, 115—130 (1972).

6. Drăghici, G., Ivan, N.V. und Jacobs, H.J., Die Boolesche Algebra als Hilfsmittel zur optimalen Gestaltung diskreter technologischer Prozesse der mechanischen Teillfertigung (Prozessoptimierung). Wiss. Z. Techn. Universität Dresden, 23, 2, 385—391 (1974).
7. Drăghici, G., Ivan, N.V., Programarea liniară mai bună ca programarea convexă în unele cazuri de determinare a parametrilor optimi de exploatare a seulelor. Construcția de mașini, XXIX, 12, 598—601, (1977).
8. Drăghici, G., Ivan, N.V., Despre modelarea subsistemului de caleul al regimurilor de aşchiere utilizate la strungurile automate multiax și luarea deciziilor optime cu ajutorul calculatoarelor electronice. Comunicările conferinței din domeniul proceselor și utilajelor de prelucrare la rece. Institutul politehnic „Traian Vuia“ Timișoara, 49—58 (1970).
9. Drăghici, G., Ivan, N.V., Stroe, M., Berechnung des Ausgangsdurchmessers beim Gewinderollen. Werkstatt und Betrieb, 104, 10, 755—758 (1971).
10. Drăghici, G., Ivan, N.V., Stroe, M., Determinarea abaterilor admisibile de la diametrul semifabricatelor în cazul rulării filetelor metalice. Construcția de mașini, XXV, 8, 471—474 (1973).
11. Drăghici, G., Chiriacescu, T.S., Calculul durabilității seulei și regimului de aşchiere folosind programarea matematică. St. Cerc. Mec. Apl., 30, 4, 975—999 (1971).
12. Ivan, N.V., Contribuții asupra sistemelor de concepție a proceselor tehnologice. Modele și rezolvări cu ajutorul calculatoarelor electronice. Teză de doctorat. Universitatea din Brașov, 1977.
13. Ivan, N.V., Bazele optimizării proceselor tehnologice în construcția de mașini. Curs, Universitatea din Brașov, 1983.
14. Ivan, N.V., Calculul regimurilor optime de aşchiere prin programare convexă. Construcția de mașini, XXV, 3, 151—156 (1973).
15. Ivan, N.V., Programarea algoritmului SIMPLEX pe calculatoare electronice, în vederea calculului regimurilor optime de aşchiere. Buletinul Universității din Brașov, A, XIV, 407—416 (1972).
16. Ivan, N.V., Asupra rentabilității utilizării calculatoarelor în proiectarea proceselor tehnologice. Noutăți în mecanica aplicativă și în construcția de mașini. Universitatea din Brașov, XVII, B, 87—94 (1975).
17. Ivan, N.V., Un sistem informatic de proiectare a proceselor tehnologice de prelucrare pe mașinile-unelte aşchietoare. Buletinul Universității din Brașov, A, XX, C, 127—132 (1978).
18. Ivan, N.V., Proiectarea asistată de calculator a adaosurilor optime de prelucrare și dimensiunilor tehnologice intermediare. A IV-a Conferință de procese și utilaje de prelucrare la rece. Institutul politehnic „Traian Vuia“ Timișoara, 1, 277—282 (1981).
19. Ivan, N.V., Modelarea matematică a datelor experimentale necesare rezolvării prin calculator a problemei adaosurilor de prelucrare. Construcția de mașini, XXIV, 2, 91—94 (1982).
20. Ivan, N.V., Subprogram de editare a operațiilor și/sau fazelor în ordinea succesiunii tehnologice optime. Lucrări științifice, 3, B, 147—152 (1981), Institutul de învățământ superior Tg. Mureș.
21. Ivan, N.V., Verificarea în producție a proceselor tehnologice proiectate cu ajutorul calculatoarelor electronice. Universitatea din Brașov, 1979.
22. Ivan, N.V., Bazele optimizării proceselor tehnologice. Îndrumar de laborator. Universitatea din Brașov, 1982.
23. Ivan, N.V., Viteza de avans, parametru important de optimizare în unele cazuri de prelucrare cu multiscule. Buletinul Universității din Brașov, XXIII, A, 255—257 (1981).
24. Ivan, N.V., Bonci, Gh., Privitor la calculul regimurilor de aşchiere pe strunguri automate multiax. Buletinul Institutului politehnic Brașov, A, XII, 427—438 (1970).
25. Ivan, N.V., Tănase, E., Determinarea parametrilor regimurilor de aşchiere utilizate la strunguri automate. Construcția de mașini, XXVII, nr. 2—3, 129—131 (1975).
26. Ivan, N.V., Drăghici, G., PAPT-1, un sistem de proiectare automată a proceselor tehnologice optime de prelucrare pe strunguri automate multiax. A II-a Conferință Națională de Mașini-unelte, decembrie, București, 82—89 (1976).
27. Ivan, N.V., Drăghici, G., Privitor la asigurarea unei exploatări optime a seulelor prin proiectarea automată a proceselor tehnologice. Noutăți în proiectarea, tehnologia și exploatarea seulelor aşchietoare. Universitatea din Brașov, 209—214 (1977).
28. Ivan, N.V., Stratulat, P., Tipizarea, mijloc de optimizare prin calculator a proceselor tehnologice. A III-a Conferință de mașini-unelte și scule, mai, București, 51—55, (1979).

