I. INTRODUCERE

I.1 MODELUL MATEMATIC : DEFINITIE, CLASIFICARE

Modelul matematic al unui sistem este un ansamblu de relații matematice, ecuații şi inecuații, care caracterizeaza şi descriu interdependențele dintre parametrii constructivi şi funcționali ai sistemului. Prezența inecuațiilor în model se datorează unor restricții cu caracter fizico-chimic, tehnologic sau constructiv.

Modelarea matematică este utilă în toate fazele de dezvoltare ale unei tehnologii, ea aducând cu sine o serie de avantaje certe[3]: aprofundarea cunoaşterii şi înțelegerii procesului (trebuiesc luate în considerare secvențe complexe cauză-efect, interdependențele dintre variabile), proiectarea optimală a instalațiilor (dimensionarea utilajelor şi evaluări ale parametrilor pe baza datelor obținute pe instalații pilot, studiul efectelor modificărilor în dimensiuni, structura optimă a fluxului tehnologic, etc.), optimizarea exploatării instalațiilor în funcțiune, controlul optimal, etc.

In construcția modelului se adoptă, în general, o linie de compromis între cerințele legate de o descriere riguroasă a procesului (ecuații complexe) şi posibilitățile de simulare numerică. Nu este necesar ca modelul să constituie o descriere extrem de amănunțită a mecanismelor reale din sistem. El trebuie să aibă gradul de complexitate minim cerut de scopul pentru care a fost construit. Iată câteva dintre problemele care se pot pune la elaborarea unui model matematic[1] :

- este necesară descrierea interdependenței dintre variabile şi în regim dinamic sau este suficientă descrierea în regim staționar?
- ce considerații tehnico-economice trebuie avute în vedere?
- cum se pot folosi cunoştințele care le avem despre proces şi cum se pot dobândi cunoştințele care ne lipsesc?
- care va fi criteriul de apreciere a calității modelului?
- cum pot fi simplificate relațiile?

In ceea ce priveşte clasificarea, există mai multe criterii utilizabile: forma ecuațiilor (liniare – neliniare, parametri concentrați – parametri distribuiți), gradul de cunoaştere al parametrilor modelului (deterministe – când fiecărui parametru sau variabile independente i se poate atribui o valoare bine definită; stohastice – parametri sau variabile ale procesului au valori care se pot exprima doar probabilistic).

Din punct de vedere al modului de deducere al relațiilor dintre variabile, se deosebesc următoarele **tipuri de modele** [2,5]:

- modele analitice, bazate pe ecuații de conservare. Ele se obțin prin scrierea ecuațiilor de bilanț (masă, energie) pentru întregul reactor sau o porțiune infinitezimală a reactorului (funcție de tipul acestuia) ;
- modele statistice (numite şi empirice) - bazate pe corelarea datelor experimentale; forma ecuațiilor poate să nu fie în legatură cu semnificația fizică a variabilelor care caracterizează procesul iar domeniul de valabilitate al modelului se rezumă la domeniul în care au fost modificate variabilele.
- modele stohastice - sunt utilizate pentru situații în care o descriere probabilistică este mai aproape de comportarea reală a procesului. Exemple tipice sunt procesele în care trebuie luată în considerare distribuția duratelor de staționare sau distribuția vârstei particulelor.

In multe cazuri, deducerea modelului se realizează mixt : pe baza relațiilor dintre variabile se stabileşte structura modelului iar prin prelucrarea statistică a datelor experimentale se obțin coeficienții din ecuații.

Figura I.1.1 relevă corelarea dintre cunoştințele analitice şi cele obținute pe bază de experiment în elaborarea modelului matematic [1].

Pe baza legilor fizico-chimice care guvernează procesul, se stabileşte structura modelului (forma ecuațiilor care descriu relațiile dintre variabilele procesului). In cazul în care structura este prea complexă pentru scopul pentru care a fost construit modelul, se trece la liniarizarea şi reducerea ecuațiilor cu derivate parțiale. O astfel de necesitate poate apare în cazul conducerii procesului cu calculatorul : dacă se utilizează un model prea complex, calculatorul ar pierde prea mult timp cu soluționarea ecuațiilor, soluția fiind obținută prea târziu, în proces putând avea loc între timp alte evolutii.

Liniarizarea se poate face prin dezvoltarea ecuațiilor în serie Taylor-Lagrange în jurul punctului de funcționare normală. Reducerea se poate realiza prin scrierea unor ecuații de bilant pe zone ale utilajului (de exemplu, în lungul coordonatei axiale), fiecare ecuație cu derivate parțiale fiind transformată într-un sistem de ecuații diferențiale ordinare.

Coeficienții ecuațiilor se vor determina prin prelucrarea datelor obținute experimental. Datele experimentale trebuiesc supuse mai întâi unui proces de validare în vederea eliminării seturilor care au fost afectate de erori (îndeplinirea bilanțurilor de materiale şi termic – în cazul în care este vorba de regimul staționar, teste statistice). Pe baza structurii stabilite pentru ecuațiile modelului şi pe baza datelor experimentale validate, se trece la determinarea parametrilor modelului: prin estimare se urmăreşte ca diferenta între valorile mărimilor de iesire în cazul procesului şi modelului să fie cât mai mică. Se folosesc criterii de eroare cum ar fi eroarea medie pătratică. Un astfel de model este valabil doar în limitele în care au fost modificați parametrii.

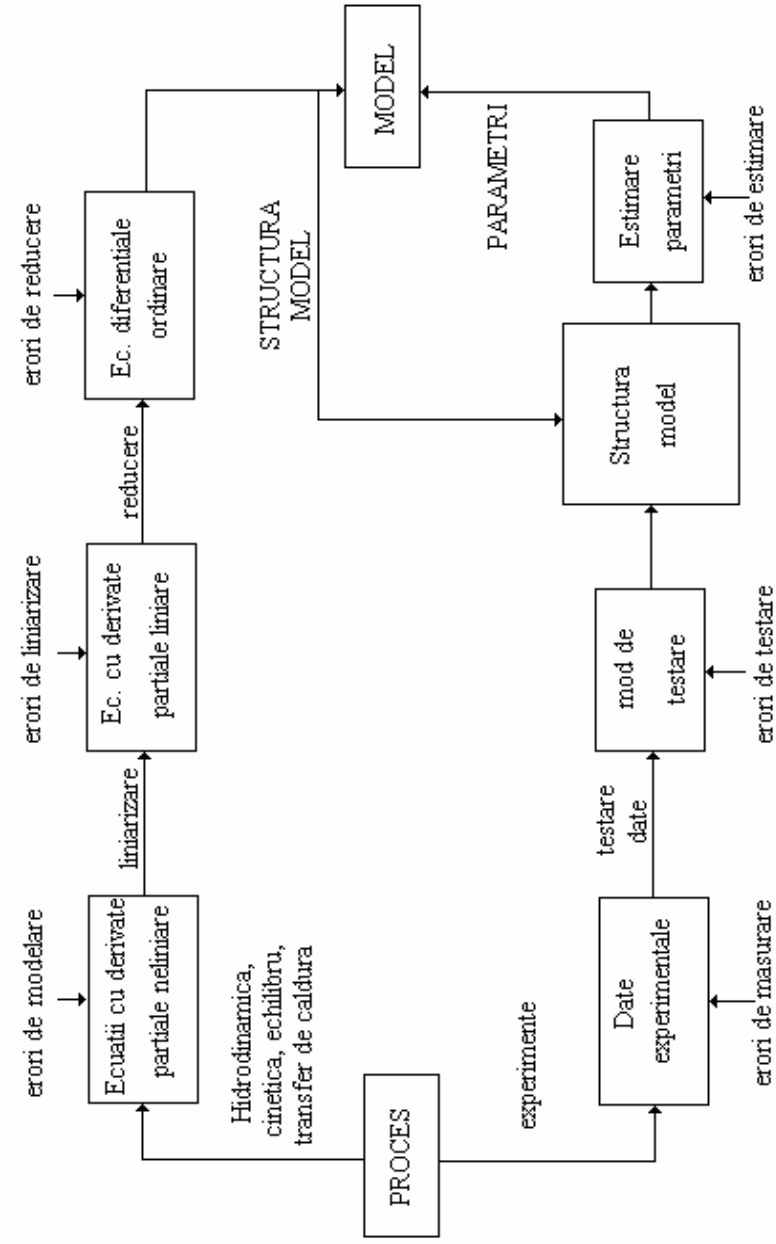

Figura I.1.1 Corelarea dintre cunostintele teoretice si cele experimentale in elaborarea modelului matematic

I.2 MATLAB : NOTIUNI DE BAZA

MATLAB este un mediu de programare bazat pe operații asupra tablourilor (vectori si matrici), de unde și numele (« MATrix LABoratory »). Elementul de bază în programare este matricea, fapt ce face din MATLAB un puternic mijloc de calcul numeric și de reprezentare grafică (operatori matriciali, programare simplificată pentru rezolvarea sistemelor de ecuații liniare si neliniare, mari facilități de reprezentare grafică, etc.). Alături de funcțiile elementare de bază (trigonometrice, exponențială, logaritm, etc.), peste 500 de funcții speciale (statistica si analiza datelor, solvere de ecuații liniare si diferențiale, transformata Fourier rapidă, etc.) sunt la dispoziția utilizatorului.

MATLAB include și aplicații specifice numite TOOLBOX-uri. Acestea sunt colecții de fișiere de tip « funcție », cu extensie « .m », dedicate rezolvării unor probleme inginereşti specifice : comunicații, controlul sistemelor, analiza în domeniul frecvențelor și identificarea sistemelor, analiza financiara, logica fuzzy, analiza spectrală, procesarea imaginii, control predictiv după model, micro-analiza și sinteza pentru control robust, analiza informației geografice, biblioteca « Numerical Algorithms Group - NAG Foundation» , analiza și vizualizarea datelor, rețele neurale, optimizare, ecuații cu derivate parțiale, proiectarea sistemelor de control, control robust, procesarea și tratarea semnalului, funcții « spline », matematica simbolică, statistică, etc.

