Capitolul 1
INTRODUCERE

1.1. Scurt istoric

Mediile de modelare şi simulare sunt folosite de o mare varietate de utilizatori (ingineri, cercetători, specialişti în diverse domenii, studenţi). Utilizarea acestor medii specializate s-a impus ca o etapă preliminară obligatorie în cadrul conceperii, elaborării şi implementării unor sisteme sau soluţii tehnice.

Încă de la începutul anilor '70, au existat o serie de tendinţe de creare a unor limbaje specializate destinate rezolvării de probleme specifice anumitor domenii de activitate. Cleve Moler, director al Departamentului de Ştiinţa Calculatoarelor, din cadrul Universitatii New Mexico, Albuquerque, SUA, a coordonat dezvoltarea mediilor LINPACK, respectiv EISPACK, urmărind posibilitatea de a facilita accesul şi utilizarea de către studenţi a acestor medii fără a fi necesară însuşirea limbajului Fortran (extrem de răspândit şi utilizat în acea perioadă). Mediile LINPACK şi EISPACK reprezentau colecţii de rutine software pentru rezolvarea unor probleme de algebră liniară, implementate în limbajul Fortran. Ca urmare a unor astfel de preocupări, tot Moler a dezvoltat şi limbajul MATLAB, prezentând o prima versiune unui grup de studenţi de la Universitatea Stanford în 1979. În forma primară, MATLAB permitea realizarea unor apeluri de rutine optimizate (codate în Fortran) utilizând un interpretor interactiv în mod text. Se evita astfel problema compilărilor, respectiv link-editărilor succesive.

În 1983, în cadrul unei prezentări a MATLAB-ului la Universitatea Stanford, inginerul automatist Jack Little a sesizat potenţialul comercial al acestui limbaj şi împreună cu Moler şi Steve Bangert au fondat compania MathWorks Inc, având ca obiect de activitate principal dezvoltarea şi asigurarea suportului tehnic pentru utilizatorii mediului MATLAB. Actualmente Moler este preşedinte executiv al companiei MathWorks Inc. Odată cu evoluţia limbajelor de programare de nivel înalt mediul MATLAB a fost rescris utilizând limbajul C (unul din cele mai utilizate limbaje şi astăzi). Acesta s-a răspândit în cadrul mediului universitar, fiind în primul rând utilizat de inginerii din domeniul conducerii automate. Bibliotecile MATLAB rescrise în limbajul C, cunoscute iniţial sub denumirea de JACKPACK, au fost ulterior utilizate în o gamă largă de domenii, numărul acestora crescând neîncetat.

În prezent, limbajul MATLAB este livrat ca şi parte integrantă a unui mediu software avansat care îmbină calculul numeric, grafica, vizualizarea şi limbajul de programare propriu-zis. Principial există două versiuni: *standard* şi *student*. Versiuni mai vechi ale MATLAB (sub sistemele DOS 3, 4) sunt încă disponibile gratuit pe Internet. Aceste versiuni prezintă o interfaţă în mod text şi o serie de rutine pentru rezolvarea problemelor de algebră liniară. De-a lungul anilor au mai apărut o serie de limbaje specializate similare MATLAB, cum ar fi de exemplu:

- Octave, un limbaj de nivel înalt, destinat calculului numeric.
- SciLab, un pachet software de rutine bazate pe calcul matriceal care include un interpretor de nivel înalt, un macrolimbaj şi grafică.
- NumPy, un set de extensii numerice/matriceale al limbajului Python, care oferă posibilitatea de a opera cu obiecte de tip vector multidimensional.

Similar primelor versiuni de Perl sau Awl, MATLAB era în principiu o colecţie de instrumente software – interpretorul prezentând un singur spaţiu de lucru global (workspace). De asemenea, simple comenzi de afişare grafică erau înglobate alături de rutine sofisticate de inversări de matrici.

Prima versiune de MATLAB (sub DOS) are un singur tip de dată - tipul de matrice bidimensională, cu numere complexe în dublă precizie - şi un set de funcţii şi operatori de creare, respectiv de calcul matricial. Limbajul a fost vectorizat, astfel încât declaraţii simple de atribuire (ex: y=a+b), în mod implicit ciclează pentru toate elementele matricilor. Astfel, un număr scalar (exemplu „56") este tratat ca şi o matrice de dimensiune (1x1). Vectorii sunt consideraţi ca matrici de dimensiune Nx1 sau 1xN, conform utilizării acestora. În alte limbaje precum C/C++ sau Perl, vectorii sunt trataţi sub forma unor şiruri continue de valori, lăsând la latitudinea utilizatorului forma de reprezentare a datelor. Acest fapt reprezintă o deficienţă majoră în cazul limbajelor destinate inginerilor, respectiv matematicienilor. Structurile din MATLAB sunt similare celor din limbajul C, cu deosebirea ca acestea sunt dinamice, putând fi adăugate/editate/şterse pe parcursul execuţiei programului.

Toolbox-uri (biblioteci de funcţii) specializate au fost adăugate MATLAB-ului pe parcursul anilor '80, păstrându-se nucleul de bază al limbajului. În anii '90, tendinţele din ingineria programării au influenţat corespunzător şi programarea în limbajul MATLAB. Astfel, în concordanţă cu doleanţele utilizatorilor, dezvoltatorii MATLAB au decis să facă o serie de îmbunătăţiri ale tipurilor de date (vectori, structuri şi obiecte). Fiecare dintre aceste facilităţi au modificat structural limbajul.

Un concurent al mediului MATLAB, este mediul Mathematica care este un limbaj orientat pe calcul simbolic, având ca şi atu o manipulare simbolică superioară limbajului MATLAB, fiind foarte popular printre cercetătorii din domeniul fizicii. O altă deosebire marcantă, este faptul că MATLAB lucrează cu programe script, scrise într-un limbaj foarte apropiat de limbajul C, în timp ce Mathematica utilizează propriul său limbaj simbolic.

Similar oricărui limbaj de programare, MATLAB a fost conceput ca o modalitate simplă de a rezolva probleme complexe, dar în timp a devenit o modalitate relativ complicată de a rezolva probleme cu un grad foarte mare de complexitate. Limbajul MATLAB, chiar dacă a evoluat în paralel cu alte limbaje de programare (gen Visual Basic), a devenit foarte diferit de acestea. Variabilele în MATLAB sunt versatile şi dinamice, definite în spaţiul de lucru, aceasta fiind modalitatea de operare cu ecuaţii matematice, simultan fiind permise şi spaţii de lucru structurate în conformitate cu tehnicile specifice limbajelor de programare moderne, orientate pe obiecte. Toate limbajele de programare sunt utile, fiecare în sine, tocmai datorită caracterului specific al fiecăruia, orientat spre un anumit domeniu aplicativ.

Oficial, prima versiune de MATLAB a apărut în 1984, oferind inginerilor şi cercetătorilor posibilitatea de a utiliza un mediu de calcul cu înaltă productivitate, ca o alternativă mult mai simplă la utilizarea unor limbaje de programare gen Fortran sau C. Astfel, MATLAB a devenit un mediu de calcul extrem de utilizat în mediile ştiinţifice şi inginereşti, combinând capabilităţi extensive de calcul matematic, reprezentare grafică şi caracteristici ale unui limbaj de înalt nivel puternic şi relativ uşor de utilizat. Analizând vastă listă a componentelor MATLAB se poate observa că dezvoltarea limbajului urmează principiul modularităţii. Baza mediului de modelare şi simulare MATLAB o reprezintă un motor de calcul şi facilităţile funcţiilor de intrare-ieşire, integrând o largă gamă de toolbox-uri, care permit utilizatorului din diverse domenii de activitate să programeze cu uşurinţă aplicaţii utilizând elemente predefinite pentru crearea unor modele ale sistemelor.

Pragmatismul acestei abordări modularizate este specific mentalităţii inginereşti. MATLAB-ul încorporează continuu noi toolbox-uri, iar compania producătoare MathWorks Inc dezvoltă în continuare aplicaţii specializate în colaborare cu parteneri din domeniul universitar şi industrial.

În prezent mediul MATLAB reprezintă un pachet de programe destinat rezolvării de probleme specifice tuturor domeniilor inginereşti, putând fi considerat un ***mediu de referinţă*** *pentru simularea şi analiza sistemelor dinamice*. Numele de **MATLAB** este un acronim pentru **MAT**rix **LAB**oratory, fapt care conduce la ideea că elementele de bază cu care se operează sunt matricile, pretându-se în mod natural la orice aplicaţie care presupune în ultimă instanţă utilizarea algebrei liniare.

Ca o extensie importantă a mediului MATLAB, a fost realizat şi integrat **SIMULINK-ul**, un pachet de programe pentru *modelarea, analiza şi simularea vizuală interactivă* a unei largi categorii de sisteme (inclusiv sistemele care conţin elemente neliniare) atât în domeniul continuu cât şi în cel discret. Pe lângă faptul că extinde facilităţile oferite de MATLAB, Simulink-ul reprezintă şi o interfaţă grafică

şi interactiva a acestuia, conducând la o simplificare a modului de operare în rezolvarea diferitelor probleme.

Fără a avea pretenţia de a acoperi integral plaja largă a subiectelor abordate, lucrarea doreşte să ofere o viziune generală asupra mediului de modelare şi simulare MATLAB, prezentând totodată şi un marcant caracter aplicativ, adresându-se unui cititor preocupat de dezvoltarea unor aplicaţii utilizând acest mediu software.

Capitolul 2

PREZENTAREA MEDIULUI DE PROGRAMARE MATLAB, INSTRUMENT SOFTWARE PENTRU SIMULAREA FUNCȚIONĂRII SISTEMELOR

2.1. Considerații generale

MATLAB-ul deține supremația în domeniul aplicațiilor inginereşti în care algebra liniară joacă un rol important, cum ar fi prelucrarea semnalelor, identificarea proceselor, conducerea proceselor, etc. datorită faptului că acesta înglobează una dintre cele mai elaborate colecții de funcții specifice respectivelor domenii. Aceste funcții sunt grupate în biblioteci numite "**toolbox**"-uri, specifice fiecărui domeniu, în cadrul fiecărui toolbox existând implementate o serie de proceduri/componente specializate pentru rezolvarea unor problematici dintr-un anumit domeniu.

Dintre cele mai importante şi utilizate toolbox-uri se pot aminti:

- **CONTROL SYSTEM TOOLBOX** - destinat problemelor de conducere automată;
- **SIGNAL PROCESSING** – conținând instrumente pentru prelucrarea digitală a semnalelor;
- **SYSTEM IDENTIFICATION** - destinat problemelor de identificare şi estimare a parametrilor sistemelor automate;
- **OPTIMIZATION** – destinat abordării şi rezolvării problemelor de optimizare;
- **NEURAL NETWORK** – destinat operării cu rețele neuronale;
- **FUZZY LOGIC** - destinat operării cu sisteme fuzzy;
- **NONLINEAR CONTROL** – dedicat conducerii sistemelor neliniare;
- **STATISTICS** – specializat în operarea cu mărimi statistice;
- **SYMBOLIC MATH** – dedicat calculului simbolic.

Modul de lucru al MATLAB-ului este interactiv, sistemul interpretând fiecare linie introdusă de utilizator de la consolă, existând totodată următoarele facilități suplimentare:

- extinderea nelimitată a setului de comenzi existente;
- lansarea în execuție a unor fişiere de comenzi (aşa numitele "fişiere script" sau programe) create de utilizator.

Utilizarea limbajului MATLAB este simplă, expresiile fiind foarte apropiate de limbajul matematic comun.

În cadrul capitolului de față se urmăreşte realizarea unei prezentări generale a mediului de programare MATLAB, a tipurilor de date ce pot fi utilizate şi a principalelor instrucțiuni MATLAB de uz general, frecvent utilizate, în scopul familiarizării cititorului cu acest puternic instrument software utilizat în domeniul teoriei sistemelor, modelării şi simulării proceselor, estimării parametrilor sistemelor, reglării automate etc.

2.2. Interfața grafică

Interfața grafica a mediului MATLAB, versiunea 6.0, prezintă trei tipuri de ***ferestre*** şi anume:

- fereastra principală (de comenzi),
- fereastra grafică,
- fereastra de editare a programelor.

➢ ***Fereastra principală***, prezentată în figura 2.1, prezintă următoarele module (cu funcționalitățile aferente):

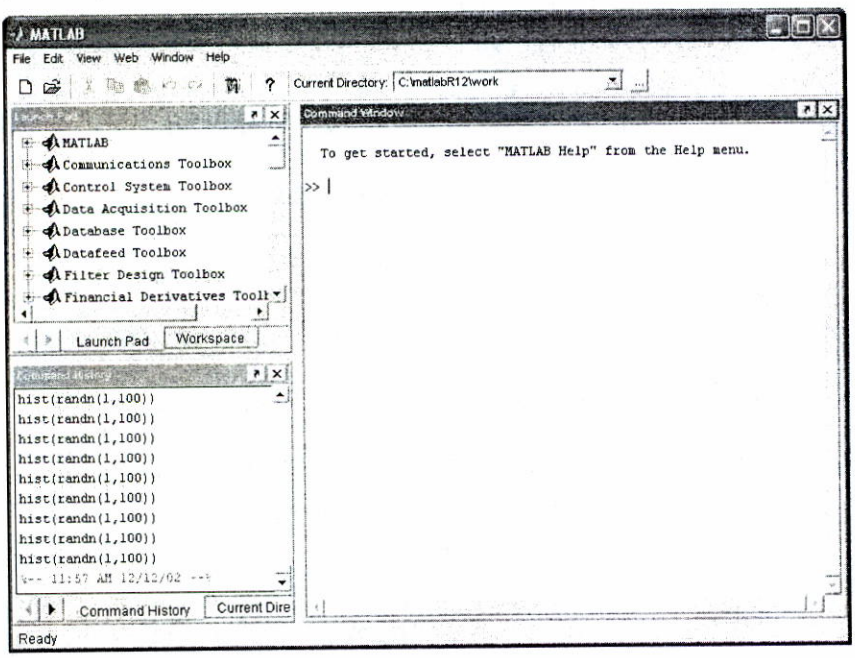

Figura 2.1. Fereastra principală MATLAB

"**Launch Pad**" – furnizează acces rapid spre programele demonstrative, instrumentele şi documentația aferentă tuturor toolboxurilor instalate.

"**Workspace**" – permite accesul rapid la zona de memorie de lucru, unde sunt stocate toate variabilele utilizate în mod curent. Se pot efectua atât vizualizarea sau modificarea valorilor acestor variabile, precum şi eliminarea lor din memorie.

"**Command Window**" – este zona în care se pot scrie comenzile care se doresc a fi executate.

"**Command History**" – furnizează o lista cu ultimele comenzi executate, şi permite reluarea acestora.

"**Curent Directory**" – permite vizualizarea, schimbarea şi listarea conţinutului directorului curent.

Aspectul ferestrei principale MATLAB poate fi particularizat, cu ajutorul opţiunii "**View > Desktop Layout**" din meniul principal.

- ***Fereastra grafică*** (exemplificată în figura 2.2) este utilizată la toate afişările grafice din MATLAB.

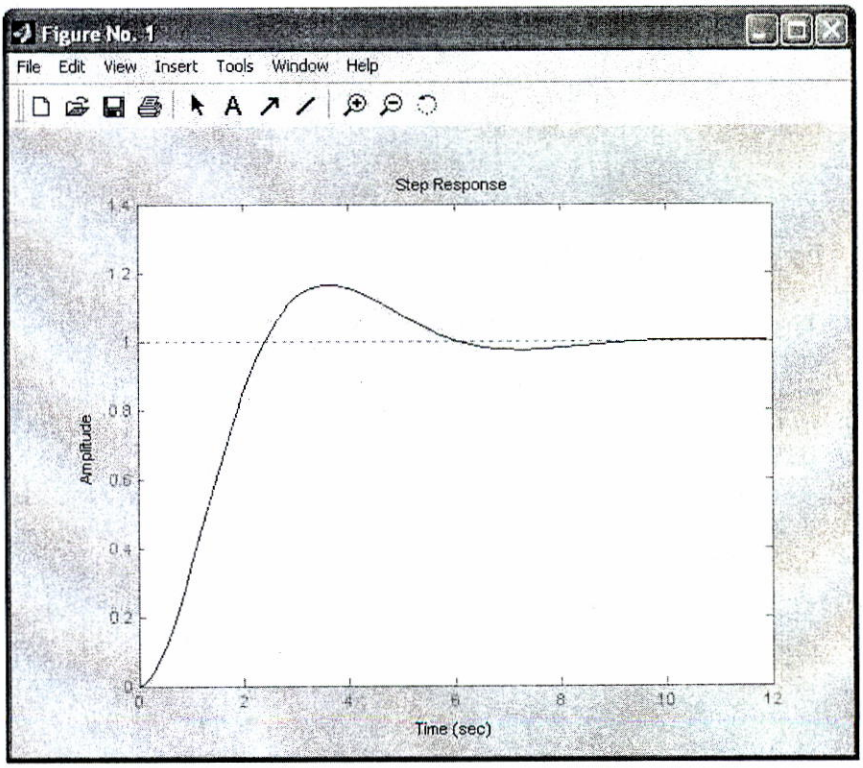

Figura 2.2. Fereastra grafică (exemplificare)

- MATLAB versiunea 6 dispune de un editor propriu de texte, care poate fi utilizat la scrierea programelor. ***Fereastra de editare*** este prezentată în figura 2.3.

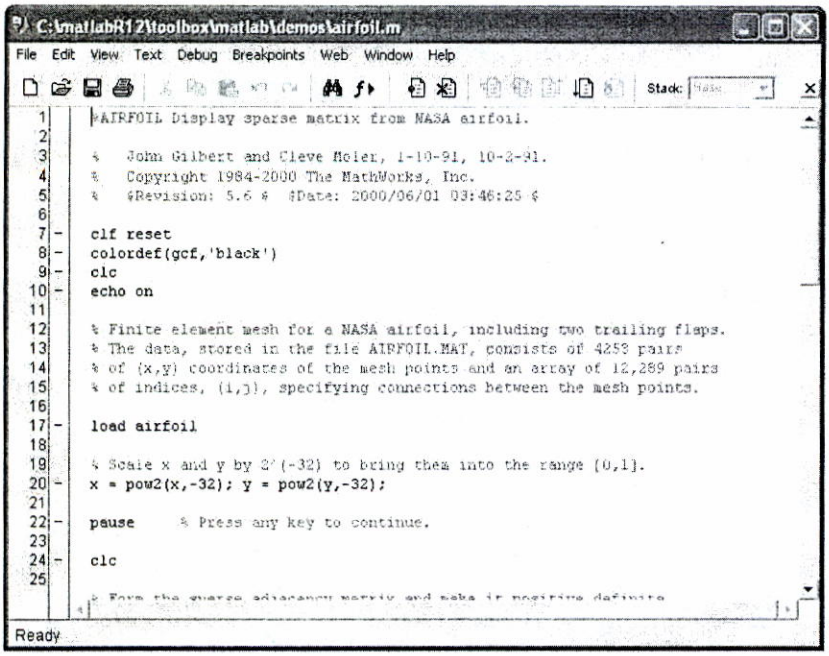

Figura 2.3. Fereastra de editare a programelor

2.3. Tipuri de date şi operatori

2.3.1. Tipuri de date. Variabile

În MATLAB se operează cu următoarele tipuri de date:

- numeric
- şir de caractere – utilizându-se ca delimitator apostroful " ' "
- matrici (tablouri) – se foloseşte ca şi delimitator paranteza dreaptă " [ ] "
- structuri complexe – se utilizează punctul " . " la specificarea câmpurilor (vezi exemplele următoare)

În mod uzual, declararea unei variabile în MATLAB se face sub forma:

```
variabila = expresie
```

Exemple:

```
>> numar=3
>> sir='unudoitrei'
>> tablou=[4 6 8]
>> S = 1 + 1/2 + 1/4
```

♦

Exemplu:
Definirea şi respectiv afişarea pe ecran a unei structuri complexe se poate face cu ajutorul următoarei secvenţe de comenzi:

```
>> structura.nume='exemplu de structura complexa'
>> structura.valoare_numerica=3
>> structura.valoare_text='trei'
```

Afişarea valorii acestei structuri se poate realiza rulând comanda:

```
>> structura
```

iar efectul este:

```
structura =
nume: 'exemplu de structura complexa'
valoare_numerica: 3
valoare_text: 'trei'
```

♦

Spre deosebire de alte limbaje de programare, în MATLAB, operarea cu variabile prezintă următoarele ***particularităţi***:

- nu este necesară declararea unei variabile înainte de a fi utilizată;
- în momentul în care se efectuează o operaţie de atribuire asupra unei variabile, MATLAB-ul alocă corespunzător memorie pentru variabila respectivă şi îi atribuie acesteia valoarea dorită;
- în cazul în care unei variabile existente i se atribuie o valoare de alt tip decât tipul variabilei, după efectuarea atribuirii, tipul variabilei se schimbă fără a se semnala eroare de tip (conversie automată de tip).
- din momentul iniţializării unei variabile, aceasta rămâne în memoria calculatorului (într-o zonă numită "***workspace***") până când se şterge în mod explicit, sau până când se părăseşte mediul de lucru.
- observaţia precedentă este valabilă şi pentru cazul variabilelor care sunt iniţializate într-un program (*fişier script*); spre deosebire de alte medii de programare, în MATLAB, la părăsirea unui program, variabilele "locale" nu sunt şterse automat din memorie.

În mod implicit MATLAB-ul face distincţie între literele mari de cele mici. Comanda **casesen** face MATLAB-ul insensibil la caractere mari sau mici. Astfel, de exemplu, **sir** şi **SIR** pot reprezenta sau nu aceeaşi variabilă.

Trebuie remarcat faptul că în MATLAB există un set de ***variabile*** şi ***constante speciale***, care sunt predefinite şi pot fi oricând accesate direct de utilizator. Cele mai utilizate sunt prezentate în tabelul următor:

Variabile speciale	
Variabila	**Semnificaţie**
Ans	Cel mai recent răspuns al sistemului
Pi	π
computer	Tipul computerului
i or j	Rădăcina pătrată din –1
Inf	Infinit
Eps	Precizia de calcul în virgulă mobilă
realmax	Cel mai mare număr în virgulă mobilă
realmin	Cel mai mare număr în virgulă mobilă
NaN	Not a number (utilizat în cazul nedeterminarilor)

De asemenea este posibilă suprascrierea acestor variabile de către utilizator. În cazul unora (de exemplu "i", "j"), suprascrierea este valabilă până când se şterge în mod explicit variabila respectivă, moment în care se reface conţinutul implicit, iar în cazul altora (de exemplu "ans", "nargin") MATLAB-ul poate oricând să refacă conţinutul iniţial, dacă este cazul. Aceste variabile speciale sunt prezentate detaliat în **Anexa 1**.

2.3.2. Matrici

În MATLAB matricile pot fi introduse în următoarele moduri:

- prin lista explicită de elemente: de exemplu, pentru a defini matricea

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

trebuie scris:

```
>> A = [1 2 3; 4 5 6; 7 8 9]
```

- matricile de dimensiuni mari se pot introduce linie cu linie:

```
>>A  = [1 2 3 <CR>
4 5 6 <CR>
7 8 9] <CR>
```

unde <CR> reprezintă tasta „ENTER".

Elementele matricilor pot fi orice expresii MATLAB (constante, variabile, funcţii sau chiar matrici).

Exemplu:

```
>> X = [-1.3   sqrt(3)   -(1+2/5)]
```

rezultă

```
X =
   -1.3000    1.7321   -1.4000
```

unde ***sqrt*** este funcția care returnează rădăcina pătrată.

♦

Elementele individuale ale unei matrici pot fi referite cu indici în interiorul unor paranteze rotunde.

Exemplu:

```
>>X(1) = abs(X(3))
```

conduce pentru X definit anterior la:

```
X =
    1.4000    1.7321   -1.4000
```

unde ***abs*** este funcția care returnează valoarea absolută.

♦

2.3.3. Operații cu matrici

❑ **transpusa**: caracterul " ' " (apostrof) aplicat unei matrici, specifică transpunerea matricii:

Exemplu:

```
>> A = [1 2 3; 4 5 6; 7 8 9]
```

conduce la

```
A =
     1     2     3
     4     5     6
     7     8     9
```

iar rulând comanda

```
>> B = A'
```

rezultă

```
B =
     1     4     7
     2     5     8
     3     6     9
```

♦

❑ **adunarea şi scăderea** matricilor se realizează cu ajutorul operatorilor " + " şi respectiv " - ".

Exemplu:

```
>> C = A + B
```

conduce la

```
C =
     2     6    10
     6    10    14
    10    14    18
```

♦

Nota: Evident, matricile trebuie sa aibă aceleaşi dimensiuni.

❑ **înmulţirea** matricilor se realizează cu ajutorul operatorului " * ".

Notă: Operaţia *X* * *Y* este permisă doar dacă dimensiunea a doua a matricii *X* (numărul de coloane) este egală cu prima dimensiune a matricii *Y* (numărul de linii).

❑ **împărţirea** matricilor: există două simboluri de împărţire în MATLAB: " \ " şi respectiv " / ", care au insa semnificaţii diferite .

Astfel, dacă A este o matrice pătratică nesingulară atunci:

- **A\B** corespunde în fapt înmulţirii la stânga a lui B cu inversa lui A, adică inv(A) * B.
- **B/A** corespunde în fapt înmulţirii la dreapta a lui B cu inversa lui A, adică B * inv(A).

❑ **ridicarea la putere** a unei matrici se realizeaza cu ajutorul caracterului "^"

Exemplu:

```
>> A^p
```

reprezintă A la puterea p (unde A este matrice pătrată şi p este un scalar).

♦

În cazul în care o anumită operaţie se efectuează individual asupra elementelor unei matrici, se utilizează caracterul " . " înaintea semnului operaţiei:

Exemplu:
Secvenţa următoare de comenzi:

```
>> X = [1 2 3]
>> Y = [4 5 6]
>> Z = X.^Y
```

are ca rezultat:

```
Z =
     1     32    729
```

Exponentul poate fi şi scalar:

```
>> Z = X.^2
```

conduce la

```
Z =
     1     32    729
```

♦

2.3.4. Generarea vectorilor

Un caz particular de tablouri utilizate frecvent în aplicaţii, îl reprezintă cazul matricilor cu o singură dimensiune (vectori).

Exemplul tipic de definire a unui ***vector*** este:

```
val_initiala:pas:val_finală.
```

Exemplu:

```
>> X = 1:5
```

Astfel se generează un vector linie, conţinând numerele de la 1 la 5, incrementul fiind unitatea:

```
X =
     1     2     3     4     5
```

Se pot utiliza şi alte incremente diferite de 1:

```
>> Y = 1:1.5:6
```

unde: 1 - reprezintă valoarea iniţială;
1.5 – reprezintă incrementul (pasul);
6 - reprezintă cea mai apropiată valoare de cea finală.
conduce la:

```
Y =
    1.0000    2.5000    4.0000    5.5000
```

♦

2.4. Instrucţii uzuale de bază ale mediului MATLAB

În acest paragraf sunt prezentate succint câteva dintre cele mai uzuale ***comenzi MATLAB***, cu caracter general, nespecifice unui anumit toolbox:

➢ ***comenzi generale ale mediului:***

intro – iniţiază o scurtă introducere în MATLAB.

help – furnizează informaţii referitoare la funcţiile MATLAB.

lookfor – permite căutarea după cuvinte-cheie doar în prima linie a help-ului tuturor funcţiilor MATLAB care se află în unul din directoarele specificate în MATLABPATH, şi respectiv în directorul curent.

Notă: MATLABPATH este o variabilă specială, care se creează automat la pornirea mediului MATLAB şi care specifică toate directoarele în care MATLAB va căuta fişierele.

demo – furnizează un set de programe demonstrative ce pot fi selectate printr-un meniu.

whatsnew – afişează conţinutul fişierelor de tip "read me" ale MATLAB-ului şi ale toolbox-urilor acestuia.

info – afişează informaţii despre MATLAB şi respectiv compania The MathWorks

subscribe – permite iniţierea unei proceduri interactive în urma căreia se generează un fişier (subscribe.log), cu ajutorul căruia se poate subscrie la MathWorks.

➢ ***comenzi de control ale variabilelor din memorie:***

disp – permite afişarea în fereastra de comenzi a unei matrici sub formă de text.

who – listează variabilele curente.

whos – listează informaţii despre variabilele curente (tipul, dimensiunea, etc.)

clear – permite ştergerea variabilelor existente în memorie.

pack – defragmentează memoria destinată stocării variabilelor (salvează toate variabilele pe disc, şterge memoria şi apoi le reîncarcă de pe disc într-o zonă continuă de memorie).

➢ ***comenzi pentru controlul directoarelor şi fişierelor:***

dir – listează conţinutul unui director (se pot folosi şi wildcard-uri).

what – listează doar fişierele ***.m**, ***.mat** şi ***.mex** dintr-un director.

cd – permite afişarea sau schimbarea directorului curent.

type – afişează în fereastra de comenzi conţinutul unui fişier text

➢ ***comenzi frecvent utilizate în programele MATLAB:***

input('TEXT') – permite introducerea de date de către utilizator (textul din paranteză este afişat pe ecran, după care se aşteaptă introducerea valorilor dorite).

Exemplu:

```
>> a=input('INTRODUCEŢI NUMĂRUL a ');
```

Pe ecran apare mesajul INTRODUCEŢI NUMĂRUL a, mediul MATLAB aşteptând introducerea valorii variabilei a (pot fi introduse şi matrici).

♦

menu('TITLE','OPTIUNE 1','OPTIUNE 2',...) - afişează pe ecran un meniu având titlul TITLE, împreună cu opţiunile scrise în continuare. Returnează numărul de ordine al opţiunii selectate.

if – instrucţiune condiţională. Poate avea clauzele "***elseif***" sau "***else***", şi în mod obligatoriu se încheie cu "***end***".

ones(M,N) – generează o matrice având M linii şi N coloane cu toate elementele iniţializate pe valoarea 1.

zeros(M,N) – generează o matrice având M linii şi N coloane cu toate elementele iniţializate pe valoarea 0.

pause(n) – provoacă suspendarea execuţiei unui program timp de **n** secunde. Dacă parametrul **n** nu este specificat execuţia programului este suspendată până la apăsarea unei taste.

2.5. Crearea şi rularea unui program

Un program MATLAB este un fişier text, având extensia **.m**, conţinând o succesiune de comenzi. Avantajul utilizării acestor fişiere este acela că se poate rula o secvenţă de instrucţii cu ajutorul unei singure comenzi, de apel a unui fisier program, ori de cate ori este nevoie, fără a fi necesară rescrierea întregii secvenţe.

Aceste fişiere program, numite şi fişiere script, pot fi create cu ajutorul oricărui editor de texte, cu condiţia ca fişierul rezultat să conţină doar text ASCII (fără caractere neafişabile) şi să poată fi salvat cu extensia **.m**. Iniţial, mediul MATLAB nu avea un editor propriu de texte, prima versiune care prezenta această facilitate fiind versiunea 5.0.

Pentru a crea un program în versiunea 6.0, se alege din meniul ferestrei principale opţiunea" **File>New>M-file**". în fereastra editorului care s-a deschis se poate introduce codul programului.

Exemplu:
Program prin care se citesc două valori de la tastatură şi apoi se afişează suma lor:

```
a=input('prima valoare=')
b=input('a doua valoare=')
c=a+b
```

Pentru a se putea rula programul de mai sus, acesta trebuie salvat, dupa care se alege opţiunea "**Debug>Run**" din meniul editorului.

♦

Notă: O altă modalitate de a rula un program în MATLAB este aceea de a scrie numele acestuia la prompter-ul MATLAB. Atenţie însă, *pentru a se putea executa programul, acesta trebuie să fie în directorul curent de lucru al mediului.*

De exemplu, dacă secvenţa de comenzi anterioară s-a salvat cu numele "**unu.m**", atunci aceasta poate fi rulată executând:

```
>> unu
```

iar efectul va fi:

```
>> unu
prima valoare=4
a =
     4
a doua valoare=3
b =
     3
c =
     7
```

♦

Dacă nu se doreşte afişarea pe ecran a valorilor de calcul intermediar, la finalul comenzilor respective se tastează semnul ";".

Comentariile se pot introduce în cadrul programelor, prin tastarea semnului "%" la începutul fiecărei linii care se doreşte a fi comentată.

Exemplu:
Secvenţa de comenzi anterioară se poate rescrie sub forma:

```
%program de insumare a 2 valori
%cele 2 valori pot fi scalari sau matrici
a=input('prima valoare=');
b=input('a doua valoare=');
%calculul sumei
c=a+b
```

iar în urma rulării se va obține:

```
>> unu
prima valoare=4
a doua valoare=3
c =
     7
```

♦

2.6. Structuri de control al fluxului informațional

În acest paragraf se vor prezenta patru dintre cele mai utilizate ***instrucțiuni pentru controlul fluxului informațional în*** MATLAB, şi anume:

- ***for***
- ***while***
- ***if***
- **case**

➢ Comanda ***for*** permite execuția repetată de un anumit număr de ori a unui grup de instrucții. Sintaxa acesteia este:

```
for variabila = expresie,
   secventa de comenzi care vor fi repetate
end
```

Exemplu:
Pentru a calcula primele 4 numere naturale impare se poate utiliza secvența de comenzi:

```
for i=0:3
 2*i+1
end
```

iar efectul ei este afişarea pe ecran a mesajului:

```
>>
ans =
     1
ans =
     3
ans =
     5
ans =
     7
```

♦

➢ Comanda ***while*** permite repetarea unui grup de instrucţii atât timp cât o anumită condiţie este îndeplinită. Sintaxa acestei comenzi este similară cu a comenzii *for* prezentată anterior:

```
while conditie
   secventa de comenzi care vor fi repetate
end
```

unde, de obicei, condiţia este un test logic.

Exemplu:
Pentru a construi un vector format din primele 4 numere impare se poate utiliza secvenţa:

```
i=0
while i<4
vector(i+1)= 2*i+1
i=i+1
end
```

♦

Pentru realizarea ***unui test logic*** se utilizează ***operatorii logici*** din tabelul următor:

Operatori logici în MATLAB	
Operator	**Semnificaţie**
==	Egal (comparare)
<	Mai mic decât
>	Mai mare decât
<=	Mai mic sau egal
>=	Mai mare sau egal
~=	Diferit
&	ŞI logic
|	SAU logic
~	NOT logic
Xor	SAU exclusiv

➢ Cu ajutorul comenzii ***if*** se pot rula diferite secvenţe de comenzi în funcţie de valoarea de adevăr a unor condiţii. Sintaxa este:

```
if  conditie1,
    primul grup de comenzi
elseif conditie2
    al doilea grup de comenzi
else
    al treilea grup de comenzi
end
```

În sintaxa comenzii se pot include oricâte clauze "***elseif***" sunt necesare.

Comanda verifică prima condiție și dacă este adevărată atunci se execută doar primul grup de comenzi, dacă nu a fost adevărată, atunci se verifică condițiile din clauzele "***elseif***". Dacă nici o condiție nu a fost validă, atunci se execută doar grupul de comenzi specificat după "***else***".

Din comanda ***"if"*** pot sa fie omise atât clauza "**elseif**", cât și "**else**".

Exemplu:

O secvență de program care preia (de la tastatură) valoarea unei temperaturi și afișează o valoare lingvistică de tipul: 'cald', 'rece' sau 'moderat'.

```
t=input('introduceti temperatura');
if t<18
    'rece'
elseif t <28
    'placut'
else
    'cald'
end
```

♦

➢ Începând cu versiunea 5.0 s-a introdus și comanda ***"case"***, similară funcțional cu cea din limbajul C, care permite executarea unui grup de comenzi în funcție de valoarea unei anumite expresii. Aceasta are sintaxa:

```
switch switch_expr
   case case_expr,
        primul grup de comenzi
   case {case_expr1, case_expr2, case_expr3,...}
        al doilea grup de comenzi
   otherwise,
        ultimul grup de comenzi
end
```

2.7. Probleme de studiat

1. Să se ruleze programele demonstrative integrate în mediul MATLAB, pe baza meniului furnizat de comanda **demo**.
2. Să se studieze sintaxa completă şi informaţiile oferite de HELP-ul limbajului, pentru cele mai uzuale instrucţii MATLAB, folosind comenzi de forma **help nume_instrucţie**.
3. Să se scrie un program MATLAB care să realizeze înmulţirea matricilor:

$$A=\begin{pmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9\end{pmatrix} \text{ şi respectiv } B=\begin{pmatrix}1 & 0 & 3\\ 5 & 1 & 2\\ 1 & 1 & 0\end{pmatrix}.$$

4. Să se scrie un program care să realizeze înmulţirea a două matrici ale căror elemente vor fi introduse de la tastatură.
5. Să se modifice programul de la punctul anterior astfel încât să fie capabil să efectueze orice operaţie aritmetică (+,-,*, :) asupra celor 2 matrici. Operaţia dorită se va specifica cu ajutorul unui meniu.

Capitolul 3

GENERAREA SOFT A UNOR SEMNALE DE TEST UTILIZATE ÎN ANALIZA ŞI SIMULAREA COMPORTĂRII SISTEMELOR TEHNICE

3.1. Considerații generale

Din punctul de vedere al analizei, respectiv simulării funcționării proceselor/sistemelor interesează în general modul cum răspunde sistemul (evoluția în timp a ieşirii lui) când este excitat cu un semnal de intrare.

Pentru a se putea analiza şi compara performanțele de regim tranzitoriu sau permanent ale proceselor/sistemelor, s-a convenit asupra utilizării unor semnale de test, având anumite forme standard de variație în timp, numite şi *semnale de intrare tipizate*.

Paragraful de față prezintă câteva modalități de generare soft a celor mai utilizate semnale de intrare tipizate, deterministe şi aleatoare, uzual folosite în analiza şi simularea comportării proceselor/sistemelor, precum şi a modurilor de reprezentare grafică şi posibilităților de scalare oferite de MATLAB.

Mediul de programare MATLAB permite generarea atât a semnalelor de intrare tipizate aperiodice (treaptă, rampă, impuls) şi periodice (semnal sinusoidal), cât şi a semnalelor de intrare compuse (utilizate atât ca semnale de test cât şi ca mărimi de prescriere).

Utilizarea altor tipuri de semnale de probă nu oferă, de regulă, informații utile suplimentare față de cazul utilizării semnalelor prezentate în continuare.

Nota: Deoarece se pune problema generării acestor semnale pe sisteme numerice, semnalele considerate în cele ce urmează vor fi definite pentru cazul discret (mai precis pentru valori discrete ale timpului).

Orice semnal poate fi generat în MATLAB, prin intermediul a doi vectori de aceeaşi dimensiune:

- un *vector bază de timp*, notat în continuare cu t, care conține momentele de timp discret pentru care se consideră valorile semnalului.