29. *Ivan, N.V., Ivan, M.C., Tiței, D., Practica proiectării asistate de calculator a proceselor tehnologice de prelucrare pe strunguri automate multiax. A 3-a sesiunea de comunicări științifice „Creația tehnică și fiabilitatea în construcția de mașini”. Probleme tehnologice de prelucrare mecanice în construcția de mașini. Institutul politehnic „Gh. Asachi”, Iași, 164—170 (1981).*
30. *Ivan, N.V., Tiței, D., Ivan, M.C., Program operativ de optimizare a regimului de așchiere la prelucrările cu multiburghie. Lucrări științifice, 3, B, 141—146 (1981). Institutul de învățământ superior, Tg. Mureș.*
31. *Ivan, N.V., Drăghici, G., Proiectarea automată a proceselor tehnologice optime în cazul prelucrărilor pe strunguri automate multiax. Universitatea din Brașov, 1976.*
32. *Ivan, N.V., Leu, A., Groos, A., Cziszser, L., Modele matematice pentru calculul adaosurilor de prelucrare. Buletinul Institutului politehnic „Gh. Asachi” Iași, 1984.*
33. *Ivănescu, P.L. Programarea pseudo-booleană și aplicații ale ei. Studii și cercetări matematice, 17, 4, 583—598 (1975).*
34. *Ivănescu, P.L., Rudeanu, S., O teorie a deciziilor binare. Cazul liniar. Studii și cercetări de calcul economic și cibernetica economică, 2, 105—127 (1966).*
35. *Ivănescu, P.L., Rosenberg, I., Rudeanu, S., Asupra determinării minimelor funcțiilor pseudo-booleene. Studii și cercetări matematice, 14, 359—364 (1963).*
36. *Jacobs, H.J., Jacob, E., Kochan, D., Spannungsoptimierung. VEB Verlag Technik, Berlin, 1977.*
37. *Kovan, V.M., ș.a. Pravocnik tehnologo mașino-stroitelila, I. Mașghiz, Moskva, 1963.*
38. *Mihăilă, M., Introducere în programarea liniară. Ed. didactică și pedagogică, București, 1970.*
39. *Niculescu, Șt., Inițiere în FORTRAN. Ed. tehnică, București, 1972.*
40. *Petriceanu, Gh., Gyenge, Cs., Morar, L., Proiectarea proceselor tehnologice și reglarea strungurilor automate. Ed. Tehnică, București, 1979.*
41. *Picos, C., Ailincăi, Gh., Bohosievici, C., Pruteanu, O., Coman, Gh., Brana, V., Paraschiv, D., Calculul adaosurilor de prelucrare și al regimurilor de așchiere. Ed. Tehnică, București, 1974.*
42. *Pisău, Gh., Toma, C., Mihăescu, I., Elaborarea și introducerea sistemelor informatice. Ed. Tehnică, București, 1975.*
43. *Rumsiski, L.Z., Prelucrarea matematică a datelor experimentale. Îndumăr. Ed. Tehnică, București, 1974 (traducere din limba rusă).*
44. *Savvas, E.S. Conducerea cu calculatoare a proceselor industriale. Ed. Tehnică, București, 1969 (traducere din limba engleză — S.U.A.).*
45. *Vlad, A., Grigorescu, N., Reglarea strungurilor automate. Ed. Tehnică, București, 1965.*