SIMULINK este o altă opțiune care poate completa MATLAB în vederea schimbării complete a interfeței utilizator – limbaj de programare : el se bazează pe utilizarea schemelor bloc pentru modelarea sistemelor dinamice.

Față de alte limbaje de programare utilizate în inginerie, MATLAB pare să aibă avantaje incontestabile atât din punct de vedere al realizării de aplicații specifice și de studii de caz cât și din punct de vedere al utilizării aplicațiilor standard achiziționate ca și « toolbox-uri » ; rutinele conținute în toolbox-uri pot fi ușor incluse în alte aplicații. Deci există atât un nivel la care utilizatorul se gaseşte foarte aproape de ecuații și de soluția lor numerică (ceea ce permite un bun control și o mare manevrabilitate a aplicației cu un efort mai mic din punct de vedere al programării) dar și un nivel la care se poate ramăne departe de soluția numerică a ecuațiilor (SIMULINK).

Acestea sunt motivele pentru care autorii au optat pentru MATLAB în vederea realizării aplicațiilor din acest volum. In cele ce urmează, se prezintă un minimum de imformații necesare pentru a putea realiza primele aplicații în acest modern mediu de programare.

I.2.1 Variabilele în MATLAB.

- Trebuie să înceapă cu o literă; MATLAB face o diferențiere între literele mari şi cele mici (Cost, COST, cost, coST sunt variabile diferite);
- Lungimea cuvântului : 19 caractere(restul sunt ignorate); nu este permisă folosirea semnelor de punctuație în interiorul unui cuvânt ce reprezintă o variabilă ;
- Variabile speciale utilizate de MATLAB:
  - « ans » - rezultatul unui calcul;
  - « pi » - raportul dintre perimetrul şi diametrul cercului = 3.14...;
  - « eps » - cel mai mic număr care, adunat la 1, crează un număr în virgula mobila mai mare decât 1 ;
  - « inf » - infinit(ex.: 1/0);
  - « NaN » - not a number(ex. 0/0);
  - « i »sau « j » = radical din -1, adică i de la numerele complexe;
  - « realmin » şi « realmax » - cel mai mic şi respectiv cel mai mare număr real pozitiv utilizabil.
- Formate de afişare a numerelor:
  - format short - patru cifre după virgulă ex.: 35.3482 - este forma standard;
  - format long - afişare pe 16 pozitii ex.: 35.34822154352415;
  - format short e - 5 digiți plus exponent ex.: 3.5348e+01;
  - format long e -16 digiți plus exponent ex.: 3.534822465369841e+1;
  - format rat - scriere sub formă de fracție ex. : 215/6;
  - alte formate : hex - hexazecimal; bank - 2 digiți după virgulă;

I.2.2. Operații matematice asupra scalarilor şi funcții elementare

Expresiile sunt evaluate de la stânga la dreapta, utilizându-se prioritățile cunoscute: puteri, înmulțire şi împărțire, adunare şi scădere.

- adunare = "+" ex.: 5+3 ;
- scădere = "-" ex.: 5-3 ;
- înmulțire = "*" ex.: 5*3 ;
- împărțire = "/" sau "\" ex.: 5/3 sau 3\5 NOTA: 5/3=3\5 "\" reprezintă împărțire la stânga.
- putere = "^" ex.: 5^3

Pentru a modifica ordinea operațiilor, se folosesc paranteze. Exemplu:

5*((3+5)^2-7*(8-2)/5)-2.

Funcții elementare:Elementary math functions.

- Trigonometrice (NOTA: pentru funcțiile trigonometrice, unghiurile sunt exprimate în radiani) :
  - *sin(x)* - sinus ;
  - *sinh(x)* - sinus hiperbolic ;
  - *asin(x)* - arcsinus (invers sinus) ;
  - *asinh(x)* – arcsinus hiperbolic (invers sinus hiperbolic) ;

  Pentru celelalte funcții trigonometrice, simbolurile sunt :
  - grupul cosinus : *cos, cosh, acos, acosh* ;
  - grupul tangentă : *tan, tanh, atan, atanh* ;
  - grupul cotangentă : *cot, coth, acot, acoth* ;
  - grupul secantă : *sec, sech, asec, asech* ;
  - grupul cosecantă : *csc, csch, acsc, acsch* .
- alte funcții elementare :
  - *exp(x)* - e^x (« e » - baza logaritmilor naturali) ;
  - *log(x)* - logaritm natural ;
  - *log10(x)* - logaritm zecimal ;
  - *sqrt(x)* - radical din x ;
  - *abs(x)* - valoarea absolută a lui x ;
  - funcții referitoare la numere complexe (*angle, conj, imag, real*) ;
  - funcții pentru aproximarea numerelor (*ceil, rem, rat, sign*, etc.).

I.2.3. Operații asupra vectorilor şi matricilor.

Dacă un tablou de variabile are aşezate elementele în linie, vorbim de un vector linie; în cazul în care acestea sunt aşezate în coloană, vorbim de un vector coloană. Dacă elementele sunt aşezate atât pe linii cât şi pe coloane, vorbim de matrici. Un vector linie se poate introduce element cu element:

```
a= [ 1 3 5 8 -2 0]
```

In acelaşi mod se poate introduce şi un vector coloană(trecerea de la o coloană la alta este marcată prin « ; »):

```
b= [ 1; 3; 5; 8; -2; 0]
```

Trecerea dintr-o formă în alta se poate realiza prin operatorul de transpunere « ' »:

```
a = b'
```

NOTA: « .' » reprezintă transpunerea iar « ' » reprezintă transpunerea complex conjugatei; pentru numere reale, cele două operații sunt identice.

In cazul vectorilor de mari dimensiuni, există o serie de alte posibilități:

```
a=0:0.1:1
```

va creea un vector care începe cu elementul 0, următoarele elemente fiind incrementate cu 0.1 până la atingerea valorii 1.

a=(0:0.1:1)*pi

va înmulți fiecare element al vectorului precedent cu π= 3.14....

Se pot utiliza şi instrucțiunile linspace şi logspace a căror formă este:

- *linspace*(prima valoare, ultima valoare, număr de valori) ;
- *logspace*(primul exponent, ultimul exponent, număr de valori).

Exemplu : *a=linspace(0,1,11)*

Alte modalităti de a crea tablouri de variabile :

- *a=1:5, b=1:2:9, c=[a b]*

va crea vectorul c=[1 2 3 4 5 1 3 5 7 9].

- *d= [a(1:2:5) 1 0 1]*

va crea vectorul linie : d=[1 3 5 1 0 1]

Adresarea elementelor este foarte facilă: d(3) fiind al treilea element, adica 5, d(3:6) fiind al treilea,...,al şaselea element al vectorului.

Introducerea matricilor este asemănătoare. Matricea:

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

se poate introduce în următoarele forme :

a). *a=[1 2 3; 4 5 6; 7 8 9]*

b). *a=[1 2 3*
4 5 6
7 8 9]

Adresarea elementelor matricii este asemănătoare cu cea pentru tablouri:

- a(2,3) reprezintă un element din linia 2, coloana 3 adica 6 ;
- a(1,:) reprezintă toate coloanele din linia 1, adică elementele 1 3 5 ;
- a(:,2) reprezintă toate liniile coloanei 2, adică 3 4 5 ;
- a(:) rearanjează matricea într-un vector coloană, preluând elementele coloană după coloană.

MATLAB dispune de mari facilități de manipulare a matricilor. Luând ca bază matricea "a" prezentată anterior, operațiile:

a(3,3)=0, a(2,5)=1

va înlocui elementul din coloana 3 şi linia 3 cu 0, va modifica dimensiunea matricii la 3 linii şi 5 coloane, elementele adăugate dar nedefinite fiind considerate 0:

$$a = \begin{bmatrix} 1 & 2 & 3 & 0 & 0 \\ 4 & 5 & 6 & 0 & 1 \\ 7 & 8 & 0 & 0 & 0 \end{bmatrix}$$

Instrucțiunea(referitoare la matricea inițială de dimensiune 3/3):

b=a(3:-1:1,:)

va crea matricea b prin luarea liniilor matricii a în ordine inversă:

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

Instrucțiunea(referitoare la matricea « a » inițială de dim. 3/3 şi b de mai sus):

c=[a b(:,[1 3])]

va crea o matrice "c" prin alipirea la matricea "a" a coloanelor 1 şi trei ale lui "b":

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

Extragerea unei matrici dintr-o matrice dată:

d=c(1:2,3:5)

are drept rezultat:

$$d=\begin{bmatrix}3 & 7 & 9\\ 6 & 4 & 9\end{bmatrix}$$

Dacă rezultatul unei instrucțiuni este constituit din două valori (sau mai multe), valorile sunt menționate într-un vector:

Instrucțiunea :

s=size(c)

va avea drept rezultat :

```
s=
     3   5
```

Dimensiunile matricii se pot afla şi astfel:

[linii coloane]=size(a)

cu rezultatul:

```
linii=
     3
coloane=
     5
```

Operatorii relaționali se pot folosi şi în cazul matricilor. Ei compară două matrici element cu element şi crează(returnează) o matrice de aceleaşi dimensiuni având elementele egale cu 1 acolo unde relația este adevarată şi 0 unde ea este falsă.