  Baza de timp se consideră de forma $t = \{i \ , \quad \text{unde } i \in \mathbf{R}_+\}$.

În general se lucrează cu momente de timp echidistante fapt care poate fi transpus în MATLAB printr-o declarare de tipul:

```
t=tinitial:pas:tfinal.
```

- Un al doilea vector, notat în continuare cu ***u***, care conţine *valorile eşantioanelor semnalului* corespunzătoare momentelor definite conform bazei de timp.

Comenzile MATLAB uzual folosite pentru *reprezentarea grafică* a semnalelor sunt:

plot – destinată reprezentării grafice bidimensionale;
grid – trasează un caroiaj în fereastra grafică curentă;
title – permite afişarea unui titlu pe grafic;
xlabel – permite afişarea unui text referitor la abscisă;
ylabel – permite afişarea unui text referitor la ordonată;
zlabel – permite afişarea unui text referitor la axa "z";
text – permite afişarea unui text în fereastra grafică.

➢ Pentru *reprezentarea grafică* a unui semnal se utilizează instrucţiunea ***plot***, a cărei sintaxă este:

```
plot(x,y) sau plot(x,y,c)
```

unde *x* şi *y* sunt doi vectori de aceeaşi dimensiune, sau respectiv două matrici având acelaşi număr de linii, iar *c* este un şir de caractere de dimensiune 1,2 sau 3 care specifică modul cum trebuie să se facă afişarea (tip de linie, culoare etc).

Nota: În cazul în care *x* şi *y* sunt matrici, se afişează în aceeaşi fereastră grafică (dar folosind culori diferite) un număr de grafice egal cu numărul de coloane al celor două matrici, considerând pe rând perechile formate din coloanele de acelaşi indice din ambele matrici (prima coloană din *x* cu prima din *y*, apoi a doua din ambele matrici ş.a.m.d.).

Şirul de caractere *c* poate conţine unul sau mai multe caractere, dar de pe coloane diferite conform tabelului de mai jos, corespunzătoare culorii, caracterului sau tipului de linie dorit.

CULOARE		CARACTER		TIP LINIE	
Y	galben	.	punct	-	continuă
M	magenta	o	cerc	:	punctată
C	albastru deschis	x	semnul "x"	-.	linie punct
R	roşu	+	plus	--	întreruptă
G	verde	*	stea		
B	albastru	s	pătrat		
W	alb	d	diamant		
K	negru	v	triunghi		
		^	triunghi		
		<	triunghi		
		>	triunghi		
		p	pentagon		
		h	hexagon		

Exemplu:

Pentru a trasa 10 puncte pe prima bisectoare, utilizând simbolul „diamond" şi culoarea albastră se poate considera următoarea secvenţa de comenzi:

```
>>x=0:9
>>y=x
>>plot(x,y,'db')
```

Efectul acestei secvenţe este prezentat în figura 3.1.

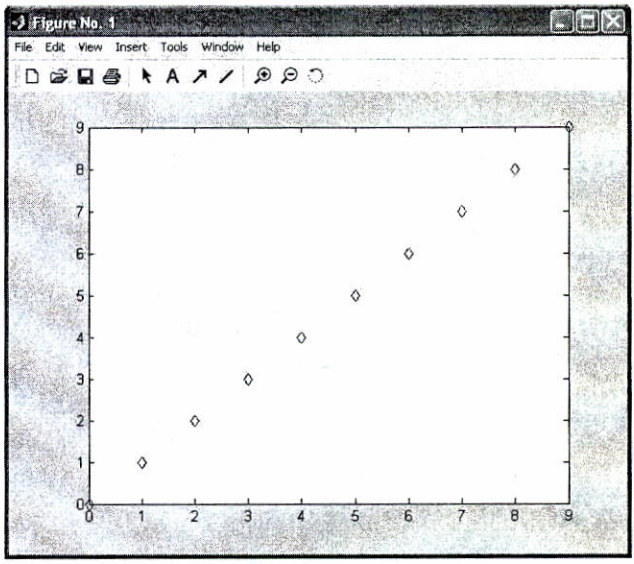

Figura 3.1. Prima bisectoare

♦

➢ Comanda care permite *trasarea unui caroiaj* în fereastra grafică curentă este ***grid***. Sintaxa acesteia este:

```
grid, grid on sau grid off.
```

Dacă funcţia ***grid*** se apelează cu parametrul "**on**" atunci ea are ca efect trasarea caroiajului în fereastra grafică curentă, iar dacă se apelează cu parametrul "**off**" efectul este ştergerea caroiajului din fereastra grafică curentă. Apelarea fără nici un parametru a comenzii *grid* are ca efect comutarea stării curente a caroiajului (dacă era ***on*** trece în ***off*** şi invers).

➢ Cu ajutorul comenzii ***title***, se poate adăuga *un titlu* în fereastra grafică curentă. Sintaxa comenzii este deosebit de simplă:

```
title(c)
```

unde ***c*** este un şir de caractere având valoarea titlului care se doreşte a fi afişat.

➢ Comenzile ***xlabel***, ***ylabel*** şi respectiv ***zlabel***, permit afişarea de texte (etichete) pe axele *x*, *y* şi respectiv *z* ale sistemului de coordonate din figura curentă. Sintaxa acestora este:

```
xlabel(c1)
ylabel(c2)
zlabel(c3)
```

unde ***c1***, ***c2*** şi respectiv ***c3*** sunt şiruri de caractere care vor fi afişate pe axele *x*, *y* şi respectiv *z*.

➢ Comanda ***text*** permite *afişarea unui text* în fereastra grafică curentă. Sintaxa acesteia permite următoarele forme de apel:

```
text(x,y,c) sau text(x,y,z,c)
```

unde ***x,y,z*** sunt coordonatele din fereastra grafică curentă unde se doreşte afişarea textului specificat de şirul de caractere ***c***.

Dacă *x,y* sau respectiv *x,y,z* sunt vectori de aceeaşi dimensiune, iar *c* este un şir de caractere, atunci mesajul conţinut în acesta va fi afişat pe ecran în toate locaţiile specificate. Există şi posibilitatea ca *c* să fie de forma unui tablou de şiruri de caractere, de aceeaşi lungime cu vectorii *x,y* sau *z*. Intr-un astfel de caz comanda afişează pe ecran fiecare element al tabloului *c* la locaţia corespunzătoare, conform exemplului de mai jos:

```
>>text(0.10*(1:10),0.10*(1:10),['aa';'bb';'cc';'dd';'ee';'ff'
;'gg';'hh';'ii';'jj'])
```

Efectul acestei comenzi poate fi observat în figura 3.2.

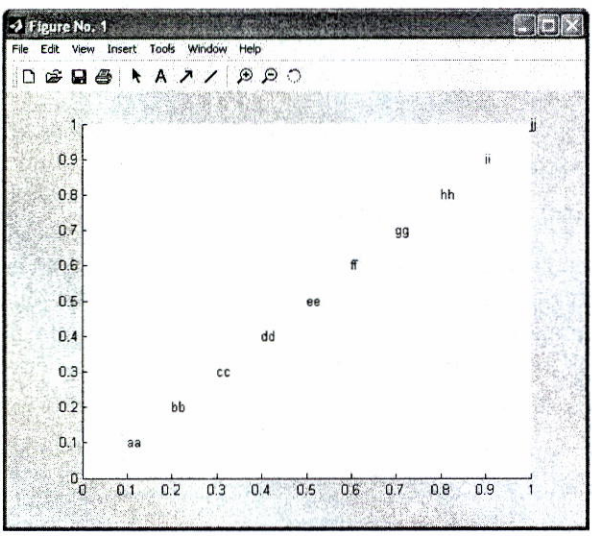

Figura 3.2. Afişarea unui text în fereastra grafică

♦

3.2. Generarea semnalelor de test tipizate

Problema generării semnalelor de test tipizate se reduce la construirea corespunzătoare a celor doi vectori: *vectorul bazei de timp* , notat cu ***t***, şi respectiv *vectorul semnalului*, notat cu ***u***.

3.2.1. Semnal treaptă

Expresia analitică de definire a unui *semnal treaptă de amplitudine "**a**"* este:

$$u(t)=\begin{cases}0 & \text{pentru } t<0\\ a & \text{pentru } t\geq 0\end{cases} \qquad (3.1)$$

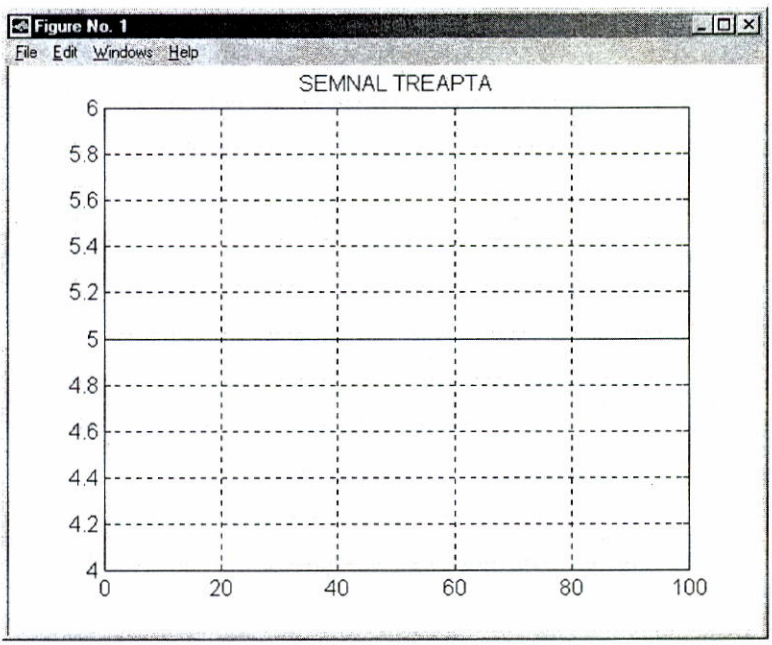

Figura 3.3. Semnal treaptă de amplitudine 5 şi durată 100

Exemplu:
Pentru generarea unui semnal treaptă având forma prezentată în figura 3.3, se poate utiliza următoarea secvenţă de comenzi:

```
>> n=100;
>> a=5;
>> u=a*ones(1,n);
>> t=1:n;
>> plot(t,u,'b')
```

```
>> grid on
>> title('SEMNAL TREAPTA')
```

♦

3.2.2. Semnal rampă

Expresia analitică de definire a unui *semnal rampă de pantă "a"* este:

$$u(t) = \begin{cases} 0 & \text{pentru } t < 0 \\ a \cdot t & \text{pentru } t \geq 0 \end{cases} \tag{3.2}$$

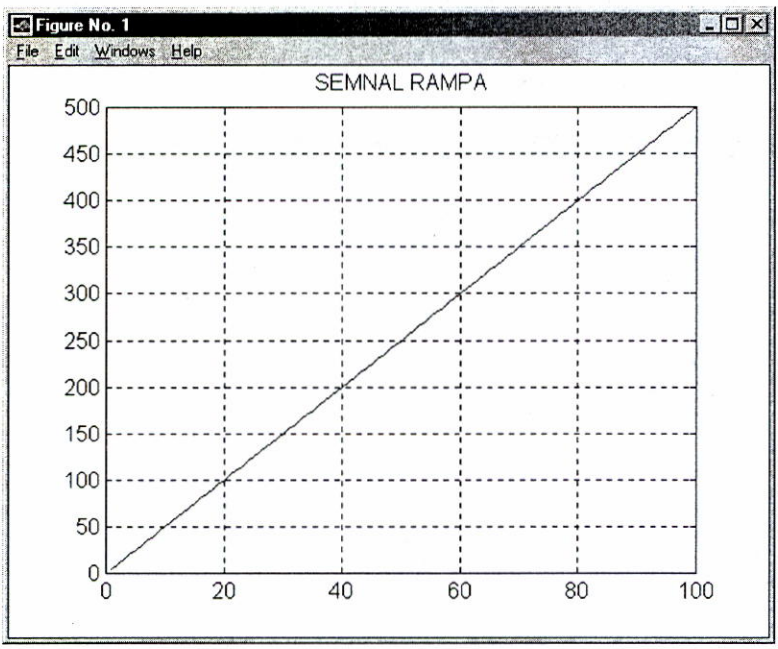

Figura 3.4. Semnal rampă de pantă 5 şi durată 100

Exemplu:
Secvenţa de comenzi cu ajutorul căreia poate fi generat un semnal de forma prezentată în figura 3.4 este detaliată în continuare:

```
>> n=100;
>> a=5;
>> t=1:n;
>> u=a*t;
>> plot(t,u,'b')
>> grid on
>> title('SEMNAL RAMPA')
```

♦

3.2.3. Semnal impuls

Expresia analitică a unui *semnal impuls de amplitudine "**a**" şi lăţime "**b**"* este:

$$u(t) = \begin{cases} a & \text{pentru } t \in [0,b] \\ 0 & \text{pentru } t \notin [0,b] \end{cases} \tag{3.3}$$

Exemplu:
Secvenţa de comenzi cu ajutorul căreia poate fi generat un semnal tip impuls de forma prezentată în figura 3.5 este următoarea:

```
>> n=100;
>> a=0.95;
>> b=30;
>> t=1:n;
>> u=zeros(1,n);
>> for i=1:min([n b])
u(i)=a;
end
>> plot(1:n,u,'b')
>> grid on
>> title('SEMNAL IMPULS')
```

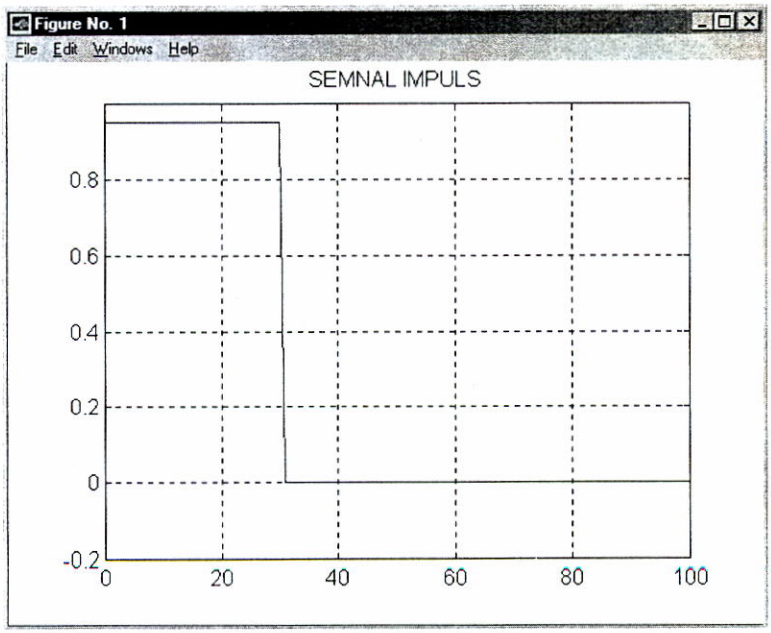

Figura 3.5. Semnal impuls de amplitudine 0.95, lăţime 30, în cadrul unei secvenţe de durată 100

♦

3.2.4. Semnal sinusoidal

Expresia analitică a *semnalului sinusoidal de amplitudine "**a**", perioadă "**T**" şi fază "**φ**"* este:

$$u(t) = A \cdot \sin\left(\frac{2\pi}{T}t + \varphi\right) \tag{3.4}$$

Exemplu:
Pentru generarea unui semnal sinusoidal de forma prezentată în figura 3.6, se poate folosi următoarea secvenţă de comenzi:

```
>> n=100;
>> a=5;
>> T=30;
>> fi=0;
>> t=1:n;
>> u=a*sin(2*pi*t/T+fi*ones(1,n));
>> plot(t,u,'b')
>> grid on
>> title('SEMNAL SINUS')
```

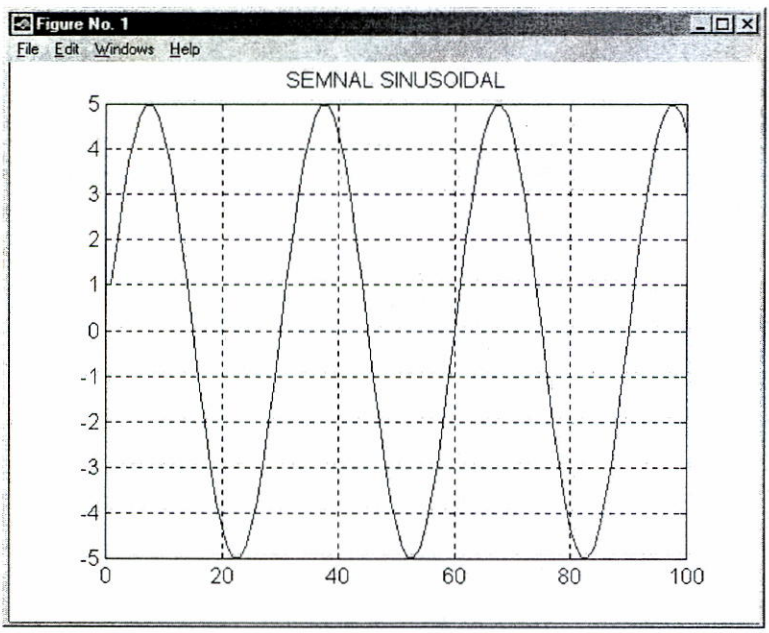

Figura 3.6. Semnal sinusoidal de amplitudine 5, perioadă 30 şi durată 100

♦

3.3. Generarea semnalelor de test aleatoare

3.3.1. Semnale aleatoare uniform distribuite

Pentru a genera *semnale aleatoare uniform distribuite* se utilizează comanda ***rand***.

Sintaxa acesteia permite următoarele forme de apel:

```
rand (n) sau rand(m,n)
```

Această comandă returnează o matrice de dimensiune [*nxn*] sau respectiv, [*mxn*], având elementele aleatoare uniform distribuite.

Exemplu:

Pentru generarea unui semnal aleator uniform distribuit, de forma prezentată în figura 3.7, se poate folosi următoarea secvență de comenzi:

```
>> n=100;
>> t=1:n;
>> u=rand(1,n);
>> plot(t,u,'b')
>> grid on
>> title('SEMNAL ALEATOR UNIFORM DISTRIBUIT')
```

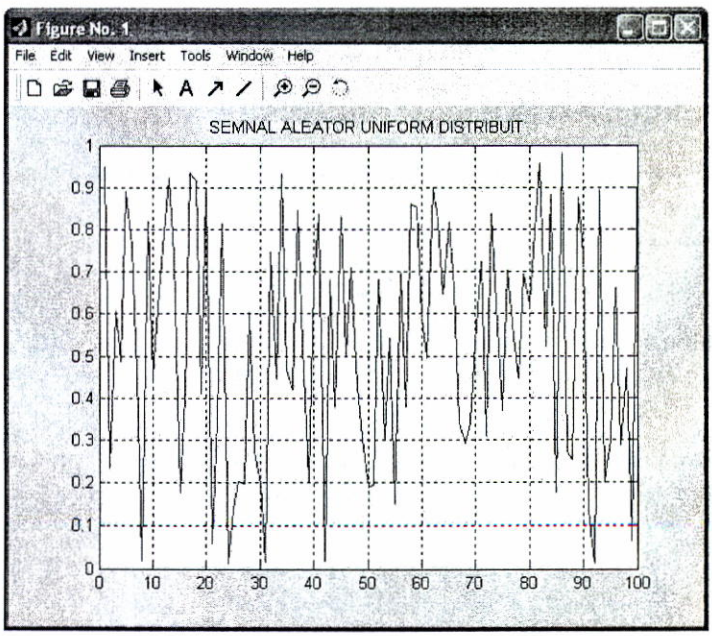

Figura 3.7. Semnal aleator uniform distribuit

♦

Pentru a studia *distribuţia semnalului* se poate trasa ***histograma*** acestuia cu ajutorul comenzii:

```
>> hist(U);
```

iar efectul acesteia este redat în figura 3.8

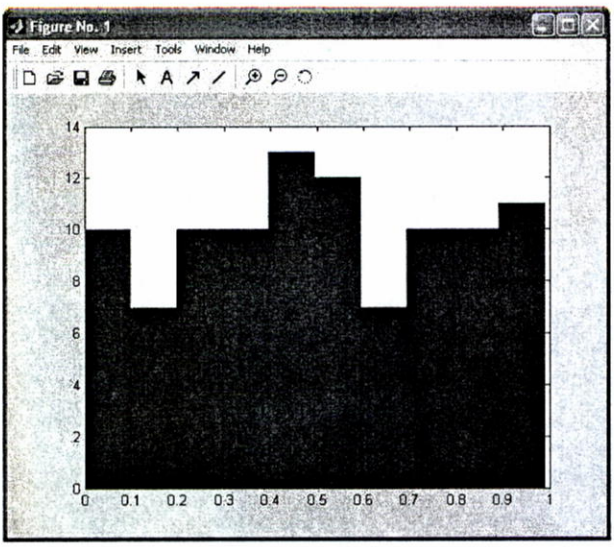

Figura 3.8. Histograma semnalului aleator uniform distribuit

♦

3.2.3. Semnale aleatoare normal distribuite

Pentru a genera semnale **aleatoare normal distribuite** se utilizează comanda ***randn***.

Sintaxa acesteia permite următoarele forme de apel:

```
randn (n) sau randn(m,n)
```

această comandă returnează o matrice având dimensiunea [n x n] sau respectiv, [m x n], având elementele aleatoare cu distribuţie normală.

Exemplu:
Pentru generarea unui semnal aleator normal distribuit, de forma prezentată în figura 3.9, se poate utiliza următoarea secvenţă de comenzi:

```
>> n=100;
>> t=1:n;
>> un=randn(1,n);
>> plot(t,un,'b')
>> grid on
>> title('SEMNAL ALEATOR NORMAL DISTRIBUIT')
```

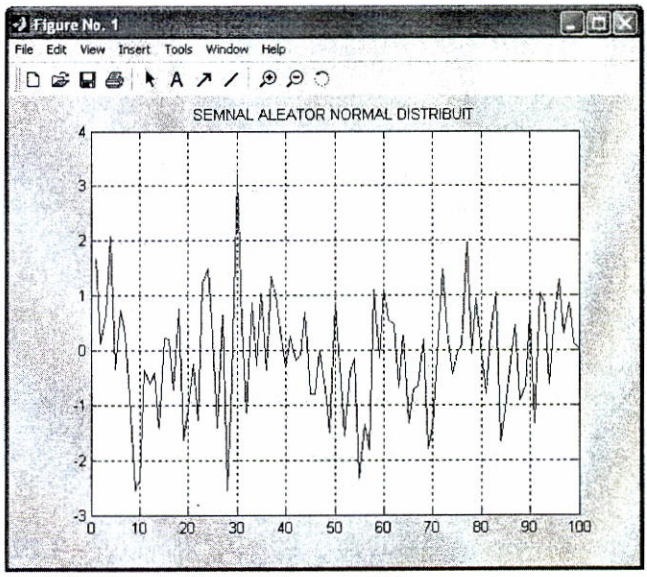

Figura 3.9. Semnal aleator normal distribuit

Pentru a studia **distribuția semnalului** se poate trasa **histograma** acestuia cu ajutorul comenzii:

```
>> hist(Un);
```

iar efectul acesteia este redat în figura 3.10.

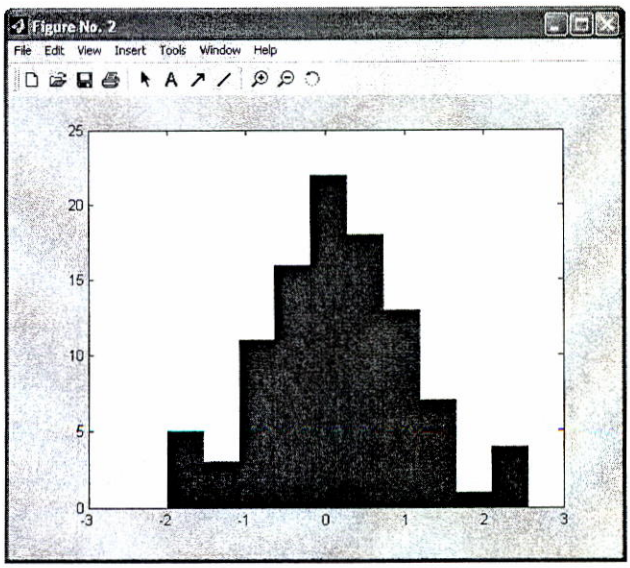

Figura 3.10. Histograma semnalului aleator normal distribuit

◆

3.3.3. Semnale pseudoaleatoare binare (SPAB)

Acest tip de semnal se utilizează cu preponderenţă în tehnicile de estimare a parametrilor sistemelor.

SPAB reprezintă de fapt semnale deterministe şi nu aleatoare (generate după o lege bine determinată) care au proprietăţi statistice apropiate de cele ale zgomotului alb.

SPAB-ul se prezintă sub forma unor succesiuni de valori binare, care se repetă cu o anumită perioadă, cele mai frecvent utilizate fiind cele de perioadă maximă $N = 2^{p-1}$ (unde p este un număr întreg), generate prin relaţii de recurenţă liniară.

Principalele ***proprietăţi*** ale unui SPAB sunt următoarele :

- semnalul se prezintă ca o succesiune de impulsuri de durata Δ şi multiplii de Δ, putând lua valorile $(+a,-a)$ sau $(+a,0)$.
- este un semnal periodic, numărul maxim de intervale elementare dintr-o perioadă este $N = 2^{p-1}$, durata maximă a unei perioade fiind deci $N\Delta$.
- în fiecare perioadă numărul de intervale în care semnalul are valoarea +a $(+a)$ excede cu o unitate numărul de intervale elementare în care semnalul are valoarea $-a$ (0).
- permutarea ciclică în cadrul unei perioade produce un SPAB întârziat faţă de cel permutat.
- suma modulo 2 (produsul) a două secvenţe întârziate una faţă de alta produce tot un SPAB de perioadămaximă.

Pentru un SPAB $(+a,-a)$ maximal, continuu, funcţia de autocorelaţie are expresia:

$$R_u(\tau) = \begin{cases} a^2\left(1 - \dfrac{|\tau|}{\Delta}\dfrac{N+1}{N}\right) & |\tau| \le \Delta \\ -\dfrac{a^2}{N} & \Delta < |\tau| \le (N-1)\Delta \end{cases} \tag{3.5}$$

şi este prezentată în figura 3.11.

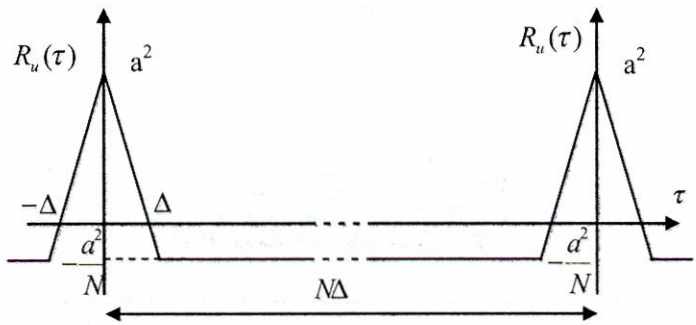

Figura 3.11. Funcţia de autocorelaţie a SPAB (+a,-a) *maximal, continuu*

În figura 3.12 este reprezentată funcția de autocorelație pentru un SPAB (+*a*,-*a*) discret.

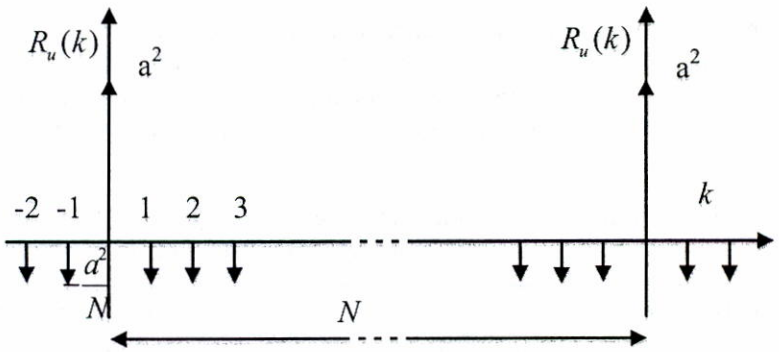

Figura 3.12. Funcția de autocorelație a SPAB (+a,-a) maximal, discret

SPAB reprezintă o bună aproximare a zgomotului alb numai dacă N este suficient de mare, iar Δ suficient de mic.

Semnalele pseudoaleatoare binare de perioadă maximă $N = 2^{p-1}$, pot fi generate foarte simplu prin utilizarea unui registru de deplasare cu reacție având *p* etaje, operația constând din suma modulo doi a semnalelor de la anumite etaje ale registrului, după cum este arătat în figura 3.13.

Semnalul de lungime maximă $N = 2^{p-1}$, generat cu un astfel de registru de deplasare, poate fi obținut doar pentru anumite combinații logice pe legătură de reacție, în general rezultând succesiuni de perioade mai reduse.

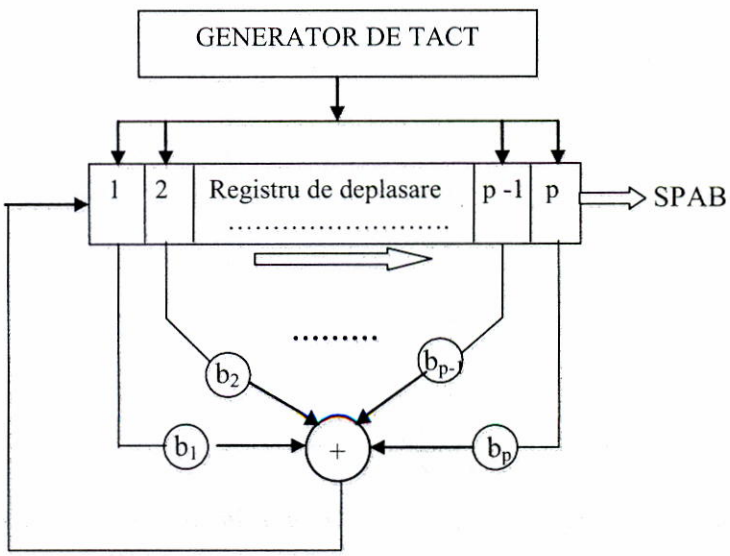

Figura 3.13. Schema de generare a unui SPAB

Parametrii b_i pot lua doar valorile 1 sau 0: valoarea 1 dacă bistabilul *i* contribuie la reacție și respectiv valoarea 0 dacă bistabilul *i* nu contribuie la reacție.

Determinarea acelor combinații care să asigure perioada de lungime maximă cu un număr minim de funcții logice de tipul suma modulo doi, constituie o problemă de minimizare logică, care poate fi soluționată prin metode analitice (exprimarea matricială și deducerea polinoamelor caracteristice) sau prin metode grafice (diagrame de stare).

Modul în care trebuie construită reacția pentru a obține o secvență maximală (valorile coeficienților b_i) pentru câteva valori ale lui *p*, este redat în următorul tabel:

Nr. celule (p)	b_1	b_2	b_3	b_4	b_5	b_6	b_7	Perioada (N)
3	0	1	1	-	-	-	-	7
4	0	0	1	1	-	-	-	15
	1	0	0	1	-	-	-	
5	0	0	1	0	1	-	-	31
	0	1	0	0	1	-	-	
	0	1	0	0	1	0	-	63
7	0	0	0	1	0	0	1	127

Astfel, generarea unei secvențe SPAB de lungime maximă N=31 se poate face utilizând schema reprezentată în figura 3.14.

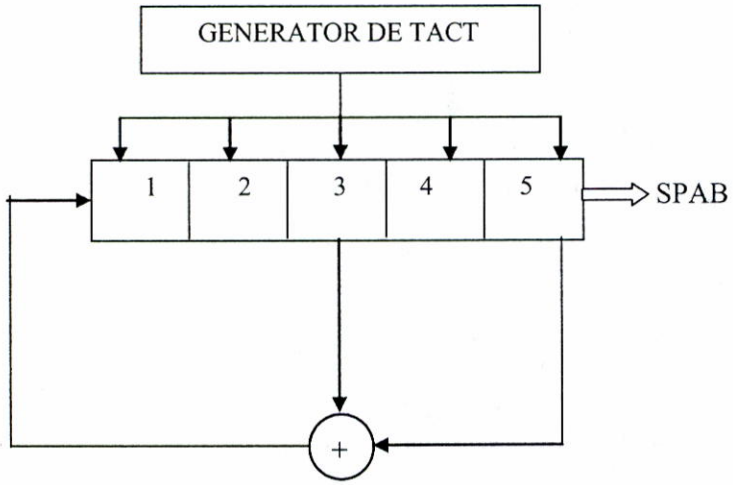

Figura 3.14. Schema de generare a unui SPAB utilizând un registru cu 5 etaje

Notă: Secvența astfel generată poate lua numai două valori (0 sau 1), de aceea se și numește SPA binară.

Exemplu:

Pentru generarea unui SPAB de lungime maximă $N = 2^{5-1} = 31$, utilizând un registru de deplasare cu 5 etaje, cu $b_3 = b_5 = 1$, de forma prezentată în figura 3.14, se poate utiliza următoarea secvența de comenzi:

```
clear
n=60
b=[1 0 0 0 0 ]
u=[]
for i=1:n
    u(i)=xor(b(3),b(5));
    b(5)=b(4);
    b(4)=b(3);
    b(3)=b(2);
    b(2)=b(1);
    b(1)=u(i);
end
```

Utilizarea unei comenzi *plot* ar avea ca efect obținerea unui grafic de tipul celui din figura 3.15, forma astfel obținuta a semnalului nereprezentând un SPAB.

```
figure
plot(u)
axis([0 n -0.5 1.5])
grid on
title('SEMNAL PSEUDOALEATOR BINAR')
```

Pentru a reprezenta grafic o formă corectă de SPAB discret se poate utiliza comanda ***stem***, având ca rezultat graficul prezentat în figura 3.16. Acest lucru presupune rularea, după ce în prealabil s-a executat secvența anterioară de generare a SPAB a următoarelor comenzi:

```
figure
stem(u)
axis([0 n -0.5 1.5])
grid on
title('SEMNAL PSEUDOALEATOR BINAR')
```

Dacă se dorește reprezentarea grafică a unui SPAB continuu (echivalentă trecerii unui SPAB discret printr-un element de reținere) se poate utiliza comanda ***stairs***:

```
figure
stairs(u)
axis([0 n -0.5 1.5])
grid on
title('SEMNAL PSEUDOALEATOR BINAR')
```

fapt care conduce la rezultatul grafic prezentat în figura 3.17.

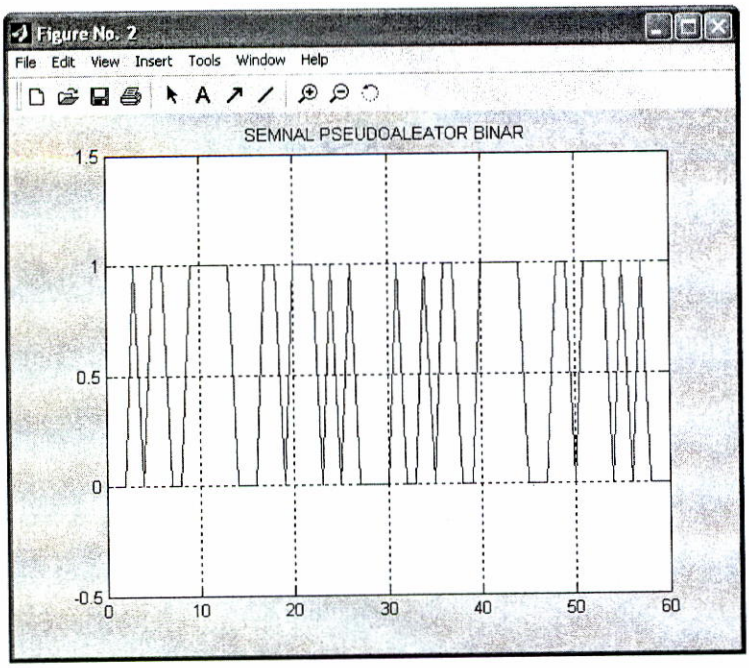

Figura 3.15. Afişare semnal generat cu comanda plot

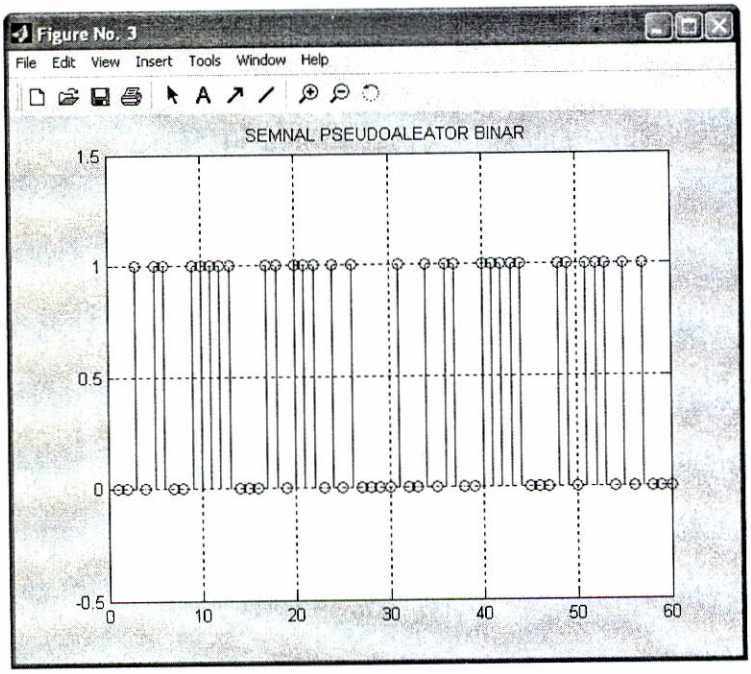

Figura 3.16. Semnal pseudoaleator binar discret

Notă: Dacă în schema prezentată în figura 3.13, operația *modulo* 2 este înlocuită cu *modulo m*, se vor obține SPA cu *m* nivele (SPAM), care însă nu se mai pot genera hardware, generarea lor software fiind relativ simplă.