CONTINUARE DE LA PAGINA 16

```

186 TYPE OPERATOR=PROCESS(ACCES:ACCES_BANDA;I:INTEGER);
187 [#####]
188 VAR T:IE TERMINAL;LIN:LINIE;DOC:DOCUMENT;
189 PROCEDURE CIT(MES:LINIE;VAR LIN:LINIE);
190 BEGIN T.SCRMESAJ(MES); T.CITMESAJ(LIN) END; [SCRIERE CU RASPUNS]
191 PROCEDURE CITDOCUMENT;
192 BEGIN
193     WITH DOC DO
194     BEGIN
195         CIT('CIMP1{:C:}','C1');
196         CIT('CIMP2{:O:}','C2');
197         CIT('CIMP3{:O:}','C3')
198     END
199 END;
200 BEGIN
201     INIT T(CHR(I+ORD('O')));
202     CYCLE
203     T.ASTBRK;
204     CITDOCUMENT;
205     ACCES.SCRARITICOL(DOC,I);
206     T.SCRMESAJ('DOCUMENT ACCEPTAT0:13:')(21:)(:0:)' );
207 END;
208 END;
209 [#####]
210 [ PROCESUL INITIAL ]
211 [#####]
212 VAR M:ACCES_BANDA;
213 CENTR:CENTRAL;
214 OPER:ARRAY(.1..NRPROCESE.) OF OPERATOR;
215 I:INTEGER;
216 BEGIN
217     INIT M,CENTR(M);
218     FOR I:=1 TO NRPROCESE DO INIT OPER(I.)(M,I);
219 END.

```

Fig. 4. (Programul 2). Exemplu de program conversațional în limbajul PASCAL CONCURENT.

3. Programe concurente. Sincronizarea și comunicarea între procese

3.1. Procese concurente

Marea majoritate a programelor realizate în limbajele de nivel înalt tradiționale (FORTRAN, COBOL, PASCAL etc.) sînt concepute de așa natură, încît acțiunile lor se desfășoară în mod strict secvențial (fiecare acțiune este inițiată doar cînd predecesoarea ei s-a terminat). Un astfel de program reprezintă un proces.

Uneori devine însă necesară conceperea unor programe care presupun executarea în paralel a mai multor procese. Aceasta conduce pe de o parte la exploatarea eficientă a resurselor sistemului de calcul (sisteme cu multiprogramare), iar pe de altă parte la realizarea unei analogii cu însăși natura problemei de rezolvat (programe pentru conducerea proceselor, programe conversaționale, programarea sistemelor multiprocesor și a sistemelor distribuite etc.) Deseori procesele paralele gestionează în comun resurse ale sistemului și între ele pot exista relații de cooperare (schimb de mesaje, transfer de date) sau conflictuale (prin solicitarea aceleiași resurse în același timp). Astfel de procese se numesc *concurente*.

Coexistența proceselor concurente care în marea majoritate a timpului se desfășoară asincron, impune în anumite momente sincronizarea în vederea realizării unei comunicări. Relațiile între ele se pot reduce în final la următoarele două situații tipice :

a) *Excluderea mutuală* între procese care folosesc în comun aceeași resursă, la care accesul este permis doar dintr-un singur proces la un moment dat. În acest caz este necesară secvențializarea cererilor de acces la resursa respectivă.

b) *Cooperarea* între procese care schimbă mesaje sau în general date cel mai des în relația producător/consumator. Informațiile produse de un proces sînt utilizate de celălalt; schimbul de mesaje presupune în anumite situații sincronizarea lor.

Spre deosebire de programarea secvențială, care permite definirea unui singur proces izolat, descrierea unor procese concurente și a relațiilor dintre ele se face prin așa-numita *programare concurentă*.