Exemplu. Vrem să aflăm elementele matricii x mai mari în valoare absolută decât 2 :

```
x=[ 1 -1 3; 2 -3 1; 2 1 4] ;
  y=abs(x)>2
```

Se obține rezultatul:

$$y = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Instrucțiunea *find* permite identificarea elementelor unei matrici care îndeplinesc o anumită condiție. Exemplu (cu referire la matricea x definită anterior):

[i,j]=find(abs(x>2))

are drept rezultat:

```
i=
     1     2     3
j=
     3     2     3
```

Alte operații asupra matricilor:

- dacă v este un vector,

a=diag(v)

va crea o matrice diagonală cu vectorul v pe diagonală;

- dacă a este o matrice,

v=diag(a)

va crea un vector format din elementele de pe diagonala matricii a;

O matrice se poate rearanja pe alte dimensiuni cu ajutorul instrucțiunii *reshape*:

```
a=[ 1 2 3 1
    4 5 6 1
    7 8 9 0] ;
b=reshape(a,4,3)
```

are drept rezultat(elementele se preiau pe coloane):

$$b = \begin{bmatrix} 1 & 5 & 9 \\ 4 & 8 & 1 \\ 7 & 3 & 1 \\ 2 & 6 & 0 \end{bmatrix}$$

Câteva matrici speciale se pot genera prin instrucțiuni simple. Exemplu :

x= zeros(3) , y= ones(2,3) , z= eye(3)

au drept rezultate:

$$x=\begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix} \qquad y=\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} \qquad z=\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$$

Operații asupra tablourilor:

Fie a=[a_1 a_2 ..... a_n], b=[b_1 b_2 .... b_n], şi c un scalar c=[c] ; operații posibile în MATLAB şi semnificația lor :

- adunarea unui scalar a+c=[a_1+c .... a_n+c] ;
- înmulțire cu un scalar a*c=[a_1*c .... a_n*c] ;
- adunarea a două tablouri a+b=[a_1+b_1 ... a_n+b_n] ;
- înmulțirea a două tablouri a.*b=[a_1*b_1 ... a_n*b_n] ;
- împărțire la dreapta, element cu elem. a./b=[a_1/b_1 ... a_n/b_n] ;
- împărțire la stânga, elem. cu element a.\b=[a_1\$b_1$ ... a_n\$b_n$] ;
- puteri a.^c=[a_1^c ... a_n^c] ;
  c.^a=[c^a_1 ... c^a_n] ;
  a.^b=[a_1^b_1 ... a_n^b_n] .

Algebra liniară şi matrici

Rangul, determinantul, transpusa şi inversa unei matrici pot fi evaluate prin câte o simplă instrucțiune.

Exemple:

- calculul rangului matricii a:
  r=rank(a)
- calculul determinantului:
  delta=det(a)
- în cazul transpusei, pentru cele care nu sunt alcătuite din nr. complexe:
  b=a.'
- dacă a este complexa, se poate calcula transpusa complex-conjugatei:
  b=a'
- calculul inversei:
  b=inv(a)

Operatorii * , /, \ (ultimii doi semnificând împărțirea) sunt folosiți în sens matricial.

Deci, pentru înmulțirea a două matrici se utilizează operatorul " * ":

*c=a*b*

unde a este o matrice m*n, b o matrice n*p iar matricea c va avea dimensiunile n*p.

Operatorul \ este un operator deosebit de puternic, soluția aleasă pentru împărțire şi deci pentru soluționarea unui sistem de ecuații liniare depinzând de forma matricii a. Operatorii \ şi / sunt echivalenți. Pentru a soluționa sistemul:

*a*x=b*

se poate scrie:

*x=a\b, x=inv(a)*b.*

Exemplul I.2.1 : să se soluționeze sistemul de ecuații :

$$\begin{aligned} &x_1 + 2\cdot x_2 + 0.5\cdot x_3 + 3\cdot x_4 = 9 \\ &0.5\cdot x_1 + 0.5\cdot x_2 + x_3 + x_4 = 4.5 \\ &2\cdot x_1 + x_2 - x_3 + 2\cdot x_4 = 4 \\ &0.5\cdot x_1 - x_2 + x_3 - 3\cdot x_4 = -2.5 \end{aligned} \qquad \text{(I.2.1)}$$

Iată secvența de program prin care se poate realiza acest lucru în MATLAB :

```
a=[1 2 0.5 3 ;0.5 0.5 1 1;2 1 -1 2;0.5 -1 1 -3];
b=[9;4.5;4;-2.5];
x=a\b;
disp('sol.sist=');disp(x) ;
```

I.2.4. Comenzi pentru execuții secvențiale

Pentru executarea repetată a unor secvențe de calcul, se pot utiliza buclele for şi while iar pentru luarea unor decizii în cursul calculului structurile if ... else.

Structura ciclului "for":

for k = tablou
    bloc de instrucțiuni (se utilizează pentru indexare k)
end;

Exemplul I.2.2: să se calculeze sinusul şi cosinusul unghiurilor 0,10, ... 90 de grade. Programul :

```
for k= 1:10
      alfa(k)=(k-1)*10;
end;
alfarad=alfa.*(pi/180);
sinalfa=sin(alfarad);
cosalfa=cos(alfarad);
disp('valoarea unghiului');disp(alfa);
disp('sinus de alfa=');disp(sinalfa);
disp('cosinus de alfa=');disp(cosalfa);
```

Secvența de program de mai sus va calcula mai întâi, printr-un ciclu « for », elementele vectorului linie alfa , unghiurile fiind exprimate în grade ; Următoarele trei linii de program calculează vectorul alfarad (unghiurile exprimate în radiani), şi apoi cei doi vectori linie care conțin valorile sinusului şi cosinusului unghiurilor (sinalfa şi cosalfa). Instrucțiunea « *disp* » (display) afişează rezultatele pe ecran: ea operează doar asupra unei singure variabile (scalar sau tablou) iar ceea ce este marcat între două apostroafe se tipăreşte întocmai.

De remarcat faptul că funcțiile elementare operează şi asupra tablourilor !

Iată rezultatul acestei secvențe de program (variabilele afişate în formatul « short »):

```
valoarea unghiului
    0   10   20   30   40   50   60   70   80   90
sinus de alfa=
  Columns 1 through 7
       0   0.1736   0.3420   0.5000   0.6428   0.7660   0.8660
  Columns 8 through 10
    0.9397   0.9848   1.0000
cosinus de alfa=
  Columns 1 through 7
    1.0000   0.9848   0.9397   0.8660   0.7660   0.6428   0.5000
  Columns 8 through 10
    0.3420   0.1736   0.0000
```

NOTA : Programele realizate sub MATLAB se redactează fie în fereastra de comenzi, situație în care fiecare linie de program este executată imediat ce se tastează « enter », fie în fişiere cu extensia « .m », caz în care programul este rulat la tastarea numelui fişierului în fereastra de comenzi. Dacă o instrucțiune nu se termină prin punct-virgulă, « ; », rezultatele sunt afişate imediat ce linia este executată.

Alte exemple de structuri for.

Ca şi în alte limbaje de programare, se pot realiza structuri "ciclu în ciclu":

```
n=10;m=5;
for k1=1:n
      for k2=1:m
            c(k1,k2)=sin((k1+k2)/(n+m)*pi);
      end;
end;
```

Incrementul poate avea şi valori negative:

```
 n=10;
disp('k       ln=') ;
for k=n+2:-1:n/2
      lognat(k)=log(k);
      disp([k     lognat(k)]) ;
end;
```

Ciclul "while"

Structura:

while expresie1 operator_relaţional expresie2
bloc de instrucţiuni
end;

Operatorii relaţionali sunt:

== egal; >= mai mare sau egal; <= mai mic sau egal;
~= diferit; < mai mic; > mai mare.

Instrucţiunile din ciclu sunt executate atât timp cât raportul dintre expresiile 1 şi 2 este adevărat.

Exemplul I.2.3 : să se calculeze $e^{\sin(\alpha)}$ pentru α=0, 10, …,90°:

```
alfa=0; dalfa=10;
disp('alfa      exp(sin(alfa))') ;
while alfa<=90
      expsinalfa=exp(sin(alfa)) ;
      disp([alfa expsinalfa]) ;
      alfa=alfa+dalfa ;
end;
```

Structura if ... else ... end:

if expresie_relaţională1
bloc de instrucţiuni executat dacă expresie_ relatională1 este adevarată
elseif expresie_relatională2
bloc de instrucţiuni executat dacă prima expresie relaţională este falsă iar a doua adevarată
elseif expresie_relatională3
bloc de instrucţiuni executat dacă primele două expresii relaţionale sunt false iar expresie_relatională3 este adevarată
.....
else
bloc de instr. executat dacă expresiile relationale anterioare nu sunt adevarate
end;

Exemplul I.2.4: Mărimea de intrare « u » a unui proces are valoarea 0 în primele 100 secunde, valoarea 1 în următoarele 100 de secunde după care revine din nou la zero :

```
tau=0 ;
while tau<250
      if tau <= 100
            u=0 ;
      elseif tau<=200
            u= 1;
      else
            u=0 ;
      end;
      disp([tau u]) ;
      tau=tau+1 ;
end ;
```

I.2.5 Funcții ale utilizatorului

La funcțiile standard ale MATLAB pot fi adăugate funcții noi utilizând instrucțiunea *function :*

Structura:

function [variabile de iesire]=denumire_funcție(variabile de intrare)

... instrucțiuni ce formează corpul funcției...

Exemplul I.2.5: să se realizeze o funcție care să calculeze valoarea medie şi abaterea standard pentru un set de valori ale unei variabile x. Dacă v_m este valoarea medie şi a_{std} abaterea standard, expresiile de calcul sunt :

$$v_m = \frac{\sum_{i=1}^{n} x_i}{n}; \qquad a_{std} = \sqrt{\frac{\sum_{i=1}^{n} (x_i - v_m)^2}{n-1}} \; ;$$

Funcția va avea două variabile de ieşire şi o singură intrare, vectorul valorilor variabilei x :

```
function [vm, abstd]=abatstd(x)
   n=length(x);
   vm=sum(x)/n;
   abstd=sqrt(sum((x-vm).^2)/(n-1));
```

length(x) determină "n", numărul de elemente ce compun vectorul x;

sum(x) calculează suma elementelor ce compun vectorul x ;

(sum((x-vm).^2) scade din fiecare element al vectorului x valoarea medie v_m , ridică rezultatul la pătrat apoi având loc însumarea.