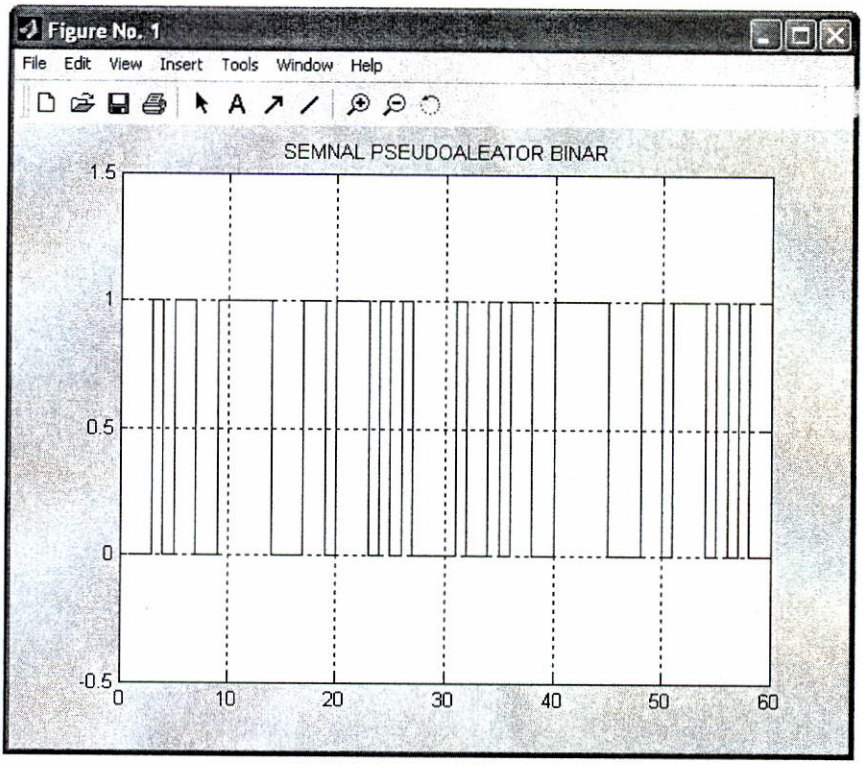

Figura 3.17. Semnal pseudoaleator binar

♦

3.4. Generarea semnalelor de intrare compuse

Generarea unor semnale mai complexe se poate realiza prin compunerea unor semnale simple, de genul celor prezentate în paragraful anterior şi nu numai. Această compunere se poate realiza în două moduri:

- însumarea (punct cu punct) a semnalelor;
- concatenarea semnalelor.

❑ Compunerea mai multor semnale prin *însumare*, se realizează adunând seturile de eşantioane ale semnalelor corespunzătoare aceluiaşi moment de timp.

Astfel, dacă se consideră două semnale, *u1*, *u2*, respectiv baza de timp *t*, având formele următoare:

u1=[a1 a2 a3 a4 ... an]
u2=[b1 b2 b3 b4 ... bn]
t=[t1 t2 t3 t4 ... tn]

Prin *însumarea* celor două semnale se obţine un nou semnal, notat *us*, de aceeaşi lungime ca şi *u1* şi *u2*. Baza de timp *t* rămâne nemodificată.

us=[a1+b1 a2+b2 a3+b 3 a4+b4 ... an+bn]

Notă: Semnalele a căror însumare se realizează în vederea generării unui nou semnal trebuie să aibă aceeaşi lungime (număr de eşantioane).

- Compunerea mai multor semnale prin *concatenare*, se realizează alăturând succesiv seturile de eşantioane corespunzătoare semnalelor şi extinzând în mod corespunzător baza de timp.

În cazul *concatenării*, se obţine un nou semnal, notat *uc*, de lungime egală cu suma lungimilor semnalelor originale, iar baza de timp se va extinde corespunzător:

uc=[a1 a2 a3 a4 ... an b1 b2 b3 b4 ... bn]
tc=[t1 t2 t3 t4 ... tn tn+t1 tn+t2 tn+t3 tn+t4 ... tn+tn]

Exemplu:
Se pune problema compunerii prin însumarea (punct cu punct) a celor trei semnale prezentate în figura 3.18, adică:

- un semnal treaptă de amplitudine 3,
- un semnal rampă de pantă 0.5,
- un semnal periodic dreptunghiular de amplitudine 75, factor de umplere 3/5 şi perioadă 50.

Pentru a însuma semnalele prezentate anterior, se poate utiliza următorul program:

```
% se definesc parametrii semnalelor de insumat
n=200;
t=1:n;
amplit_tr=3;
panta=0.5;
f=3/5;
amplit_dr=75;
T=50;
% se genereaza semnalele treapta şi rampa
u1= amplit_tr*ones(1,n);
u2=panta*t;
% se genereaza semnalul dreptunghiular pentru o perioada
u31= amplit_dr*[ones(1,f*T) zeros(1,(1-f)*T)];
```

```
% se extinde secventa initiala cu un numar necesar de perioade
u3=u31
for i=1:n/T-1
    u3=[u3 u31];
end
% se insumeaza cele trei semnale punct cu punct
u=u1+u2+u3;
% se afiseaza grafic semnalul compus
plot(1:n,u,'-')
grid on
title('SEMNAL OBTINUT PRIN INSUMARE')
```

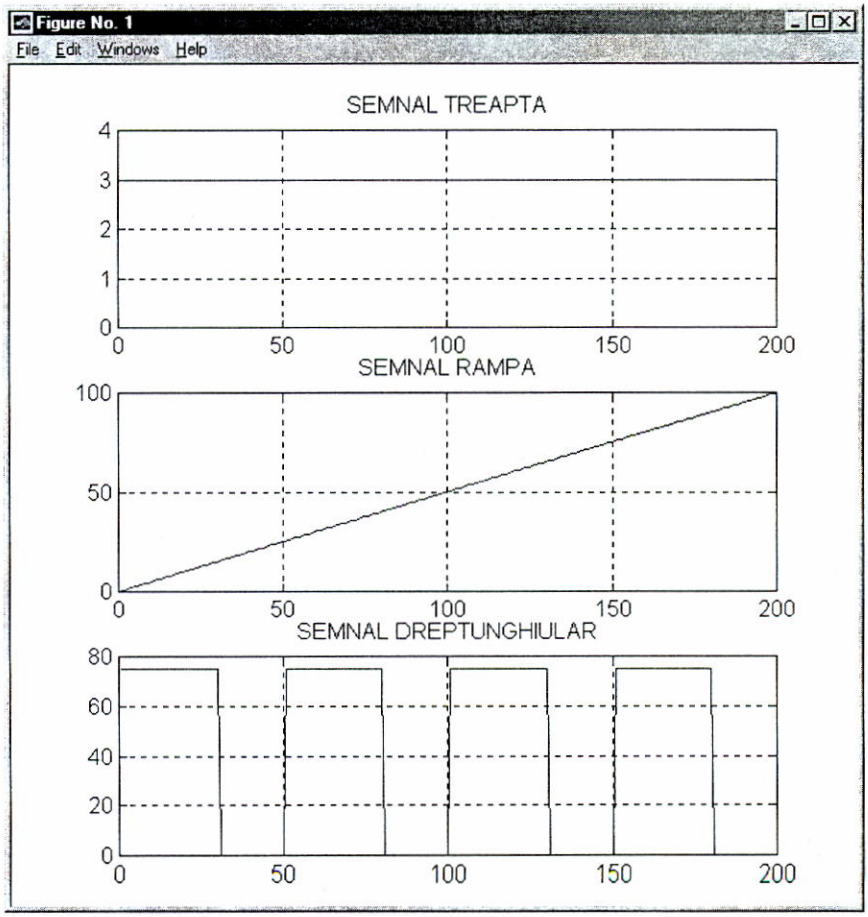

Figura 3.18. Semnale folosite la însumare

Semnalul '*u*' rezultat, se obține prin însumarea punct cu punct a celor trei semnale u1, u2, u3, așa cum este prezentat în graficul din figura 3.19.

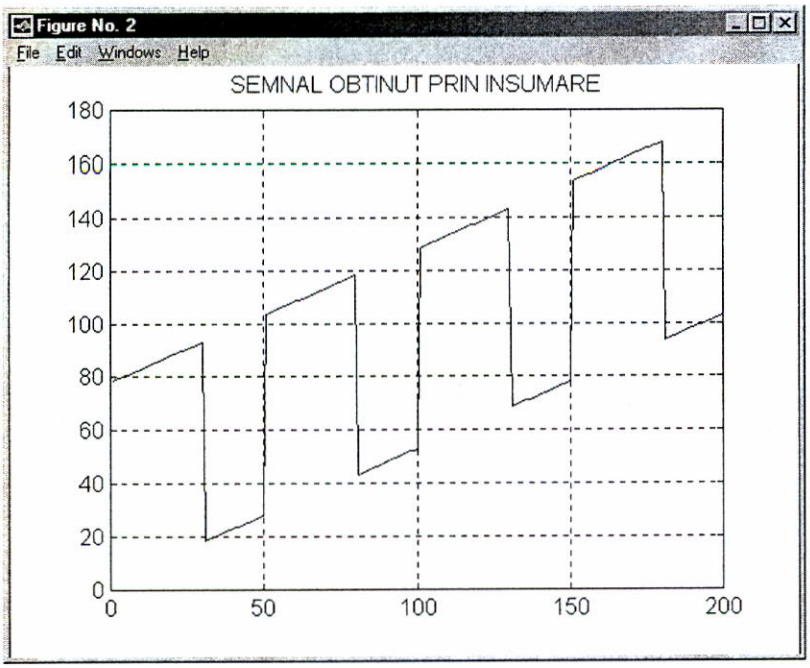

Figura 3.19. Semnalul compus prin însumare punctiformă

♦

Exemplu:
Se consideră semnalele din exemplul anterior (vezi figura 3.18) şi se doreşte concatenarea lor. Acest lucru este realizat prin programul următor:

```
% se definesc parametrii semnalelor de insumat
n=200;
t=1:n;
amplit_tr=3;
panta=0.5;
f=3/5;
amplit_dr=75;
T=50;
% se genereaza semnalele treapte şi rampa
u1= amplit_tr*ones(1,n);
u2=panta*t;
% se genereaza semnalul dreptunghiular
u31= amplit_dr*[ones(1,f*T) zeros(1,(1-f)*T)];
u3=u31
for i=1:n/T-1
    u3=[u3 u31];
end
```

3.5. Probleme de studiat

1. Să se scrie un program de generare a unui semnal periodic, simetric, triunghiular de amplitudine maximă a = 3, perioadă T = 20 şi lungime a secvenţei l = 100;
2. Să se scrie un program de generare a unui semnal obţinut prin însumarea unui semnal sinusoidal cu un semnal rampă (caz general, deci se va prevedea posibilitatea introducerii parametrilor semnalelor de la tastatură).
3. Să se explice de ce semnalele care se însumează în vederea generării unui semnal compus, utilizând mediul de programare MATLAB, trebuie sa aibă aceeaşi lungime.
4. Să se scrie un program pentru generarea unui semnal dreptunghiular cu factorul de umplere 1/3, perioada T = 6, amplitudinea a = 3 şi lungimea l = 18.
5. Să se scrie un program pentru generarea unui semnal obţinut prin suprapunerea unui semnal rampă de pantă 2, peste un semnal sinusoidal de amplitudine a = 10, perioadă T = 2 şi lungime l = 20.

```
% se concateneaza cele trei semnale
u=[u1 u2 u3];
% se extinde baza de timp
te=[t t+n t+2*n];
% se afiseaza grafic semnalul compus
plot(te,u,'-')
grid on
title('SEMNAL OBTINUT PRIN CONCATENARE')
```

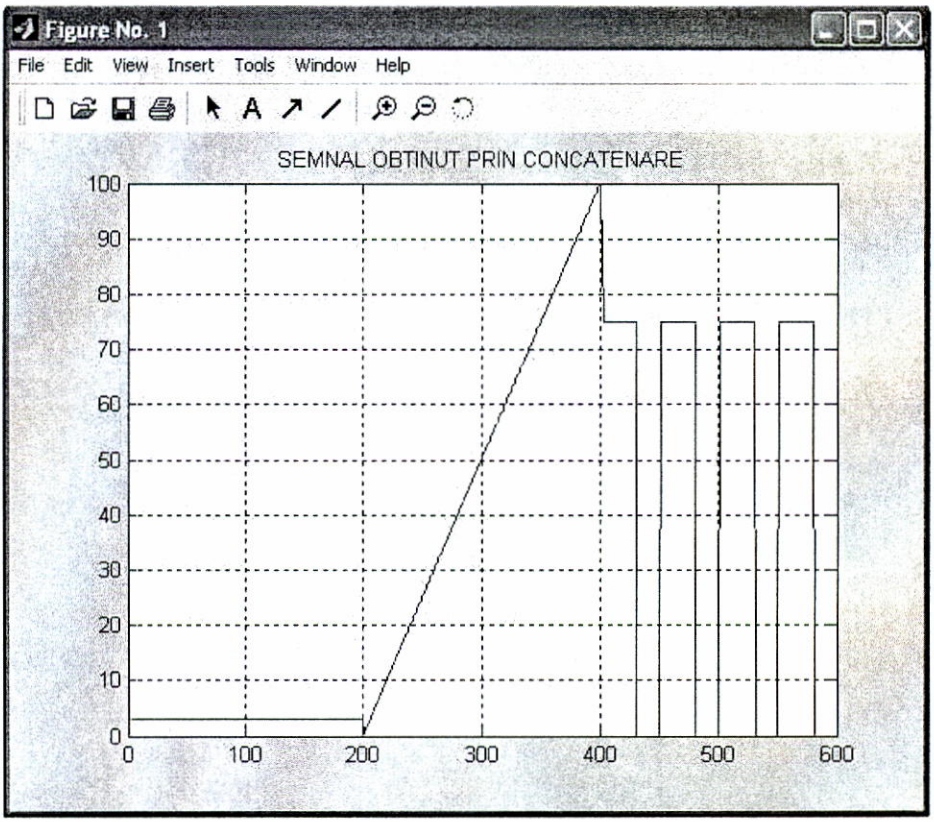

Figura 3.20. Semnalul compus prin concatenare

Semnalul '*u*' obținut prin concatenarea celor trei semnale *u1, u2, u3*, este prezentat în figura 3.20.

♦

Capitolul 4

MODELAREA SISTEMELOR LINIARE INVARIANTE

4.1. Considerații teoretice

Paragraful de față își propune o trecere în revistă a principalelor modalități de reprezentare matematică a sistemelor liniare, invariante (modele matematice), respectiv a posibilităților de efectuare a conversiilor de la o formă de reprezentare la alta pentru același sistem.

Se vor considera numai modele **liniare** (putându-se aplica principiul superpoziției), **continue** (mărimile variază continuu în timp) respectiv **discrete** (sunt disponibile doar valori eșantionate) și **invariante** (parametri constanți în timp).

În continuare se vor prezenta succint câteva elemente de bază privind principalele modele matematice (MM) din domeniul timp și din domeniul operațional. Modelele din domeniul frecvență (caracteristicile de frecvență) se vor trata distinct în paragraful 8.

În cadrul abordării sistemice, în domeniul timp (t – variabilă explicită) se operează cu două categorii de modele "tipizate" considerate în formă standard, ceea ce permite abordarea unitară a oricăror probleme de analiză/sinteză a sistemelor (de orice natură) și anume ***MM intrare-ieșire*** și respectiv ***MM intrare-stare-ieșire***.

a. Modele matematice intrare–ieșire: MM-II

MM-II reprezintă o relație doar între mărimile de intrare u și ieșire y ale sistemului S (vezi figura 4.1) și derivatele lor de diferite ordine.

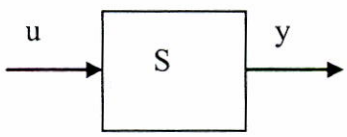

Figura 4.1. Orientarea sistemului S

Forma generală a modelului matematic intrare-ieșire, ***MM-II****, pentru sisteme* ***monovariabile (u și y scalari), continue și invariante*** *este o* ***ecuație diferențială cu coeficienți constanți****:*

$$a_n y^{(n)}(t) + \ldots + a_1 y^{(1)}(t) + a_0 y(t) = b_m u^{(m)}(t) + \ldots + b_1 u^{(1)}(t) + b_0 u(t) \qquad \text{MM-II}$$

(caz continuu)

sau

$$\sum_{i=0}^{n} a_i y_i^{(i)}(t) = \sum_{j=0}^{m} b_j u_j^{(j)}(t) \tag{4.1}$$

sau în cazul prezentei timpului mort Tm:

$$\sum_{i=0}^{n} a_i y_i^{(i)}(t) = \sum_{j=0}^{m} b_j u_j^{(j)}(t - Tm) \tag{4.2}$$

Pentru sisteme fizic realizabile ***n>m****,* ***n*** *reprezentând* ***ordinul sistemului****.*

Algoritmul de determinare al unui MM-II este următorul:
1. Se scriu ecuaţiile corespunzătoare legilor fizicii care descriu funcţionarea sistemului respectiv
2. Se stabilesc variabilele terminale (mărimile de intrare/ieşire)
3. Se elimină succesiv variabilele intermediare (care nu sunt terminale) din sistemul de ecuaţii determinat la punctul 1, până se obţine o relaţie de dependenţă numai între mărimile de intrare/ieşire şi derivatele lor (de forma 4.1).

Pentru descrierea comportării dinamice a sistemelor în care sunt disponibile numai valori eşantionate ale semnalelor de intrare şi de ieşire (sisteme discrete), în locul ecuaţiilor diferenţiale se utilizează ecuaţii cu diferenţe (recurente), forma generală a MM-II discret fiind următoarea:

$$\begin{aligned} &y(t) + a_1 y(t-1) + \ldots + a_{na-1} y(t-na+1) + a_{na} y(t-na) = \\ &= b_1 u(t-1) + \ldots + b_{nb-1} u(t-nb+1) + b_{nb} u(t-nb) \end{aligned} \qquad \text{MM-II}$$

sau (caz discret)

$$\left(y(t) + \sum_{i=1}^{na} a_i y(t-i) \right) = \sum_{j=1}^{nb} b_j u(t-j), \tag{4.3}$$

unde $t = 0,1,2,\ldots$ *timp discret normalizat*

sau în cazul prezenţei unui timp mort k:

$$\left(y(t) + \sum_{i=1}^{na} a_i y(t-i) \right) = \sum_{j=0}^{nb} b_j u(t-k-j) \tag{4.4}$$

Observaţie:
În MATLAB, MM-II nu este tratat în mod explicit, neexistând funcţii corespunzătoare acestui tip de model. Prezentarea lui însă s-a considerat necesară pentru o înţelegere corectă, de ansamblu, a problemei modelării sistemelor.

b. Modele matematice intrare-stare-ieşire: MM-ISI

Nevoia de a dispune de cât mai multă informaţie despre sistem, în special când se pune problema conducerii lui, pune în multe cazuri în evidenţă, insuficienţa caracte-

rizării prin relația intrare-ieşire – impunând introducerea unor mărimi suplimentare – mărimile de stare.

***Mărimile de stare**, definite prin intermediul **variabilelor de stare** – sunt mărimi interne ale sistemului – care descriu complet „starea" sistemului la un anumit moment de timp, pe baza lor putând fi determinată evoluția viitoare a sistemului. Ele conțin informații despre istoria trecută şi prezentă a sistemului.*

Variabilele de stare constituie componentele ***vectorului de stare*** notat cu ***x(t)***:

$x(t) = [x_1(t) \quad x_2(t) \quad \ldots \quad x_n(t)]^T$ - vectorul mărimilor de stare

Alături de mărimile de intrare $u(t)$ şi de ieşire $y(t)$, mărimile de stare $x(t)$ reprezintă ***mărimile caracteristice*** ale sistemului.

Mărimile de stare pot fi în acelaşi timp mărimi de ieşire, în timp ce mărimile de intrare sunt mărimi exogene sistemului.

Dacă în cadrul MM-II s-a evidențiat o dependență directă intrare-ieşire: **u→y**, în cadrul caracterizării de stare dependența intrare-ieşire este defalcată în două prin introducerea ***mărimii de stare*** **u→x→y**.

Forma standard a modelului de stare este următoarea:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \quad ; \quad x(0_+) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \qquad \text{MM-ISI,}$$

(caz continuu) (4.5)

cu $x(0_+)$ – stare inițială.

- în cazul sistemelor monovariabile – SISO (Single Input Single Output), $u(t)$ şi $y(t)$ sunt scalari, iar:
  A – matricea sistemului, dim[nxn]
  B – vector de intrare, dim[nx1]
  C – vector de ieşire, dim[1xn]
  D – dim[1x1], *pentru sisteme fizic realizabile: D=0*
  n – defineşte ordinul sistemului (este egal cu numărul componentelor vectorului de stare, adică cu numărul variabilelor de stare)
- în cazul sistemelor multivariabile – MIMO (Multi Input Multi Output), $u(t)$ şi $y(t)$ sunt vectori, pentru un sistem cu q intrări şi r ieşiri, rezultând:

  $u(t)=[u_1(t), u_2(t), \ldots, u_q(t)]^T$

  $y(t)=[y_1(t), y_2(t), \ldots, y_r(t)]^T$

Coeficienții A, B, C, D devin matrici de următoarele dimensiuni:
$A[nxn]$ - matricea sistemului, $B[nxq]$ - matricea de intrare (sau de controlabilitate), $C[rxn]$ - matricea de ieşire (sau de observabilitate), $D[rxq]$ – matricea de interconexiune (pentru sisteme fizic realizabile $D=0$).

Pentru ambele cazuri, vectorul de stare are aceeaşi formă:

$$x(t)=[x_1(t), x_2(t), \ldots, x_n(t)]^T$$

Rezultă deci că problema determinării MM-ISI se reduce în fapt la problema determinării parametrilor {A, B, C, D}, sau pentru sisteme fizic realizabile, a parametrilor {A, B, C }.

Algoritmul de determinare a unui **MM-ISI** este următorul:

1. Se scriu ecuaţiile corespunzătoare legilor fizicii care guvernează dinamica procesului respectiv
2. Se definesc variabilele terminale şi respectiv variabilele de stare
3. Ecuaţiile determinate la punctul 1, se rearanjează şi se aduc la forma standard a MM-ISI dată de relaţiile (4.5), ceea ce permite determinarea parametrilor {A,B,C,D} care definesc complet modelul de stare.

Observaţie:
Există diverse moduri de alegere a vectorului de stare ***x(t)*** pentru un acelaşi sistem – între aceşti vectori de stare existând relaţii de izomorfism. Dacă mărimile de stare sunt astfel alese încât vectorul de stare să aibă un număr minim de componente – se spune că sistemul este adus la ***forma canonică minimală*** – sau ***forma canonică***.

Pentru descrierea comportării dinamice a sistemelor în spaţiul stărilor, în cazul sistemelor discrete, MM-ISI are următoarea formă:

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t) \quad ; \quad x(0_+) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \qquad \text{MM-ISI, (caz discret)}$$

t – timp discret normalizat (4.6)

unde observaţiile referitoare la dimensiunile matricilor/vectorilor sunt aceleaşi ca şi cazul continuu.

c. Caracterizarea sistemelor în domeniul operaţional. Funcţia de transfer

Funcţia de transfer (f.d.t.) reprezintă un model matematic intrare-ieşire în domeniul operaţional.

Instrumentul de lucru în cadrul calculului operaţional îl reprezintă transformarea (operaţională) Laplace.

Transformarea Laplace *reprezintă o aplicaţie liniară de la mulţimea* ***funcţiilor original*** *(din domeniul timp) la mulţimea funcţiilor complexe de variabilă complexă "s", care asociază fiecărei funcţii original, imaginea sa Laplace – numită* ***funcţie imagine.***

Trecerea în domeniul complex, prin utilizarea transformatei Laplace, conduce la simplificări semnificative prin algebrizarea tuturor calculelor – ecuaţiile diferenţiale transformându-se în ecuaţii algebrice.

Convenție: Pentru funcțiile imagine se vor utiliza majusculele literelor utilizate în notarea funcției original, astfel:

$$\boldsymbol{L}u(t) = U(s),$$

unde:

$\boldsymbol{L}$ – este operatorul Laplace

$U(s)$ este imaginea Laplace a funcției original $u(t)$

s – variabilă complexă: $s=\sigma+j\omega$, $\sigma=Re[s]$, $\omega=Im[s]$

Calculele care implică studiul unui sistem de reglare automata (SRA) impun cunoașterea principalelor transformate Laplace (Anexa 2).

Transformata Laplace a derivatei de ordin „n" a unei funcții f(t) are expresia:

$$\boldsymbol{L}[f^{(n)}(t)] = s^n F(s) - \underbrace{s^{n-1} f(0_+) - s^{n-2} f'(0_+) - \ldots - s f^{(n-2)}(0_+) - f^{(n-1)}(0_+)}_{p.c.i.} \qquad (4.7)$$

unde *p.c.i.* – polinom al condițiilor inițiale.

Observație: ***Condițiile inițiale*** *(CI) pentru un sistem – reprezintă ansamblul valorilor mărimilor caracteristice ale sistemului la momentul de timp considerat inițial* $(t=0_+)$.

Considerând ecuația diferențială (3.1), care reprezintă forma generală a MM-II în domeniul timp, pentru un sistem liniar, invariant, monovariabil (SISO) și prin aplicarea transformatei Laplace (termen cu termen), în cazul condițiilor inițiale nule (CI=0), (deci p.c.i.=0, în (4.7)), se obține (datorită liniarității operatorului Laplace):

$$a_n s^n Y(s) + \ldots + a_1 s Y(s) + a_0 Y(s) = b_m s^m U(s) + \ldots + b_1 U(s) + b_0 U(s)$$

unde:

$$\boldsymbol{L}u(t) = U(s),\ \boldsymbol{L}y(t) = Y(s)$$

sau:

$$(a_n s^n + \ldots + a_1 s + a_0) Y(s) = (b_m s^m + \ldots + b_1 s + b_0) U(s) \qquad (4.8)$$

Funcția de transfer (f.d.t.) *a unui sistem* ***liniar, continuu*** *și* ***monovariabil,*** *notată cu* ***H(s)****, se definește ca raportul dintre imaginea Laplace a mărimii de ieșire* $Y(s)$ *și imaginea Laplace a mărimii de intrare* $U(s)$*, sistemul fiind considerat în condiții inițiale nule (CI=0).*

Deci: ținând cont de relația (4.8):

$$H(s) \stackrel{\Delta}{=} \left.\frac{Y(s)}{U(s)}\right|_{CI=0} = \frac{b_m s^m + \cdots + b_1 s + b_0}{a_n s^n + \cdots + a_1 s + a_0} = \frac{\sum_{j=0}^{m} b_j s^j}{\sum_{i=0}^{n} a_i s^i} = \frac{B(s)}{A(s)} \qquad (4.9)$$

Rădăcinile numărătorului f.d.t. reprezintă ***zerourile sistemului*** şi se obţin ca soluţii ale ecuaţiei $B(s) = \sum_{j=0}^{m} b_j s^j = 0$, şi se notează cu z_j.

Polinomul de la numitorul funcţiei de transfer – deci polinomul $A(s) = \sum_{i=0}^{n} a_i s^i$ – se numeşte ***polinom caracteristic*** al sistemului, iar $A(s) = 0$ reprezintă ***ecuaţia caracteristică*** a sistemului.

Rădăcinile ecuaţiei caracteristice se numesc ***polii sistemului***, sau ***valorile proprii*** ale sistemului (obţinându-se ca soluţii ale ecuaţiei $A(s)$=0), şi se notează cu p_i.

Ordinul "**n**" al polinomului caracteristic defineşte ***ordinul sistemului***. Pentru *sisteme fizic realizabile n>m.*

Rezultă deci că, funcţia de transfer (f.d.t.) se constituie de cele mai multe ori ca o expresie algebrică în domeniul operaţional, a unui raport de două polinoame în variabila complexă "s" (funcţie raţională).

Atât polii cât şi zerourile sistemului au un rol deosebit de important în analiza şi sinteza sistemului respectiv.

Cunoaşterea funcţiei de transfer a unui sistem este extrem de importantă în problemele de analiză şi sinteză a sistemelor automate, pe baza ei putându-se studia cu exactitate evoluţia în timp a sistemului, performanţele acestuia în regim staţionar şi dinamic, stabilitatea sistemului, acordarea regulatoarelor, optimizarea funcţionării sistemului în funcţie de anumite deziderate impuse, etc.

Funcţia de transfer reprezintă totodată una dintre modalităţile uzuale de a exprima informaţia despre dinamica sistemului în interiorul unei scheme bloc (Fig.4.2):

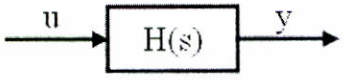

Figura 4.2. Schema bloc a unui sistem având f.d.t. H(s)

Funcţia de transfer, care reprezintă o relaţie doar între mărimile de intrare-ieşire, poate fi obţinută şi pornind de la forma generală a MM-ISI (4.6) pe baza următoarei relaţii de calcul:

$$\boxed{H(s) = \frac{Y(s)}{U(s)} = C^T [sI - A]^{-1} B + D} \qquad (4.10)$$

Remarcă: Calculul inversei unei matrici se face cu relaţia cunoscută $[\cdot]^{-1} = \dfrac{\text{adjunctă}\,[\cdot]}{\text{determinant}\,[\cdot]}$

Calculul adj[·] implică operația de transpunere $[\cdot]^T$, iar în matricea transpusă se înlocuiesc elementele ei cu complementul algebric corespunzător: $\Delta_{ij} = (-1)^{i+j} M_{ij}$, unde M_{ij} reprezintă minorul respectiv.

În consecință, relația efectivă de calcul a f.d.t. plecând de la MM-ISI devine:

$$\boxed{H(s) = C^T \frac{\text{adj}[sI - A]}{\det[sI - A]} \cdot B + D} \tag{4.11}$$

Se observă din relația (4.11) că *polinomul caracteristic al sistemului* (numitorul lui (4.11)) se obține ca:

$$\boxed{A(s) = \det[SI - A]} \tag{4.12}$$

Observație: Indiferent care model se utilizează pentru calculul f.d.t., MM-II relația (4.1) sau MM-ISI relația (4.5), trebuie să se ajungă la o aceeași expresie pentru *f.d.t.*

d. Matricea de transfer.

În cazul sistemelor multivariabile (MIMO) locul f.d.t. este luat de ***matricea de transfer***.

Astfel, considerând un sistem cu q intrări și r ieșiri, acesta trebuie înțeles în sensul figurii 4.3.

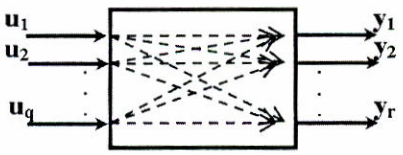

Figura 4.3. Schema bloc a unui sistem MIMO

sau:

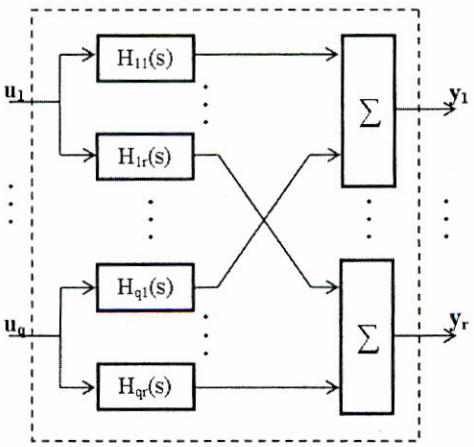

Figura 4.4. Sistem MIMO

Sistemul MIMO din figura 4.3 trebuie văzut în sensul figurii 4.4, deci existând qxr căi de transfer între cele q intrări şi cele r ieşiri, caracterizate fiecare de către o funcţie de transfer. Deci există qxr f.d.t. Intrarea şi ieşirea (în operaţional) sunt vectori de forma:

$$U(s)=[U_1(s) \ldots U_q(s)]^T \quad ; \quad [qx1]$$
$$Y(s)=[Y_1(s) \ldots Y_r(s)]^T \quad ; \quad [rx1]$$

Matricea de transfer se defineşte ca:

$$H(s)=\begin{bmatrix} H_{11}(s) & H_{12}(s) & \ldots & H_{1r}(s) \\ H_{21}(s) & H_{22}(s) & \ldots & H_{2r}(s) \\ \ldots\ldots & \ldots\ldots & \ldots & \ldots\ldots \\ H_{q1}(s) & H_{q2}(s) & \ldots & H_{qr}(s) \end{bmatrix}; [qxr] \tag{4.13}$$

unde: $H_{ij}(s)=\left.\dfrac{Y_j(s)}{U_i(s)}\right|_{\substack{i=\overline{1,q} \\ j=\overline{1,r}}}$ reprezintă *f.d.t.* ale canalelor $u_i \rightarrow y_j$

Pentru cazul MIMO, matricea de transfer poate fi calculată din MM-ISI cu relaţia (4.10), unde (A,B,C,D) – sunt matrici de dimensiuni corespunzătoare.

Expresia funcţiei de transfer poate fi rescrisă funcţie de rădăcinile numărătorului (zerouri) respectiv a numitorului (poli), obţinându-se reprezentarea poli-zerouri sau sub forma unei sume de fracţii simple obţinând reprezentarea poli-reziduuri.

În practică, pentru sistemul în studiu se poate dispune la un moment dat de un anumit tip de model matematic, dar rezolvarea problemei concrete de analiză a funcţionării lui sau de sinteză a sistemului de conducere automată necesită utilizarea unui alt tip de model.

Astfel de exemplu, dacă pentru proiectarea sistemului de conducere sunt utilizate tehnicile "clasice", modelul necesar este funcţia de transfer. Aplicarea rezultatelor moderne ale teoriei conducerii poate solicita însă modelul de stare, sau se dispune pentru sistem de un model matematic continuu dar se doreşte proiectarea unui sistem de reglare discret.

În aceste situaţii transformările de modele matematice (trecerea de la un model matematic ce descrie un sistem la un alt model matematic ce descrie într-o altă formă acelaşi sistem) devin un imperativ.
În continuare se prezintă modul în care pot fi realizate aceste implementări şi transformări de modele în MATLAB utilizând comenzile din biblioteca Control System Toolbox; consideraţiile fiind valabile atât pentru sisteme în timp continuu cât şi pentru sisteme în timp discret.

e. Reprezentarea poli-zerouri

Funcția de transfer a unui sistem liniar, invariant în raport cu timpul, poate fi reprezentată prin evidențierea rădăcinilor numărătorului (zerourile), respectiv ai numitorului (polii).

După cum s-a menționat deja expresia funcției de transfer pentru un sistem liniar de ordin n este dată de relația (4.9). În cazul în care f.d.t. are zero-uri și poli reali, complex conjugați și respectiv în origine, f.d.t. poate fi rescrisă sub forma generală:

$$H(s) = K \frac{\prod_{j=1}^{m_a}(T_j s + 1)\prod_{q=1}^{m_b}(T_q^2 s^2 + 2\xi_q T_q s + 1)}{s^{\alpha}\prod_{i=1}^{n_a}(T_i s + 1)\prod_{l=1}^{n_b}(T_l^2 s^2 + 2\xi_l T_l s + 1)} \tag{4.14}$$

în care:

- K - coeficientul de transfer al sistemului
- $\alpha>0$ evidențiază prezența polilor în origine
- $\alpha<0$ evidențiază prezența zerourilor în origine
- polinoamele de ordin 1 generează polii, respectiv zerourile reale, T_i, $i = \overline{1, n_a}$, T_j, $j = \overline{1, m_a}$ sunt constantele de timp ale sistemelor de ordin 1 (de temporizare, respectiv de anticipare)
- polinoamele de ordin 2 generează polii, respectiv zerourile complex conjugate $T_l = \frac{1}{\omega_{0l}}; l = \overline{1, n_b}$, $T_q = \frac{1}{\omega_{0g}}; q = \overline{1, m_b}$ reprezintă constantele de timp ale sistemelor de ordin 2 (de temporizare, respectiv de anticipare) ω_{0g} și ω_{0l} – pulsațiile naturale (proprii) ale sistemelor ξ_l, $l = \overline{1, n_b}$, $\xi_q, q = \overline{1, m_b}$ – reprezintă factori de amortizare

  $m_a+m_b=m$

  $\alpha+n_a+n_b=n$

  F.d.t. (4.14) poate fi rescrisă funcție de ω_0 (ținând cont de relația $T = \frac{1}{\omega_0}$).

Factorii din componența formei generale (4.14) pot fi interpretați ca funcții de transfer ale unor subsisteme (elemente) tipizate.

Dacă se cunosc caracteristicile logaritmice de frecvență (diagramele Bode) ale subsistemelor tipizate, se pot construi cu ușurință caracteristicile logaritmice de frecvență pentru orice sistem (cu o f.d.t. de forma (4.14)), prin însumarea grafică a caracteristicilor subsistemelor componente (considerarea unităților logaritmice are ca efect transformarea produselor din cadrul f.d.t. în sume).

f. Reprezentarea poli-reziduuri

O altă reprezentare a funcției de transfer (4.9) a unui sistem se poate obține rescriind termenii sub forma unor fracții simple:

$$H(s)=\frac{r_1}{s-p_1}+\frac{r_2}{s-p_2}+\cdots+\frac{r_{n_x}}{s-p_{n_x}}+k(s) \tag{4.15}$$

unde s-a presupus că numitorul nu are rădăcini multiple.

Această formă poate fi definită în termenii unui vector coloană *pol* care conține polii sistemului, un vector coloană *res* care conține reziduurile și respectiv a unui vector linie *k* care conține coeficienții polinomului definind partea improprie a sistemului (acest vector nu trebuie definit dacă numărul zerourilor este mai mic sau egal cu numărul polilor).

În cazul funcțiilor de transfer care au poli multipli, în relația (4.15) vor apărea termeni de forma:

$$\frac{r_i}{s-p_i}+\frac{r_{i+1}}{(s-p_i)^2}+\cdots+\frac{r_{i+m-1}}{(s-p_i)^m} \tag{4.16}$$

unde m reprezintă ordinul de multiplicitate a polului p_i.

Reprezentarea sistemului va fi identică, cu excepția vectorului *pol* unde polul p_i va apărea de m ori.

4.2. Reprezentări de modele matematice în MATLAB

4.2.1. Modele în spațiul stărilor

Exemplu:
Se consideră un sistem format din trei circuite integratoare, conectate în serie, conform figura 4.5. Considerând notațiile din figura menționată, se poate scrie direct, conform formei standard a modelului de stare:

$$\dot{x}_1 = 0\cdot x_1 + 0\cdot x_2 + 0\cdot x_3 + 1\cdot u$$

$$\dot{x}_2 = 1\cdot x_1 + 0\cdot x_2 + 0\cdot x_3 + 0\cdot u$$

$$\dot{x}_3 = 0\cdot x_1 + 1\cdot x_2 + 0\cdot x_3 + 0\cdot u$$

$$y = 0\cdot x_1 + 0\cdot x_2 + 1\cdot x_3$$

sau, în forma vectorial matricială:

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases} \tag{4.17}$$

unde: $A = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$, $B = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$, $C = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}$, $D = \begin{pmatrix} 0 \end{pmatrix}$, $x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$

Implementarea în MATLAB, a acestui tip de model, presupune definirea a 4 matrici, uzual notate cu *A, B, C, D*. Acestea vor fi interpretate drept model de stare doar de anumite comenzi specifice, cum ar fi, spre exemplu, cele de simulare (*lsim, step, initial* etc.).