Limbajul PASCAL CONCURENT [2] oferă mijloace de rezolvare într-un limbaj de nivel înalt a categoriei cu totul noi de probleme puse de programarea concurentă.

Respectînd întru totul principiile și forma limbajului PASCAL, s-au introdus cîteva concepte noi, cum ar fi tipurile sistem, sincronizarea programată și automată, timpul real, intrări/ieșiri la nivel fizic [2, 7, 9] etc.

Programul 2 (fig. 4) este un program concurent care permite introducerea în paralel de la mai multe terminale a unor înregistrări, concentrate în final pe o bandă magnetică. Un operator central are posibilitatea de a porni și opri sesiunea de lucru la fiecare terminal, respectiv de a încheia lucrul la toate terminalele, după care închide fișierul creat. În paragrafele următoare se vor face referiri repetate la acest program.

3.2. Procesele — elemente active ale programului concurent

În PASCAL CONCURENT procesele unui program sînt variabile de tip *process*. Acțiunile descrise în corpul de instrucțiuni din blocul fiecărui proces reprezintă activitățile paralele ce au loc în program. Datele declarate într-un proces sînt proprii acestuia și sînt inaccesibile din alte componente sistem. Procese concurente pot gestiona în comun resurse prin intermediul altor componente de tip sistem, la care au drept de acces. Acestea se descriu ca parametri formali ai tipului *process* și se explicitează ca argumente la lansarea (inițializarea) procesului.

În programul 2 tipul proces „central” descrie activitatea operatorului central (liniile 153—183). Tipul proces „operator” descrie activitatea operatorului de la terminal (185—207).

Din listele de parametri ale tipurilor proces „central” și „operator” rezultă că ambele au acces la un monitor de tip „acces-banda”.

3.3. Monitoarele — elemente de sincronizare și comunicare.

Sincronizarea automată

Descrierea unor zone de date folosite în comun de mai multe procese și a tuturor operațiilor permise asupra acestora se face printr-o definiție de tip *monitor*. Procesele nu pot acționa direct asupra datelor declarate în monitor. Acestea sînt accesibile din exterior doar în mod indirect, prin intermediul unor rutine externe ale monitorului.

Datele dintr-o variabilă de tip monitor reprezintă deci resurse partajate între procese. Protecția acestora față de accesul simultan din mai multe procese (excluderea mutuală) se rezolvă în mod automat, după cum urmează :

- un proces solicită acces la un monitor apelînd una din rutinele acestuia ;
- pe parcursul executării rutinei apelate, procesul dobîndește acces exclusiv la monitor, care devine astfel ocupat ;
- la terminarea rutinei apelate, monitorul se eliberează ;
- cererile de acces la un monitor ocupat au ca rezultat punerea în așteptare a proceselor respective ; cererile se achită pe rînd, la eliberarea monitorului de către procesul precedent.

Prin acest mecanism, transparent pentru programator, se realizează sincronizarea automată a proceselor.

În programul 2, prin intermediul monitorului de tip „acces-banda“ se realizează accesul proceselor la banda magnetică.

3.4. Clase

O clasă are rolul de a descrie o structură de date și accesul controlat, prin intermediul unor operații precizate (rutinele clasei) la aceste date. Referirile din afara clasei la datele și rutinele ei pot avea loc, prin definiție, doar dintr-o singură componentă sistem (spre deosebire de monitoare). Aceasta se verifică la compilarea programului.

Descriînd în mod unitar date și acțiuni, clasa este un mijloc de modularizare și structurare a programului, fără a implica aspecte privind concurența și sincronizarea în cadrul sistemului concurent (cum este cazul proceselor și monitoarelor).

În programul 2 tipurile clasă „ie-terminal“ (liniile 22—75), „ie-bandă“ (77—94), „ie-consola“ (96—127) furnizează proceduri pentru introducerea și extragerea de mesaje la terminal respectiv consolă, pentru scrierea pe bandă și închiderea fișierului, pentru punerea în așteptare a procesului „central“ pînă la intervenția operatorului etc. [9].