Funcția trebuie salvată într-un fişier ce are ca nume numele funcției(în acest caz : abatstd.m). Exemplu de apelare (variabilele de ieşire pot să aibă şi nume diferite față de ieşirile definite în fişierul funcției):

```
 x=[5.1 6 5.7 6.2 5.5 5.8 6.05 5.3 5.8 5.4 6.05 5.75 5.9 6.15
5.38 ];
 [vm, abstd]=abatstd(x)
```

I.2.6 Grafica în MATLAB

In cazul bidimensional, se utilizează instrucțiunea "*plot*". Structura:
plot(y1,x1,'tip_linie1',y2,x2,'tip_linie2', ...);
- "tip_linie" reprezintă tipul liniei sau al punctului precum şi culoarea.
Tipul liniei poate fi: « - », « _ » , « -. » . Drept puncte se pot folosi: « * », « + » , « x », « o » ,etc.

Pentru culori, codurile sunt: r - rosu; g - verde; y - galben; b - albastru; w - alb, etc.

Pe figură se pot adăuga:
- titlul figurii : *title*(' text ce reprezintă titlul');
- eticheta(denumirile) axelor : *xlabel*(' .... '), *ylabel*(' ... ') ;
- grila (caroiajul): *grid*;
- un text amplasat în punctul de coordonate x,y: *text*(x,y,'....').

Coordonatele se dau în unitățile axelor;
- limitele valorilor pe fiecare axă: *axis*([valminx valmaxx valminy valmaxy]).

Exemplul I.2.6: să se realizeze graficul funcțiilor sinus şi cosinus în intervalul 0 - 360°.

```
x=0:pi/50:2*pi;
y1=sin(x);
y2=cos(x);
xgrd=x.*(180/pi) ;
plot(xgrd, y1, 'r+', xgrd, y2, 'go'), grid,...
 title('Graficul functiilor sinus si cosinus'),...
xlabel('valoare unghiului[grade]'),ylabel('valoarea functiei');
```

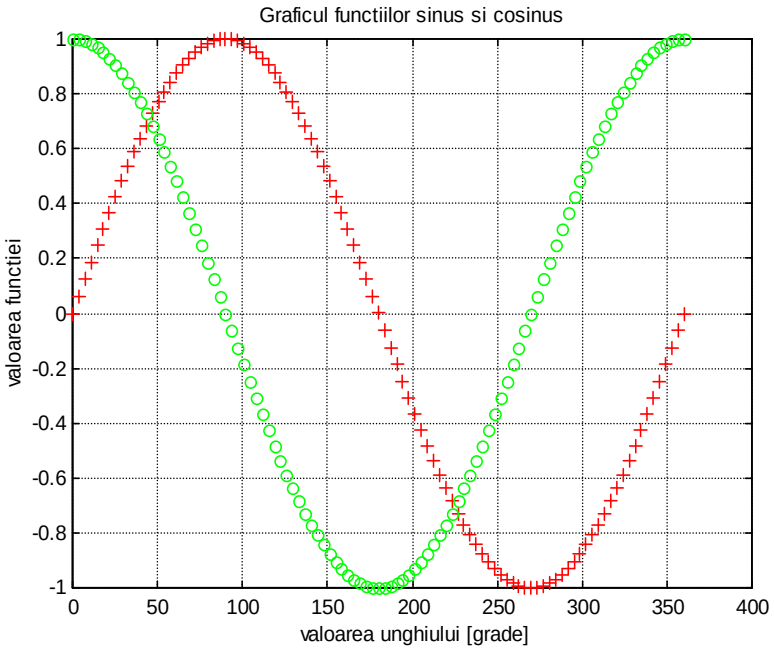

Figura I.2.1 Eemplu de utilizare a instrucțiunii « plot » : graficul funcțiilor sinus şi cosinus.

NOTA: în cazul de mai sus, fiindcă sin şi cos sunt funcții din biblioteca MATLAB, se poate folosi şi forma: *plot(xgrd, sin(x), 'r+', xgrd, cos(x), 'go')*.

In cazul tridimensional, se foloseşte instrucțiunea *plot3*.

Exemplul I.2.7: să se construiască graficul funcției z=x*exp(-x^2-y^2), unde x,y⊂[-2,2];

Vectorii ce conțin valorile luate în considerare pentru x şi y vor fi definiți cu instrucțiunea *linspace.* Instrucțiunea *« [X,Y]=meshgrid(x,y) »* transformă vectorii x,y în matricile X,Y, necesare pentru definirea punctelor (x_i,y_i) pentru care se va face evaluarea lui z în vederea reprezentării grafice (liniile matricii X sunt copii ale vectorului x iar coloanele matricii Y sunt copii ale vectorului y) :

```
x=linspace(-2,2,20);
y=linspace(-2,2,20);
[X,Y]=meshgrid(x,y);
Z=X.*exp(-X.^2-Y.^2);
plot3(X,Y,Z,'go'),grid;
```

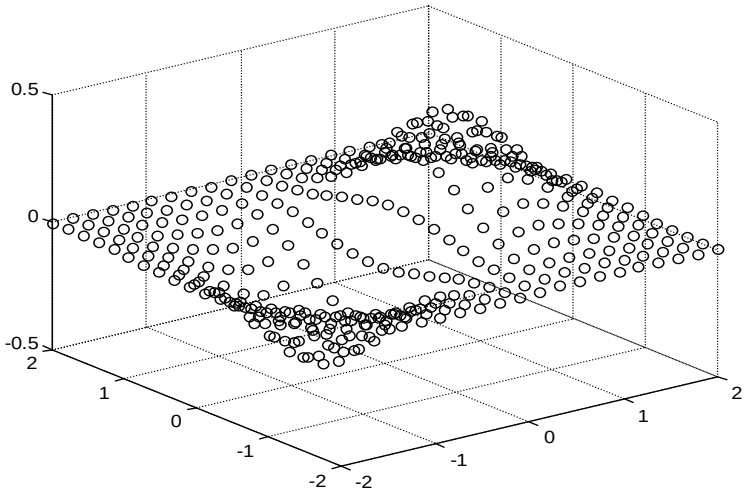

Figura I.2.2. Exemplu de utilizare a instrucțiunii plot3 : graficul funcției $z=x*e^{-x\wedge 2-y\wedge 2}$.

Suprafața spatială de răspuns se poate construi şi cu instrucțiunea *surfl*(X,Y,Z) iar contururile de valoare constantă cu instrucțiunea *contour3*(X,Y,Z,nr_de_contururi). In cazul în care se dispune de valorile unei variabile în funcție de valorile altora două , dacă acestea din urmă sunt plasate echidistant, se poate completa setul de date cu instrucțiunea *interp2*.

Exemplul I.2.8: pornind de la vectorii x, y şi matricea td (a se vedea tabelul III.1.1), se completează prin interpolare setul de date realizând pe axele x şi y un increment de 0.2 în loc de 1 ;"*zi*" reprezintă rezultatul interpolării. Instrucțiunile *shading interp* şi *colormap(gray)* vor duce la realizarea suprafeței spațiale sub forma de "lumini-umbre" utilizând setul de culori "gray" (figura I.2.3).

Tabelul I.1.1 Tabloul de date referitor la variabilele x,y si td :

y \ x	1	2	3	4	5
1	82	81	80	82	84
2	79	63	61	65	81
3	84	84	82	85	86

```
[1 2 3 4 5];
y=[1 2 3];
td=[ 82 81 80 82 84
     79 63 61 65 81
     84 84 82 85 86];
xi=1:.2:5;yi=1:.2:3;
zi=interp2(x,y',td,…
xi,yi',’cubic');
surfl(xi,yi,zi);
shading interp;
colormap(hot);
```

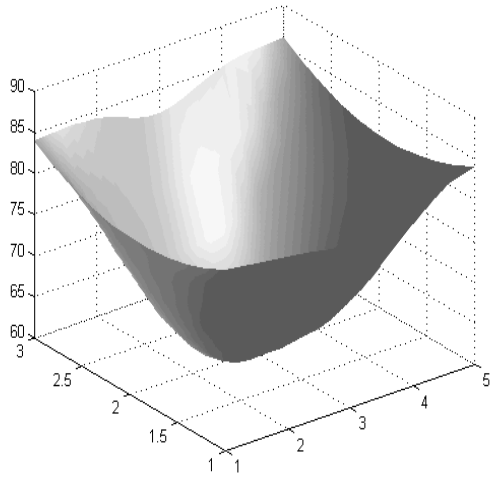

Figura I.2.3 Reprezentarea grafica a variabilei « td » în functie de « x » si « y » (tabelul I.1.1).

Explicații referitoare la alte instrucțiuni şi funcții ale MATLAB, inclusiv cele conținute în « toolbox »-uri, vor fi date pe măsură ce acestea vor fi utilizate în textul cărții.

I.3 SOLUTIONAREA SISTEMELOR DE ECUATII ALGEBRICE NELINIARE

Una din cele mai cunoscute metode de soluționare a ecuațiilor neliniare este metoda Newton-Raphson, metodă bazată pe proprietățile tangentei la curbă. Algoritmul Newton[13], prezentat în continuare, se bazează pe generalizarea metodei Newton – Raphson pentru sistemele de ecuații neliniare.

I.3.1 Algoritmul Newton

a. Cazul unui sistem neliniar de două variabile .

Fie sistemul :

$$\begin{cases} f_1(x_1, x_2) = 0 \\ f_2(x_1, x_2) = 0 \end{cases} \qquad \text{(I.3.1)}$$

si fie x_1^0 si x_2^0 aproximația inițială a variabilelor x_1 şi x_2.