Pentru a defini cele 4 matrici, conform ecuațiilor (4.17), se utilizează instrucțiunile următoare :

```
>> A = [0 0 0 ; 1 0  0; 0 1 0];
>> B = [1; 0; 0];
>> C = [0 0 1];
>> D = 0 ;
```

Notă: În fapt, instrucțiunile reprezintă simple atribuiri de valori unor matrici, notate în acest caz cu A,B,C,D – notații consacrate formei standard MM-ISI.
În contextul unei simulări, utilizând de exemplu instrucțiunea *lsim*, cele patru matrici având dimensiunile corespunzătoare au semnificația matricilor unui model de tip MM-ISI.
Nota este valabilă în continuare şi la introducerea altor tipuri de modele.

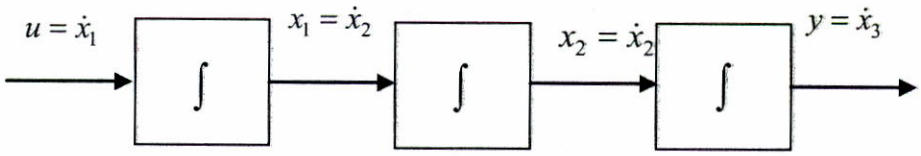

Figura 4.5. Schema a trei elemente integratoare conectate în serie

Exemplu:
Se consideră sistemul mecanic în translație masa-resort-amortizor, prezentat în figura 4.6.
Sistemul mecanic este alcătuit dintr-un corp de masă *m*, un resort de constantă elastică *k* şi un amortizor având coeficientul de frecare vâscoasă *r*.
Se doreşte determinarea modelului matematic intrare-stare-ieşire şi realizarea unui program MATLAB de citire a parametrilor *m, r, k,* şi respectiv de definire a matricilor *A, B, C, D.*

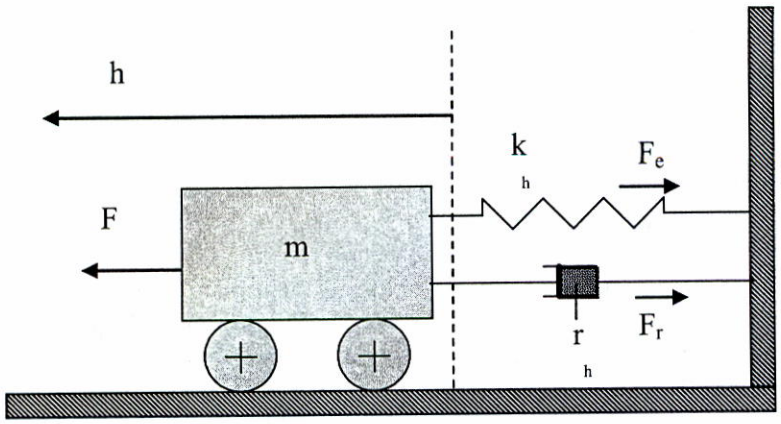

Figura 4.6. Sistem masă-resort-amortizor

Corpul de masă m se găseşte sub acţiunea unei forţe externe F. În aceste condiţii, asupra masei m acţionează următoarele forte:

$F_e = k \cdot h$ forţa elastică (proporţională cu deplasarea);

$F_r = r \cdot v = r \cdot \dot{h}$ forţa vâscoasă (proporţională cu viteza);

$F_a = m \cdot a = m \cdot \ddot{h}$ forţa de acceleraţie (proporţională cu acceleraţia);

F forţa externă (mărimea de intrare a sistemului).

Ecuaţia de echilibru a forţelor pentru sistemul mecanic considerat este:

$$F = F_a + F_e + F_r = m \cdot \ddot{h} + r \cdot \dot{h} + k \cdot h \tag{4.18}$$

unde h este deplasarea căruciorului.

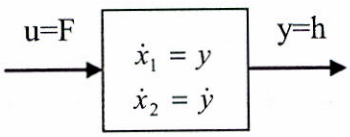

Figura 4.7. Sistemul masă-resort-amortizor

Astfel, considerând mărimile de intrare şi ieşire, respectiv variabilele de stare conform schemei bloc din figura 4.7, deci:

$$u = [F],\ y = [h] \text{ şi respectiv } x = \begin{bmatrix} h \\ \dot{h} \end{bmatrix} \tag{4.19}$$

rezultă:

$$\dot{x}_1 = \dot{h} = x_2$$

$$\dot{x}_2 = \ddot{h}$$

Ţinând cont de (4.18) se poate scrie:

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = \frac{1}{m}(-kx_1 - rx_2) + \frac{1}{m}u$$
$$y = x_1$$

Rescriind în forma standard vectorial matricială:

$$\begin{cases} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{r}{m} \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u \\ y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \end{bmatrix} + 0\,u \end{cases} \tag{4.20}$$

rezultă matricile A,B,C,D: $A = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{r}{m} \end{bmatrix}$ $B = \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix}$

$$C = \begin{bmatrix} 1 & 0 \end{bmatrix} \quad D = 0$$

Secvenţa MATLAB care realizează citirea parametrilor m,r,k, şi definirea celor 4 matrici este:

```
m=input('masa');
r=input('coef. amortizare');
k=input('coef. elastic');
A = [0 1 ; -k/m -r/m]
B = [0; 1/m]
C = [1 0]
D = 0
```

♦

4.2.2. Funcţia de transfer

Implementarea în MATLAB a unei funcţii de transfer este simplu de realizat. Astfel, este suficient să se asigneze doi vectori formaţi de coeficienţii polinomului de la numărătorul, respectiv numitorul funcţiei de transfer, sortaţi în ordinea descrescătoare a puterilor variabilei s.

Notă: în fapt, cei doi vectori creaţi prin aceste simple atribuiri de valori, au semnificaţia unui model de tip funcţie de transfer doar în funcţie de context, prin asocierea lor cu anumite instrucţiuni (de simulare, de transformări de modele etc.).

De exemplu, în cazul unei simulări utilizând instrucţiunea lsim, cei doi vectori, având dimensiunile corespunzătoare, au semnificaţia unui model de tip funcţie de transfer.

Exemplu:

Dacă se doreşte reprezentarea funcţiei de transfer (4.21) în vederea unei simulări:

$$H(s) = \frac{s+2}{3s^2 + 4s + 5} \tag{4.21}$$

trebuie iniţial asignate valorile corespunzătoare către 2 vectori, spre exemplu *num* şi *den*:

```
>> num = [1 2];
>> den = [3 4 5];
```

după care se utilizează o instrucţiune *lsim* avand *num* şi *den* ca parametri (vezi paragraful 5.5)

♦

Exemplu:

Se consideră sistemul prezentat în figura 4.6, şi se doreşte calcularea funcţiei de transfer aferente.

Pornind de la ecuaţia de echilibru a forţelor care acţionează în sistem, adică relaţia (4.18) şi cu alegerea (4.19), se poate scrie:

$$m \cdot \ddot{y} + r \cdot \dot{y} + k \cdot y = u \tag{4.22}$$

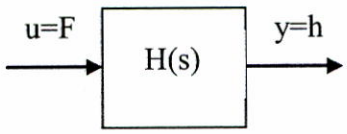

Figura 4.8. Schema bloc a sistemului masă-resort-amortizor din figura 4.6

Ecuaţia (4.22) reprezintă MM-II asociat sistemului. Funcţia de transfer se poate obţine aplicând direct transformata Laplace MM-II dat de relaţia 4.22, rezultând:

$$H(s) = \frac{y(s)}{u(s)} = \frac{1}{ms^2 + rs + k} \tag{4.23}$$

Aceeaşi funcţie de transfer se poate obţine şi pornind de la MM-ISI (cunoscând matricile *A, B, C, D*) aplicând relaţia:

$$y(s) = C^T[şi - A]^{-1}Bu(s) + Du(s) \tag{4.24}$$

Programul MATLAB care realizează citirea celor trei parametri ai sistemului (*m, r, k*), şi calculează cei doi vectori care compun funcţia de transfer este următorul:

```
m=input('masa');
r=input('coef. amortizare');
k=input('coef. elastic');
num = 1
den= [m r k]
%lsim avand num şi den ca parametri
```

Pentru sistemele de tipul SIMO (o singură intrare, mai multe ieşiri) numărătorul lui *H(s)* este un vector coloană, având ca elemente polinoame, fiecare polinom corespunzând unei singure ieşiri. În acest caz *num* este o matrice având numărul de linii egal cu numărul ieşirilor din sistem, iar numărul de coloane trebuie să fie egal cu maximul dintre ordinele acestor polinoame plus 1.

Exemplu:
Se doreşte implementarea în MATLAB a unui sistem cu o intrare şi două ieşiri, având următoarea matrice de transfer:

$$H(s)=\frac{\begin{pmatrix} 11s+12 \\ 21s^3+22s^2+23 \end{pmatrix}}{31s^3+32s+33} \tag{4.25}$$

Matricea de transfer poate fi descompusă în două f.d.t. asociate celor două ieşiri, Y_1 şi respectiv Y_2 :

$$H_1(s)=\frac{11s+12}{31s^3+32s+33}$$

şi respectiv

$$H_2(s)=\frac{21s^3+22s^2+23}{31s^3+32s+33}$$

În acest caz, se poate defini în MATLAB matricea de transfer asociată sistemului prin definirea a două matrici (de exemplu notate cu *num* şi *den)* după cum urmează:

```
>> num = [0 0 11 12;21 22 0 23];
>> den = [31 0 32 33];
```

Liniile matricelor sunt formate din coeficienţii polinoamelor în ordinea descrescătoare a puterilor variabilei complexe *s*. În plus trebuie inserate zerouri pentru realizarea consistenţei vectorului (în cazul în care un termen al puterii lui s nu apare în polinom).

Notă: Trebuie reţinut că în MATLAB nu se pot implementa direct sisteme MI (multi-input), *den* neputând fi decât un vector linie. Acest lucru nu limitează plaja de aplicabilitate a MATLAB-ului, un sistem de tip MIMO putând fi descompus în mai multe subsisteme SIMO.

4.2.3. Modele de tipul poli-zerouri

Funcția de transfer poate fi scrisă punând în evidență rădăcinile numărătorului (zerourile sistemului), respectiv pe cele ale numitorului (polii sistemului).

În MATLAB se folosesc vectori linie pentru a indica coeficienții unui polinom, respectiv vectori coloană pentru a indica rădăcinile aferente.

Exemplu:
Reprezentarea în MATLAB a funcției de transfer:

$$H(s)=2\frac{(s-3)(s+4)}{(s+5)(s-6)} \tag{4.26}$$

în vederea utilizării ulterioare a acestui model într-o simulare (utilizând de exemplu instrucțiunea *lsim*) poate fi obținută cu ajutorul a trei atribuiri de genul:

```
>> k=2;
>> zer=[3;-4];
>> pol=[-5; 6];
```

Notă: La asignarea vectorilor *zer* și *pol* trebuie avute în vedere semnele corespunzătoare rădăcinilor.

Exemplu:
Se consideră un sistem având funcția de transfer dată de relația (4.27):

$$H(s)=\frac{2}{(s+3-4j)(s+3+4j)} \tag{4.27}$$

Notă: Pentru o funcție de transfer care nu are zerouri, vectorul corespunzător numărătorului *zer* nu poate rămâne nedefinit, fiind necesară o declarație de forma:

```
zer=inf;
```

Astfel, pentru a obține reprezentarea funcției de transfer considerată în forma poli-zerouri, secvența de atribuiri necesare este următoarea:

```
>> k=1;
>> pol=[-3+4j;-3-4j];
>> zer=inf;
```

După cum s-a menționat deja, în cazul sistemelor SIMO numărătorul expresiei lui *H(s)* este un vector de polinoame. Este important de remarcat faptul că rădăcinile polinomului numărător, corespunzător celei de a *i*-a ieșire, trebuie introduse în a *i*-a coloană a matricii *zer*.

Exemplu:
Se consideră un sistem cu o intrare şi trei ieşiri, a cărui matrice de transfer este:

$$H(s)=\frac{\begin{pmatrix}1\\ 2s\\ 3(s+4)(s+5)\end{pmatrix}}{(s+6)\ (s+7)^2} \tag{4.28}$$

Reprezentarea poli-zerouri a acestui sistem, se poate realiza prin efectuarea următoarelor asignări:

```
>> zer =[inf  0 -4;inf inf -5];
>> pol =[-6; -7; -7];
>> k =[1; 2; 3];
```

rămânând valabile observaţia referitoare la elementul *inf.*

4.2.4. Modele de tip poli-reziduuri

Exemplu:
Se consideră un sistem având următoarea funcţie de transfer:

$$H(s)=\frac{s^2+2s+4}{s(s+3)^2(s^2-4s+5)} \tag{4.29}$$

Descompunând în fracţii simple se poate obţine forma:

$$H(s)=\frac{0.0889}{s}-\frac{0.0131}{s+3}-\frac{0.0897}{(s+3)^2}-\frac{0.0379+0.1009j}{s-1-2j}-\frac{0.0379-0.1009j}{s-1+2j} \tag{4.30}$$

Modelul poli-reziduuri corespunzător poate fi obţinut prin efectuarea următoarelor asignări:

```
>> res=[ -0.0131;  -0.0897;  -0.0379 - 0.1009i;  -0.0379 +
0.1009i; 0.0889];
>> pol=[ -3; -3;2+1j; 2-1j;0];
>> k=[];
```

Acelaşi model poate fi obţinut utilizând comanda ***residue:***

```
>> [res,pol,k]=residue([1 2 4], conv(conv([1 0],conv([1 3],
   [ 1 3])),[1 -4 5]))
```

care returnează direct vectorii ce conţin reziduurile, polii, respectiv partea improprie a sistemului folosind polinoamele numărător şi numitor a funcţiei de transfer considerate ca argumente ale comenzii *residue*, adică:

```
res =

  -0.0131
  -0.0897
  -0.0379 - 0.1009i
  -0.0379 + 0.1009i
   0.0889

pol =

  -3.0000
  -3.0000
   2.0000 + 1.0000i
   2.0000 - 1.0000i
        0

k =

     [ ]
```

Nota: Comanda *conv* realizează înmulţirea a doua polinoame.

4.3. Conversii de modele matematice în MATLAB

Odată ce un sistem a fost reprezentat printr-unul din modelele enumerate anterior, se poate realiza direct conversia într-un alt tip de model, utilizând comenzi MATLAB dedicate din ***Control System Toolbox***.

Comenzile cele mai utilizate pentru realizarea conversiei dintr-un tip de model matematic în altul, pentru acelaşi sistem liniar invariant, sunt:

- *ss2ss* – transformare de la un MM-ISI la un alt MM-ISI
- *ss2tf* – transformare din MM-ISI în funcţie de transfer
- *ss2zp* – transformare din MM-ISI în reprezentare poli-zerouri
- *tf2ss* – transformare din funcţie de transfer în MM-ISI
- *tf2zp* – transformare din funcţie de transfer în reprezentare poli-zerouri
- *zp2ss* – transformare din reprezentare poli-zerouri în MM-ISI
- *zp2tf* – transformare din reprezentare poli-zerouri în funcţie de transfer
- *canon** – aducere la forma canonică

unde (*) indică faptul că această comandă nu poate fi utilizată pentru sisteme în timp discret.

4.3.1. Conversia din spațiul stărilor

❑ *ss2ss, cannon*

Fiind dată o reprezentare de tipul intrare-stare–ieşire, pentru o anumită alegere a variabilelor de stare, se poate obține uşor o altă reprezentare în acelaşi spațiu, pentru o altă alegere a variabilelor de stare, folosind o **transformare de stare**.

Exemplu:
Se consideră sistemul mecanic prezentat în figura 4.6. Mărimile caracteristice ale sistemului se aleg conform schemei bloc din figura 4.9.

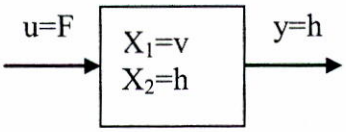

Figura 4.9. Schema bloc a sistemului masă-resort-amortizor

Adică:
- forța de tracțiune F – mărimea de intrare
- deplasarea h – mărimea de ieşire
- deplasarea h şi respectiv viteza v căruciorului – variabilele de stare.

O posibilă reprezentare intrare-stare-ieşire pentru acest sistem este:

$$\begin{cases} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = A \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + B\,u \\ y = C \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + D\,u \end{cases} \quad \text{unde} \quad A = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{r}{m} \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 \end{bmatrix} \quad D = 0 \tag{4.31}$$

cu vectorul de stare $X = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$

Dacă se doreşte obținerea unei reprezentări în alt spațiu al stărilor, alegând ca şi variabile de stare viteza, şi respectiv suma dintre viteza şi dublul deplasării, se poate utiliza o transformare de stare de forma:

$$\begin{pmatrix} z1 \\ z2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \tag{4.32}$$

Considerând, pentru simplificarea calculelor, $k=r=m=1$, această nouă reprezentare se poate obține în MATLAB folosind următoarele instrucțiuni:

```
>> A=[0  1; -1  -1];
>> B= [0; 1];
>> C=[1  0];
>> D=[0];
>> T=[0  1;2  1];
>> [At, Bt, Ct, Dt] = ss2ss (A, B ,C ,D ,T)
```

unde ultima comandă returnează matricea modelului transformat, adică:

```
At =

   -0.5000   -0.5000
    1.5000   -0.5000

Bt =

     1
     1

Ct =

   -0.5000    0.5000

Dt =

     0
```

Rezulta un MM-ISI de forma:

$$\begin{cases} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = A_t \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + B_t \; u \\ y = C_t \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + D_t \, u \end{cases} \quad \text{unde} \quad A_t = \begin{bmatrix} -0.5 & 0.5 \\ 1.5 & -0.5 \end{bmatrix} \; B_t = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad (4.33)$$

$$C_t = [-0.5 \quad 0.5] \; D_t = 0$$

cu vectorul de stare $Z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$

În general, comanda ***ss2ss*** poate să realizeze o transformare de stare definita printr-o relație:

$$z=Tx$$

unde T este matricea de transformare, rezultând pentru noul model MM-ISI următoarele matrici:

$$A_t = TAT^{-1}, \quad B_t = TB, \quad C_t = CT^{-1}, \quad D_t = D$$

Comanda ***canon*** poate fi considerată un caz special al comenzii *ss2ss*. Această comandă permite obținerea a două reprezentări canonice de tipul intrare-stare-ieşire plecând de la un model intrare-stare-ieşire dat.

Prima formă de reprezentare este numită ***modală*** şi se caracterizează prin aceea că valorile proprii ale sistemului apar în matricea sistemului pe diagonala principală.

A doua formă, numită şi ***companion***, se caracterizează prin aceea că termenii din polinomul caracteristic apar în ultima coloană a matricii sistemului.

Tipul modelului returnat se poate specifica prin utilizarea unui argument de tip şir de caractere la apelul comenzii, de forma: 'modal' sau respectiv 'companion'. Dacă acest argument lipseşte, se returnează modelul matematic în forma modală.

Notă: Se recomandă să se lucreze pe cât posibil cu forma canonică modală.

Exemplu:
Considerând sistemul din figura 4.6., comanda:

```
>> [At ,Bt ,Ct ,Dt] = canon (A, B, C, D)
```

returnează matricele care compun modelul canonic de stare în forma modala:

```
At =

   -0.5000    0.8660
   -0.8660   -0.5000

Bt =

         0
    1.6330

Ct =

    0.7071         0

Dt =

      0
```

De remarcat că matricea A_t are pe diagonala principală partea reală a valorilor proprii. În cazul în care toate valorile proprii sunt reale matricea A_t este diagonală.

Valorile proprii ale matricii A se pot calcula cu ajutorul comenzii:

```
>> eigs(A)
```

rezultatul fiind:

```
ans =

  -0.5000 - 0.8660i
  -0.5000 + 0.8660i
```

Exemplu:

În continuare se va prezenta cazul în care valorile proprii ale unui sistem nu sunt toate reale. Se consideră un sistem ale cărui valori proprii reale sunt –3 şi –6, iar cele complexe sunt -9±15j, având o reprezentare în spaţiul stărilor de forma:

$$\begin{cases} \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} 97.5 & -48 & -63 & -52 \\ 83 & -43 & -56 & -43.5 \\ 67 & -31 & -47 & -36 \\ 67.5 & -36 & -27 & -34.5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} + \begin{pmatrix} 1 \\ 2 \\ 3 \\ 0 \end{pmatrix} \cdot U \\ Y = \begin{pmatrix} 1 & 2 & 3 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} + 0 \cdot U \end{cases} \tag{4.34}$$

Secvenţa următoare de instrucţiuni returnează matricele modelului de stare în forma canonică modală

```
>> A=[97.5 -48 -63 -52.5;83.5 -43 -56 -43.5; 67 -31 -47 -36;
67.5 -36 -27 -34.5]
>> B=[1;2;3;0]
>> C=[1 2 3 4]
>> D=0
>> [At, Bt ,Ct ,Dt ] =canon (A, B, C, D)
```

Deci:

```
At =

   -9.0000   15.0000         0         0
  -15.0000   -9.0000         0         0
         0         0   -6.0000         0
         0         0         0   -3.0000

Bt =

   -1.3642
   17.7349
  -10.6014
    9.5263
```

```
Ct =

   -3.8850    0.9041   -2.9399   -4.0415

Dt =

     0
```

Prin utilizarea comenzii *canon* şi a folosirii argumentului *'companion'* în locul lui *'modal'* se obține reprezentarea de stare pentru care perechea de matrici (A_t, B_t) este adusă la forma canonică observabilă de tipul specificat:

$$A_t = \begin{pmatrix} 0 & 0 & 0 & \dots & -a_n \\ 1 & 0 & 0 & \dots & -a_{n-1} \\ 0 & 1 & 0 & \dots & \vdots \\ \vdots & \vdots & \vdots & \ddots & -a_2 \\ 0 & \cdots & \cdots & 1 & -a_1 \end{pmatrix}, B_t = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \end{pmatrix}^T \tag{4.35}$$

unde a_i sunt coeficienții polinomului caracteristic:

$$|sI - A| = s^n + a_1 s^{n-1} + a_2 s^{n-2} + \cdots + a^{n-1} s + a_n \tag{4.36}$$

În cazul sistemului dat de relația (4.34), forma canonică observabilă se obține cu ajutorul comenzii:

```
>> [At, Bt ,Ct ,Dt ] =canon (A, B, C, D,'companion' )
```

care conduce la următorul rezultat:

```
At =

  1.0e+003 *

   -0.0000    0.0000   -0.0000   -5.5080
    0.0010         0         -0.0000   -3.0780
    0.0000    0.0010   -0.0000   -0.4860
    0.0000    0.0000    0.0010   -0.0270

Bt =

     1
     0
     0
     0
```

```
Ct =

  1.0e+004 *

    0.0014   -0.1278    1.5974    9.0085

Dt =

     0
```

❑ ***ss2tf***

În multe situații din practică, cum ar fi analiza în domeniul frecvență, proiectarea regulatoarelor s.a.m.d. sunt necesare modele matematice de tipul funcție de transfer sau reprezentări poli-zerouri.

Obținerea unor astfel de modele, pornind de la o reprezentare de stare a sistemului, implică utilizarea comenzii ***ss2tf***, respectiv ***ss2zp***.

Exemplu:

Fie sistemul constituit din următorul circuit electric cu surse comandate de curent reprezentat în figura 4.10.

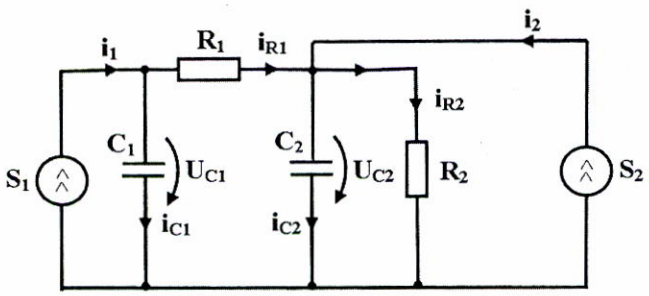

Figura 4.10. Circuit $C_1R_1C_2R_2$

Cele două surse ideale de curent S_1 şi S_2 injectează curenții i_1 şi i_2 în rețeaua electrică $C_1R_1C_2R_2$.

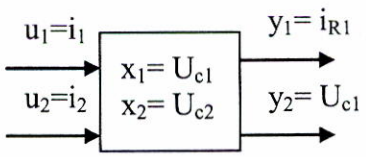

Figura 4.11. Schema bloc a circuitului $C_1R_1C_2R_2$

Circuitul considerat în figura 4.10 este un sistem multivariabil (MIMO) cu două intrări şi două ieşiri.

Se cere determinarea MM-ISI şi a matricii de transfer.

1) *Ecuațiile fizice ale sistemului sunt:*

$$\begin{cases} i_{C1} = i_1 - i_{R1} \\ i_{C2} = i_{R1} - i_{R2} + i_2 \\ i_{R1} = \dfrac{U_{C1} - U_{C2}}{R_1} \\ i_{R2} = \dfrac{U_{C2}}{R_2} \\ i_{C1} = \dfrac{dU_{C1}}{dt} C_1 \\ i_{C2} = \dfrac{dU_{C2}}{dt} C_2 \end{cases} \tag{4.37}$$

2) *Conform figurii 4.11 mărimile caracteristice ale sistemului sunt:*

- vectorul mărimilor de stare: $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} U_{C1} \\ U_{C2} \end{bmatrix}$
- vectorul mărimilor de intrare: $u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} i_1 \\ i_2 \end{bmatrix}$
- vectorul mărimilor de ieşire: $y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} i_{R1} \\ u_{C2} \end{bmatrix}$

3) *Ținând cont de paşii 1) şi respectiv 2), se pot scrie ecuațiile:*

$$\begin{cases} C_1 \dot{x}_1 = u_1 - \dfrac{1}{R_1}(x_1 - x_2) \\ C_2 \dot{x}_2 = \dfrac{1}{R_1}(x_1 - x_2) - \dfrac{x_2}{R_2} + u_2 \end{cases}$$

care rearanjate conduc la obținerea MM-ISI în formă detaliată:

$$\begin{aligned} \dot{x}_1 &= -\frac{1}{R_1 C_1} x_1 + \frac{1}{R_1 C_1} x_2 + \frac{1}{C_1} u_1 \\ \dot{x}_2 &= \frac{1}{R_2 C_2} x_1 - \frac{1}{C_2}\left(\frac{1}{R_1} + \frac{1}{R_2}\right) x_2 + \frac{1}{C_2} u_2 \\ y_1 &= \frac{1}{R_1} x_1 - \frac{1}{R_2} x_2 + 0\, u_2 \\ y_2 &= 0\, x_1 + 0\, x_2 + 0\, u_2 \end{aligned} \tag{4.38}$$

Prin rescriere în formă vectorial matriceală rezultă:

$$\overbrace{\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix}}^{\dot{x}} = \overbrace{\begin{bmatrix}-\frac{1}{R_1C_1} & \frac{1}{R_1C_1}\\ \frac{1}{R_1C_2} & -\frac{1}{C_2}\left(\frac{1}{R_1}+\frac{1}{R_2}\right)\end{bmatrix}}^{A}\overbrace{\begin{bmatrix}x_1\\ x_2\end{bmatrix}}^{x} + \overbrace{\begin{bmatrix}\frac{1}{C_1} & 0\\ 0 & \frac{1}{C_2}\end{bmatrix}}^{B}\overbrace{\begin{bmatrix}u_1\\ u_2\end{bmatrix}}^{u} \qquad (4.39)$$

$$\underbrace{\begin{bmatrix}y_1\\ y_2\end{bmatrix}}_{y} = \underbrace{\begin{bmatrix}\frac{1}{R_1} & -\frac{1}{R_1}\\ 0 & 1\end{bmatrix}}_{C}\underbrace{\begin{bmatrix}x_1\\ x_2\end{bmatrix}}_{x} + \underbrace{\begin{bmatrix}0 & 0\\ 0 & 0\end{bmatrix}}_{D}\underbrace{\begin{bmatrix}u_1\\ u_2\end{bmatrix}}_{u}$$

Matricea de transfer (de dimensiune 2x2) se obține utilizând relația de calcul 4.10:

$$H(s)=[C]\underbrace{\begin{bmatrix}s+\frac{1}{R_1C_1} & -\frac{1}{R_1C_1}\\ -\frac{1}{R_2C_2} & s+\frac{1}{C_2}\left(\frac{1}{R_1}+\frac{1}{R_2}\right)\end{bmatrix}^{-1}}_{[sI-A]}[B]=\begin{bmatrix}H_{11}(s) & H_{12}(s)\\ H_{22}(s) & H_{22}(s)\end{bmatrix} \qquad (4.40)$$

Alegând C_1=1mF, R_1=1KΩ şi respectiv C_2=2mF, R_2=2KΩ ca şi parametri ai circuitului considerat, matricele corespunzătoare modelului MM-ISI pot fi definite executând secvența:

```
>> c1=0.001;
>> r1=1000;
>> c2=0.002;
>> r2=2000;
>> A=[-1/(r1*c1) 1/(r1*c1); 1/(r1*c2) -(1/r1+1/r2)/c2]
>> B=[1/c1 0; 0 1/c2]
>> C=[1/r1 -1/r1; 0 1]
>> D=[0 0; 0 0]
```

având ca rezultat:

```
A =

   -1.0000    1.0000
    0.5000   -0.7500

B =

        1000           0
           0         500
```

```
C =

    0.0010   -0.0010
         0    1.0000

D =

     0     0
     0     0
```

Deci MM-ISI particularizat pentru valorile alese anterior ale parametrilor C_1, R_1, C_2, R_2 este:

$$\begin{cases} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -1 & 1 \\ 0.5 & -0.75 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1000 & 0 \\ 0 & 500 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \\ \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 0.0010 & -0.0010 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \end{cases} \tag{4.41}$$

Notă: Având în vedere că pentru calculul matricii de transfer, un sistem de tip MIMO trebuie descompus în mai multe subsisteme SIMO (câte unul pentru fiecare intrare) procedura implică considerarea pe rând a câte unei intrări presupunându-se toate celelalte intrări nule. Rezultă, astfel, pentru fiecare intrare o matrice de transfer de tip coloană.

Matricea de transfer a sistemului considerat având două intrări şi două ieşiri, se descompune pe rând în două matrici coloană de dimensiune 2x1, câte una corespunzătoare pentru fiecare intrare.

Acestea sunt furnizate de instrucţiunile:

```
>> [num1, den1]=ss2tf(A, B, C, D,1)
>> [num2, den2]=ss2tf(A, B, C, D,2)
```

care returnează matricele :

```
num1 =

         0    1.0000    0.2500
         0         0  500.0000

den1 =

    1.0000    1.7500    0.2500
```

şi respectiv:

```
num2 =

         0   -0.5000         0
         0  500.0000  500.0000
```

```
den2 =

    1.0000    1.7500    0.2500
```

Notă: Ultimul argument din instrucțiunile precedente este indicele intrării considerate, putând fi omis în cazul unui sistem cu o singură intrare. Matricea *num* are numărul de linii egal cu numărul ieşirilor sistemului (vector linie în cazul unui sistem cu o singură ieşire) şi are atâtea coloane câte elemente are vectorul *den*, corespunzător polinomului de la numitorul funcției de transfer.

Pentru exemplul considerat, matricea de transfer conține patru elemente, fiecare dintre ele reprezentând o funcție de transfer, corespunzătoare unui canal intrare-ieşire (vezi figura 3.4). Acestea pot fi afişate pe ecran cu ajutorul comenzilor:

```
>> printsys(num1(1,:),den1)
>> printsys(num1(2,:),den1)
>> printsys(num2(2,:),den2)
>> printsys(num2(1,:),den2)
```

Rezulta astfel funcțiile de transfer:

$$H_{11}(s) = \frac{s + 0.25}{s^2 + 1.75s + 0.25}$$

$$H_{12}(s) = \frac{500}{s^2 + 1.75s + 0.25}$$

$$H_{21}(s) = \frac{500s + 500}{s^2 + 1.75s + 0.25}$$

$$H_{22}(s) = \frac{-0.5s}{s^2 + 1.75s + 0.25}$$

▶

❑ ***ss2zp***

Transformarea dintr-o reprezentare de tip intrare-stare-ieşire a sistemului la o formă de tipul poli-zerouri poate fi realizată cu instrucțiunea ***ss2zp***.

Exemplu:
Comanda pentru determinarea modelului de tip poli-zerouri asociat sistemului din figura 4.10, în condițiile în care $u_2 = 0$ este:

```
>> [zer, pol, k] =ss2zp(A, B, C, D, 1)
```

rezultând:

```
zer =

   -0.2500         Inf

pol =

   -1.5931
   -0.1569

k =

    1.0000
  500.0000
```

►

Notă: în cazul în care funcția de transfer nu are zerouri, vectorul *zer* nu va fi vid, ci va conține elemente de tip *Inf*.

Exemplu:
Determinarea modelului de tip poli-zerouri pentru sistemul dat de relația (4.35) se realizează cu următoarea secvență de comenzi:

```
>> A=[97.5 -48 -63 -52.5;83.5 -43 -56 -43.5; 67 -31 -47 -36;
   67.5 -36 -27 -34.5]
>> B=[1;2;3;0]
>> C=[1 2 3 4]
>> D=0
>> [zer, pol, k] =ss2zp(A, B, C, D, 1)
```

care returnează următorii vectori:

```
zer =

  -5.8654 + 4.3621i
  -5.8654 - 4.3621i
  76.0521

pol =

  -9.0000 +15.0000i
  -9.0000 -15.0000i
  -6.0000
  -3.0000

k =

   14.0000
```

►

4.3.2. Conversia din matricea de transfer

❑ *tf2ss*

Conversia modelului unui sistem din forma matrice de transfer în MM-ISI este realizată de comanda ***tf2ss***.

Exemplu:
Fie sistemul modelat prin următoarea funcţie de transfer:

$$H_{11}(s) = \frac{s + 0.25}{s^2 + 1.75s + 0.25}$$

Pentru a obţine MM-ISI, se executa următoarea secvenţă de comenzi:

```
>> num=[500 0.25]
>> den=[1 1.75 0.25]
>> [A, B, C, D] = tf2ss(num,den)
```

care returnează matricile:

```
A =

   -1.7500   -0.2500
    1.0000         0

B =

     1
     0

C =

  500.0000    0.2500

D =

     0
```

▶

Notă: Se poate observa ca matricele A şi respectiv B furnizate de această comandă nu coincid însă cu matricele aferente MM-ISI (3.41) pentru $u_2=0$. Acest lucru se datorează faptului că transformarea *tf2ss* conduce pentru matricile A şi B la forme particulare de tipul:

$$A = \begin{pmatrix} -a_1 & -a_2 & \cdots & -a_{n-1} & -a_n \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix} \qquad (4.42)$$

$$B = (1 \quad 0 \quad 0 \quad \cdots \quad 0)^T$$

Notă: Este de remarcat faptul că aceasta comanda *tf2ss* utilizează numai sisteme având o singură intrare şi, în consecinţă, matricile B şi D vor fi întotdeauna vectori coloană.

❑ ***tf2zp***

Conversia dintr-o reprezentare de tipul funcţiei de transfer la reprezentarea poli-zerouri este realizată de instrucţiunea ***tf2zp***.

Exemplu:
Pentru a obţine modelul poli-zerouri corespunzător funcţiei de transfer a sistemului din exemplul anterior, trebuie rulată secvenţa de comenzi următoare:

```
>> num=[500 0.25]
>> den=[1 1.75 0.25]
>> [zer, pol, k] = tf2zp(num, den)
```

Aceasta returnează:

```
zer =

 -5.0000e-004

pol =

   -1.5931
   -0.1569

k =

   500
```

care, pentru exemplul considerat, returnează aceiaşi vectori *pol* şi *k* ca şi cei obţinuţi în secţiunea anterioară prin comanda *ss2zp. (vezi exemplul referitor la sistemul prezentat în figura 4.10).* În acel caz aveam un zero =-0.25 (prima coloană din matricea z) şi respectiv un coeficient k =1 (prima linie din matricea k), lucru care este identic cu a avea un zero egal cu-$5*10^{-4}$ şi un coeficient k = 500.

De asemenea, se obţin aceleaşi valori pentru modelul poli-zerouri, rulând comanda ss2zp, plecând de la modelul de stare din exemplul anterior.

```
>> [zer, pol, k] = ss2zp(A,B,C,D)
```

▶

4.3.3. Conversia din poli-zerouri

Pentru obținerea MM-ISI respectiv a matricei de transfer, pornind de la o reprezentare de tip poli-zerouri, se folosesc instrucțiunile ***zp2ss*** respectiv ***zp2tf***.

❑ ***zp2ss***

Exemplu:

Se consideră un sistem liniar, invariant având un zero egal cu –1, coeficientul de transfer egal cu 2 şi 3 poli, cu valorile: 1, 2+3i şi respectiv 2-3i.

Pentru a calcula modelul matematic de tip intrare-stare-ieşire asociat acestui sistem, se rulează comenzile:

```
zer=-1
pol=[1 2+3i 2-3i]
k=2
[A, B, C, D] =zp2ss (zer, pol, k)
```

Acestea vor returna:

```
A =

    1.0000         0              0
    2.0000    4.0000     -3.6056
         0    3.6056              0

B =

     1
     1
     0

C =

         0         0    0.5547

D =

     0
```

❑ ***zp2tf***

Funcția de transfer asociata acestui sistem se poate calcula cu ajutorul comenzii:

```
>> [num, den] =zp2tf (zer, pol, k)
```

care conduce la:

```
num =

     0     0     2     2

den =

     1    -5    17   -13
```

Notă: Mediul MATLAB pune la dispoziția utilizatorului o comandă care permite afișarea pe ecran a unei funcții de transfer într-un format mai ușor de interpretat. Această comandă este ***printsys***, iar apelul ei este de forma următoare:

```
>> printsys(num,den)
```

Efectul acesteia este o afișare pe ecran de forma:

```
num/den =

            2 s + 2
   ----------------------
   s^3 - 5 s^2 + 17 s - 13
```

►

4.4. Conversia continuu-discret

Comenzile uzuale pentru realizarea trecerii din domeniul continuu la cel discret sunt:

- ***c2d*** - conversia de la continuu la discret
- ***c2dm*** - conversia de la continuu la discret, cu specificarea metodei de discretizare

Se consideră un sistem în timp continuu modelat în spațiul stărilor:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \quad \text{cu } t \text{ –timp continuu} \qquad (4.43)$$

Sintaxa comenzii ***c2d*** , pentru discretizarea modelului continuu (4.43) este:

```
[Ad, Bd] = c2d(A,B,Te)
```

unde A, B sunt matricile MM-ISI continuu, iar Te este perioada de eșantionare constantă.