3.5. Componentele programului concurent. Drepturi de acces

Tipurile sistem introduse mai sus sînt după cum s-a arătat tipuri abstracte cuprinzînd descrierea unor date și a unor operații posibile. Componentele efective ale programului concurent, procese, monitoare, și clase sînt variabile de aceste tipuri. În cazul în care în structura programului intră mai multe componente definind operații identice asupra unor date de aceeași structură, acestea se vor declara ca variabile de același tip sistem.

În prog. 2 s-a declarat un tablou de procese „oper“ cu „nrproces“ elemente de tip „operator“ (linia 213), care se pun în lucru (se inițializează) pe linia 217. În continuare, activitățile acestor procese se vor desfășura în paralel.

O componentă sistem are acces la parametrii și variabilele ei proprii. De asemenea, componenta poate avea acces la alte variabile de tip sistem explicitate ca argumente la inițializarea ei.

Astfel, la definirea tipurilor proces din programul 2 se indică accesul la un monitor de tip „acces-banda“ (liniile 153 și 185). La inițializarea variabilelor proces respective (liniile 216 și 217) se concretizează monitorul (variabila „m“).

Parametrii formali de tip sistem dintr-o definiție de tip *process* sau *monitor* pot fi doar monitoare. Astfel, resursele partajate între procese sînt gestionate exclusiv prin monitoare.

Accesul din exterior la o componentă sistem se face apelînd rutine ale ei. Apelul rutinei se realizează prin numele acesteia care se va califica (v. § 2.2.3.2) cu identificatorul componentei din care face parte.

Reprezentarea grafică a componentelor unui program concurrent și a legăturilor dintre ele se poate face cu ajutorul unui graf de acces. Componentele sistemului se reprezintă prin cercuri, iar drepturile de acces (legăturile dintre componente) prin săgeți. Un graf de acces este prin urmare un graf orientat, organizat ierarhic pe baza drepturilor de acces ale componentelor sistemului. El nu poate conține circuite, evitîndu-se astfel blocările datorate unor apelări recursive ale monitoarelor. Respectarea organizării în acest mod a drepturilor de acces se verifică riguros la compilarea programului. În fig. 5 se prezintă graful de acces corespunzător programului 2.

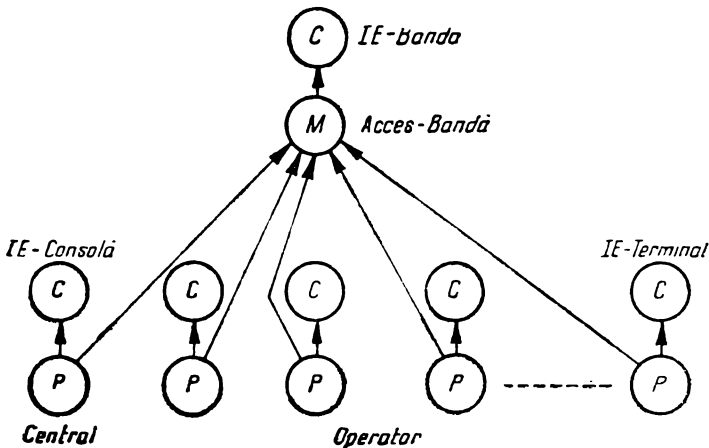


Fig. 5. Graful de acces pentru programul 2

Fiecare proces de tip „operator“ are acces la o componentă de tip „ie-terminal“, declarată ca variabilă în interiorul procesului (linia 187). În mod asemănător tipul proces „central“ conține o variabilă (clasă) de tip „ie-consola“ iar tipul monitor „acces-banda“ conține o clasă de tip „ie-banda“. Toate procesele au acces la monitorul de tip „acces-banda“.

3.6. Sincronizarea programată a proceselor

Sincronizarea automată realizată prin excluderea mutuală a apelurilor procedurilor de monitor, nu permite rezolvarea tuturor problemelor care apar la interacțiunea proceselor.