Noile aproximații :

$$x_1^1 = x_1^0 + \Delta x_1^0$$
$$x_2^1 = x_2^0 + \Delta x_2^0$$

trebuie astfel alese încât ele să apropie de zero valorile funcțiilor $f_1 si f_2$:

$$\begin{cases} f_1(x_1^1, x_2^1) = 0 \\ f_2(x_1^1, x_2^1) = 0 \end{cases}$$

adică :

$$\begin{cases} f_1(x_1^0 + \Delta x_1^0, x_2^0 + \Delta x_2^0) = 0 \\ f_2(x_1^0 + \Delta x_1^0, x_2^0 + \Delta x_2^0) = 0 \end{cases} \quad \text{(I.3.2)}$$

Dezvoltând sistemul I.3.2 în serie Taylor, se obține:

$$\begin{aligned} f_1(x_1^0, x_2^0) + \left(\frac{\partial f_1}{\partial x_1}\right)^0 .\Delta x_1^0 + \left(\frac{\partial f_1}{\partial x_2}\right)^0 .\Delta x_2^0 + ... = 0 \\ f_2(x_1^0, x_2^0) + \left(\frac{\partial f_2}{\partial x_1}\right)^0 .\Delta x_1^0 + \left(\frac{\partial f_2}{\partial x_2}\right)^0 .\Delta x_2^0 + ... = 0 \end{aligned} \quad \text{(I.3.3)}$$

Prin neglijarea termenilor ce conțin $\Delta x_1^0 si \Delta x_2^0$ la puteri mai mari decât 1 ,se obține un sistem de două ecuații liniare cu două necunoscute $\Delta x_1^0 si \Delta x_2^0$ .Prin rezolvarea sistemului ,se obțin noile aproximări ale variabilelor: $x_1^1 si x_2^1$. Procedeele se repetă până când se atinge convergența .Unul din criteriile de convergență utilizate este :

$$\sqrt{\left(\Delta x_1^k\right)^2 + \left(\Delta x_2^k\right)^2} < \varepsilon \quad \text{(I.3.4)}$$

unde k este numărul iterației iar ε este o valoare pozitivă aleasă în funcție de precizia dorită de calcul.

b. Cazul unui sistem de „n" ecuații.

Fie: $f_i(x_1, x_2, x_3, \ldots, x_n) = 0$ i=1,2,3,....,n

sistemul de n ecuații neliniare.

In forma matricială:

$$\varphi(X)=0 \qquad \text{(I.3.5)}$$

unde $\varphi=\left[f_1, f_2, \ldots . f_n\right]^T$,$X=\left[x_1, x_2, \ldots x_n\right]^T$

Fie X^{k+1} valoarea lui X la a « k+1 » iterație:

$$X^{k+1}=X^k+\Delta X^k$$

Daca X^{k+1} este valoarea cautată, atunci $\varphi\left(X^{k+1}\right)=0$ sau altfel :

$$\varphi\left(X^k+\Delta X^k\right)=0 \qquad \text{(I.3.6)}$$

Prin dezvoltare în serie Taylor, se obține:

$$\varphi\left(X^k+\Delta X^k\right)=\varphi\left(X^k\right)+\nabla\varphi\left(X^k\right).\Delta X^k+\ldots . \qquad \text{(I.3.7)}$$

unde ∇ este operatorul derivatelor parțiale de ordinul întâi în raport cu componentele lui X . Prin neglijarea termenilor ce conțin $\left(\Delta X^k\right)^2$ și superiori,se obține, ținând cont de (I.3.6):

$$\varphi\left(X^k\right)+J_k.\Delta X^k=0 \qquad \text{(I.3.8)}$$

unde $J_k=\nabla\varphi\left(X^k\right)$ este numită matrice Jacobiana:

$$J_k=\begin{bmatrix} \dfrac{\partial f_1(X)}{\partial x_1} & \ldots\ldots\ldots & \dfrac{\partial f_1(X)}{\partial x_n} \\ \ldots\ldots & \ldots\ldots & \ldots\ldots \\ \dfrac{\partial f_n(X)}{\partial x_1} & \ldots\ldots\ldots & \dfrac{\partial f_n(X)}{\partial x_n} \end{bmatrix}$$

Deci : $\Delta X^k=-J_k^{-1}.\varphi\left(X^k\right)$

Prin rezolvarea lui (I.3.8) ,se obține aproximația „mai bună”:

$$X^{k+1}=X^k+\Delta X^k \qquad \text{sau}$$

$$X^{k+1} = X^k - J_k^{-1}\varphi\left(X^k\right) \tag{I.3.9}$$

Dezavantajele principale ale metodei:

- soluția poate să nu conveargă dacă aproximația inițială nu este bună.

-metoda face necesară calculul derivatelor fiecărei funcții în raport cu fiecare din variabile.

Exemplul I.3.1 : să se rezolve sistemul de ecuații neliniare :

$$\begin{aligned} x^3 + x \cdot y^2 &= 9.375 \\ x \cdot y + y^2 &= 7 \end{aligned} \tag{I.3.10}$$

In acest caz, matricea Jacobiana este :

$$J = \begin{bmatrix} 3 \cdot x^2 + y^2 & 2 \cdot x \cdot y \\ y & x + 2 \cdot y \end{bmatrix} \tag{I.3.11}$$

iar ca vector de start vom utiliza $X=[1\ 1]^T$.

Vectorul funcțiilor şi matricea Jacobiana se definesc fiecare în câte un fişier de tip funcție .

a) Vectorul funcțiilor « f_i » ale sistemului de ecuații :

```
function F=fsysex(X)
x=X(1);y=X(2);
F=zeros(2,1);
F(1)=x^3+x*y^2-9.375;
F(2)=x*y+y^2-7;
```

b). Matricea Jacobiana :

```
function JF=jfsysex(X)
x=X(1);y=X(2);
JF=zeros(2,2);
JF(1,:)=[3*x^2+y^2 2*x*y];
JF(2,:)=[y x+2*y];
```

Ecuația specifică algoritmului Newton (I.3.9) este implementată printr-un ciclu « while », ieşirea din ciclu fiind condiționată de atingerea toleranței admise :

```
function [vx,it]=nrsysex(X,F,JF,n,tol)
it=0;vx=X;
fr=feval(F,vx);
while norm(fr)>tol
      jr=feval(JF,vx);
      vx1=vx-jr\fr;vx=vx1;
      fr=feval(F,vx);
      it=it+1;
end;
```

Rutina de mai sus utilizează două noi instrucțiuni MATLAB :

- *feval(F,vx)* - evaluează vectorul F (sau matricea JF) utilizând valorile din vectorul « vx » ;
- norm(fr) – returnează cea mai mare valoare absoluta din vectorul « fr ».

Apelarea rutinei Newton se face precizând vectorul de start şi fişierele unde sunt definite vectorul « F » şi matricea Jacobiana « JF » :

```
[solsis iteratii]=nrsysex([1
1]','fsysex','jfsysex',2,0.0001);
x=solsis(1);
y=solsis(2);
disp('solutia sistemului de ecuatii neliniare');
disp('x, y =');disp([x y]);
disp('numar iteratii:');disp(iteratii);
```

iată şi rezultatul apelării :

```
» sisnlex
soluția sistemului de ecuații neliniare
x, y =
   1.5000    2.0000

număr iterații:
   11
```

NOTA : alți algoritmi, cum ar fi de exemplu algoritmul Broyden şi algoritmul Levenberg-Marquart evită definirea Jacobianului prin tehnici de evaluare a acestuia, tehnici specifice fiecărui algoritm. Un exemplu de utilizare a algoritmului Levenberg-Marquart este prezentat în cazul regimului staționar al unei coloane de distilare (solverul « fsolve » al toolbox-ului de « optimizări »).

I.4 SOLUTIONAREA NUMERICA A ECUATIILOR DIFERENTIALE

I.4.1 Ecuații diferențiale ordinare

Rezolvarea numerică a unei ecuații diferențiale de forma generală:

$$f\left(x, y, \frac{dy}{dx}, \ldots \frac{d^n y}{dx^n}\right) = 0 \qquad \text{(I.4.1)}$$

presupune a determina y(x) care, împreună cu cele n derivate ale sale, satisfac condiția (I.4.1).

Pentru aflarea soluției unice, este necesar să se cunoască valorile lui y şi ale primelor sale n - 1 derivate într-un anumit punct. Când acestea sunt cunoscute în punctul inițial, x_0, suntem în cazul unei probleme cu condiții inițiale (problema Cauchy). In altă situație, vorbim de o problemă cu condiții la limită. Soluția numerică constă în a calcula (aproxima), pe baza unui algoritm specific fiecărei metode numerice, valorile lui y pentru diverse valori ale variabilei independente x.

I.4.1.1 Metode explicite

a. *Metoda Euler*

Una din cele mai simple metode de integrare numerică este metoda Euler[10].
Fie ecuația diferențialã de ordinul I şi condiția la limită:

$$\begin{cases} \dfrac{dy}{dx} = f(x, y) \\ y(x_0) = y_0 \end{cases} \qquad \text{(I.4.2)}$$

Dezvoltarea în serie Taylor:

$$y(x + \Delta x) = y(x) + \Delta x \cdot y'(x) + \frac{\Delta x^2}{2} \cdot y''(x) + \frac{\Delta x^3}{3!} \cdot y'''(x) + \ldots \qquad \text{(I.4.3)}$$

poate fi scrisă pentru $x = x_i$:

$$y_{i+1} = y_i + \Delta x \cdot f_i + \frac{\Delta x^2}{2} \cdot f_i' + \frac{\Delta x^3}{3!} \cdot f_i'' + \ldots \qquad \text{(I .4.4)}$$

Neglijând termenii în Δx^2 şi superiori se ajunge la formula lui Euler :

$$y_{i+1} = y_i + \Delta x \cdot f_i \qquad \text{(I.4.5)}$$

Exemplul I.4.1 : ecuația diferențiala care descrie dinamica evoluției nivelului (h(t)) într-un rezervor cu scurgere liberă în cazul modificării debitului de intrare (în forma q(t)) este (T_t- constanta de timp şi K – coeficientul de transfer) :

$$T_t*h'(t) + h(t)=K*q(t) \qquad \text{(I.4.6)}$$

Ne propunem să determinăm evoluția nivelului pentru o modificare sub forma de semnal treaptă a debitului : q(t)=0.2 (valorile constantelor - T=20, K=2). La t=0, h(0)=0.5 iar limita superioară în timp până la care se urmăreşte evoluția lui h este de 100 secunde.