Notă: Matricile C și D nu sunt influențate de această conversie și de aceea nu trebuie specificate.

Ad și respectiv Bd sunt matricile corespunzătoare modelului discret obținut pentru pasul de eșantionare Te, considerând intrarea u trecută printr-un element de reținere de ordinul zero.

Forma MM-ISI discret corespunzător este dată de relaţiile:

$$x(t+1) = A_d x(t) + B_d u(t)$$
$$y(t) = Cx(t) + Du(t) \tag{4.44}$$

cu t – timp discret normalizat, t=0,1,2,...

Exemplu:
Pentru a evidenţia relaţia dintre modelele continuu şi discret, se consideră un sistem SISO de ordinul întâi, descris de ecuaţiile:

$$\dot{x} = -x + 2u$$
$$y = x \tag{4.45}$$

unde $A=-1$, $B=2$, $C=1$, $D=0$

Ieşirea sistemului, pentru un semnal de intrare de tip treaptă unitară (răspunsul indicial) în condiţii iniţiale nule este:

$$y(t) = x(t) = 2(1-e^{-t})u(t) \tag{4.46}$$

MM-ISI discret, având perioada de eşantionare $T_e=1$ secundă, poate fi obţinut prin executarea programului:

```
A=[-1];
B=[2];
C=[1];
D=0;
Te=1;
[Ad, Bd]=c2d(A,B, Te);
disp ('modelul discret este:')
Ad,Bd,C,D
```

Aceasta conduce la:

```
modelul discret este:

Ad =

    0.3679

Bd =

    1.2642

C =

     1

D =

     0
```

Folosind relația (4.46), răspunsul indicial al sistemului pe intervalul continuu t = 0,...,6 secunde, poate fi calculat cu ajutorul comenzilor:

```
>>  t = 0:0.1:6;
>>  xc=2*(1-exp(-t));
```

Pe de altă parte, răspunsul în domeniul timp al sistemului discretizat poate fi calculat folosind secvența de instrucțiuni:

```
>> xd(1)=0;
>>  for i = 1:6/Te
xd(i+1) =Ad*xd(i)+Bd;
end
```

Notă: Este important de subliniat că acest răspuns poate fi de asemenea obținut folosind comanda *dlsim* sau o schemă SIMULINK.

Reprezentarea grafică a celor două răspunsuri (continuu și discret) este obținută prin instrucțiunea:

```
>> plot(t, xc, 0:Te:6, xd, 'o');
```

care furnizează graficul prezentat în figura 4.12.

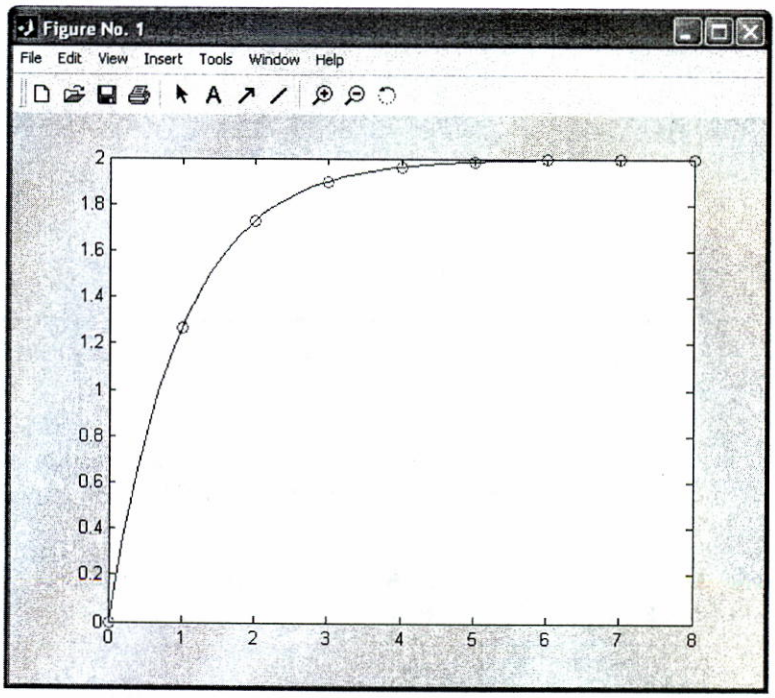

Figura 4.12. Semnalul de iesire continuu (linia continua), respectiv discret (punctele marcate cu cerc)

O variantă cvasicontinuală (în forma de trepte) a semnalului discret (obținuta prin trecerea ieşirii printr-un element de reținere de ordinul zero) se poate genera cu o secvență de instrucțiuni de forma:

```
>> [tx, xdc] = stairs(0:Te:8,xd);
>> plot(t, xc,':', tx,xdc)
>> axis([0 8 -0.5 2.5])
```

rezultatul grafic fiind prezentat în figura 4.13.

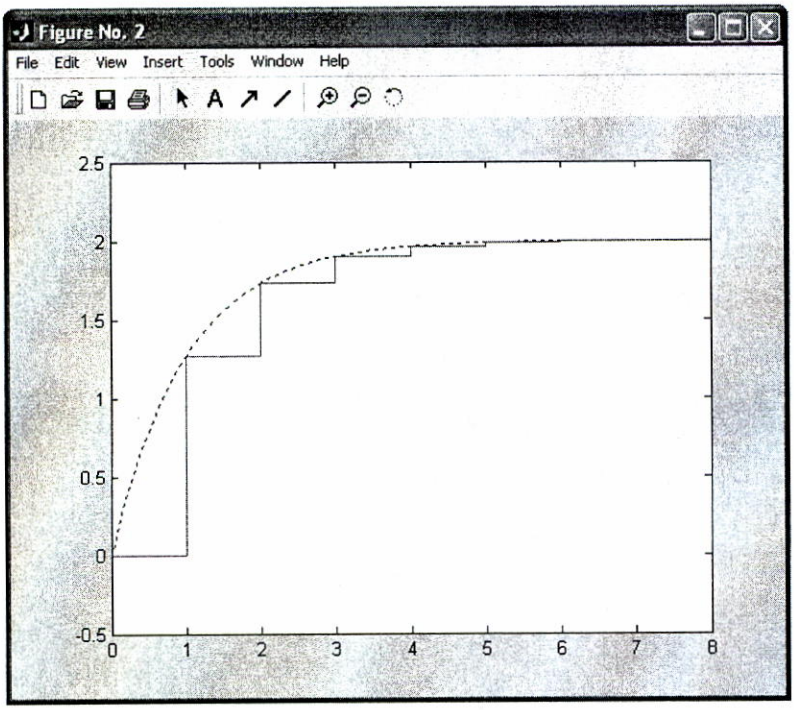

Figura 4.13. Semnalul de ieşire continuu (linia întreruptă) respectiv cvasicontinuu (linia continuă)

▶

Pentru a realiza conversia unui sistem din timp continuu în timp discret, cu specificarea metodei de discretizare, se poate utiliza comanda ***c2dm***, a cărei sintaxă este:

```
[Ad,Bd,Cd,Dd] = c2dm(A,B,C,D,Te,'method') sau
[NUMd,DENd] = c2dm(NUM,DEN,Te,'method')
```

unde

- *A,B,C,D* respectiv *Ad,Bd,Cd,Dd* reprezintă parametrii MM-ISI continuu, respectiv discret;

- *NUMd,DENd* respectiv *NUM,DEN* reprezintă numărătorul, respectiv numitorul funcției de transfer a sistemului continuu, respectiv discret;
- *Te* este pasul de eşantionare,
- *method* reprezintă metoda utilizată la discretizare:
  'zoh' – metoda consideră un element de reținere de ordinul 0 aplicat intrărilor;
  'foh' – metoda consideră un element de reținere de ordinul 1 aplicat intrărilor;
  'tustin' – metoda utilizează aproximarea biliniară (Tustin);
  'prewarp' – metoda consideră aproximarea biliniară (Tustin) şi presupune precizarea frecventei critice (ex: *c2dm(A,B,C,D,Ts,'prewarp',Wc)*);
  'matched' – metoda bazată pe alocare de poli şi zerouri.

Exemplu:
Se va considera un sistem SISO de ordinul întâi, descris de MM-ISI (4.45).
Se prezintă un program care realizează discretizarea modelului considerat, utilizând toate cele cinci metode enumerate.
Se calculează şi afişează de asemenea, mărimile de stare şi respectiv ieşire, pentru o intrare de tip treaptă unitara, corespunzătoare celor cinci modele discrete obținute.

```
%MM-ISI continual şi pasul de esantionare
A=[-1];
B=[2];
C=[1];
D=0;
Tc=1

%calculul MM-ISI prin diferite metode
[Ad1,Bd1,Cd1,Dd1] = c2dm(A,B,C,D,Tc,'zoh')
[Ad2,Bd2,Cd2,Dd2] = c2dm(A,B,C,D,Tc,'foh')
[Ad3,Bd3,Cd3,Dd3] = c2dm(A,B,C,D,Tc,'tustin')
omegan=2*pi*5e-3
[Ad4,Bd4,Cd4,Dd4] = c2dm(A,B,C,D,Tc,'prewarp', omegan)
[Ad5,Bd5,Cd5,Dd5] = c2dm(A,B,C,D,Tc,'matched')

%calculul raspunsului discret (starea şi iesirea) pentru
modelele ISI obtinute
xd1(1)=0;
for i = 1:8/Tc
    xd1(i+1) =Ad1*xd1(i)+Bd1;
    yd1(i)=Cd1*xd1(i)+Dd1;
end

xd2(1)=0;
for i = 1:8/Tc
    xd2(i+1) =Ad2*xd2(i)+Bd2;
```

```
    yd2(i)=Cd2*xd2(i)+Dd2;
end

xd3(1)=0;
for i = 1:8/Tc
    xd3(i+1) =Ad3*xd3(i)+Bd3;
    yd3(i)=Cd3*xd3(i)+Dd3;
end

xd4(1)=0;
for i = 1:8/Tc
    xd4(i+1) =Ad4*xd4(i)+Bd4;
    yd4(i)=Cd4*xd4(i)+Dd4;
end

xd5(1)=0;
for i = 1:8/Tc
    xd5(i+1) =Ad5*xd5(i)+Bd5;
    yd5(i)=Cd5*xd5(i)+Dd5;
end

%afisarea grafica a marimilor de stare şi iesire pentru MM-
ISI discrete
plot(0:8, [xd1;xd2;xd3;xd4;xd5])
figure
plot(0:7, [yd1;yd2;yd3;yd4;yd5])

%calculul şi afisarea raspunsului indicial continuu
t = 0:0.1:7;
xc=2*(1-exp(-t));
hold on
plot(t, xc,':')
axis([0 7 -0.2 2.2])

%afisarea pe ecran a valorilor marimilor de iesire
format short e
iesiri=[yd1;yd2;yd3;yd4;yd5]
```

Evoluțiile celor cinci sisteme discrete în spațiul stărilor sunt prezentate în figura 4.14.

În figura 4.15 se prezintă răspunsul indicial al celor cinci sisteme discrete (linie continuă), respectiv răspunsul sistemului continuu (linie întreruptă).

În figura 4.15 se disting numai trei grafice cu linie continuă în loc de cinci deoarece în acest caz, răspunsurile obținute prin metodele *'zoh'*, *'matched'* şi respectiv *'tustin'*, *'prewarp'* se suprapun.

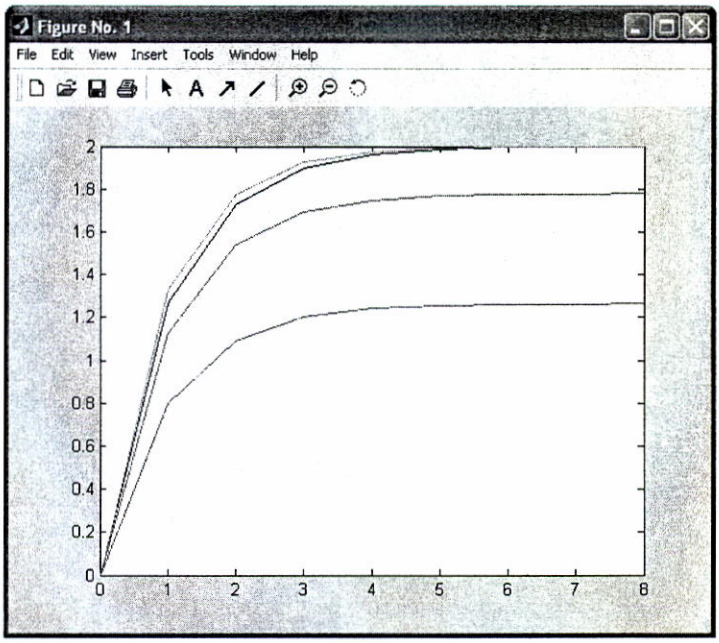

Figura 4.14. Mărimile de stare corespunzătoare celor 5 modele discrete

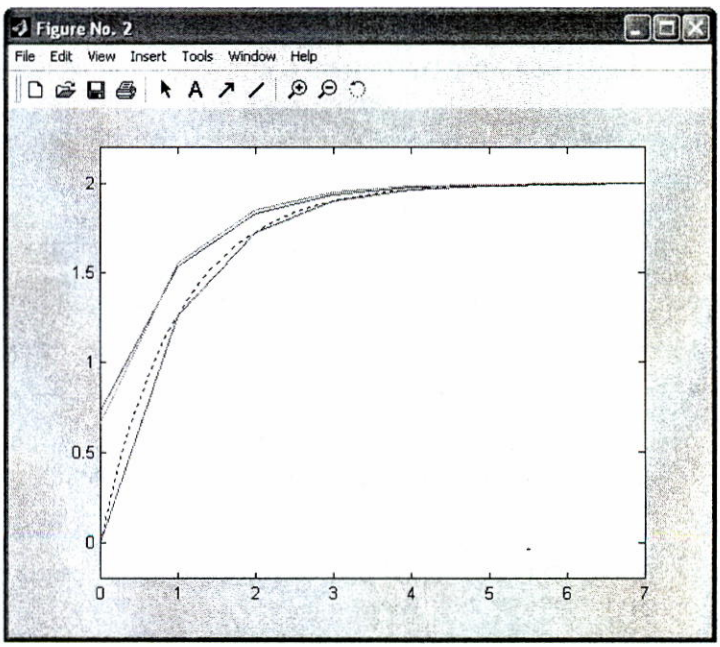

Figura 4.15. Răspunsul indicial al celor 5 sisteme discrete (linie continuă), respectiv răspunsul indicial al sistemului continuu (linie întreruptă)

Valorile numerice calculate pentru cele cinci ieşiri demonstrează acest lucru:

```
iesiri =

  Columns 1 through 5
            0         1.2642e+000  1.7293e+000  1.9004e+000
1.9634e+000
  7.3576e-001  1.5349e+000  1.8289e+000  1.9371e+000
1.9768e+000
  6.6667e-001  1.5556e+000  1.8519e+000  1.9506e+000
1.9835e+000
  6.6670e-001  1.5556e+000  1.8519e+000  1.9506e+000
1.9835e+000
            0     1.2642e+000  1.7293e+000  1.9004e+000
1.9634e+000

  Columns 6 through 8
  1.9865e+000  1.9950e+000  1.9982e+000
  1.9915e+000  1.9969e+000  1.9988e+000
  1.9945e+000  1.9982e+000  1.9994e+000
  1.9945e+000  1.9982e+000  1.9994e+000
  1.9865e+000  1.9950e+000  1.9982e+000
```

4.5. Conversia discret-continuu

Pentru realizarea trecerii de la domeniul discret la cel continuu, comenzile disponibile în MATLAB sunt:

- ***d2c*** - conversia de la discret la continuu
- ***d2cm*** - conversia de la discret la continuu cu specificarea metodei utilizate

Sintaxa comenzii ***d2c*** , utilizată pentru trecerea în domeniul continuu a unui sistem discret, este:

```
[Ac, Bc] = d2c(Ad,Bd,Te)
```

unde *Ad*, *Bd* sunt matricele MM-ISI al sistemului discret, *Ac*, *Bc* sunt matricele MM-ISI al sistemului continuu obținut, iar *Te* este perioada de eşantionare.

Exemplu:
Se consideră un sistem SISO de ordinul unu, în timp continuu, descris de modelul (4.45).

Modelul obținut prin discretizarea sistemului considerat se obține cu secvența de comenzi prezentată în exemplul din paragraful 4.4:

```
A=[-1];
B=[2];
C=[1];
D=0;
Tc=1;
[Ad, Bd]=c2d(A,B, Tc);
Cd=C;
Dd=D;
disp ('modelul discret este:')
Ad,Bd,Cd,Dd
```

Pentru a obține din nou modelul continuu, pornind de la MM-ISI discret *{Ad,Bd,C,D}* calculat prin secvența anterioară, se execută comanda:

```
>> [Ac,Bc]=d2c(Ad,Bd,Tc)
```

rezultatul fiind:

```
Ac =

   -1.0000

Bc =

     2
```

Practic se obțin valori identice cu cele ale modelului continuu inițial.

În mod analog comenzii *c2dm*, prezentată în paragraful 4.4, există comanda ***d2cm***, care permite specificarea metodei prin care se va realiza trecerea de la domeniul discret la cel continual. Apelul acesteia se face la fel ca și cel al comenzii *c2dm*, metoda specificată putând fi una dintre:

'zoh' – conversie considerând un element de reținere de ordinul 0 aplicat intrărilor;

'tustin' – conversie utilizând aproximarea biliniară (Tustin);

'prewarp' – metoda consideră aproximarea biliniară (Tustin) și presupune precizarea frecvenței critice (ex: *d2cm(A,B,C,D,Ts,'prewarp',Wc)*);

'matched' – conversie prin alocare de poli și zerouri.

Exemplu:

Se consideră același sistem SISO de ordinul unu, din cadrul exemplului precedent și se calculează modelul continuu, pe baza aproximării Tustin aplicată modelului discret descris de matricile *Ad, Bd, Cd, Dd*:

```
>> [Act,Bct,Cct,Dct]=d2cm(Ad,Bd,Cc,Dd,Tc,'tustin')
```

Matricile modelului continuu obținut sunt:

```
Act =

   -0.9242

Bct =

    1.8485

Cct =

    1.4621

Dct =

   -0.9242
```

4.6. Exemple de programe privind conversii de modele matematice

a) Exemplu de program care realizează trecerea de la MM-ISI la f.d.t. utilizând funcția *ss2tf*.

```
% afişare mesaj explicativ
disp('--------------------------')
disp('TRECEREA MM-ISI --->  f.d.t.')
disp('--------------------------')
disp(' ')
% stabilire format afişare - suprima liniile vide la afisare
format compact
disp('MM-ISI')
% introducere matrici corespunzatoare MM-ISI
A=input('A=');
B=input('B=');
C=input('C=');
D=input('D=');
% se stabileşte numărul de intrări
iu=1;
% se realizează transformarea propriu-zisă
[num,den]=ss2tf(A,B,C,D,iu);
% afişare coeficienţi numărător f.d.t.
disp('Coeficienţii numărătorului f.d.t.')
num
% afişare coeficienţi numitor f.d.t.
disp('Coeficienţii numitorului f.d.t.')
den
```

b) Exemplu de program care realizează o transformare a unui MM-ISI din domeniul timp continuu în domeniul timp discret, utilizând funcția ***c2d***.

```
format compact
disp('----------------------------')
disp('PROGRAM PENTRU TRECEREA CONTINUU ---> DISCRET')
disp('----------------------------')
disp(' ')
disp('MM-ISI continual')
A=input('A=');
B=input('B=');
C=input('C=');
D=input('D=');
T=input('perioada de discretizare');
[Ad,Bd]=c2d(A,B,T)
disp('MM-ISI discret')
Ad
Bd
Cd=C
Dd=D
```

c) Exemplu de program de aducere a unei funcții de transfer la forma poli-zerouri.

```
format compact
disp('f,d.t.')
% introducere coeficienți numărător, numitor f.d.t.
num=input('num= ');
den=input('den= ');
% trecere de la f.d.t.-> configuratie poli-zerouri
i=sqrt(-1);
[Z,P,K]=tf2zp(num,den);
% afişare POLI f.d.t.
Z,P,K
```

Notă: amplasarea în planul complex "s" a polilor sistemului, permite aprecierea stabilității acestuia utilizând teorema fundamentală a stabilității: un sistem continuu este stabil dacă şi numai dacă toți polii f.d.t. sunt amplasați strict în semiplanul stâng al planului complex "s".

4.7. Probleme de studiat

1. Să se scrie un program pentru trecerea de la MM-ISI la configuraţia "poli-zerouri" respectiv pentru trecerea din domeniul timp discret în domeniul timp continuu.
2. Utilizând transformările de modele considerate, să se studieze stabilitatea sistemului reprezentat printr-un circuit serie RLC funcţionând în gol (figura 4.16) având funcţia de transfer:

$$H(s)=\frac{u_e(s)}{u_i(s)}=\frac{1}{LCs^2+RCs+1}$$

pentru următoarele valori numerice: R=1 K , C=1 F , L=1 H (vezi teorema fundamentală a stabilităţii).

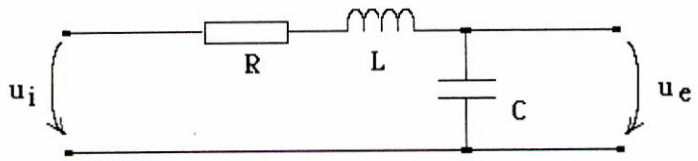

Figura 4.16. Circuit serie RLC

3. Se consideră sistemul caracterizat prin MM-ISI având:

$$A=\begin{pmatrix}1 & 0\\ 1 & 1\end{pmatrix},\ B=\begin{pmatrix}1\\ 0\end{pmatrix},\ C=\begin{pmatrix}1 & 1\end{pmatrix},\ D=0$$

Să se determine analitic $H(s)$, folosind relaţia :

$$H(s)=C(s\cdot I_n-A)^{-1}B+D$$

şi apoi să se verifice rezultatul obţinut prin rularea programului de trecere de la MM-ISI la f.d.t. prezentat în paragraful 4.6.

Capitolul 5

SIMULAREA FUNCȚIONĂRII SISTEMELOR LINIARE CONTINUE ŞI DISCRETE

5.1. Considerații generale

Acest paragraf are ca obiectiv prezentarea modalităților de simulare în MATLAB, a sistemelor liniare atât pentru semnale tipizate aplicate la intrare cât şi pentru semnale oarecare. Simularea se realizează în ambele domenii: continuu şi discret.

Înainte de a trece la prezentarea instrucțiunilor MATLAB care permit analiza răspunsului în domeniul timp al unui sistem liniar invariant, se aminteşte că acest răspuns este exprimat analitic de expresia:

$$y(t) = Ce^{At}x_0 + \int_0^t (Ce^{A(t-\tau)}B + D)u(\tau)d\tau \tag{5.1}$$

unde x_0 reprezintă condițiile inițiale, $u(t)$ este intrarea sistemului şi $\{A,B,C,D\}$ sunt parametrii modelului de stare MM-ISI (vezi paragraful 4.1).

În relația de mai sus, primul termen din membrul drept descrie evoluția liberă a sistemului (răspunsul liber), iar al doilea evoluția forțată de semnalul de intrare aplicat sistemului (răspunsul forțat).

Pornind de la relația de definiție a funcției de transfer, răspunsul forțat al sistemului poate fi obținut ca:

$$y(t) = L^{-1}(Y(s)) = L^{-1}(H(s)\cdot U(s)) \tag{5.2}$$

unde L^{-1} este operatorul transformării Laplace, $H(s)$ este funcția de transfer a sistemului, iar $U(s)$ reprezintă imaginea Laplace a semnalului de la intrarea sistemului.

Pentru exemplificarea celor de mai sus se consideră sistemul caracterizat de următorul model matematic (5.3), la intrarea căruia se aplică un semnal tip treaptă unitară.

$$\begin{cases} \dot{x} = \begin{pmatrix} -0.1 & 3 & 0 \\ -3 & -0.1 & 0 \\ 0 & 0 & -1 \end{pmatrix} x + \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} u \\ y = \begin{pmatrix} 1 & 0 & 1 \end{pmatrix} x \end{cases} \tag{5.3}$$

funcţia de transfer a sistemului poate fi obţinută cu ajutorul relaţiei:

$$H(s) = C(s \cdot I_n - A)^{-1} B + D \tag{5.4}$$

unde $\hat{I}n$ este matricea identitate de dimensiune (nxn)
sau apelând instrucţiunile MATLAB:

```
>> A=[-0.1 3 0; -3 -0.1 0; 0 0 -1]
>> B=[0;1;1]
>> C=[1 0 1]
>> D=0
>> [numarat,numit]=ss2tf(A,B,C,D)
```

Astfel se obţine:

```
numarat =
         0    1.0000    3.2000   12.0100

numit =
    1.0000    1.2000    9.2100    9.0100
```

adică:

$$H(s) = \frac{s^2 + 3.2s + 12.01}{s^3 + 1.2s^2 + 9.21s + 9.01} \tag{5.5}$$

Considerând intrarea de tipul treaptă unitară, din tabelele de transformate Laplace (vezi Anexa 2) rezulta:

$$U(s) = \frac{1}{s} \tag{5.6}$$

Pe baza relaţiilor (5.2), (5.5) şi (5.6), şi descompunând apoi rezultatul în fracţii simple se obţine:

$$Y(s) = \frac{-0.166 \pm 0.005j}{s + 0.1 \mp 3j} + \frac{-1}{s+1} + \frac{1.333}{s} \tag{5.7}$$

Folosind transformata Laplace inversă se obţine răspunsul în domeniul timp:

$$y(t) = (0.332\, e^{-0.1t} \cos(3t + 3.108) - e^{-t} + 1.333) \cdot 1(t) \tag{5.8}$$

unde 1(t) este funcţia treaptă unitară.

Principalele comenzi MATLAB folosite pentru determinarea răspunsului sistemelor liniare invariante, pentru diverse tipuri de semnale de intrare, pentru cazul continuu, respectiv discret sunt:

- continuu:
  - ***initial*** – răspunsul liber;
  - ***impulse*** – răspunsul la impuls;
  - ***step*** – răspunsul la treaptă unitară;
  - ***lsim*** – răspunsul la orice semnal de intrare;
- discret:
  - ***dinitial*** – răspunsul liber;
  - ***dimpulse*** – răspunsul la impuls;
  - ***dstep*** – răspunsul la treaptă unitară;
  - ***dlsim*** – răspunsul la orice semnal de intrare.

5.2. Răspunsul liber

Pentru determinarea răspunsului liber al unui sistem liniar, invariant în raport cu timpul, pentru orice condiții inițiale x_0 se folosește comanda ***initial***.

Exemplu:
Se consideră sistemul descris de ecuațiile (4.3), având următoarea stare inițială:

$$x_0=(2\ \ 1\ \ 0)^T \tag{5.9}$$

Secvența de comenzi MATLAB care calculează răspunsul liber al sistemului pentru MM-ISI (5.3), în condițiile inițiale considerate este:

```
>> A=[-0.1 3 0; -3 -0.1 0; 0 0 -1];
>> B=[0; 1; 1];
>> C=[1 0 1];
>> D=0;
>> x0=[2; 1; 0];
>> initial(A,B,C,D,x0);
```

Rezultatul acestei secvențe are ca efect **trasarea graficului** evoluției libere a sistemului considerat pentru starea inițiala x_0 (5.9) (vezi figura 5.1).

Notă: Pentru sistemele multi-output, apelul comenzii ***initial*** trasează în aceeași fereastră câte un grafic pentru fiecare ieșire.
Dacă se dorește furnizarea valorilor numerice ale mărimii de ieșire respectiv stare, corespunzătoare unor momente de timp t, se utilizează o comanda având sintaxa:

```
>> [out, state, t]= initial(A,B,C,D,x0)
```

unde *out*, și *state* returnează vectorul valorilor ieșirii, respectiv matricea valorilor variabilelor de stare corespunzătoare momentelor de timp t (vector) generate.
Apelul comenzii în aceasta forma nu generează un grafic, însă aceste valori pot fi ulterior reprezentate în formă grafică, utilizând comanda ***plot***.

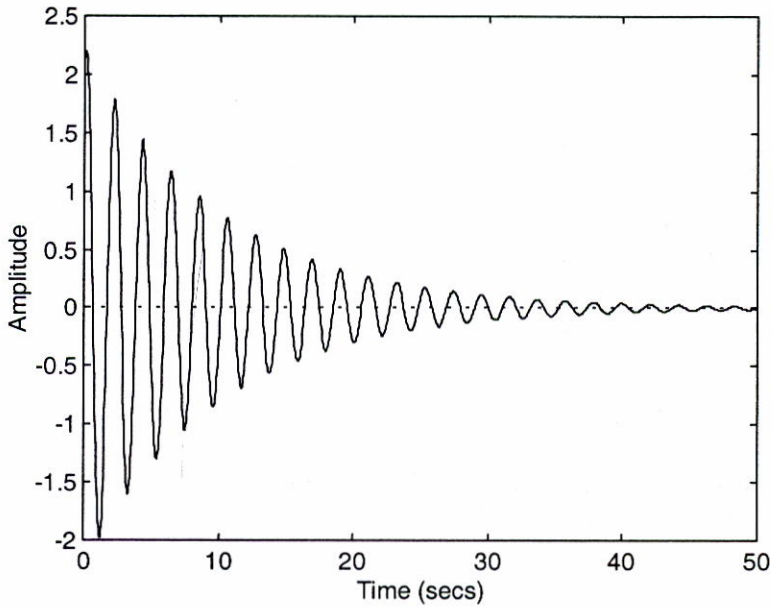

Figura 5.1. Evoluţia liberă a sistemului

Dacă se doreşte ca valorile calculate pentru ieşire, respectiv pentru starea sistemului, să corespundă unor anumite momente de timp, bine precizate, acestea se specifică printr-un vector t, iar comanda va fi apelată cu sintaxa:

```
>> [iesire, stare]= initial(A,B,C,D,x0,t)
```

Este important de menţionat că vectorul t trebuie sa conţină valori echidistante (pas constant) şi strict crescătoare.

Pentru calculul răspunsului liber (evoluţiei libere) a sistemelor în domeniul discret, comanda folosită este ***dinitial***, sintaxa fiind identică cu a comenzii corespunzătoare din cazul continuu.

Considerând acelaşi MM-ISI şi aceeaşi stare iniţială ca şi în exemplul precedent (cazul continuu) se pune problema discretizării modelului şi calculul răspunsului liber discret. Astfel primele 5 comenzi ale programului anterior se completează cu următoarea secvenţa:

```
>> Te=0.5;
>> [Ad, Bd]= c2d(A,B,Te);
>> dinitial(Ad,Bd,C,D,x0);
```

având ca rezultat răspunsul liber discret reprezentat în figura 5.2.

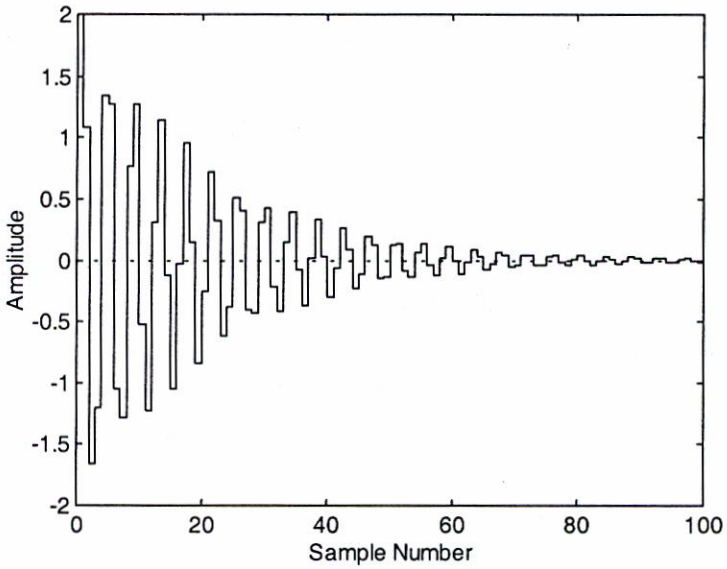

Figura 5.2. Răspunsul liber discret.

5.3. Răspunsul indicial

Răspunsul indicial reprezintă răspunsul sistemului pentru un semnal de intrare de tip treapta unitara în condiții inițiale nule.

Comanda MATLAB care permite calculul mărimilor de ieşire respectiv stare pentru un semnal treaptă unitara aplicat în intrare este ***step***, sintaxa acesteia fiind destul de flexibilă, putând lua una din formele:

```
step(A,B,C,D) sau step(num,den)
[iesire,stare] = step(A,B,C,D,t)
[iesire,stare] = step(num,den,t)
[iesire,stare,t] = step(A,B,C,D)
[iesire,stare,t] = step(num,den)
```

În mod similar comenzii ***initial***, dacă apelul se face doar cu parametri de intrare şi fără specificarea explicita a parametrilor de ieşire, rezultatul va fi reprezentat grafic. În celelalte cazuri, valorile ieşirii şi ale stărilor sunt memorate în variabilele specificate, graficul putându-se obține cu ajutorul comenzii ***plot***. Dacă se dorește specificarea momentelor de timp în care să se facă evaluarea ieşirii şi a stării sistemului, atunci se construieşte un vector *t*, echidistant, care se transmite la apelul funcției. Dacă apelul se face fără specificarea timpului, acesta este determinat şi returnat automat la execuția comenzii.

Exemplu:
Se consideră sistemul descris de modelul (5.3), pentru care se determina răspunsul indicial:

```
>> A=[-0.1 3 0; -3 -0.1 0; 0 0 -1];
>> B=[0; 1; 1];
>> C=[1 0 1];
>> D=0;
>> step(A,B,C,D);
```

Rezultatul obținut este prezentat în figura 5.3.

În cazul sistemelor multi-input, trebuie specificat care intrare să fie de tipul treaptă, restul fiind considerate nule. Acest lucru este posibil folosind una din sintaxele următoare:

```
step(A,B,C,D,iu)
[iesire,stare,t] = step(A,B,C,D,iu)
[iesire,stare] = step(A,B,C,D,iu,t)
```

unde *iu* este numărul intrării nenule.

Dacă intrarea nenulă nu este specificată, comanda construieşte răspunsul considerând toate perechile intrare-ieşire posibile în parte.

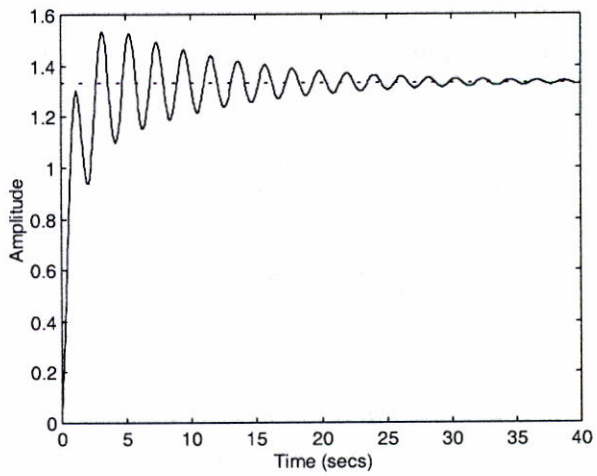

Figura 5.3. Răspuns indicial continuu

Pentru determinarea răspunsului indicial în cazul sistemelor discrete, comanda folosită este ***dstep***, sintaxa fiind identică cu a comenzii pentru cazul continual. Pentru exemplificarea celor prezentate, se poate completa programul anterior cu următoarele instrucţii:

```
>> Te=0.5;
>> [Ad,Bd]= c2d(A,B,Te);
>> dstep(Ad,Bd,C,D);
```

conducând la răspunsul din figura 5.4:

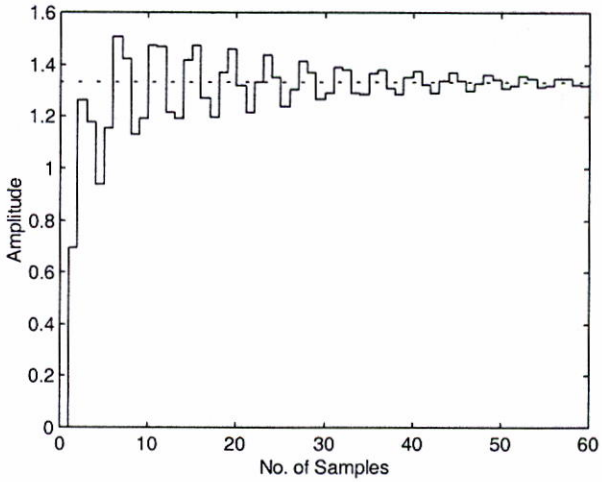

Figura 5.4. Răspuns indicial discret

5.4. Răspunsul la impuls

Comanda MATLAB care determină răspunsul unui sistem liniar, invariant, pentru un semnal de intrare de tip impuls unitar (impuls Dirac), pentru condiții inițiale nule – numit *funcție pondere*, este ***impulse***. Sintaxa acestei comenzi este similară comenzii ***step*** prezentate în paragraful anterior, putând lua una din formele:

```
impulse(A,B,C,D)
impulse(num,den)
[iesire,stare] = impulse(A,B,C,D,T)
[iesire,stare = impulse (num,den,T)
[iesire,stare,T] = impulse (A,B,C,D)
 [iesire,stare,T] = impulse (num,den)
impulse(A,B,C,D,iu)
[Y,X,T] = impulse(A,B,C,D,iu)
[iesire,stare] = impulse(A,B,C,D,iu,T)
```

Toate precizările legate de modul de specificare a timpului și a intrărilor nenule (în cazul sistemelor multi-input) se păstrează la fel cu cele ale comenzii ***step***.

Pentru a determina răspunsul la impuls al sistemului modelat de ecuațiile (5.3) se scrie secvența următoare:

```
>> A=[-0.1 3 0; -3 -0.1 0; 0 0 -1];
>> B=[0; 1; 1];
>> C=[1 0 1];
>> D=0;
>> impulse(A,B,C,D);
```

obținându-se rezultatul prezentat în figura 5.5.