O astfel de situație apare spre exemplu în cazul a două procese p_1 și p_2 care cooperează prin intermediul unui tampon T , de dimensiune finită. Procesul p_1 (producătorul) introduce articole în tampon, iar p_2 (consumatorul) le extrage. Cele două procese funcționează în paralel, asincron și independent unul de celălalt. Se pot ivi însă două situații limită în care tamponul s-a golit sau a devenit plin. Astfel, devine necesar ca procesul p_1 să fie pus în așteptare prin program când găsește tamponul plin; în mod analog este necesar ca p_2 să fie pus în așteptare când găsește tamponul gol. După modificarea stării tamponului, care a necesitat punerea în așteptare, tot prin program procesele trebuie relansate. S-a ilustrat astfel, într-o situație tipică, necesitatea ca programatorul, în funcție de anumite condiții, să pună în așteptare un proces care apoi să poată fi relansat tot din program.

Procesele se pun în așteptare în variabile de tipul standard `queue`. Acestea reprezintă locații de așteptare care, la un moment dat, pot conține un singur proces.

Variabile de tip `queue`, sau structuri cu componente de acest tip, se declară doar în monitoare. Prin urmare toate acțiunile de punere în așteptare și relansare a proceselor au loc în monitoare. Ele se realizează prin intermediul procedurilor standard `delay` și `continue`. Ambele se apelează cu un argument de tip `queue` și execută următoarele operații:

`delay(x)` — Procesul activ (care a apelat rutina de monitor) se pune în așteptare în variabila „ x ”. Procesul eliberează monitorul, care astfel devine accesibil pentru alt proces.

`continue(x)` — Dacă în locația „ x ” așteaptă un proces, acesta se reactivează din punctul în care a executat `delay`. În continuare acest proces va avea acces exclusiv la monitor. Variabila „ x ” ia valoarea: locația vidă. Procesul care a apelat `continue` revine din rutina monitorului și pierde accesul la acesta. Dacă locația „ x ” a fost vidă, se execută doar eliberarea monitorului și revenirea din rutină.

Mecanismul de sincronizare prin `delay` și `continue` reprezintă tehnica de sincronizare programată a proceselor în PASCAL CONCURRENT.

În programul 2 procesele se pun în așteptare respectiv se lansează la cererea operatorului central. Aceste operații se realizează în monitorul de tip „acces-banda” ca urmare a unor apeluri din procesul de tip „central” (liniile 178, 179, 180). Punerea în așteptare a procesului cu indice „ i ” se face în locația corespunzătoare din tabloul „locații”, cu elemente de tip `queue` (linia 132).

4. Concluzii

S-au prezentat câteva aspecte fundamentale privind programarea structurată și concurrentă și două limbaje, PASCAL și PASCAL CONCURRENT, care facilitează în mare măsură aplicarea în practică a acestor tehnici. Aceste limbaje sînt accesibile programatorului din țara noastră nu numai pe calculatoarele din familia Felix C, ci și pe calculatoarele Felix M-18, Independent și Coral.

BIBLIOGRAFIE

1. *Wirth, N.* The Programming Language Pascal, Acta Informatica, nr. 1, 1971.
2. *Hansen, P.B.* The Programming Language Concurrent Pascal, IEEE Transaction on Software Engineering, vol. 1, nr. 2, 1975.
3. *Jilari, M.* Îndrumător de limbaje de programare, Ed. Tehnică, București, 1978.
4. *Văduva, I.* Programarea structurată, Ed. Tehnică, București 1978.
5. *Niculescu, S* Fortran – înțelegere în programarea structurată, Ed. tehnică, București, 1979.
6. * * * Programarea și utilizarea calculatoarelor – studii și aplicații, Ed. FACLA, 1981.
7. * * * Limbajul Pascal. Manual de utilizare, CCE al IPT, 1981.
8. * * * Limbajul Pascal. Manual de operare, CCE al IPT, 1982.
9. * * * Limbajul Pascal Concurrent. Manual de utilizare, CCE al IPT, 1980.
10. * * * Limbajele Pascal și Pascal Concurrent, V2, Manual de utilizare și operare, CCE al IPT, 1983.
11. *Dahl, O.J.* Structured Programming, Academic Press London, New York, 1972.
12. *Niculescu, S.* Macroprocesoare și limbaje extensibile, Editura științifică și enciclopedică, București, 1982.