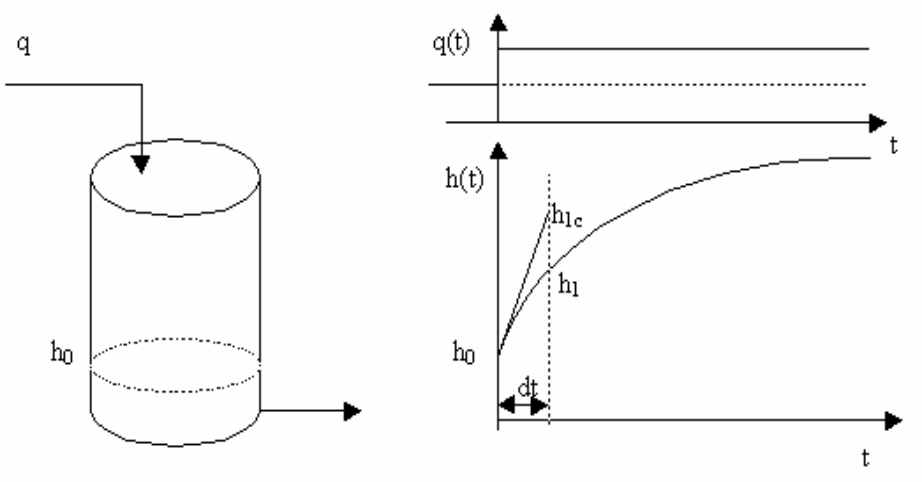

Figura I.4.2 Rezervor hidraulic cu scurgere liberă : evoluția nivelului ca urmare a modificării debitului sub forma de semnal treaptă. Principiul metodei Euler.

In acest caz, expresia funcției f din ecuația I.4.2 este :

$$f(h, t)= (K*q(t)-h(t))/T_t \qquad \text{(I.4.7)}$$

Dacă se face pe axa t un pas de valoare dt, deci t= dt, conform metodei Euler se poate calcula (aproxima) valoarea lui h cu relația:

$$h(dt) = h(0) + \left(\frac{dh(t)}{dt} \right)_{t=0} \cdot dt \qquad \text{(I.4.8)}$$

$$h_1 = h_0 + f(h_0, dt) \cdot dt \qquad \text{(I.4.9)}$$

Dacă pasul dt este suficient de mic, valoarea calculată, h_{1c}, va fi foarte aproape de valoarea adevărată h_1.
Diferența $\varepsilon = h_{1c} - h_1$ este eroarea de integrare; în cazul metodei Euler, eroarea de integrare este proporționalã cu dt^2. Se poate deci remarca importanța alegerii corecte a pasului de integrare : cu cât acesta este mai mic, cu atât precizia de calcul este mai buna dar şi timpul de calcul devine mai mare.

Generalizând, la pasul n + 1 se obține:

$$h_{n+1} = h_n + f(h_n, t_n) \cdot dt \qquad \text{(I.4.10)}$$

$$\text{unde :} \quad t_n = n \cdot dt$$

Programul de mai jos, construit pe baza unui ciclu « while », reține valorile succesive ale timpului şi nivelului în vectorii hvect şi tvect. Algoritmul de soluționare numerică este prezentat în figura I.4.2.

```
%val. initiale, pas, contor, timp final
h0=0.5 ;h=0;t=0;dt=2;k=1;tf=100 ;
% Coef.transf., const. de timp,  val. salt
K=2 ;Tt=20 ;dq=0.2 ;
while   t<=tf
       % derivata
        hder=(dq*K-h)/Tt ;
        htot=h0+h;hvect(k)=htot;
        tvect(k)=t;
       % Euler :
        h=h+hder*dt;
        t=t+dt;
        k=k+1;
end;
plot(tvect,hvect,'*');
```

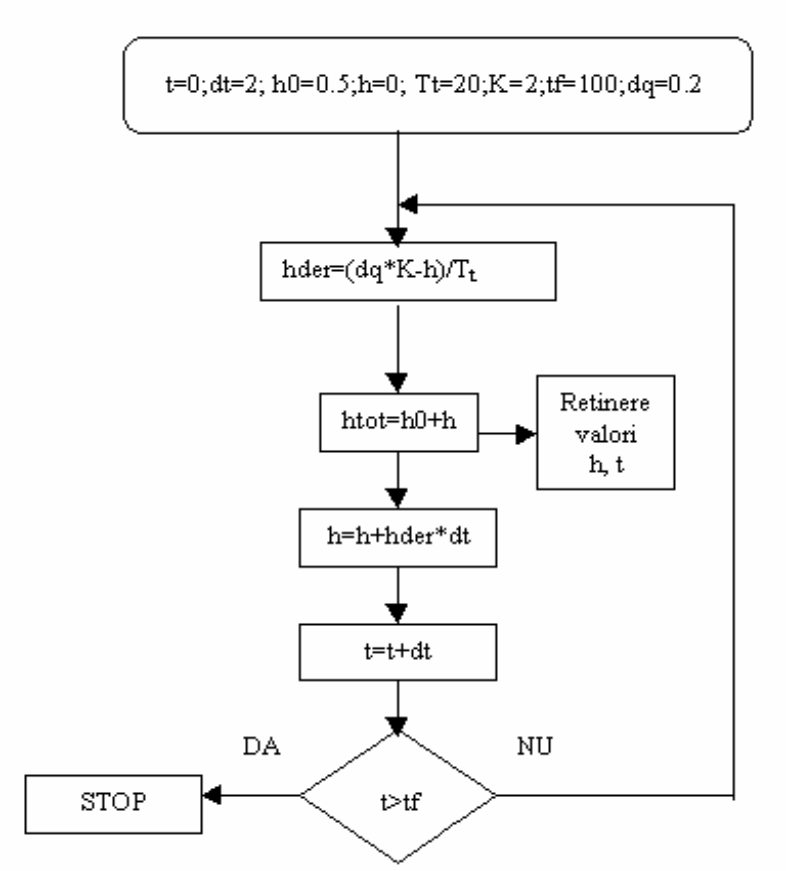

Figura I.4.3 Ordinograma de soluționare prin metoda Euler a ecuației I.4.3

In figura I.4.4 sunt prezentate, comparativ, rezultatele obținute prin utilizarea a doi paşi de integrare diferiți(2 şi 10). Se pot constata erorile generate de un pas de integrare mare .

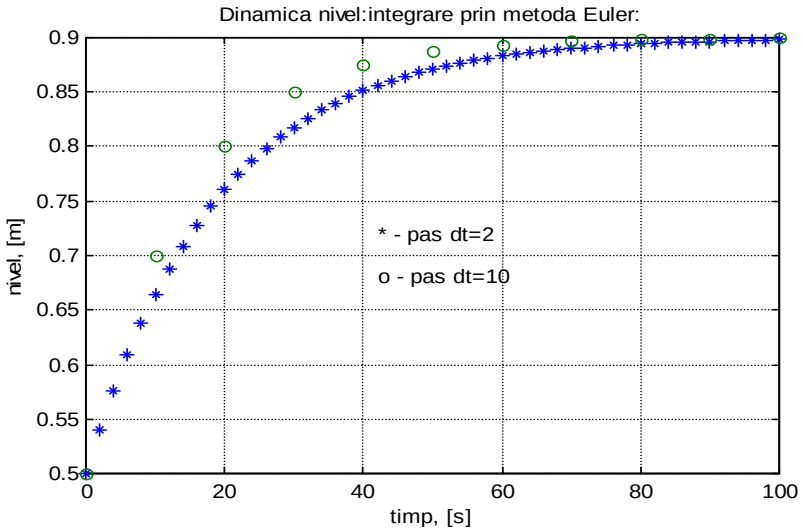

Figura I.4.4. Efectul pasului de integrare asupra preciziei metodei Euler.

Eroarea globala în cazul metodei Euler.

Pentru o corectă alegere a pasului de integrare, să examinăm problema stabilității soluției pentru ecuația diferențială :

$$\frac{dy}{dx} = a \cdot y \qquad \text{(I.4.11)}$$

cu condiția : pentru $x=0$, $y(0)=y_0$

Pentru o valoare Δx a pasului, valoarea lui y estimată prin algoritmul Euler este :

$$y_{i+1}^c = y_i^c + \Delta x \cdot a \cdot y_i^c \qquad \text{(I.4.12)}$$

şi deci :

$$y_{i+1}^c = (1+\Delta x \cdot a) \cdot y_i^c = (1+\Delta x \cdot a)^2 \cdot y_{i-1}^c = \ldots = (1+\Delta x \cdot a)^{i+1} \cdot y_0 \qquad \text{(I.4.13)}$$

Soluția analitică :

$$y(x_{i+1}) = y_0 \cdot e^{a \cdot x_{i+1}} = y_0 \cdot e^{(1+i) \cdot \Delta x \cdot a} \qquad \text{(I.4.14)}$$

Cu alte cuvinte, se aproximează $e^{a\cdot\Delta x}$ prin $(1+a\cdot\Delta x)$. Eroarea de integrare se va compune deci din două parți [10,15]:

- una cauzată de aproximarea exponențialei prin primul termen al expansiei în serie Taylor ;
- o a doua parte cauzată de propagarea erorii de la un pas la altul .

Din acest motiv, pentru o soluție stabilă, este necesar ca :

$$|(1+\Delta x\cdot a)|\leq 1 \qquad \text{(I.4.15)}$$

Pasul Δx trebuie să fie deci mai mic decat *2/a.* Acuratețea soluției necesită însă paşi de cateva ori mai mici decât această valoare.