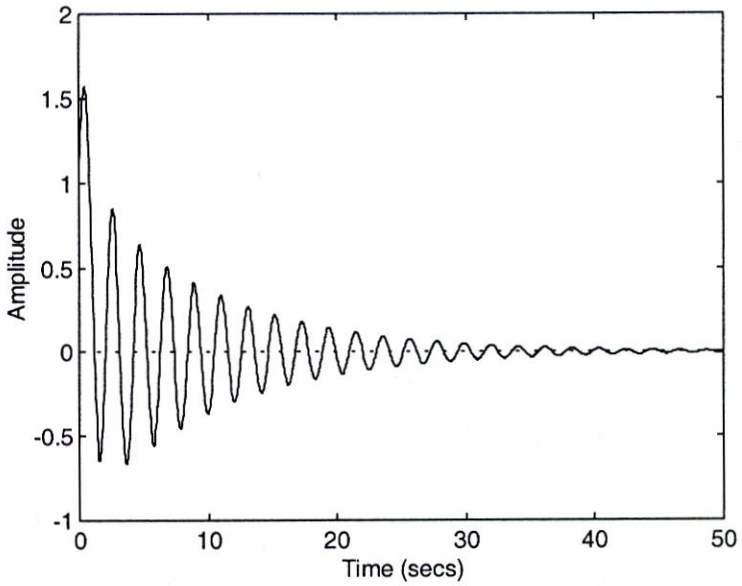

Figura 5.5. Răspuns la impuls continual

Pentru determinarea în domeniul discret a răspunsului sistemelor pentru un impuls unitar aplicat în intrare, în condiții inițiale nule, numit *secvența de ponderare*, comanda care poate fi folosită este ***dimpulse***, sintaxa fiind identică cu a comenzii pentru cazul continuu.

Pentru exemplificarea acestei situații, se poate completa programul anterior cu următoarele instrucții:

```
>> Te=0.5;
>> [Ad,Bd]= c2d(A,B,Te);
>> dimpulse(Ad,Bd,C,D);
```

rezultatul fiind arătat în figura 5.6.

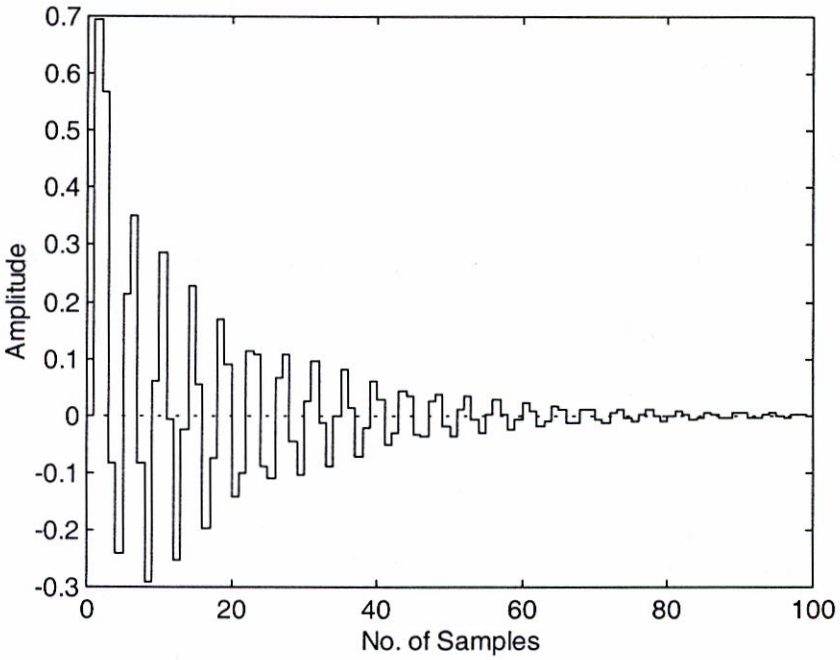

Figura 5.6. Răspuns la impuls discret

5.5. Răspunsul la orice tip de semnal de intrare

Un caz mai general îl constituie determinarea răspunsului unui sistem liniar, invariant în raport cu timpul, generat de aplicarea la intrare a unui semnal oarecare, pentru orice condiții inițiale cunoscute. Acest răspuns este obținut folosind comanda MATLAB ***lsim***.

Semnalul de intrare trebuie "construit" folosind comenzi MATLAB, rezultatul constituindu-se sub forma unui vector care conține eșantioanele semnalului luate la momente de timp echidistante. Mărimea de ieșire și respectiv starea sistemului vor fi returnate de comanda ***lsim*** sub formă de matrici bidimensionale, care vor avea o dimensiune egală cu lungimea vectorului semnalului de intrare, cealaltă dimensiune fiind dată de numărul de ieșiri și respectiv numărul de variabile de stare ale sistemului considerat (ordinul sistemului).

În mod evident, momentele de timp pentru care sunt calculate valorile ieșirii și ale stării sistemului corespund acelora pentru care au fost generate eșantioanele semnalului de intrare.

Sintaxa comenzii este:

```
lsim(A,B,C,D,u,t)
lsim(num,den,u,t)
[iesire,stare] = lsim(A,B,C,D,T)
[iesire,stare] = lsim(num,den,T)
[Y,X,T] = lsim(A,B,C,D)
[Y,X,T] = lsim(num,den)
lsim(A,B,C,D,T,X0)
[iesire,stare] = lsim(A,B,C,D,T,X0)
```

unde *u* este vectorul corespunzător semnalului de intrare, *t* vectorul momentelor de timp echidistante pentru care s-au calculat eşantioanele din *u*, iar *X0* este vectorul care conţine starea iniţială a sistemului. Dacă vectorul *X0* lipseşte se presupune că starea iniţială este nulă.

Exemplu:
Se consideră sistemul modelat de ecuaţiile (5.3). Se presupune de asemenea că semnalul de intrare este de tipul tren de impulsuri dreptunghiulare, cu factor de umplere ½ şi perioadă 80. Simularea comportării sistemului pe intervalul a trei perioade se poate efectua prin următoarea secvenţă de comenzi:

```
>> A=[-0.1 3 0; -3 -0.1 0; 0 0 -1];
>> B=[0; 1; 1];
>> C=[1 0 1];
>> D=0;
>> Ut=[zeros(1,40) ones(1,40)];
>> U=[Ut Ut Ut];
>> lsim(A,B,C,D,U,1:240);
```

În urma rulării programului se obţine rezultatul prezentat în figura 5.7.

Pentru a analiza răspunsul unui sistem, pentru orice tip de semnal aplicat la intrare şi orice condiţii iniţiale cunoscute a priori, în cazul discret, se foloseşte comanda ***dlsim***. Sintaxa este asemănătoare cu a comenzii corespunzătoare pentru cazul continuu, diferenţa constând în faptul că specificarea momentelor de timp nu mai este necesară.

Exemplu:
Considerând din nou sistemul modelat prin MM-ISi (5.3), pentru simularea în domeniul discret, la programul din cazul continuu prezentat în exemplul anterior trebuie adăugate următoarele comenzi:

```
>> Te=0.5;
>> [Ad, Bd]= c2d(A,B,Te);
>> Ut=[zeros(1,40) ones(1,40)];
>> U=[Ut Ut Ut];
>> dlsim(Ad,Bd,C,D,U);
```

răspunsul discret obţinut este prezentat în figura 5.8.

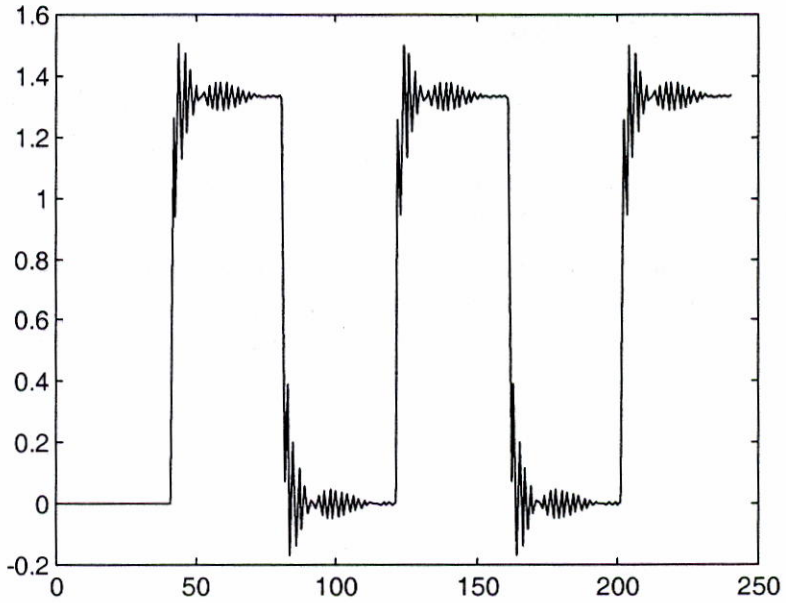

Figura 5.7. Răspuns la semnal dreptunghiular

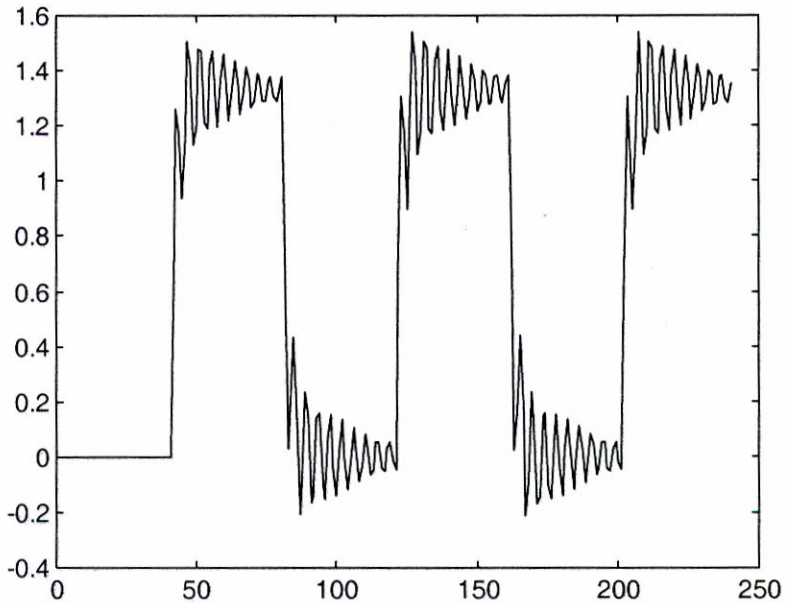

Figura 5.8. Răspuns discret la semnal dreptunghiular

5.6. Probleme de studiat

1. Să se scrie un program de simulare a unui sistem dintre cele prezentate în capitolul 4.
2. Să se studieze influenţa parametrilor m, C şi k asupra comportării sistemului mecanic masă-resort-amortizor, referit în paragraful 4.2.1. Se consideră că mărimea de intrare este de tipul treaptă unitară.
3. Să se scrie un program de simulare a sistemului mecanic masă-resort-amortizor, referit în paragraful 4.2.1, considerând ca acestuia i se aplica o forţă constantă egală cu 10 N, iar la momentul zero, arcul este întins cu 1 m, iar căruciorul se deplasează spre dreapta cu 2 m/s.

Capitolul 6

SIMULAREA FUNCȚIONĂRII SISTEMELOR CU INTERCONEXIUNI

6.1. Considerații generale

În acest paragraf se prezintă câteva modalități de simulare a funcționării sistemelor compuse din subsisteme, ale căror modele matematice sunt disponibile a priori, conectate între ele pe baza unei topologii cunoscute.

Principial, determinarea răspunsului unor astfel de sisteme se rezolvă în doi pași:

- determinarea unui model matematic echivalent întregii structuri;
- simularea funcționării sistemului pe baza modelului matematic obținut la pasul anterior.

Comenzile MATLAB necesare pentru determinarea modelului echivalent sunt:

- ***append*** – extinde starea sistemului;
- ***augstate*** – extinde ieşirea sistemului pe baza stării;
- ***blkbuild*** – construieşte un sistem în spațiul stărilor;
- ***cloop*** – modelează un sistem în buclă închisă cu reacție unitară;
- ***connect*** – calculează modelul unui sistem interconectat;
- ***feedback*** – modelează un sistem în buclă închisă;
- ***parallel*** – modelează un sistem configurat într-o conexiune paralel;
- ***series*** – modelează un sistem configurat într-o conexiune serie.

Toate aceste comenzi sunt valabile atât pentru sistemele continue, cât şi pentru sistemele în timp discret.

Trebuie subliniat că implementarea comenzilor de mai sus conduce la o acuratețe mai mare a rezultatelor dacă sistemele se modelează în spațiul stărilor. Modelele obținute sunt de cele mai multe ori de formă neminimală. Pentru înlăturarea acestui neajuns este necesară utilizarea comenzii ***minreal*** care aduce modelul de stare la o forma minimală.

Odată ce modelul matematic echivalent a fost obținut, simularea se poate face cu oricare din comenzile MATLAB prezentate în capitolul anterior: ***initial, impulse, step*** sau ***lsim*** şi respectiv corespondentele lor discrete.

6.2. Conexiunea serie

Comanda MATLAB care determină modelul echivalent a unui sistem compus din două subsisteme conectate în serie (vezi figura 6.1) este ***series.*** Sintaxa acesteia permite următoarele forme de apelare:

```
[A,B,C,D] = series(A1,B1,C1,D1,A2,B2,C2,D2)
[A,B,C,D] = series(A1,B1,C1,D1,A2,B2,C2,D2,OUTPUTS1,
            INPUTS2)
[NUM,DEN] = series(NUM1,DEN1,NUM2,DEN2)
```

unde *A1,B1,C1,D1,A2,B2,C2,D2* reprezintă MM-ISI al celor două subsisteme, şi respectiv *NUM1, DEN1, NUM2, DEN2* funcţia de transfer asociată subsistemelor. *OUTPUTS1* şi *INPUTS2* sunt vectori ce conţin indicii ieşirilor sistemului 1 şi ale intrărilor sistemului 2 ce "participă" la conexiune.

În mod implicit, comanda returnează modelul matematic echivalent al unei conexiuni serie a două sisteme, conectând toate ieşirile sistemului 1 la intrările sistemului 2, adică :

$$u2 = y1.$$

Sistemul ce rezultă în acest mod are aceleaşi intrări ca şi sistemul 1, iar ieşirile sunt cele ale sistemului 2.

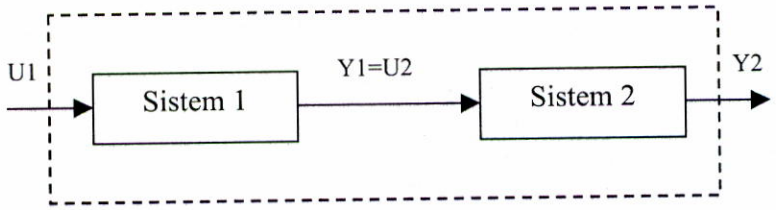

Figura 6.1. Conexiune serie a 2 sisteme

Dacă se doreşte modelarea unei conexiuni parţial conectate (vezi figura 6.2) este necesară utilizarea a încă doi parametri la apelul funcţiei, *OUTPUTS1* şi respectiv *INPUTS2*. În acest mod se obţine modelul matematic al conexiunii pentru cazul în care doar ieşirile specificate în vectorul *OUTPUTS1* ale sistemului 1 sunt conectate la intrările specificate în *INPUTS2* ale sistemului 2.

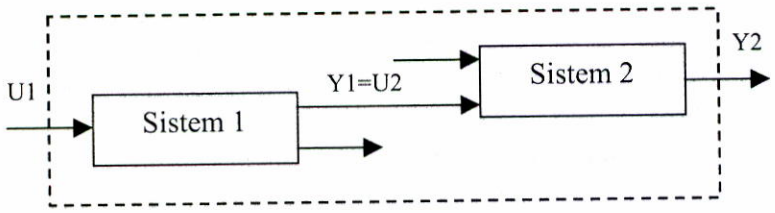

Figura 6.2. Conexiune serie parţială a 2 sisteme

În cazul conexiunii serie parţial singurele intrări ale sistemului echivalent sunt cele ale sistemului 1, iar ieşirile sunt cele ale sistemului 2.

Exemplu:
Se consideră două sisteme de tipul SISO caracterizate prin funcţiile de transfer $H_1(s)$ şi $H_2(s)$.

$$H_1(s)=\frac{1}{2s},\quad H_2(s)=\frac{3}{4s+5} \tag{6.1}$$

Modelul matematic echivalent conexiunii serie a celor două sisteme poate fi ușor obținut analitic pe baza relației:

$$H(s)=H_1(s)\cdot H_2(s)=\frac{1}{2s}\cdot\frac{3}{4s+5}=\frac{3}{8s^2+10s} \tag{6.2}$$

Același rezultat este furnizat și de comanda MATLAB:

```
>> [num, den]=series(1,[2 0],3,[4 5])
```

6.3. Conexiunea paralel

Pentru a determina modelul matematic echivalent unei conexiuni paralel a două sisteme (vezi figura 6.3) se folosește comanda ***parallel***, care permite următoarele forme de apel:

```
[A,B,C,D] = parallel(A1,B1,C1,D1,A2,B2,C2,D2)
[NUM,DEN] = parallel (NUM1,DEN1,NUM2,DEN2)
```

unde parametrii de intrare și respectiv de ieșire au aceeași semnificație ca și în comanda ***series*** prezentată anterior.

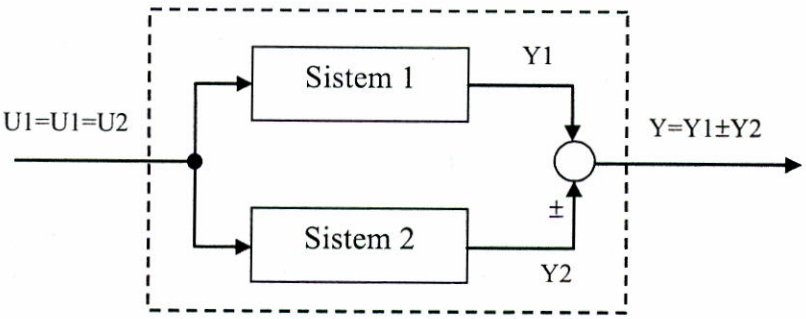

Figura 6.3. Conexiune paralelă a două sisteme

În cazul unor sisteme paralele parțial conectate (vezi figura 6.4), la apelul instrucțiunii *parallel* mai sunt necesare patru argumente de intrare, respectiv vectori care specifică indicii intrărilor și respectiv ieșirilor care "participă" la conexiune.

În acest caz comanda MATLAB trebuie apelată sub forma:

```
>> [A,B,C,D]= parallel(A1,B1,C1,D1,A2,B2,C2,D2,i1,i2,o1,o2)
```

unde numai intrările a căror indici apar în *i1* și respectiv *i2* se consideră coincidente și respectiv doar ieșirile specificate în vectorii *o1* și *o2* se însumează în vederea obținerii ieșirii sistemului. În mod evident *i1* și *i2*, pe de o parte, la fel ca și *o1* și *o2*

pe de altă parte, trebuie să aibă aceeaşi dimensiune. Intrările şi ieşirile care nu sunt specificate la apelul comenzii, spre deosebire de comanda ***series*** prezentată anterior, sunt considerate intrări şi respectiv ieşiri pentru sistemul paralel echivalent.

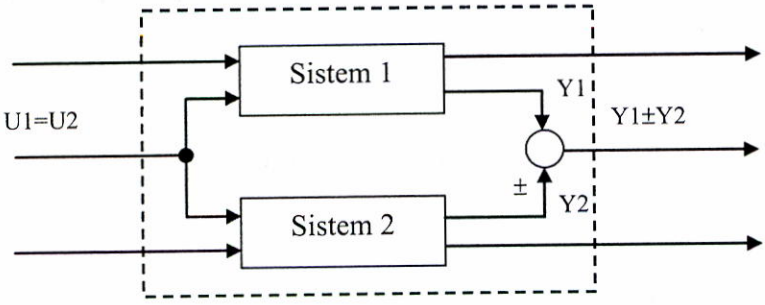

Figura 6.4. Conexiune paralel parţială a 2 sisteme

Exemplu:
Se consideră două sisteme de tipul SISO caracterizate prin funcţiile de transfer $H_1(s)$ şi $H_2(s)$ prezentate în relaţia (6.1).
Modelul matematic echivalent conexiunii paralel a celor două sisteme se obţine pe baza relaţiei:

$$H(s) = H_1(s) + H_2(s) = \frac{1}{2s} + \frac{3}{4s+5} = \frac{10s+5}{8s^2+10s} \tag{6.3}$$

Acelaşi rezultat este furnizat şi de comanda MATLAB corespunzătoare:

```
>> [num, den]=parallel(1,[2 0],3,[4 5])
```

6.4. Conexiunea cu reacţie

În figura 6.5 se prezintă o conexiune cu reacţie (un sistem în buclă închisă) a cărui model matematic poate fi calculat cu ajutorul comenzii ***feedback***, care permite urmatoarele forme de apel:

```
[A,B,C,D] = feedback(A1,B1,C1,D1,A2,B2,C2,D2,sign)
[NUM,DEN] = feedback(NUM1,DEN1,NUM2,DEN2,sign)
```

unde parametrii de intrare şi respectiv de ieşire au aceeaşi semnificaţie ca şi în cazul comenzilor prezentate anterior, mai puţin ultimul argument care specifică semnul reacţiei (luând valoarea 1 sau –1). Argumentul *sign* poate fi omis, caz în care se consideră reacţia negativă.

În cazul unor sisteme cu reacţie parţială, de genul celui prezentat în figura 6.6, este necesar să se utilizeze încă doi parametri la apelarea funcţiei ***feedback***, iar sintaxa devine:

```
[A,B,C,D] = feedback(A1,B1,C1,D1,A2,B2,C2,D2,in1,out1)
```

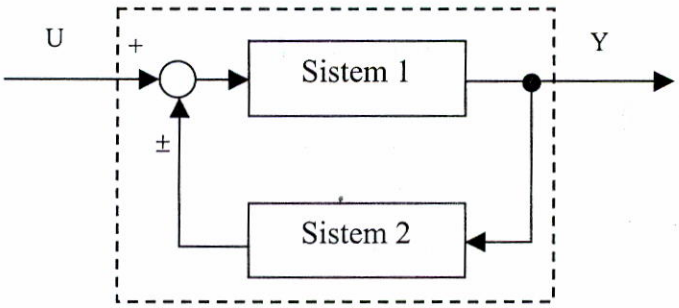

Figura 6.5. Conexiune cu reacție

Modelul returnat în acest caz este obținut considerând conectate toate ieșirile sistemului 2 la intrările sistemului 1 specificate în vectorul *in1* și respectiv ieșirile sistemului 1 specificate de *out1* cu toate intrările sistemului 2.

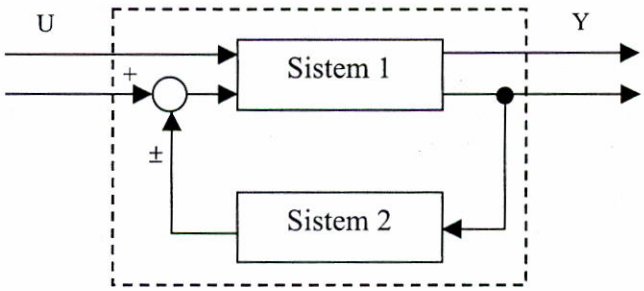

Figura 6.6. Conexiune cu reacție parțială a 2 sisteme

Implicit se consideră o reacție pozitivă, pentru a realiza o conexiune cu reacție negativă se pot folosi valori negative în cadrul vectorului *in1*.

În cazul unor sisteme cu reacție unitară, adică în situațiile în care avem:

$$H_2(s) = \pm 1,$$

se poate utiliza comanda ***cloop***, sintaxa acesteia fiind mai simplă decât a comenzii ***feedback*** datorită faptului că nu mai este necesară specificarea modelului matematic asociat reacției la apelul funcției:

```
[A,B,C,D] = cloop(A1,B1,C1,D1,sign)
[NUM,DEN] = cloop(NUM1,DEN1,sign)
[A,B,C,D] =cloop(A1,B1,C1,D1,out,in)
```

În primele două cazuri, sistemul echivalent are toate intrările și ieșirile la fel ca și sistemul în buclă deschisă (sistemul 1).

În ultimul caz doar ieșirile a căror indice este înscris în vectorul *out* se conectează la intrările corespunzătoare specificate de vectorul *în*. Implicit se consideră reacția unitară pozitivă, pentru o reacție negativă este necesară utilizarea valorilor negative în matricea *în*.

Exemplu:
Se consideră o structură de tipul celei din figura 6.5, unde subsistemele componente au funcţiile de transfer din relaţia (6.1). Analitic, expresia funcţiei de transfer echivalente se obţine aplicând formula:

$$H(s)=\frac{H_1(s)}{1\mp H_1(s)\cdot H_2(s)}=\frac{4s+5}{8s^2+10s\mp 3} \tag{6.4}$$

considerând reacţia negativă, funcţia de transfer se poate obţine cu ajutorul comenzii MATLAB:

```
>> [NUM,DEN] = feedback(1,[2 0],3,[4 5],-1)
```

6.5. Conexiune oarecare

Pentru a determina modelul matematic echivalent asociat unui sistem compus din mai multe subsisteme conectate între ele după o topologie oarecare, dar cunoscută a priori, trebuie urmat ***algoritmul*** prezentat în continuare:

PASUL 1: se completează schema cu blocuri fictive pentru generarea semnalelor de intrare în sistem şi apoi se numerotează subsistemele într-o ordine arbitrar aleasă.
PASUL 2: se defineşte numărul de blocuri din componenţa sistemului compus.
PASUL 3: se defineşte pentru fiecare bloc modelul matematic sub forma de f.d.t sau MM-ISI.
PASUL 4: se construieşte un model intermediar în spaţiul stărilor cu ajutorul comenzii ***blkbuild***.
PASUL 5: se specifică modul de interconectare al blocurilor prin intermediul unei matrici de interconexiuni q.
PASUL 6: se specifică intrările şi ieşirile sistemului echivalent (indicii asociaţi blocurilor de intrare şi respectiv ieşire) prin intermediul a doi vectori : *iu* şi respectiv *iy*.
PASUL 7: se determină un model matematic, în spaţiul stărilor, care ţine cont de topologia sistemului, cu ajutorul comenzii ***connect***.
PASUL 8: se aduce modelul obţinut la pasul anterior la forma minimală prin utilizarea comenzii ***minreal***.

Exemplu:
Se consideră sistemul de reglare automată (SRA) reprezentat în figura 6.7.

Modelele matematice ale celor patru subsisteme sunt funcţiile de transfer date în relaţiile 6.5:

$$H_{PC}(s)=\frac{5}{8s^2+10s+3},\quad H_{RG}(s)=5\left(1+\frac{1}{8s}\right),$$

$$H_{TR}(s)=3,\quad H_P(s)=3 \tag{6.5}$$

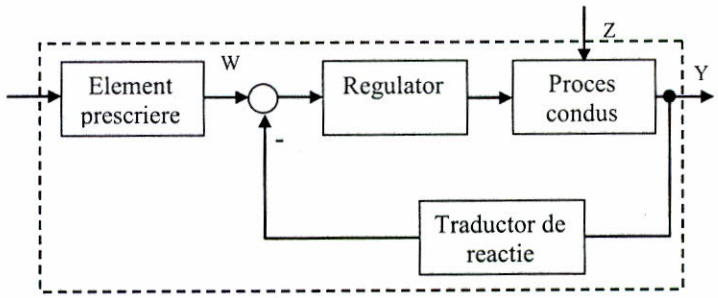

Figura 6.7. Schema bloc a unui sistem de reglare automată

Se dorește simularea structurii SRA pentru o mărime de referință (W) de tipul treaptă unitară în condițiile în care asupra sistemului acționează o perturbație (Z) de tipul impuls cu amplitudine unitară și durată 3 secunde.

În continuare se parcurge pas cu pas algoritmul prezentat anterior:

Pasul 1: se completează schema cu blocuri fictive pentru generarea semnalelor de intrare în sistem și apoi se numerotează subsistemele într-o ordine arbitrar aleasă, obținându-se schema din figura 6.8:

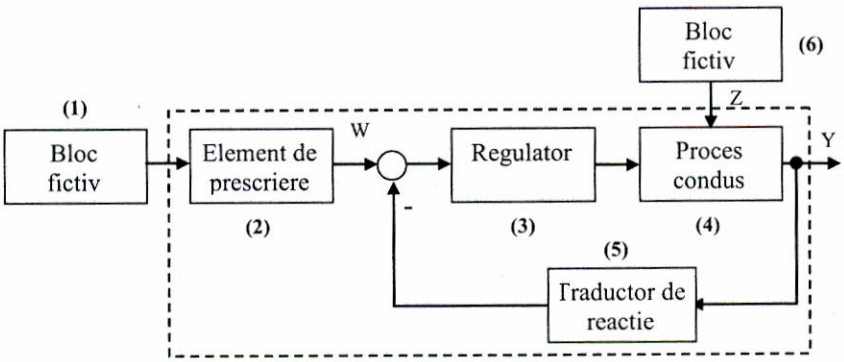

Figura 6.8. Schema bloc completată a unui sistem de reglare automată

Pasul 2: se definește numărul de blocuri din componența sistemului compus.

```
>> nblocks=6;
```

Pasul 3: se definește pentru fiecare bloc modelul matematic sub forma de f.d.t sau MM-ISI.

Aceste modele matematice se introduc fie sub forma de perechi (*ai,bi,ci,di*), fie perechi (*ni,di*), unde *i=1...nblocks*, după cum modelul disponibil este de forma MM-ISI sau f.d.t.

```
>> n1=1;
>> d1=1;
```

```
>> n2=3;
>> d2=1;
>> n3=[5*8 5];
>> d3=[8 0];
>> n4=5;
>> d4=[8 10 3];
>> n5=3;
>> d5=1;
>> n6=1;
>> d6=1;
```

Pasul 4:se construieşte un model intermediar în spaţiul stărilor cu ajutorul comenzii ***blkbuild***.

Este important de remarcat că deşi aparent apelul comenzii ***blkbuild*** este fără parametri de intrare şi respectiv fără returnare de parametri, comanda are nevoie de toţi parametrii definiţi la paşii 2 şi 3.

Din acest motiv, numele acestor parametri (respectiv variabile) este bine fixat. În urma executării acestei comenzi, modelul intermediar este returnat în variabile *a,b,c,d,* care dacă nu există în memorie sunt create automat de această comandă.

```
>> blkbuild;
```

Pasul 5:se specifică modul de interconectare al blocurilor prin intermediul unei matrici de interconexiuni *q*. Această matrice are un număr de linii egal cu numărul blocurilor din sistem şi un număr de coloane egal cu numărul maxim de intrări într-un bloc plus unu.

Matricea se completează pe linie, primul element fiind numărul de ordine al blocului curent, iar următoarele elemente sunt indicii blocurilor ale căror ieşiri sunt intrări pentru blocul curent, luate cu semn.

```
>> q=[1 0 0
2 1 0
3 2 -5
4 3 6
5 4 0
6 0 0];
```

Pasul 6: se specifică intrările şi ieşirile sistemului echivalent (indicii asociaţi blocurilor de intrare şi respectiv ieşire) prin intermediul a doi vectori : iu şi respectiv iy.

```
>> iu=[1 6];
>> iy=[4];
```

Pasul 7:se determină un model matematic, în spaţiul stărilor, care ţine cont de topologia sistemului, cu ajutorul comenzii ***connect***.

```
>> [A,B,C,D]=connect(a,b,c,d,q,iu,iy);
```

Pasul 8:se aduce modelul obținut în pasul anterior la forma minimală prin utilizarea comenzii ***minreal***.

Acest pas este opțional, se execută datorită faptului că modelul obținut la pasul anterior poate avea un număr de variabile de stare mai mare decât ordinul sistemului.

```
>> [Am,Bm,Cm,Dm]=minreal(A,B,C,D);
```

În continuare, simularea sistemului se face pe baza modelului obținut la Pasul 8, cu următoarea secvență de comenzi:

```
>> t=0.1:0.1:40;
>> r=ones(length(t),1);
>> p=[ones(length(0.1:0.1:3),1) ; zeros(length(3:0.1:40)-1,1)];
>> y=lsim(Am,Bm,Cm,Dm,[r p],t);
>> plot(t,p,'w-.',t,r,'w:',t,y,'w' )
```

Rezultatul acestei simulări, făcută pe o perioadă de 40 secunde cu un pas de 0.1 secunde, este prezentat în figura 6.9:

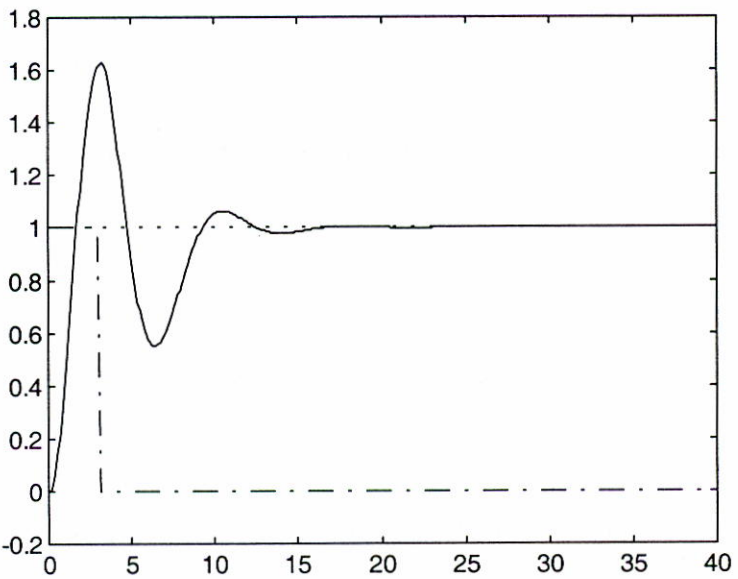

Figura 6.9. Răspunsul SRA (linie continuă), referința (linie întreruptă), perturbația (linie punct).

6.6. Sisteme extinse

Cu ajutorul comenzii ***append*** este posibil să se "extindă" starea unui sistem dat pe baza stării unui alt sistem.

Exemplu:
Se consideră două sisteme cu MM-ISI corespunzătoare ecuațiilor (6.6):

$$\begin{cases} \dot{x}_1 = A_1 x_1 + B_1 u_1 \\ y_1 = C_1 x_1 + D_1 u_1 \end{cases}, \quad \begin{cases} \dot{x}_2 = A_2 x_2 + B_2 u_2 \\ y_2 = C_2 x_2 + D_2 u_2 \end{cases} \tag{6.6}$$

comanda ***append***, apelată cu sintaxa de mai jos:

```
>> [A,B,C,D]=append(A1,B1,C1,D1,A2,B2,C2,D2);
```

furnizează MM-ISI asociat sistemului care surprinde dinamica celor două sisteme definite mai sus, model care este prezentat în relația:

$$\begin{cases} \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} A_1 & 0 \\ 0 & A_2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} B_1 & 0 \\ 0 & B_2 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \\ \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} C_1 & 0 \\ 0 & C_2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} D_1 & 0 \\ 0 & D_2 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \end{cases} \tag{6.7}$$

O particularizare a comenzii ***append*** este comanda ***augstate***.
Concret, comanda:

```
>> [Aa,Ba,Ca,Da]=augstate(A,B,C,D);
```

returnează modelul matematic asociat aceluiași sistem, în condițiile în care ieşirea sa este extinsă pe baza stării sistemului, adică:

$$\begin{cases} \dot{x} = Ax + Bu \\ \begin{pmatrix} y \\ x \end{pmatrix} = \begin{pmatrix} C \\ I \end{pmatrix} x + \begin{pmatrix} D \\ 0 \end{pmatrix} u \end{cases} \tag{6.8}$$

Această comandă poate fi utilă la modelarea sistemelor cu reacție după stare.

6.7. Probleme de studiat

1. Să se scrie un program de simulare a sistemului de reglare automată (SRA) reprezentat în figura 6.7 folosind instrucțiuni specifice tipurilor standard de conexiuni sistemice.
2. Să se scrie un program de simulare a unui sistem din cele prezentate în Anexa 3, folosind instrucțiuni specifice tipurilor standard de conexiuni sistemice.
3. Să se scrie un program de simulare a unui sistem din cele prezentate în Anexa 3, folosind algoritmul general de rezolvare a unei conexiuni sistemice oarecare.

Capitolul 7

INDICATORI DE CALITATE AI SRA, DEFINIȚI PE BAZA RĂSPUNSULUI INDICIAL

7.1. Considerații generale

Aprecierea modului în care un sistem de reglare automata (SRA) satisface cerințele de calitate ale reglării se face pe baza unor indicatori de calitate, care reprezintă mărimi prin intermediul cărora se apreciază cantitativ calitatea unui SRA. Ei caracterizează în principal precizia şi rapiditatea reglării.

Cel mai frecvent procedeu de apreciere a performanțelor este cel bazat pe analiza comportării SRA la o solicitare de tip treaptă (răspunsul indicial).

În principal se utilizează două categorii de indicatori de calitate, definiți pe baza:

- variației treaptă a mărimii de referință (prescriere);
- variației treaptă a mărimii perturbatoare.

Se consideră sistemul de reglare automată prezentat în figura 7.1.

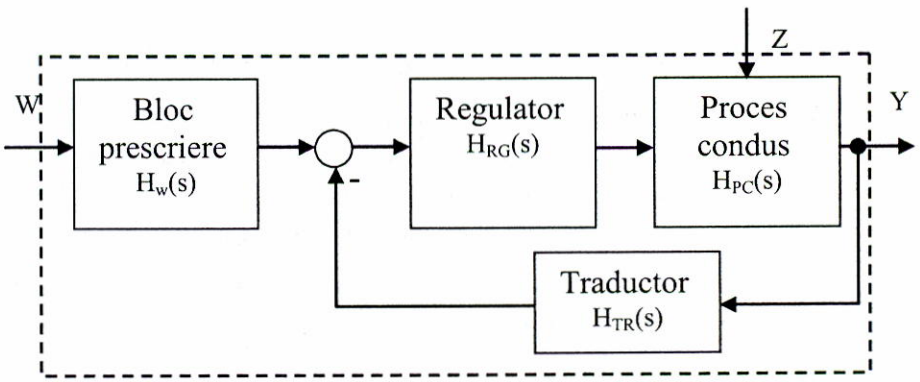

Figura 7.1. Schema bloc a unui sistem cu reglare automată

Aprecierea calității unui SRA se face pe baza acestor indicatori, prin compararea lor cu niște valori impuse inițial prin proiectare, o reglare considerându-se corespunzătoare atunci când sunt îndeplinite condiții de tip limitativ.