VĂ RECOMANDĂM :

Din seria Automatică-Management-Calculatoare (AMC) apar în trimestrul III 1985 volumele :

AMC 48, AMC 49, AMC 50, AMC 51, în cuprinsul cărora sînt prezente module și cicluri de foarte mare actualitate, de înaltă calitate și de un interes deosebit, și anume :

— Congresul mondial trienal al Federației Internaționale de Automatizare (IFAC) „O punte între știință și tehnologie”, Budapesta 1984, reprezentat prin plenare, studii de caz și sinteze pentru toate secțiunile (autori străini și români).

— „Societatea informatică” note de lectură după cartea japonezului Masuda, „Resursele informaționale naționale” și „Fenomenul calculatoarelor personale” după sovieticul Gromov.

— „Memento de teleprelucrare”, cu toate informațiile necesare pentru echipamentele și sistemele teletinformatică românești.

— „Minicalculatoarele INDEPENDENT și CO-RAL”. Manual de utilizare din ciclul SERVICE pentru CALCULATOARE.

— „BASIC pentru începători, cu calculatorul personal”, un manual practic din ciclul „CALCULATOARE PERSONALE ȘI PROGRAMAREA LOR”.

— Ciclul „PROIECTAREA ASISTATĂ DE CALCULATOR” reprezentat prin articole de direcționare în domeniu și articole prezentate la o primă sesiune națională.

— Microcalculatoarele personale românești, Student-HC 80, PRAE (pentru acesta și limbajul său BASIC) și microcalculatorul profesional-personal românesc Felix PC, în prezentări sintetice — în premieră într-o carte.

În trimestrul IV 1985 apar și volumele AMC 52-53-54, cu ciclurile amintite, dar și cu automatizarea flexibilă, roboții, limbajul BASIC pentru WANG VS, ghidul analistului (continuare la AMC 45-46), jocuri de întreprindere ș.a.

Fiecare dintr-unul volumelor AMC este de aproximativ 25 lei.

- Seria AMC se procură din librăriile cu carte tehnică.
- Din cursul anului 1984, de la volumul AMC 38, pe coperți apar cele câteva domenii atacate, volumele căpătînd, astfel, titluri. Această tematizare s-a bucurat de succes nu numai în rîndul informaticienilor ci și al altor specialiști, ca și a unor cercuri de studenți, elevi ș.a. Cicluri continue ca : proiectarea asistată de calculator, analiza sistemelor, calculatoare personale, rețele de calculatoare, note de lectură, servicii pentru calculatoare, congrese ș.a., realizate de specialiști, atrag noi cititori permanenți.
- Pentru a vă asigura obținerea seriei AMC în noile condiții, recomandăm întreprinderilor să ne trimită pe adresa : EDITURA TEHNICA, Piața ȘCINTEII 1, comenzi ferme anticipate, semnate de director și contabil șef, care să colecteze atît volumele necesare documentării unității cît și cererile ferme exprimate de oamenii muncii prin organe sindicale (FUS, CIT ș.a.). Și cititorii individuali ne pot scrie, indicînd adresa exactă. Noi înaintăm toate aceste comenzi, cu indicații de prioritate, centrelor de librării, prin Întreprinderea de difuzare a cărții (I.D.C.).
- Plata se face la primirea volumelor, nu anticipat.
- În 1984 au apărut volumele 35-44, iar în 1985 (trim. I-II) AMC 45-47. În trim. III apar AMC 48-51, iar în trim. IV, AMC 52-54. Prețul unui volum este de 22-25 lei.
- În 1986 sînt prevăzute AMC 55-68.
- Eventualele lămuriri la tel. 18.06.28 sau 17.68.10/2100.
- Volum realizat prin cooperare cu Comisia de cibernetică a Academiei R. S. România, Institutul de tehnică de calcul și informatică, Institutul de proiectări automatizări, Întreprinderea pentru întreținerea și repararea utilajelor de calcul și cu specialiști din aceste foruri și unități.
- Cu sprijinul Federației Internaționale de Automatizare (IFAC) și cu concursul unor specialiști din R. P. Ungară.
- Cu concursul și al unor specialiști de la Institutul Politehnic București, Universitatea Brașov, Institutul Politehnic Timișoara, Institutul de Meteorologie și Hidrologie, Institutul de Cercetări și Proiectări pentru Gospodărirea Apelor.