In cazul sistemelor de ecuații diferențiale de ordinul I, metoda Euler îşi păstrează simplitatea. Fie de exemplu sistemul:

$$\begin{aligned}\frac{dy_1(x)}{dx} &= f_1(y_1, y_2, x)\\ \frac{dy_2(x)}{dx} &= f_2(y_1, y_2, x)\end{aligned} \qquad \text{(I.4.16)}$$

relațiile de recurentă pentru calculul valorilor succesive pentru y_1 şi y_2 sunt:

$$\begin{aligned}y_{1,n+1} &= y_{1n} + \Delta x\cdot f_1(y_{1n}, y_{2n}, x_n)\\ y_{2,n+1} &= y_{2n} + \Delta x\cdot f_2(y_{1n}, y_{2n}, x_n)\end{aligned} \qquad \text{(I.4.17)}$$

NOTA : în cazul în care condițiile la limită nu sunt precizate doar pentru $x=x_0$, (de exemplu pentru y_2 se precizează valoarea la $x=x_F$ – limita superioară a intervalului de integrare – integrarea trebuie să inceapă tot de la $x=x_0$ dar cu o valoare presupusă pentru y_2 ; în cazul în care la $x=x_F$ diferența dintre valoarea lui y_2 rezultată din calcul şi cea reală depaşeşte eroarea admisă, se corecteaza valoarea inițial presupusă pentru y_2 la $x=x_0$) .

In cazul sistemelor descrise de ecuații diferențiale liniare de ordin superior, acestea pot fi reduse, prin substituții adecvate, la un sistem de ecuații diferențiale de ordinul I.

Exemplul I.4.2: Fie ecuația diferențială de ordinul I:

$$a_2\cdot y''(t) + a_1\cdot y'(t) + a_0\cdot y(t) = i(t) \qquad \text{(I.4.18)}$$

cu condițiile la limită :

pentru $t=0, \quad y(0)=0 ; \quad y'(0)=0$ (I.4.19)

Se definesc două noi variabile:

$$y_1(t) = y(t)$$
$$y_2(t) = y'(t)$$

Ecuatia (I.4.18) devine:

$$a_2 \cdot y'_2(t) + a_1 \cdot y_2(t) + a_0 \cdot y_1(t) = i(t)$$

Si deci sistemul de mai jos este echivalent cu ecuația I.4.18:

$$\begin{cases} y'_1(t) = y_2(t) \\ y'_2(t) = \dfrac{1}{a_2}\left[i(t) - a_1 \cdot y_2(t) - a_0 \cdot y_1(t)\right] \end{cases} \quad \text{(I.4.20)}$$

Condițiile la limită pentru sistemul I.4.20:

pentru $t=0, \quad e_1(0)=0 ; \; e_2(0)=0$ (I.4.21)

b. *Metoda Runge-Kutta*

Dintre algoritmii de integrare de tip Runge – Kutta, cel mai cunoscut este algoritmul Runge-Kutta de ordinul IV numit simplu algoritmul Runge-Kutta[10]. In acest caz, la fiecare pas de integrare se fac patru evaluări ale derivatei, în final utilizându-se pentru calculul valorii funcției în noul punct o medie ponderată a valorilor obținute pentru derivată. Avantajul metodei este dat de precizia ei, eroarea de integrare fiind proporțională cu puterea a 4-a a intervalului de integrare.

Pentru o ecuație diferențială de ordinul I:

$$\frac{dy(t)}{dx} = f(x, y) \quad \text{(I.4.22)}$$

cu condiția inițială: $y(x_0) = y_0$

Derivata se evaluează atât la capetele intervalului Δx (în punctele x_i şi x_{i+1}) cât şi – de două ori - la mijlocul intervalului (în punctul $x_i+\Delta x/2$). Relația de recurență este urmatoarea [10]:

$$y_{i+1} = y_i + (k_1 + 2k_2 + 2k_3 + k_4)/6 \quad \text{(I.4.23)}$$

unde:

$k_1 = \Delta x \cdot f(x_i, y_i)$
$k_2 = \Delta x \cdot f(x_i + \Delta x/2, y_i + k_1/2)$
$k_3 = \Delta x \cdot f(x_i + \Delta x/2, y_i + k_2/2)$
$k_4 = \Delta x \cdot f(x_i + \Delta x, y_i + k_3)$

Programul de mai jos soluționeaza prin metoda Runge-Kutta ecuația diferențiala I.4.6 în aceleaşi condiții cu cele prezentate în cadrul metodei Euler :

```
h0=0.5 ;h=0;t=0;dt=10;k=1;tf=100 ;
K=2 ;Tt=20 ;dq=0.2 ;
while   t<=tf
   htot=h0+h;hvect(k)=htot;
   tvect(k)=t;
   k1=dt*(dq*K-h)/Tt;
   k2=dt*(dq*K-(h+k1/2))/Tt;
   k3=dt*(dq*K-(h+k2/2))/Tt;
   k4=dt*(dq*K-(h+k3))/Tt;
   h=h+(k1+2*k2+2*k3+k4)/6;
   t=t+dt;
   k=k+1;
end;
```

Referindu-ne din nou la ecuația I.4.11, valoarea calculată a lui y la pasul i+1 este[10, 14] :

$$y_{i+1}^c = (1 + a \cdot \Delta x + \frac{a^2 \cdot \Delta x^2}{2} + \frac{a^3 \cdot \Delta x^3}{6} + \frac{a^4 \cdot \Delta x^4}{24}) \cdot y_i^c \qquad \text{(I.4.25)}$$

Expresia din paranteză corespunde primilor patru termeni din dezvoltarea în serie Taylor a funcției $e^{a \cdot h}$, eroarea de aproximare fiind deci mai mică decât în cazul algoritmului Euler.

Metoda Runge-Kutta poate fi generalizată şi pentru un sistem de *n* ecuații diferențiale:

$$\begin{cases} \frac{dy_1}{dx} = f_1(x, y_{1,} y_2, ..., y_n) \\ ..... \\ \frac{dy_n}{dx} = f_n(x, y_1, y_2, ..., y_n) \end{cases} \qquad \text{(I.4.26)}$$

cu condițiile inițiale:

$$\begin{cases} y_1(0) = y_{10} \\ ..... \\ y_n(0) = y_{n0} \end{cases} \qquad \text{(I.4.27)}$$

sau, altfel:

$$\frac{dy_j}{dx} = f_j\left(x, y_1, \ldots\ldots, y_n\right) \qquad \text{(I.4.28)}$$

cu condițiile la limită : $y_j(0) = y_{j0}$ $\qquad j = 1, \ldots\ldots, n$

Folosind exprimarea vectorială (Y, K si F), algoritmul Runge-Kutta are forma:

$$Y_{i+1} = Y_i + (K_1 + 2K_2 + 2K_3 + K_4)/6 \qquad \text{(I.4.29)}$$

Solverele MATLAB pentru ecuații diferențiale ordinare (sau pentru sisteme de ecuații) folosesc diverşi algoritmi de soluționare (Runge-Kutta de diverse ordine, Runge-Kutta Fehlberg, etc.) putându-se opta şi pentru pas variabil – situație în care valoarea pasului este ajustată în funcție de valoarea gradientului : la un gradient mic, pasul este mărit iar la un gradient mare este micşorat. Solverele folosesc drept argumente obligatorii vectorul derivatelor – vector ce trebuie definit într-un fisier de tip « function », valoarea inițială şi valoarea finală ale variabilei x, şi vectorul valorilor inițiale ale necunoscutelor y.

Exemplul I.4.3 :Ecuația comportării dinamice a unui schimbător de căldură (« *y* » – temperatura produsului la ieşire iar « *i* » - debitul de apă de racire) este :

$$10 \cdot y''(t) + 7 \cdot y'(t) + y(t) = K \cdot i(t) \qquad \text{(I.4.30)}$$

Să se determine evoluția temperaturii în cazul modificării debitului sub forma de semnal treaptă unitară (i(t) = 1, K=10).

Utilizând substituțiile :

$$y_1(t) = y(t)$$
$$y_2(t) = y'(t)$$

ecuația I.4.23 este echivalentã cu sistemul:

$$\begin{cases} y'_1(t) = y_2(t) \\ y'_2(t) = \left[\left(10 - y_1(t) - 7 \cdot y_2(t)\right)\right]/10 \end{cases} \qquad \text{(I.4.31)}$$

Condițiile inițiale: la $t = 0$, $y(0) = 30$ şi $y'(0) = 0$ integrarea având loc până la $t_f = 50$.