7.2. Indicatori de calitate definiți în raport cu variația treapta a mărimii de referință (prescriere)

În figura 7.2 este reprezentat răspunsul unui SRA la variația treaptă a mărimii de prescriere pe baza căruia se definesc următorii indicatori de calitate:

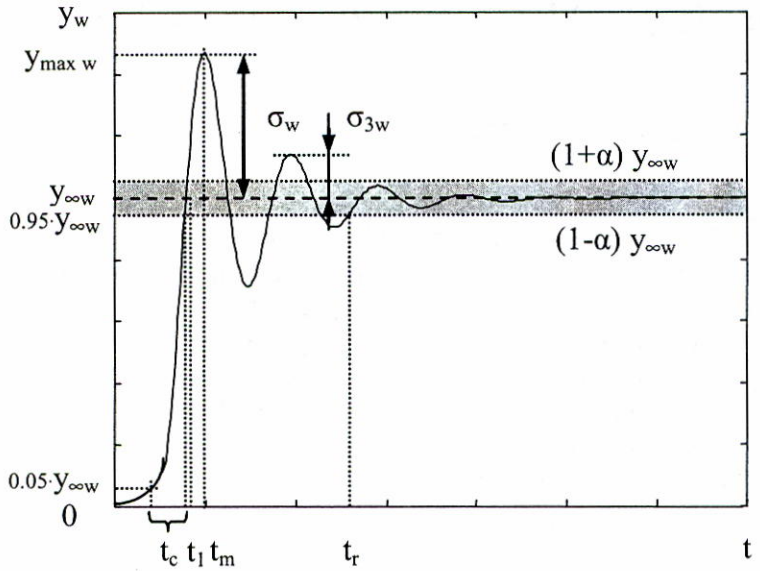

Figura 7.2. Răspunsul unui SRA la variația treapta a mărimii de prescriere

- **σ_w – suprareglajul** – în raport cu variația treaptă a mărimii de prescriere.
  Este un indicator de bază în analiza SRA şi se defineşte ca fiind abaterea maximă a răspunsului în regim tranzitoriu față de valoarea staționară $y_{\infty w}$ (în raport cu prescrierea) care se stabileşte după terminarea acestui regim:

$$\sigma_w = y_{max\ w} - y_{\infty w} \tag{7.1}$$

De regulă suprareglajul se exprimă procentual:

$$\sigma_w\,\% = \frac{\sigma_w}{y_{\infty w}} \cdot 100 = \frac{y_{max\ w} - y_{\infty w}}{y_{\infty w}} \cdot 100 \tag{7.2}$$

Având în vedere că depăşirile semnificative ale valorii staționare y_∞ pot conduce la suprasolicitări ale elementelor de execuție, sau ale procesului tehnologic reglat, valoarea suprareglajului trebuie limitată prin proiectare, printr-o condiție de tip limitativ

$$\sigma_w\,\% \le \sigma_{w\,lim}\,\% \tag{7.3}$$

unde $\sigma_{w\lim}\%$ reprezintă valoarea maximă admisibilă impusă suprareglajului în funcție de tipul şi condițiile solicitate de funcționare ale procesului tehnologic.

- **t_r – timpul de reglare** – numit şi *timp de răspuns* sau *durata regimului tranzitoriu* al SRA reprezintă timpul de la care răspunsul sistemului rămâne doar în zona de liniştire.

  În practică **zona de liniştire** se adoptă în general $\pm\,\alpha = \pm(2\% - 5\%)$ față de $y_{\infty w}$. Pentru ca procesul de reglare să se desfăşoare cu suficientă rapiditate, trebuie satisfăcută condiția limitativă :

$$t_r \le t_{r\lim} \tag{7.4}$$

  unde $t_{r\lim}$ este durata maxim admisibilă a regimului tranzitoriu.

- **t_1 – timpul de primă reglare** – reprezintă timpul în care mărimea reglată atinge pentru prima dată valoarea de stabilizare $y_{\infty w}$.

$$t_1 = \{ \min t \mid y(t) = y_{\infty w} \} \tag{7.5}$$

- **t_m – timpul primului extrem** (timpul atingerii primului maxim)

$$t_m = \{ \min t \mid \dot{y}(t) = 0,\ t = 0 \} \tag{7.6}$$

- **t_c – timpul de creştere** – reprezintă timpul în care răspunsul sistemului creşte de la valoarea $0{,}05y_{\infty w}$ la valoarea $0{,}95y_{\infty w}$. t_c, oferă deci un indiciu asupra vitezei de creştere a mărimii reglate pe parcursul primei oscilații, cea mai periculoasă de altfel pentru procesul tehnologic.

- **Φ – gradul de amortizare** – (indicele de oscilație), reprezentând variația relativă a amplitudinilor primelor două oscilații de polaritate pozitivă în raport cu $y_{\infty w}$:

$$\Phi = \frac{\sigma_w - \sigma_{3w}}{\sigma_w} = 1 - \frac{\sigma_{3w}}{\sigma_w} = 1 - \delta \tag{7.7}$$

  unde $\delta = \sigma_{3w}/\sigma_w$ este *decrementul oscilațiilor*

Exemplu

În continuare se prezintă o variantă propusă de program MATLAB care calculează o serie de indicatori de calitate, definiți în raport cu referința, pentru un SRA cu structura similară celei prezentate în figura 7.1.

Modelele matematice asociate subsistemelor componente sunt date de relațiile:

$$H_{PC}(s) = \frac{5}{8s^2 + 10s + 3},\quad H_{RG}(s) = 5\left(1 + \frac{1}{8s}\right),$$
$$H_T(s) = 3,\quad H_w(s) = 3 \tag{7.11}$$

Simularea sistemului se face utilizând algoritmul prezentat în capitolul anterior.

```
%PROGRAM PENTRU SIMULAREA FUNCŢIONARII UNUI SISTEM
%DE REGLARE AUTOMATA ŞI DETERMINAREA UNOR INDICATORI
%DE CALITATE DEFINIŢI PE BAZA RĂSPUNSULUI INDICIAL

nblocks=6;
n1=1;d1=1;n2=3;d2=1;n3=[5*8 5];d3=[8 0];
n4=5;d4=[8 10 3];n5=3;d5=1;n6=1;d6=1;
blkbuild;
q=[1 0 0;2 1 0;3 2 -5;4 3 6;5 4 0;6 0 0];
iu=[1 6];iy=[4];
[A,B,C,D]=connect(a,b,c,d,q,iu,iy);
[Am,Bm,Cm,Dm]=minreal(A,B,C,D);
[y,x,timp]=step(Am,Bm,Cm,Dm,1);
plot(timp,y,'r'),grid,title('Raspuns indicial al SRA')

%URMEAZA CALCULUL INICATORILOR DE CALITATE

n=length(timp);
for j=1:n
    if y(j)==max(y)
        disp('TIMPUL PRIMULUI MAXIM: '),t1m=timp(j)
        jmax=j;
    end
end
disp('SUPRAREGLAJUL ESTE: '), f=max(y)-y(n)
disp('SUPRAREGLAJ ÎN %: '), fp=f*100/y(n)
for i=1:jmax
    if y(i+1)>=0.95*y(n) & y(i)<=0.95*y(n)
        disp('TIMPUL DE PRIMA REGLARE:'),t1=timp(i+1)
    end
end
k=0;
for i=1:n-1
    if y(i+1)<=y(n) & y(i)>=y(n) & k= =0
        jmax2=i;k=1;
    end
end
for i=jmax2:n
    v(i)=y(i);
end
ga=1-(max(v)-y(n))/(max(y)-y(n));
disp('GRADUL DE AMORTIZARE: ');ga
```

7.3. Indicatori de calitate definiți în raport cu variația treapta a mărimii de perturbație

O altă categorie de indicatori de calitate sunt indicatorii definiți în raport cu variația treaptă a mărimii perturbatoare. În acest caz, sistemul se consideră adus intr-o stare inițială staționară, valoarea staționară a ieșirii sistemului fiind egală cu $y_{\infty w}$, după care, păstrând referința neschimbată (constantă) se aplică o treaptă unitară pe intrarea perturbatoare (z). Răspunsul sistemului în aceste condiții este reprezentat grafic în figura 7.3.

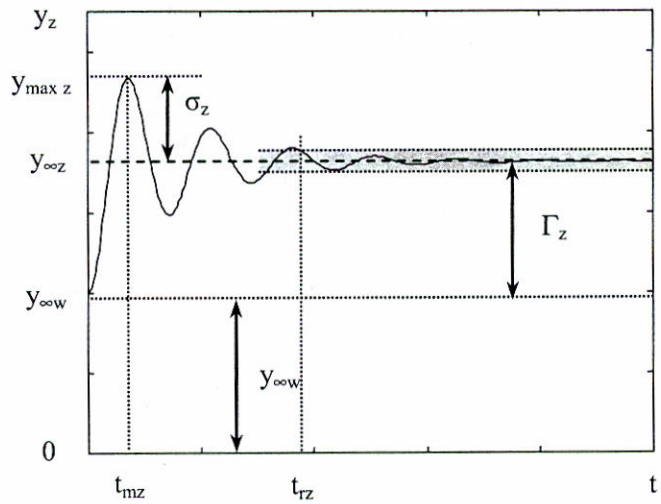

Figura 7.3. Răspunsul unui SRA la o variație treaptă a mărimii perturbatoare

În mod asemănător definirii indicatorilor din paragraful precedent, se definesc următorii indicatori raportați la perturbație:

- **Γ_z – statismul** în raport cu perturbația considerată, reprezintă variația staționară a mărimii reglate (ieșirea sistemului), sub acțiunea perturbației, în condițiile păstrării constante a mărimii de conducere (prescriere):

$$\Gamma_z = y_{\infty z} - y_{\infty w}\ ,\ w = \text{constant} \tag{7.8}$$

- **σ_z – suprareglajul** – abaterea maximă în raport cu mărimea perturbatoare;

$$\sigma_z = y_{max\,z} - y_{\infty z} \tag{7.9}$$

- **t_{mz} – timpul atingerii suprareglajului,** momentul atingerii lui σ_z;
- **y_{zr} – abaterea reziduală maximă:**

$$y_{zr} = \max\{\, y_z(t) - y_{\infty z} \mid t \geq t_{rz} \,\} \tag{7.10}$$

7.4. Probleme de studiat

1. Să se simuleze funcţionarea sistemului de reglare automată a cărui structură este prezentată în figura 7.4 şi apoi să se calculeze principalii indicatori de calitate aferenţi, definiţi în raport cu referinţa.

   Se consideră următoarele funcţii de transfer asociate blocurilor componente:

   $$H_{PC}(s)=\frac{1}{s^2+2s+3},\quad H_{RG}(s)=\frac{3}{0.01s+3}$$

2. Să se completeze programul din paragraful anterior cu o secvenţă de comenzi pentru calculul timpului de reglare şi a timpului de creştere.

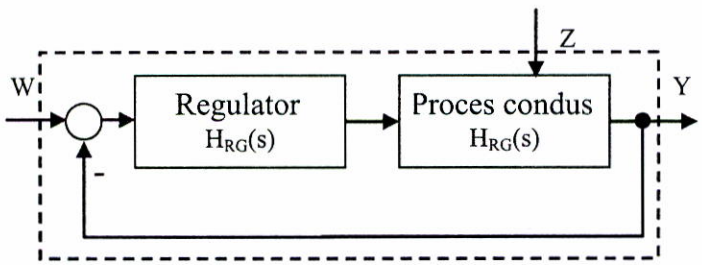

Figura 7.4. Schema bloc simplificată a unui sistem cu reglare automată cu reacţie unitară

3. Să se scrie un program MATLAB pentru calculul indicatorilor de calitate definiţi în raport cu variaţia treaptă a perturbaţiei.

Capitolul 9

UTILIZAREA MEDIULUI MATLAB ÎN IDENTIFICAREA ŞI ESTIMAREA PARAMETRILOR SISTEMELOR

9.1. Consideraţii privind identificarea sistemelor

Problema identificării poate fi privită ca problemă inversă analizei sistemelor, şi anume, fiind cunoscute semnalele din intrare şi ieşire trebuie determinat *modelul matematic* (MM) care descrie comportarea sistemului (Figura 9.1). Modelul matematic obţinut poate fi ecuaţia diferenţială / ecuaţia recursivă, funcţia pondere / secvenţă de ponderare, funcţia de transfer / funcţia de transfer discretă ş.a.m.d.

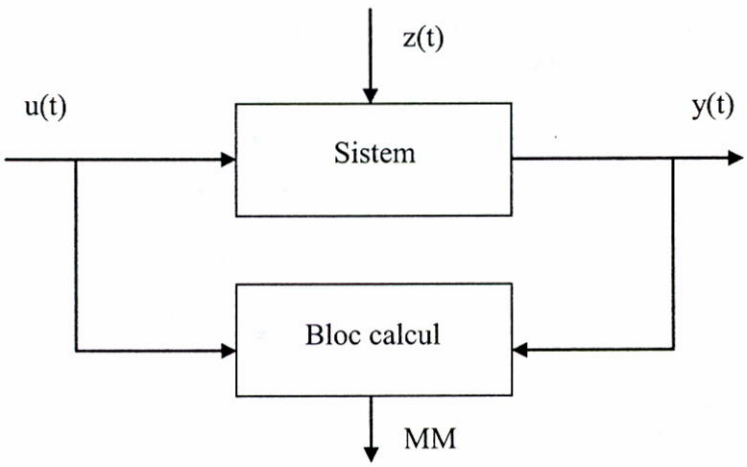

Figura 9.1. Schema bloc după care se desfăşoară un proces de identificare

Identificarea poate fi definită [Zad62] ca: *determinarea, pe baza intrării şi ieşirii, a unui sistem dintr-o clasă determinată de sisteme, faţă de care sistemul care se încearcă este echivalent.*

În aceasta definiţie elementele clasei de sisteme sunt modelele.

În cele mai multe din cazurile practice, din cunoaşterea parţială a funcţionării unui proces se poate dispune de o anumită cantitate de cunoştinţe apriorice care să faciliteze fixarea structurii modelului, ceea ce mai rămâne de făcut este determinarea valorilor numerice ale parametrilor, deci practic problema identificării se reduce la problema *estimării parametrilor.*

Identificarea trebuie abordată întotdeauna în corelație cu scopul final, care ar putea fi de exemplu: analiza comportării dinamice a sistemului sau proiectarea unui sistem de reglare convențional sau sinteza unei strategii de conducere adaptivă ş.a.m.d.

Este evident că modelul necesar este altul pentru fiecare dintre cazuri (şi alta poate fi şi precizia cerută modelului).

Procesul de determinare a modelului poate fi surprins algoritmic conform Figurii 9.2.

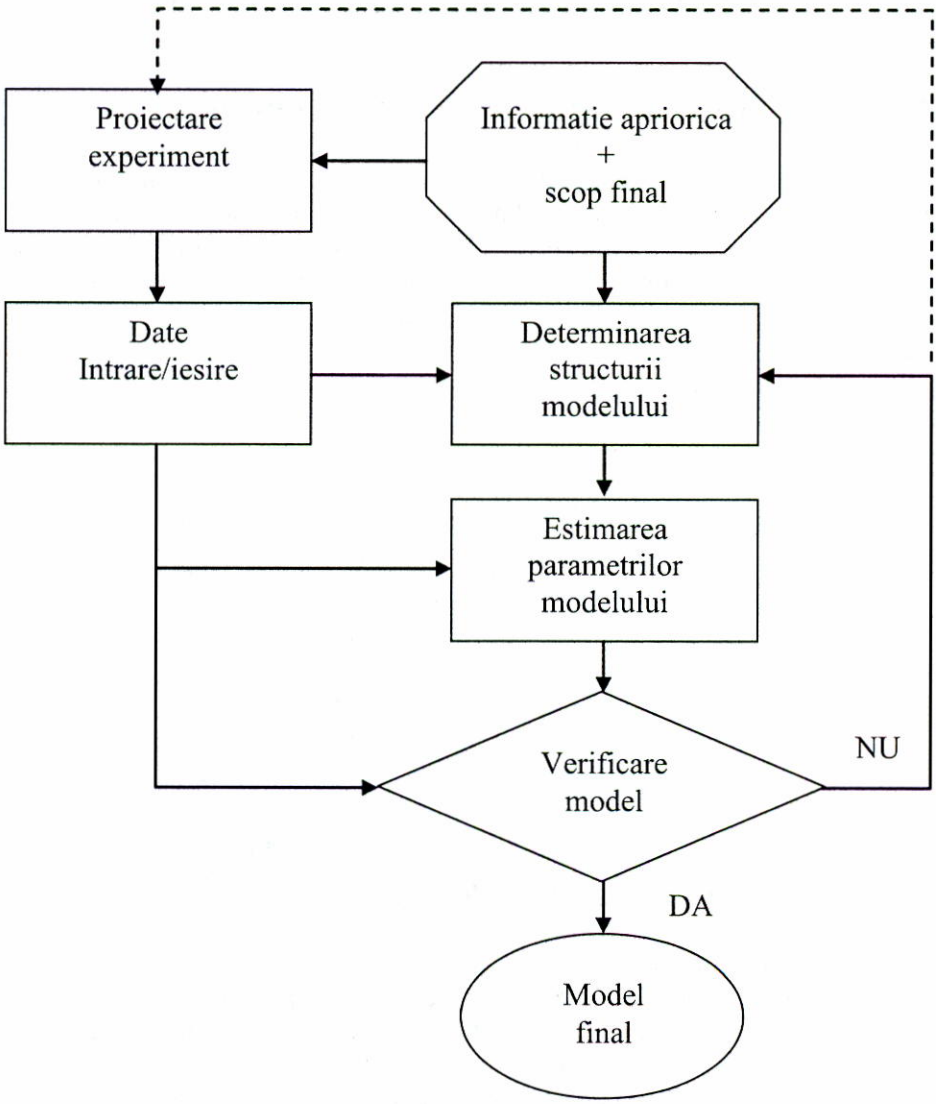

Figura 9.2. Algoritm general de identificare

Prin proiectarea experimentului se urmăreşte alegerea celui mai „optim" semnal de testare.

Pentru verificarea modelului se pot efectua mai multe tipuri de teste, în cazul celui mai frecvent comparându-se ieşirea modelului cu ieşirea sistemului de identificat, pentru acelaşi tip de semnal aplicat în intrare.

În principal, procedurile de identificare se împart în metode active, şi respectiv metode pasive de identificare.

- *Metodele active de identificare* presupun aplicarea unor semnale de test, urmărindu-se obţinerea unor informaţii care, fără un efort de calcul deosebit, sa furnizeze modelul căutat. În general, printr-o metodă activă se determină un model neparametric (funcţia pondere, răspunsul indicial, caracteristici de frecvenţă) punându-se problema, dacă este nevoie, de transformare într-un model parametric (ecuaţia diferenţială, funcţia de transfer etc.) necesare în general în probleme de sinteză.

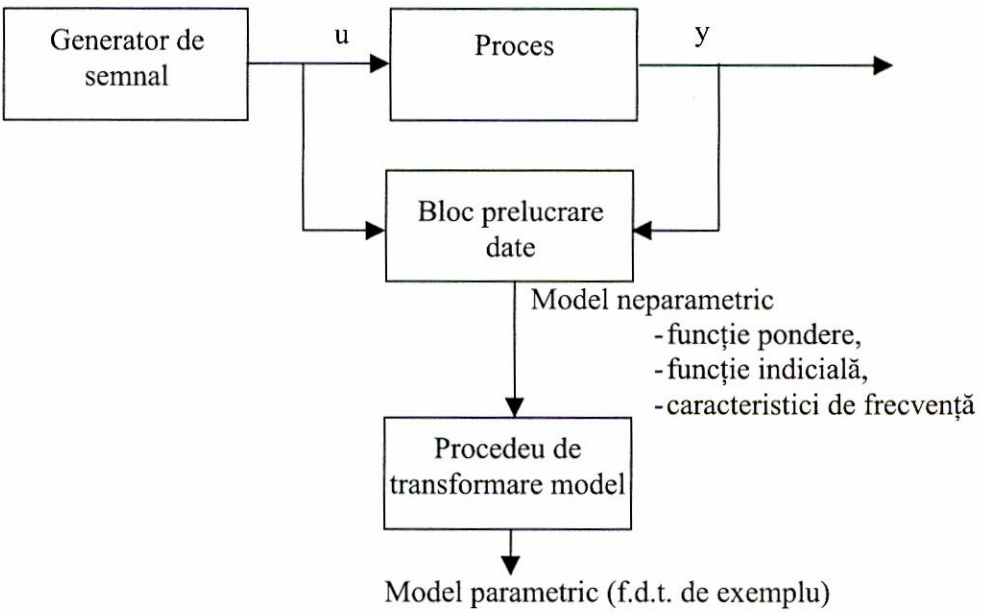

Figura 9.3. Identificare utilizând metode active

- *Metodele pasive* utilizează variaţiile aleatoare ale mărimilor procesului în funcţionarea lui normală. Ele conduc aproape întotdeauna la obţinerea de modele parametrice (Figura 9.4)

Pentru a putea aborda şi rezolva probleme specifice identificării şi estimării parametrilor sistemelor, mediul Matlab dispune de un toolbox specializat, numit **System Identification Toolbox**.

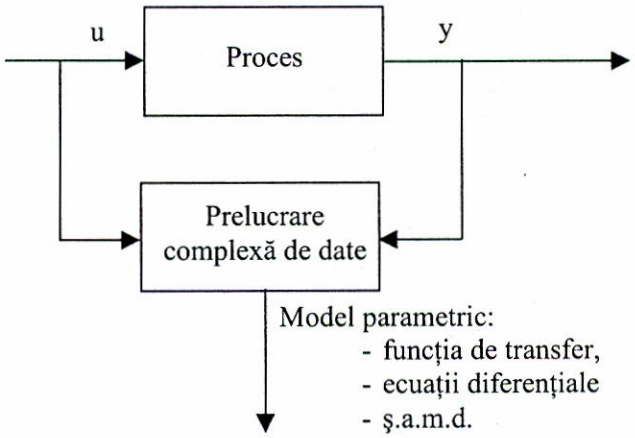

Figura 9.4. Identificare utilizând metode pasive

Capitolul 10

CARACTERISTICI ALE SEMNALELOR (PROCESELOR) STOCHASTICE (ALEATOARE) UTILIZATE ÎN PRELUCRAREA SEMNALELOR, RESPECTIV IDENTIFICAREA SISTEMELOR

10.1. Semnale (procese) stochastice (aleatoare)

Un semnal (proces) stochastic (aleator) reprezintă un proces care se desfăşoară în timp şi este guvernat, cel puţin în parte de legi probabilistice.

Din punct de vedere matematic, un semnal aleator este o funcţie de două variabile $X(k,t)$, unde k ia valori în spaţiul realizărilor (eşantioanelor) posibile, iar t ia valori pe axa reală a timpului.

Pentru fiecare k se obţine o realizare particulară a procesului stochastic (o funcţie particulară) $X(k,t) = X_k(t)$, numită şi **funcţie eşantion** sau **realizare**, care este în fapt rezultatul unui experiment. Experimentele realizate asupra unui proces aleatoriu sunt puse de regulă în evidenţă prin înregistrări, o înregistrare constituind **o realizare** a procesului aleator. Un număr finit, însă suficient de mare, de eşantioane (realizări) constituie **spaţiul eşantioanelor (realizărilor)**.

În ultima analiză, procesele aleatoare sunt funcţii de timp. Prin fixarea variabilei independente t, la o valoare particulară t_1, se obţine un ansamblu de valori, $x_i(t_1)$, $i = \overline{1,k}$, care constituie o mărime (variabilă) aleatoare $X(t_1)$ (poate să ia diverse valori cu diverse probabilităţi).

În consecinţă, procesele aleatoare pot fi caracterizate prin două tipuri distincte de caracteristici:

- ***Caracteristici de ansamblu*** (la momentul t) – reprezintă caracteristicile unei variabile aleatoare care este compusă din ansamblul realizărilor posibile ale procesului la momentul dat t.
- ***Caracteristici de timp*** (temporale) – constituie caracteristicile unei variabile aleatoare care reprezintă o realizare particulară a procesului studiat.

Se numesc procese stochastice ***staţionare*** – procesele a căror proprietăţi statistice sunt invariante la schimbarea arbitrară a originii timpului. Una dintre consecinţe este ca pentru acest tip de procese *valorile medii sunt constante.*

Procesele stochastice staţionare, pentru care valorile medii de ansamblu (obţinute pe ansamblul realizărilor unui proces stochastic) sunt egale cu mediile temporale (ale unei singure realizări) se numesc ***procese ergodice.***

Proprietatea de ergodicitate permite deci să se înlocuiască un ansamblu statistic de eşantioane cu un singur eşantion, considerat reprezentativ.

Importanța practică a acestei proprietăți, verificată de altfel de procesele stochastice din realitate este esențială, deoarece permite utilizarea relațiilor de calcul a mediilor temporale de diferite ordine în locul celor de calcul al mediilor statistice de ansamblu, care sunt de multe ori inoperante.

10.2. Principalele caracteristici utilizate în studiul şi prelucrarea semnalelor precum şi în estimarea parametrilor sistemelor

În continuare se prezintă relațiile de calcul ale principalelor caracteristici utilizate în studiul şi prelucrarea semnalelor stochastice, precum şi în problemele referitoare la estimarea parametrilor (stărilor) sistemelor.

De asemenea se vor considera comenzile mediului Matlab dedicate calculului acestor mărimi.

Notă: Având în vedere că se consideră verificată ipoteza de ergodicitate, se vor prezenta doar relațiile de calcul referitoare la caracteristicile de timp (temporale), în varianta discretă.

- **Media** (temporală) sau **speranța matematică** (Media de ordinul 1)

$$E[X] = m_x = \frac{1}{N}\sum_{t=1}^{N} x(t) \tag{10.1}$$

unde t - timp discret normalizat, (t=0,1,2...)

$E[.]$ - operatorul de mediere

X - semnal discret (secvența de valori numerice)

$x(t)$ - valoarea secvenței pentru momentul t

N - lungimea secvenței

Media poate fi privita ca "centru de greutate" al distribuției valorilor posibile ale procesului (semnalului).

- **Valoarea medie pătratică** (Media de ordinul II)

$$E\left[x^2(t)\right] = \frac{1}{N}\sum_{t=1}^{N} x^2(t) \tag{10.2}$$

Procesul stochastic centrat se definește ca:

$$\overset{0}{x}(t) = x(t) - m_x \tag{10.3}$$

În consecință:

$$E\,\overset{0}{x}(t) = 0$$

Notă: Operația de centrare se justifică prin aceea că, în general, interesează fluctuațiile față de valoarea medie şi nu față de originea (arbitrară).

- Valoarea medie pătratică a variabilei centrate defineşte **dispersia** sau **varianta** σ_x^2:

$$\sigma_x^2 = E\overset{0}{x}(t) = \frac{1}{N}\sum_{t=1}^{N}(x(t)-m_x)^2 = \frac{1}{N}\sum_{t=1}^{N}\overset{0}{x}^2(t) \tag{10.4}$$

- **Abaterea medie pătratică** sau **abaterea standard** se defineşte ca:

$$\sigma_x = \sqrt{\sigma_x^2} = \sqrt{\frac{1}{N}(x(t)-m_x)^2} \tag{10.5}$$

- **Funcţia de autocorelaţie** (temporală) se defineşte prin relaţia:

$$R_x(k) = E[x(t)x(t+k)] = \frac{1}{N}\sum_{t=1}^{N}x(t)x(t+k)\,,\quad t=0,1,2,\ldots \tag{10.6}$$

- **Funcţia de intercorelaţie** (temporală) se defineşte similar, dar relativ la două procese stochastice care evoluează în paralel:

$$R_{xy}(k) = E[x(t)y(t+k)] = \frac{1}{N}\sum_{t=1}^{N}x(t)y(t+k)\,,\quad t=0,1,2,\ldots \tag{10.7}$$

- **Funcţiile de autocovarianţă şi intercovarianţă**

Se definesc analog funcţiilor de auto/intercorelaţie dar, în acest caz, sunt implicate procesele stochastice centrate.

Funcţia de autocovarianţă:

$$C_x(k) = E\left[\overset{0}{x}(t)\overset{0}{x}(t+k)\right] \tag{10.8}$$

Se observă că:

$$C_x(0) = \sigma_x^2$$

Se poate demonstra cu uşurinţă relaţia:

$$C_x(k) = R_x(k) - m_x^2$$

Funcţia de intercovarianţă este definită ca:

$$Cxy(k) = \left[E\overset{0}{x}(t)\overset{0}{y}(t+k)\right] \tag{10.9}$$

În acest caz:

$$C_{xy}(k) = R_{xy}(k) - m_x m_y$$

Sensul fizic al autocorelaţiei (autocovarianţei) trebuie căutat în influenţa valorilor trecute ale procesului asupra valorilor de la un moment de timp considerat, exprima deci o legătură internă, intrinsecă a procesului însuşi şi care, în final, se reduce la un aspect energetic.

Funcţiile de intercorelaţie (intercovarianţă) a doua procese aleatoare, care au o evoluţie în paralel şi se influenţează reciproc sunt legate şi ele de energia mutuală de interacţiune dintre ele.

Notă: În cazul variabilelor aleatoare relaţiile de calcul sunt asemănătoare celor prezentate pentru procese aleatoare, dar în acest caz se operează cu secvenţe de valori care nu sunt dependente de timp.

În continuare vor fi prezentate **comenzile mediului Matlab** dedicate calculului mărimilor anterior prezentate.

❑ **Media (speranţa matematică)**

Comanda Matlab care calculează aceasta mărime este: ***mean***. Sintaxa acesteia permite două forme de apel:

```
- mean(X),
- mean(X,DIM)
```

Dacă argumentul este un vector, atunci *mean(X)* returnează valoarea medie a elementelor din vectorul X.

Dacă X este o matrice, atunci *mean(X)* va returna un vector linie, care conţine valorile medii ale elementelor matricei X, calculate pe fiecare coloană în parte.

A doua formă de apelare permite specificarea dimensiunii. Astfel, *mean(X,DIM)* calculează media de a lungul dimensiunii DIM a lui X.

Exemplu:

Fie vectorul X, introdus cu ajutorul comenzii:

```
>> X = [0 1 2
     3 4 5]
```

Comanda:

```
>> mean(X,1)
```

returnează [1.5 2.5 3.5] , iar

```
>> mean(X,2)
```

returnează

```
[1
4]
```

❑ **Mărimea aleatoare centrată**

Nu există o comandă dedicată pentru calculul acestei mărimi, dar evident ea poate fi calculată utilizând comanda:

```
>> X0=X-mean(X)
```

❑ **Dispersia sau varianţa**

Comanda Matlab care calculează această mărime este ***var***. Sintaxa acestei comenzi permite trei forme de apel:

```
- var(X),
- var(X,1),
- var(X,w).
```

În mod analog comenzii *mean*, dacă argumentul X este un vector, atunci *var(X)* returnează varianța acestuia, iar dacă X este o matrice, atunci rezultatul returnat este un vector linie care conține varianța fiecărei coloane din matricea X.

VAR(X) realizează împărțirea la *N-1* (unde N este lungimea secvenței) în cadrul relației (10.4) de definire a variației. Aceasta face ca *VAR(X)* să ofere cea mai bună estimare a variației în cazul în care X are o distribuție normală.

VAR(X,1) realizează împărțirea variației la N.

VAR(X,W) calculează varianța folosind vectorul de ponderare W. Numărul elementelor vectorului W trebuie să fie egal cu numărul de linii ale lui X, excepție făcând cazul $W = 1$ care este interpretat ca o formă scurta de apel pentru vectorul unitate. Elementele vectorului W trebuie să fie pozitive, iar comanda *var* realizează împărțirea fiecărui element din W la suma tuturor elementelor vectorului.

❑ **Abaterea medie pătratică sau deviația standard**

Deviația standard este rădăcina pătrată a dispersiei.

Comanda corespunzătoare ei este ***std***, permițând trei forme de apel:

```
- std(X),
- std(X,flag),
- std(X,flag,dim).
```

Deviația standard se poate defini în două moduri, corespunzătoare celor două forme prezentate în cadrul comenzilor Matlab referitoare la varianță, și anume:

$$std = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - m_x)^2}\,, \tag{10.10}$$

sau

$$std = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - m_x)^2}\,, \tag{10.11}$$

unde N este numărul de eșantioane, iar $m_x = \frac{1}{N}\sum_{i=1}^{N} x_i$ este valoarea medie a vectorului X.

Apelul:

```
>> s = std(X)
```

unde X este un vector, returnează deviația standard pe baza relației (10.10), iar un apel de forma:

```
>> s = std(X,1)
```

va calcula deviația standard folosind relația (10.11).

Dacă X este o matrice, atunci *std(X)* returnează un vector linie conținând deviațiile standard calculate pe fiecare coloană a matricii X.

Forma de apel: *s = std(X,flag)* în cazul în care *flag = 0*, conduce la acelaşi rezultat ca şi *std(X)*. Cazul *flag = 1* este identic cu *s=std(X,1)*.

s = std(X,flag,dim) calculează deviaţia standard a matricii *X* de-a lungul dimensiunii specificata de scalarul *dim*.

Exemplu:

Fie matricea *X*, definită conform comenzii:

```
>> X = [1     5     9
        7    15    22]
```

Comenzile :

```
>> s1 = std(X,0,1)
>> s 2= std(X,0,2)
```

vor conduce la:

```
s1 =
    4.2426    7.0711    9.1924
s2 =
     4.000
     7.5056
```

❑ **Covarianţa şi autocovarianţa**

Comanda Matlab care permite calculul acestor mărimi este ***cov***, iar sintaxa acesteia permite două forme de apel:

```
- cov(X),
- cov(X,Y).
```

C = cov(X), unde X este un vector, returnează varianţa elementelor vectorului.

În cazul matricilor, unde fiecare linie reprezintă o observaţie, iar fiecare coloană este o variabilă aleatoare, *cov(X)* este **matricea de covarianţă**.

Vectorul *diag(cov(X))* conţine varianţa pentru fiecare coloană, iar *sqrt(diag(cov(X)))* va conţine deviaţiile standard.

Dacă *x* şi *y* sunt vectori, atunci *cov(X,Y)* este echivalent cu *cov([X Y])*.

Notă: comanda *cov* calculează media fiecărei coloane a matricii înainte de a calcula rezultatul final.

Exemplu:

Considerând o matrice, notată cu *A*, de forma:

```
>> A = [-1 1 2 ; -2 3 1 ; 4 0 3]
```

se poate obţine un vector ce conţine varianţa pentru fiecare coloană a matricii A, cu ajutorul comenzii:

```
>> v = diag(cov(A))'
```

Rezultatul va fi afişat sub forma:

```
v =
   10.3333    2.3333    1.0000
```

Comparând vectorul v cu matricea de covarianţă *C*, dată de comanda:

```
>> C = cov(A);
C =
    10.3333   -4.1667    3.0000
    -4.1667    2.3333   -1.5000
     3.0000   -1.5000    1.0000
```

se poate observa că elementele de pe diagonala principală, *C(i,i)*, reprezintă varianţa coloanelor matricii *A*. Celelalte elemente *C(i,i)*, unde $i \neq j$, reprezintă covarianţa dintre coloanele *i* şi *j* ale matricii *A*.

❑ Intercorelaţia şi autocorelaţia

Comanda Matlab care permite calculul intercorelaţiei sau autocorelaţiei, este ***xcorr***, iar sintaxa acesteia permite următoarele forme de apel:

```
- xcorr(X,Y),
- xcorr(X),
- xcorr(X,Y,'option'),
- xcorr(X,'option'),
- xcorr(X,Y,maxlags),
- xcorr(X,maxlags),
- xcorr(X,Y,maxlags,'option'),
- xcorr(X,maxlags,'option'),
- [R,lags] = xcorr(...).
```

R = xcorr(x,y) returnează secvenţa de intercorelaţie de lungime 2*N-1, unde x şi y sunt 2 vectori de lungime N (N>1). Dacă x şi y au lungimi diferite, atunci vectorul de lungime mai mică este completat cu zerouri până ajunge la dimensiunea celuilalt.

În mod implicit, comanda *xcorr* calculează intercorelaţia fără a face împărţirea la N.

$$R_{xy}(m) = \begin{cases} \sum_{i=0}^{N-i-1} x_{i+m} y_i & m \geq 0 \\ R_{yx}(-m) & m < 0 \end{cases} \qquad (10.12)$$

Elementele vectorului *R* respectă relaţia: *R(i) = R(i-N)*, unde *i=1, ..., 2*N-1*

Calculul funcţiei de corelaţie impune însă în general efectuarea unei împărţiri în scopul obţinerii unei mai mari acurateţi a rezultatului.

Pentru a calcula secvenţa de autocorelaţie a vectorului *X* se poate folosi comanda: *R = xcorr(X)*. Dacă x este o matrice de dimensiune *N* x *P*, atunci R va fi la rândul său o matrice cu *2*N-1* linii, a căror coloane, în număr de P^2, conţin secvenţele de intercorelaţie calculate pentru toate combinaţiile posibile dintre coloanele lui *x*.

La apelul comenzii de calcul a corelației se poate specifica dacă se face o împărțire a rezultatului, şi ce tip de împărțire se aplică. Acest lucru se realizează cu sintaxa: *R = xcorr(X,Y,'optiune')*, unde 'optiune' poate fi:

- **'biased'**: caz care implementează următoarea relație de calcul a corelației:

$$R_{xy,biased}(m) = \frac{1}{N} R_{xy}(m)$$

- **'unbiased'**: caz care implementează următoarea relație de calcul a corelației

$$R_{xy,unbiased}(m) = \frac{1}{N-|m|} R_{xy}(m)$$

- **'coeff'**: Realizează normalizarea secvenței astfel încât, la momentul 0, aceasta să aibă valoarea 1.0
- **'none'**, calculează corelația fără a face o normalizare a acesteia (implicit)

Exemplu:
Se consideră două secvențe: x1 – secvență cu valori linear crescătoare şi respectiv x2 – secvența aleatoare cu distribuție uniformă. Programul care calculează şi afişează funcțiile de autocorelație corespunzătoare celor două secvențe x1, respectiv x2, considerând cele 4 tipuri de normalizare implementate în Matlab, este următorul:

```
X1=1:10
X2=rand(1,10)
Rb1=xcorr(X1,X1,'biased');
Ru1=xcorr(X1,X1,'unbiased');
Rc1=xcorr(X1,X1,'coeff');
Rn1=xcorr(X1,X1,'none');
Rb2=xcorr(X2,X2,'biased');
Ru2=xcorr(X2,X2,'unbiased');
Rc2=xcorr(X2,X2,'coeff');
Rn2=xcorr(X2,X2,'none');
figure
semilogy(Rb1)
hold on
semilogy(Ru1,':')
semilogy(Rc1,'--')
semilogy(Rn1,'-.')
figure
semilogy(Rb2)
hold on
semilogy(Ru2,':')
semilogy(Rc2,'--')
semilogy(Rn2,'-.')
```

În figurile 10.1.a. şi respectiv 10.1.b. se pot observa graficele funcțiilor de autocorelație calculate pentru două secvențe, x1 şi respectiv x2, considerând cele 4 tipuri de normalizare implementate în Matlab. Pe abscisă s-a reprezentat defazajul, iar pe ordonată s-a reprezentat logaritmul funcției de autocorelație.

În mod analog, sintaxa *R = xcorr(x,'option')* permite calculul autocorelației cu specificarea modului în care se va face normalizarea rezultatului. Opțiunile permise sunt aceleași ca cele prezentate mai sus, în cazul intercorelației.

Pentru a calcula intercorelația pentru un domeniu de variație a decalajului m între limitele [-maxlags:maxlags], se poate folosi sintaxa *R = xcorr(x,y,maxlags)*. În acest caz, vectorul intercorelației are dimensiunea 2*maxlags+1.

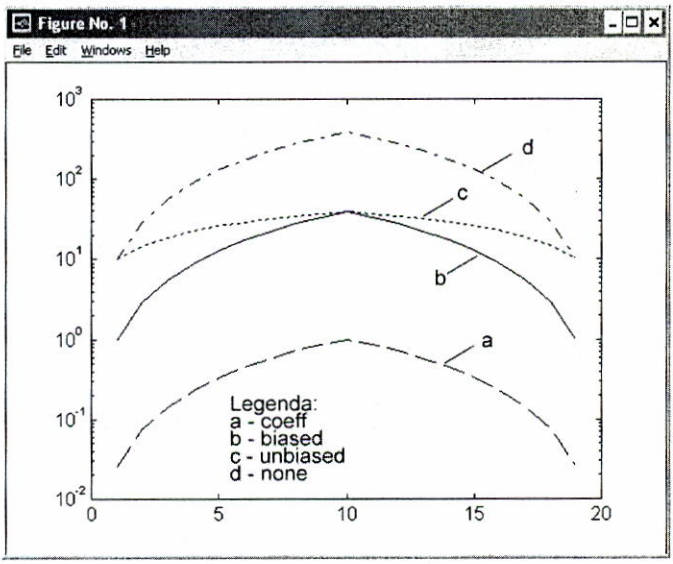

Figura 10.1.a. Autocorelația calculată pentru secvența x1

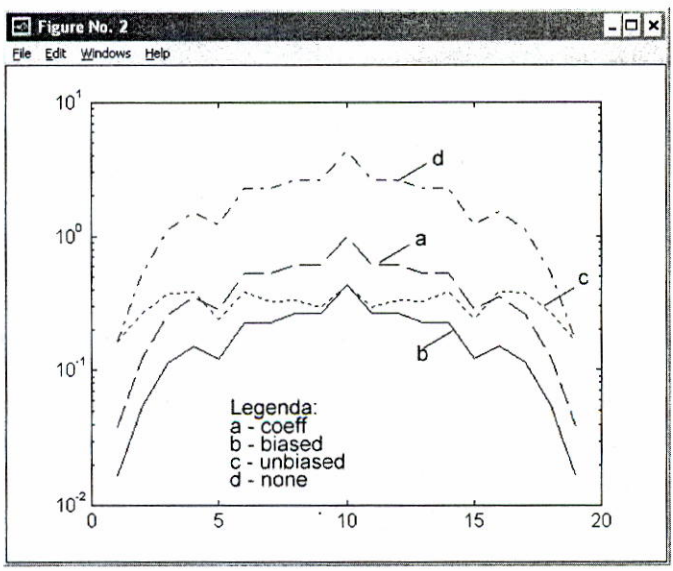

Figura 10.1.b. Autocorelația calculată pentru secvența x2

Autocorelaţia poate fi de asemenea calculată pe un domeniu de variaţie a decalajului, cu ajutorul comenzii: *R = xcorr(x,maxlags)*, caz în care rezultatul va avea forma unui vector de dimensiune *2*maxlags+1*. Dacă *x* este o matrice de dimensiune *N* x *P*, atunci rezultatul, *R*, va fi tot o matrice având *2*maxlags+1* linii, a căror P^2 coloane vor conţine secvenţele de autocorelaţie pentru toate combinaţiile posibile formate din coloanele matricii x.

Sintaxa: *R = xcorr(x,y,maxlags,'option')*, permite specificarea atât a intervalului maxim pe care variază decalajul, cât şi a tipului de normalizare a intercorelaţiei.

În mod similar, folosind sintaxa: *R = xcorr(x,maxlags,'option')*, se poate specifica atât intervalul maxim pe care variază decalajul, cât şi tipul de normalizare care se aplică autocorelaţiei.

În fine, sintaxa: *[R,lags] = xcorr(...)*, pe lângă valoarea corelaţiei, returnează şi un vector *lags*, care indică decalajele la care s-a efectuat calculul corelaţiei, şi a cărui elemente sunt distribuite în domeniul *[-maxlags:maxlags]*. Dacă *maxlags* nu este specificat, atunci domeniul de variaţie se consideră a fi *[-N+1:N-1]*.

În toate situaţiile, atât intercorelaţia, cât şi autocorelaţia calculate de comanda *xcorr*, au în mijlocul secvenţei, la elementul sau linia cu numărul *maxlags+1* (sau respectiv elementul sau linia *N* în cazul în care *maxlags* nu este specificat) valoarea corespunzătoare unui decalaj nul între secvenţele furnizate ca argumente.

Exemple:

1. Se consideră un semnal aleator gaussian de medie nulă, şi se doreşte obţinerea autocorelaţiei corespunzătoare unui decalaj *m* cuprins în intervalul [-10,10]. Aceasta poate fi reprezentată cu ajutorul secvenţei:

```
>> ww = randn(1000,1);
>> [R_ww,lags] = xcorr(ww,10,'coeff');
>> stem(lags,R_ww)
```

A doua mărime returnată de către comanda *xcorr*, decalajul, notat cu *lags* în cadrul secvenţei de mai sus, este foarte utilă dacă se doreşte reprezentarea grafică a intercorelaţiei şi respectiv a autocorelaţiei.

Rezultatul comenzilor este reprezentat în graficul din figura 10.2, pe abscisă fiind reprezentat defazajul, iar pe ordonată autocorelaţia semnalului *ww*.

2. Schimbând între ele argumentele de intrare *x* şi *y*, se obţine inversarea secvenţei de corelaţie. În cazul vectorilor linie, secvenţa rezultată este inversată de la stânga la dreapta, iar pentru vectorii coloană, inversarea se face de sus în jos.
   Următorul exemplul ilustrează acest caz (funcţia *mat2str* este folosită pentru a se putea face o afişare compactă a numerelor complexe).

```
x = [1,2i,3]; y = [4,5,6];
[R1] = xcorr(x,y);
figure
plot (R1,'o')
hold on
```

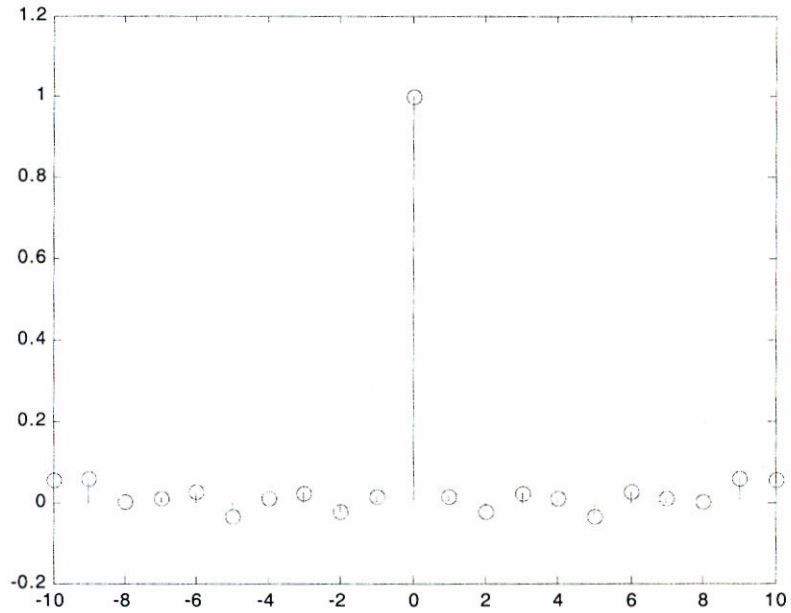

Figura 10.2. Autocorelația unui semnal aleator

```
lags=-2:2;
R1 = mat2str(R1,2), lags
R2 = xcorr(y,x);
plot (R2,'+')
R2 = mat2str(R2,2)
R3 = conj(fliplr(xcorr(y,x)));
R3 = mat2str(R3,2)
```

Rezultatul este reprezentat în figura 10.3, unde pe abscisă este reprezentată partea reală, iar pe ordonată partea imaginară a secvențelor de intercorelație.

După cum se observă, decalajul variază intre –2 şi 2:

```
lags =    [-2    -1     0     1     2]
```

iar secvențele de corelație corespunzătoare vor fi:

```
R1 = [6-i*8.9e-016  5+i*12  22+i*10  15+i*8  12+i*8.9e-016]
R2 = [12-i*8.9e-016  15-i*8  22-i*10  5-i*12  6+i*8.9e-016]
```

respectiv,

```
R3 = [6-i*8.9e-016  5+i*12  22+i*10  15+i*8  12+i*8.9e-016]
```

Mediul Matlab permite calculul **coeficientului de corelare**, cu relația:

$$corrcoef(x,y) = \frac{\text{cov}(x,y)}{\sqrt{\text{cov}(x,x)\cdot\text{cov}(y,y)}} \tag{10.13}$$

unde *cov* reprezintă covarianța.

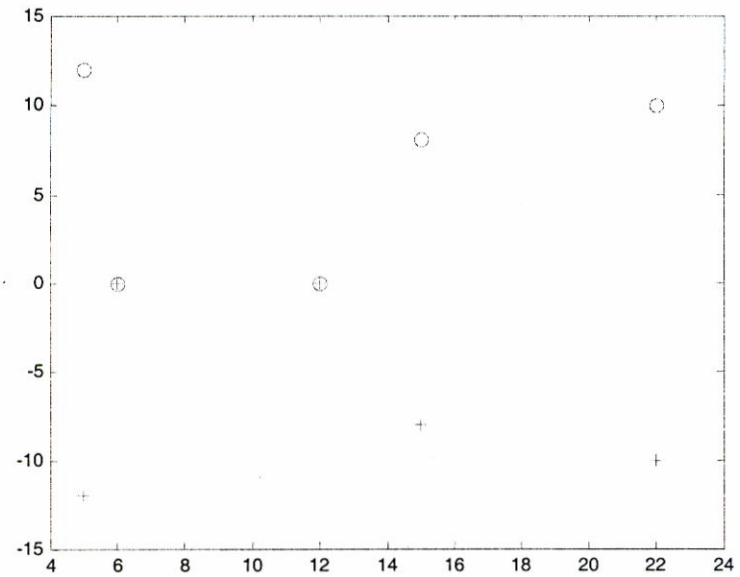

Figura 10.3. Distribuţia intercorelaţiei dintre x *şi* y

Acesta poate lua valori în intervalul [-1, 1]:

$$-1 \le corrcoef(x, y) \le 1 \qquad (10.14)$$

Modulul acestui coeficient arată gradul de corelare a celor două secvenţe, iar semnul indică daca secvenţele variază în acelaşi sens sau nu.

Comanda Matlab corespunzătoare este ***corrcoef***, putând avea ca argument o matrice sau 2 vectori.

10.3. Probleme de studiat

Se va scrie un program în Matlab care să calculeze toate mărimile prezentate în prezentul paragraf.

Mod de lucru:

a) se vor genera două semnale aleatoare unul uniform distribuit X, iar altul normal distribuit Y.
b) se vor calcula mărimile precizate utilizând relaţiile lor de definiţie prezentate, şi apoi se va efectua verificarea folosind comenzile corespunzătoare ale mediului Matlab
c) se vor reprezenta grafic următoarele mărimi:
   - X şi $X0$
   - Y şi $Y0$
   - R_{XY}, R_{XX}, R_{YY}.

Capitolul 11
FILTRAREA NUMERICĂ A DATELOR

11.1. Considerații generale

Identificarea propriu-zisă a unui model al unui sistem (utilizând un anumit algoritm de identificare) este precedată de două etape deosebit de importante, care pot influenta decisiv rezultatul identificării, şi anume, *măsurarea/achiziția datelor*, respectiv *prelucrarea primară a acestora*.

Prelucrarea primară a datelor măsurate are drept scop eliminarea eventualelor valori eronate, respectiv eliminarea influenței zgomotelor, acest al doilea obiectiv fiind în general atins printr-o *filtrare* corespunzătoare. Ea se poate realiza fie utilizând filtre electronice implementate analogic, fie utilizând algoritmi de filtrare implementați software pe echipamente numerice.

Există o categorie de *filtre de bază* frecvent utilizate în prelucrarea semnalelor: filtru trece-jos, filtru trece-sus, filtru trece-bandă, filtru opreşte bandă.

Caracteristicile ideale ale acestor filtre sunt următoarele:

- ***filtrul trece-jos*** permite trecerea informației în banda de frecvență (0,B) şi taie informația în domeniul (B,∞) - vezi figura 11.1.a.
- ***filtrul trece-sus*** implementează caracteristica complementară filtrului trece-jos (permițând trecerea doar a informației în banda de frecvență (B,∞) şi taie informația în domeniul (0,B) - vezi figura 11.1.b.
- ***filtrul trece-bandă*** permite trecerea doar a informației din banda de frecvență de la (f_c-B) la (f_c+B), unde B este lățimea de bandă, iar f_c este frecvența centrală a benzii de frecvență - vezi figura 11.1.c.
- ***filtrul opreşte-bandă*** taie toată informația din banda de frecvență de la (f_c-B) la (f_c+B), permițând trecerea informației din restul domeniului de frecvență - vezi figura 11.1.d.

Realizarea caracteristicilor ideale ale filtrelor nu este posibilă în practică, caracteristicile reale aproximând intr-o măsură mai mică sau mai mare pe cele ideale.

11.2. Filtre numerice recursive

Filtrul numeric recursiv, în cazul general, este descris de o ecuație cu diferențe de forma :

$$y(i) = \sum_{k=0}^{m} b_k x(i-k) - \sum_{j=1}^{m} a_j y(i-j) \qquad (11.1)$$

unde a_j, b_k sunt coeficienții filtrului, iar m este ordinul filtrului.

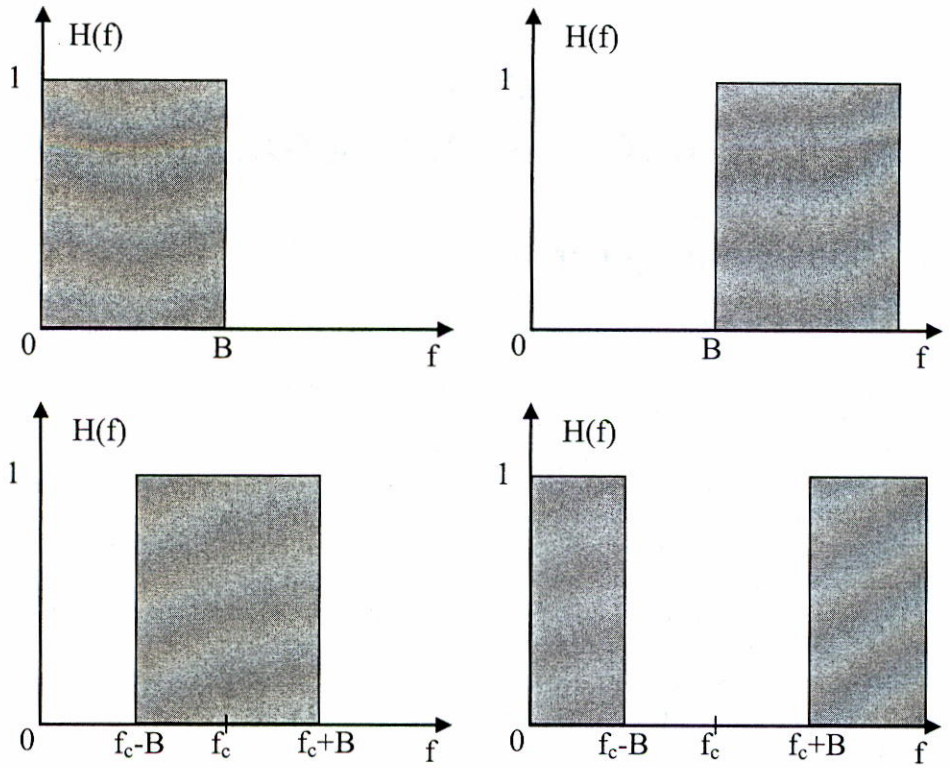

Figura 11.1. Caracteristici ideale de transfer ale filtrelor: a - filtru trece-jos; b - filtru trece-sus; c - filtru trece-bandă; d - filtru opreşte-bandă.

În mediul Matlab există funcţia "***filter***" care implementează relaţia (11.1). Aceasta permite următoarele forme de apel:

```
y = filter(b,a,X)
[y,zf] = filter(b,a,X)
[y,zf] = filter(b,a,X,zi)
```

unde *X* reprezintă secvenţa de intrare, iar *a* şi *b* sunt vectori care conţin coeficienţii filtrului. *zi* este un vector care conţine condiţiile iniţiale ale filtrului, având dimensiunea egală cu maximul dintre dimensiunile vectorilor *a* şi *b* din care se scade 1 - adică *max(length(a),length(b))-1). zf* este tot vector, de dimensiune egală cu *zi* - adică *max(length(a),length(b))-1*, care conţine condiţiile finale ale filtrului.

Comanda *filter* principial implementează o structură de tipul celei prezentate în figura 11.2, unde z^{-1} este operatorul de întârziere cu un pas.

În continuare sunt prezentate câteva considerente referitoare la filtrele recursive de ordinul 1 şi 2.

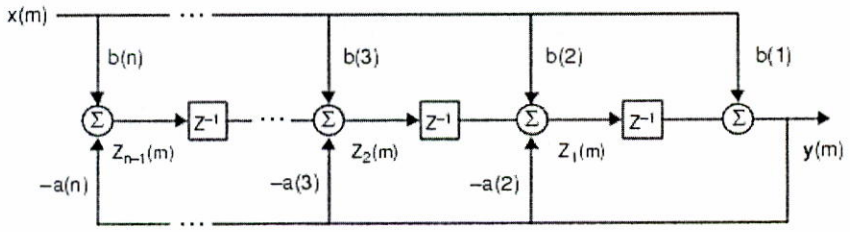

Figura 11.2. Structura filtrului recursiv

11.2.1. Filtre de ordinul I

Forma generală a unui filtru de ordinul I este descrisă de o ecuație cu diferențe de forma (11.1) unde m = 1:

$$y(i) = \sum_{k=0}^{1} b_k x(i-k) - a_1 y(i-1) = b_0 x(i) + b_0 x(i-1) - a_1 y(i-1) \qquad (11.2)$$

Valorile coeficienților pentru câteva categorii importante de filtre sunt date în tabelul următor :

Tipul filtrului	b_0	b_1	a_1
Integrator	T	0	-1
Diferențiator	$\frac{1}{T}$	$-\frac{1}{T}$	0
Filtru trece-jos	$(1-\alpha)$	0	$-\alpha$
Filtru trece-sus	$(1-\alpha)$	0	α
Filtru Cornell (trece-sus)	$1-\frac{\alpha}{2}$	$-\left(1-\frac{\alpha}{2}\right)$	$-(1-\alpha)$

unde $0 < \alpha < 1$.

Exemplu

În continuare se prezintă un program care realizează filtrarea semnalului perturbat prezentat în figura 11.4, obținut prin suprapunerea peste semnalul util sinusoidal reprezentat în figura 11.3 a semnalului perturbator reprezentat în figura 11.3, folosind un filtru recursiv de ordinul I, implementat de comanda *filter*.

```
% Program Matlab pentru filtrare cu functia filter
%Generare semnale
%generare semnal util (sinusoidal)
i=0:0.1:30;
su=sin(i);
%generare semnal perturbator (aleatoriu uniform distribuit)
perturbatie=0.2*rand(1,length(i))-0.1;
```

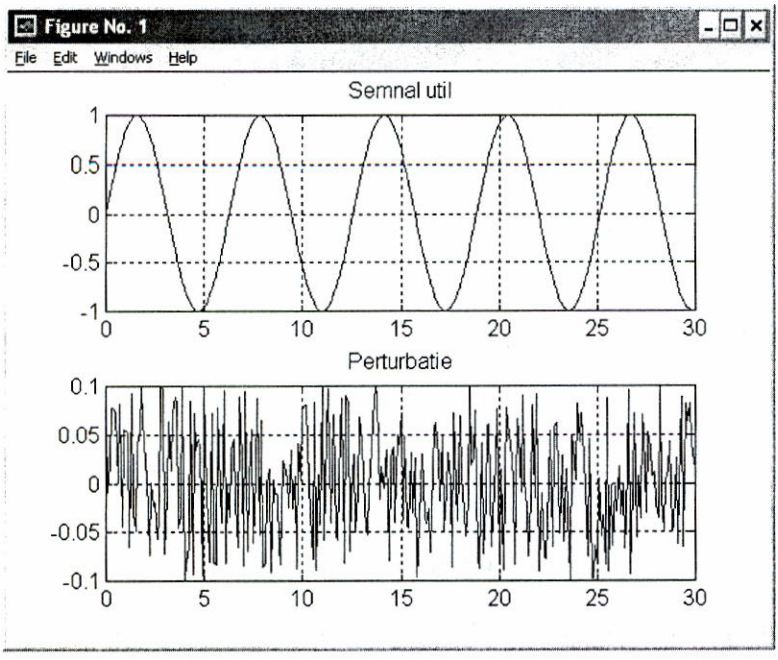

Figura 11.3. Semnalul util, respectiv semnalul perturbator

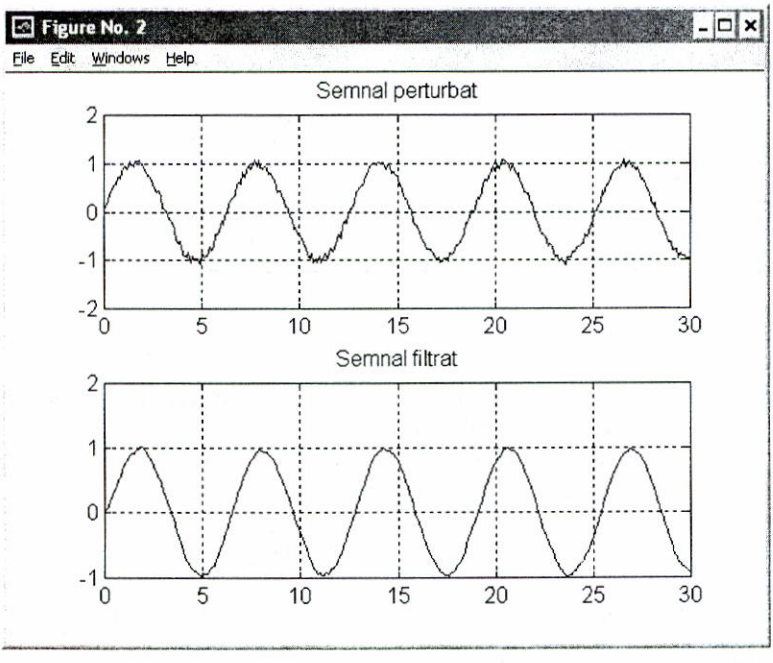

Figura 11.4. Semnalul perturbat, respectiv semnalul obţinut prin filtrare cu alfa = 0.7

```
%obtinere semnal perturbat  prin insumarea celor doua semnale
x=su+perturbatie;
figure
subplot(211)
plot(i,su);,grid,title('Semnal util');
subplot(212)
plot(i,perturbatie);,grid,title('Perturbatie');
figure
subplot(211)
plot(i,x),grid,title('Semnal perturbat');
%Filtrare cu ajutorul unui filtru recursiv de ordinul 1
alfa=input('Introduceti alfa (subunitar): ');
a=[1 -alfa];
b=[(1-alfa) 0];
y=filter(b,a,x);
subplot(212)
plot(i,y),grid,title('Semnal filtrat');
```

Pentru alfa egal cu 0.7, semnalul care rezultă în urma filtrării este reprezentat în figura 11.4.

11.2.2. Filtre de ordinul II

Forma generală a unui filtru de ordinul II se defineşte printr-o ecuaţie cu diferenţe de forma:

$$y(i) = \sum_{k=0}^{2} b_k x(i-k) - \sum_{j=1}^{2} a_j y(i-j)$$

adică:

$$y(i) = b_0 x(i) + b_1 x(i-1) + b_2 x(i-2) - a_1 y(i-1) - a_2 y(i-2) \qquad (11.3)$$

În practică însă se utilizează cu precădere o particularizare a formei generale (11.3) şi anume:

$$y(i) = b_0 x(i) - a_1 y(i-1) - a_2 y(i-2) \qquad (11.4)$$

Proiectarea unui filtru numeric constă în determinarea coeficienţilor a_j şi b_k pentru implementarea directă a filtrului (sau coeficienţi echivalenţi corespunzători altei forme de implementare).

11.2.3. Filtrarea semnalelor utilizând Transformata Fourier Rapidă (TFR)

Această categorie de filtre se utilizează în cazul în care zgomotul de măsurare are o putere spectrală redusă în raport cu semnalul util.

Exemplu:

Pentru exemplificare se consideră semnalul perturbat din figura 11.6 obținut prin suprapunerea unei perturbații aleatoare de medie nulă (vezi figura 11.6) peste un semnal util de forma celui din prezentat în figura 11.5.

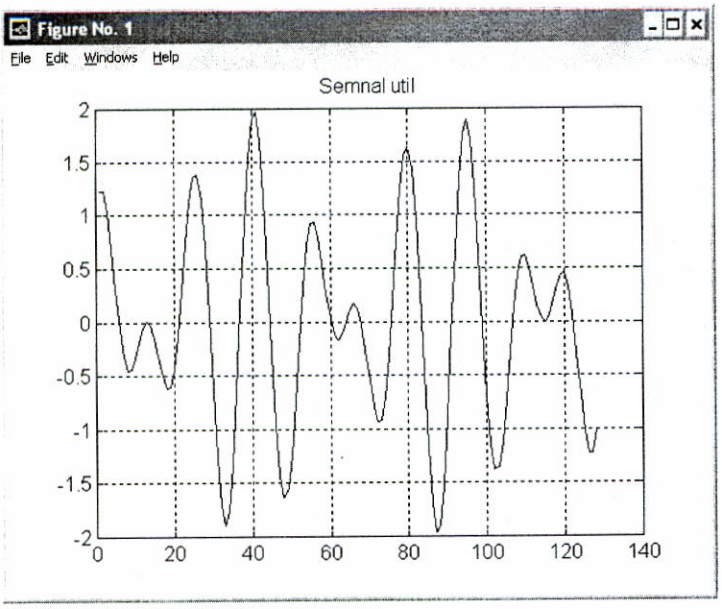

Figura 11.5. Semnal util

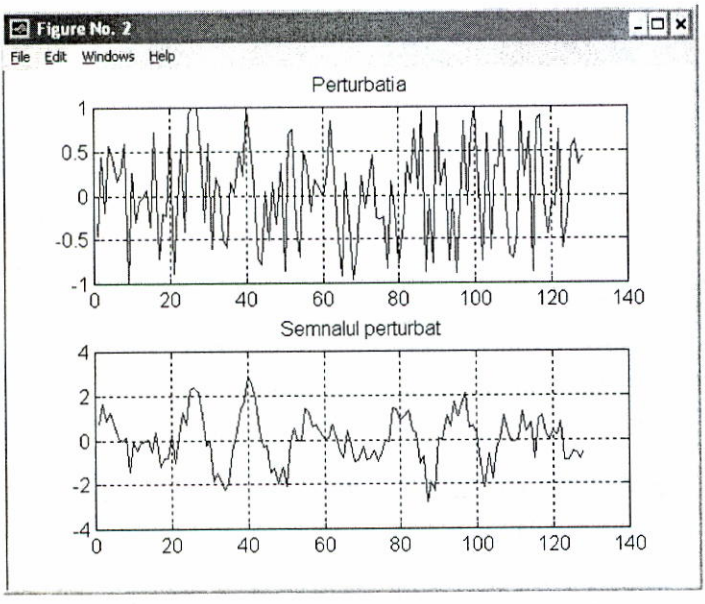

Figura 11.6. Perturbația și semnalul perturbat

Secvența de program care generează și afișează semnalul util este:

```
i=1:128;
q=sin(i*14*pi/128)+cos(i*19*pi/128);
plot(i,q); title('Semnal util'); grid
```

Pentru generarea perturbației și respectiv compunerea secvenței perturbate s-a utilizat secvența următoare:

```
p= rand(1,128)*2-1;
s=q+p;
figure
subplot(2,1,1); plot(i,p); title('Perturbatia'); grid
subplot(2,1,2); plot(i,s); title('Semnalul perturbat'); grid
```

Filtrarea semnalului perturbat presupune următoarele etape:

a) Calculul Transformatei Fourier Rapide a semnalului perturbat (semnalul s din secvența de program anterioară):

```
fs=fft(s);
```

b) Se reprezintă grafic distribuția spectrală de putere atât a semnalului util cât și a semnalului perturbat, apoi se determină valoarea corespunzătoare coeficientului k. Acesta se alege într-un asemenea mod încât să "taie" cât mai mult din componentele parazite. Pentru exemplul considerat o alegere convenabilă este $k=16$.

```
fq=fft(q);
figure
subplot(2,1,1);plot(abs(fq(i)); title(' Semnalul util')
subplot(2,1,2);plot(abs(fs(i)); title('Semnalul perturbat')
```

Rezultatul acestei secvențe de comenzi este prezentat în figura 11.7.

În continuare se poate reprezenta grafic, pentru a verifica corectitudinea alegerii coeficientului *alfa*, atât puterea spectrală a semnalului perturbat cât și coeficientul de rejecție a zgomotului (în cazul de față având valoarea 16).

```
alfa=16;
plot([abs(fs(i))' alfa*ones(1,128)'])
```

c) Se construiește secvența g astfel:

$$g(i)=\begin{cases} fs(i), & \text{pentru } |fs(i)|-k>0 \\ 0, & \text{pentru } |fs(i)|-k\le 0 \end{cases} \qquad (11.5)$$

Transformata Fourier Rapidă corespunzătoare semnalului filtrat, obținută prin aplicarea relației (11.5), trebuie să elimine componentele armonice de putere mică, dar să lase neschimbate armonicele de putere mare care definesc semnalul util, secvența de comenzi corespunzătoare fiind:

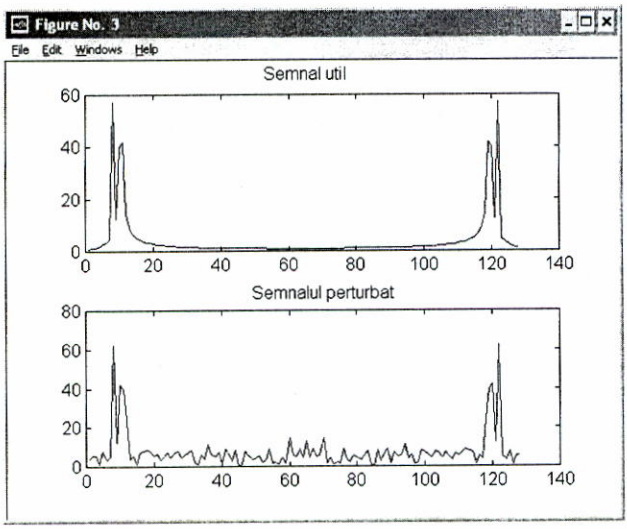

Figura 11.7. Distribuția spectrală de putere a semnalului util, respectiv perturbat

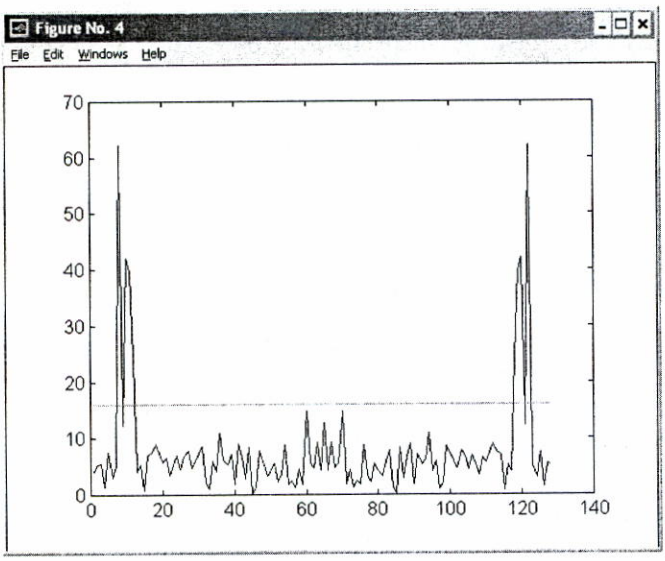

Figura 11.8. Distribuția spectrală de putere a semnalului perturbat şi coeficientul de rejecție a zgomotului (în cazul de față având valoarea 16)

```
g=zeros(1,length(fs));
for k=1:128
    a=abs(fs(k))-alfa;
    if a>0, g(k)=fs(k);
    end
end
plot(abs(g)); grid
```

Caracteristica obținută are forma celei din figura 11.9.

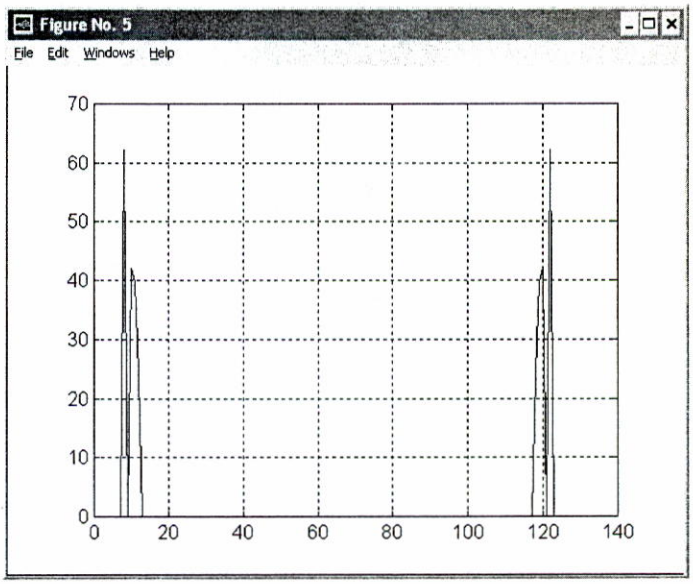

Figura 11.9. Distribuția spectrală de putere a semnalului filtrat

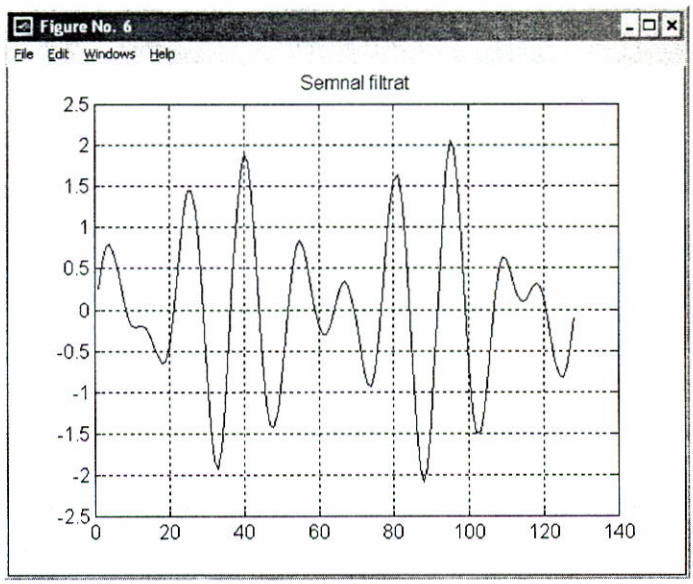

Figura 11.10. Semnalul filtrat

d) Se realizează transformata Fourier inversă a lui g rezultând semnalul filtrat, adică

```
h=ifft(g)
plot(real(h)); grid; title(,Semnal filtrat')
```

Pentru exemplul considerat mai sus, în urma filtrării semnalului cu ajutorul algoritmului prezentat, s-a obținut o rejectare foarte bună a perturbațiilor, semnalul filtrat având forma prezentată în figura 11.10.

Pentru a putea compara cu mai multă uşurință semnalul util cu cel obținut prin filtrare, acestea pot fi afişate pe aceeaşi figură cu ajutorul unor comenzi de forma:

```
figure
plot(real(h));
hold on
plot(q,':g');title('Semnal util şi semnal filtrat')
```

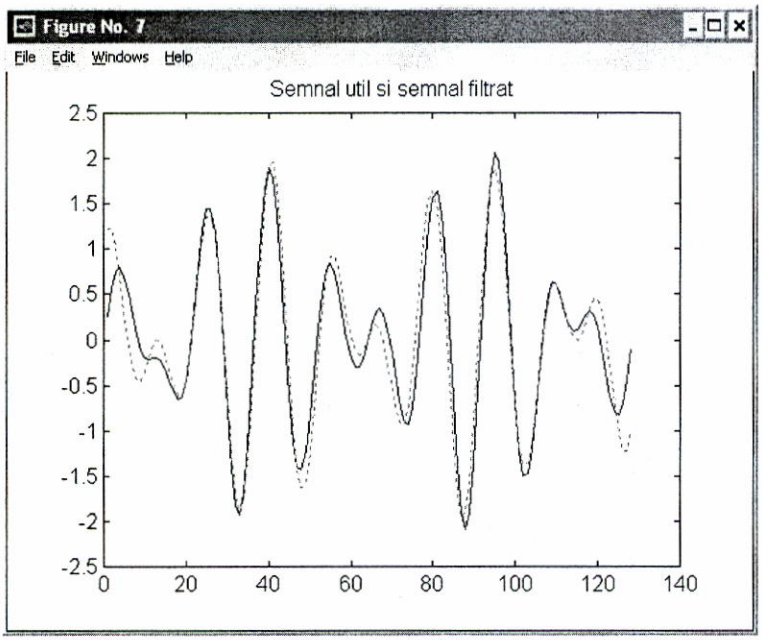

Figura 11.11. Semnalul util (linie punctată), respectiv semnalul filtrat (linie continuă)

11.3. Probleme de studiat

1. Să se realizeze filtrarea utilizând TFR a semnalului dat de următoarea ecuație discretă:

   q(i)=sin(i*10*pi/128)+cos(i*19*pi/128)+sin(i*25*pi/128)+cos(i*35*pi/128)

   peste care se suprapune un zgomot aleator de medie nulă, rezultând semnalul perturbat s.

   Mod de lucru: se vor urma etapele prezentate anterior, şi se vor reprezenta grafic secvențele q, s, h.

   Să se rezolve problema 1, pentru o perturbație aleatoare de medie nenulă.