Definirea derivatelor se face într-un fişier de tip funcție :

```
function dery=vectdery(t,vecty)
dery=zeros(2,1);
y1=vecty(1) ;
y2=vecty(2) ;
dery(1)=y2 ;
dery(2)=(10-y1-7*y2)/10 ;
```

Programul principal începe prin precizarea valorilor pe axa timpului pentru care se doreşte să se obțină valorile vectorului Y (vecty) ; se precizează valorile inițiale ale lui Y (y10 si y20) şi valoarea staționară inițială (ys) iar apelarea solverului ode45 (Runge–Kutta Fehlberg) este extrem de simplă . In vederea reprezentării grafice, la valorile y_1 se adaugă valoarea de regim stationar :

```
TSPAN=0:1:50;
ys=30; y10=0 ;y20=0;
[t,vecty]=ode45('vectdery',TSPAN,[y10 y20]');
y1=vecty(:,1)+ys;
y2=vecty(:,2);
plot(t,y1);
```

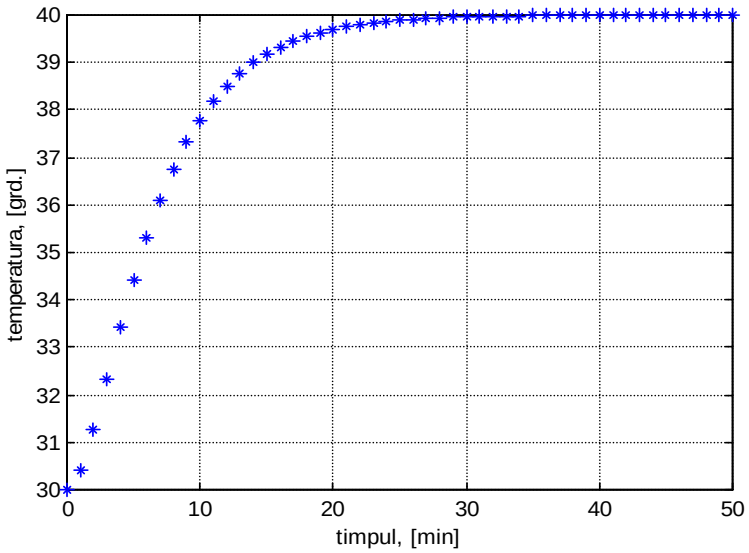

Figura I.4.5 Evoluția temperaturii (ecuația diferențială I.4.30) pentru semnal treaptă unitară (negativ) pe debit de apa de racire .Soluționare a ecuației diferențiale prin utilizarea solverului « ode45 ».

I.4.1.2 Metode implicite

I.4.1.2.1 Metode predictor – corector

La metodele prezentate până acum soluția se obtine pe baza informațiilor într-un singur punct. Referindu-ne la ecuația I.4.2, în cazul metodelor implicite (numite şi indirecte) se folosesc şi informații despre y şi despre derivata sa şi din afara intervalului de integrare considerat (x_i, x_{i+1}). Marea majoritate a metodelor din această categorie sunt de tipul *predictor-corector* [10]: valoarea lui y_{i+1} este prezisă (cu o metodă directă) după care urmează faza de corectie a rezultatului obtinut.

Pentru *prezicere* se foloseşte :

$$y_{i+1}^{(0)} = y_{i-1} + 2 \cdot \Delta x \cdot f(x_i, y_i) \tag{I.4.32}$$

Acest lucru ar însemna că metoda nu poate fi folosită pentru calculul lui y_1, fiind necesar un punct anterior lui x_0. Pentru pornire se poate utiliza una din metodele explicite prezentate anterior.

Corecția: noua valoare a lui y_{i+1} se calculează cu:

$$y_{i+1}^{(1)} = y_i + \frac{\Delta x}{2} \cdot \left[f(x_i, y_i) + f\left(x_{i+1}, y_{i+1}^{(0)}\right) \right] \tag{I.4.33}$$

Cu noua valoare y_{i+1} se poate calcula o nouă valoare a funcției. In general a k-a aproximare a lui y_{i+1} este dată de:

$$y_{i+1}^{(k)} = y_i + \frac{\Delta x}{2} \cdot \left[f(x_i, y_i) + f\left(x_{i+1}, y_{i+1}^{(k-1)}\right) \right] \tag{I.4.34}$$

Interațiile se opresc când: $\left| y_{i+1}^{(k+1)} - y_{i+1}^{(k)} \right| < \varepsilon$ (I.4.35)

unde ε este eroarea de integrare admisibilă.

Metodele implicite fiind iterative, volumul de calcule pentru fiecare pas de integrare este mai mare decât în cadrul metodelor explicite. Ele sunt însă mult mai precise decât formulele deschise de acelaşi ordin. De asemenea, ele sunt şi mai stabile. Convergența este cu atât mai rapidă cu cât estimarea inițială $y_{i+1}^{(0)}$ este mai aproape de valoarea exactă y_{i+1}.

I.4.1.2.2 Scheme cu diferențe.

Fie y(x) o funcție definită în intervalul [a,b]. Divizăm intervalul în n subintervale de valoare :

$$\Delta x = \frac{b-a}{n}$$

Fie $y_1, y_2, ..., y_{i-1}, y_i, y_{i+1}, ..., y_n$ valorile funcției în centrele celor n subintervale.
Se pot defini trei tipuri de diferențe : regresive, progresive şi centrale.

Diferențe regresive (la stânga).

Prima diferență regresivă în punctul i :

$$\nabla y_i = y_i - y_{i-1} \qquad \text{(I.4.36)}$$

a doua diferență regresivă :

$$\nabla(\nabla y_i) = \nabla^2 y_i = (y_i - y_{i-1}) - (y_{i-1} - y_{i-2}) = y_i - 2 \cdot y_{i-1} + y_{i-2} \qquad \text{(I.4.37)}$$

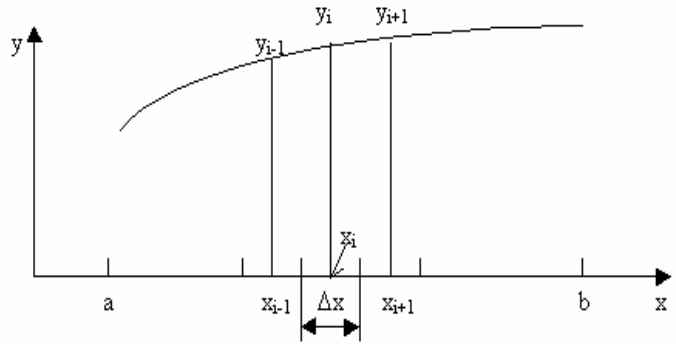

Figura I.4.6 Divizarea intervalului de definiție a funcției y(x) în « n » subintervale.

Diferențe progresive (la dreapta)

Prima diferență progresivă în punctul i :

$$\Delta y_i = y_{i+1} - y_i \qquad \text{(I.4.38)}$$

a doua diferență progresivă :

$$\Delta(\Delta y_i) = \Delta^2 y_i = (y_{i+2} - y_{i+1}) - (y_{i+1} - y_i) = y_{i+2} - 2 \cdot y_{i+1} + y_i \qquad \text{(I.4.39)}$$

Diferențe centrale.

Prima diferență centrală în punctul i :

$$\delta y_i = y_{i+0.5} - y_{i-0.5} \tag{I.4.40}$$

a doua diferență centrală :

$$\delta^2 y_i = \delta(\delta y_i) = (y_{i+1} - y_i) - (y_i - y_{i-1}) = y_{i+1} - 2 \cdot y_i + y_{i-1} \tag{I.4.41}$$

I.4.1.2.3 Metoda matricială.

Ecuațiile diferențiale de ordinul I (sau sistemele de astfel de ecuații) pun adesea probleme dificile din punct de vedere al soluționării numerice. Una din metodele posibile de soluționare constă în transformarea acestor ecuații în ecuații cu diferențe, problema transformându-se astfel într-o problemă de soluționare al unui sistem de ecuații liniare sau neliniare, în funcție de forma ecuației.

Fie ecuația diferențială de ordinul II şi condițiile la limită:

$$\begin{aligned} &A(x) \cdot \frac{d^2 y}{dx^2} + B(x) \cdot \frac{dy}{dx} + C(x) \cdot y = D(x) \\ &y(0) = y_0 \\ &Y(L) = y_L \end{aligned} \tag{I.4.42}$$

Cele două derivate pot fi scrise în forma:

$$\begin{aligned} &\frac{dy}{dx} = \frac{y_{i+1} - y_{i-1}}{2 \cdot \Delta x} \\ &\frac{d^2 y}{dx^2} = \frac{y_{i+1} - 2 \cdot y_i + y_{i-1}}{\Delta x^2} \end{aligned} \tag{I.4.43}$$

Următoarea ecuație cu diferențe înlocuieşte deci ecuația diferențială[14] :

$$\frac{A_i}{\Delta x^2} \cdot \left(y_{i+1} - 2 \cdot y_i + y_{i-1} \right) + \frac{B_i}{2 \cdot \Delta x} \cdot \left(y_{i+1} - y_{i-1} \right) + C_i \cdot y_i = D_i \tag{I.4.44}$$

Prin gruparea termenilor se obține :

$$\left[\frac{A_i}{\Delta x^2} + \frac{B_i}{2 \cdot \Delta x} \right] \cdot y_{i+1} + \left[C_i - \frac{2 \cdot A_i}{\Delta x^2} \right] \cdot y_i + \left[\frac{A_i}{\Delta x^2} - \frac{B_i}{2 \cdot \Delta x} \right] \cdot y_{i-1} = D_i \tag{I.4.45}$$

Adoptând urmatoarele notații :

$$\alpha_i = \frac{A_i}{\Delta x^2} - \frac{B_i}{2 \cdot \Delta x}; \beta_i = C_i - \frac{2 \cdot A_i}{\Delta x^2}; \gamma_i = \frac{A_i}{\Delta x^2} + \frac{B_i}{2 \cdot \Delta x}$$

obținem : $\alpha_i \cdot y_{i-1} + \beta_i \cdot y_i + \gamma_i \cdot y_{i+1} = D_i$ (1.4.46)

Scriind ecuația de mai sus pentru i = 1, 2, ..., n se obține un sistem de n ecuații cu n necunoscute y_1, y_2, ..., y_n. Având in vedere condițiile la limită, prima ecuație (i=1) şi ultima (i=n) au forma (y_0 si y_L sunt valori cunoscute !) :

$$\begin{aligned} &\alpha_1 \cdot y_0 + \beta_1 \cdot y_1 + \gamma_1 \cdot y_2 = D_1 \\ &\alpha_n \cdot y_{n-1} + \beta_n \cdot y_n + \gamma_n \cdot y_L = D_n \end{aligned} \qquad \text{(I.4.47)}$$

Sistemul devine deci (elementele neprecizate sunt 0) :

$$\begin{bmatrix} \beta_1 & \gamma_1 & & & & \\ \alpha_2 & \beta_2 & \gamma_2 & & & \\ & \alpha_3 & \beta_3 & \gamma_3 & & \\ & & & \ddots & & \\ & & & \alpha_{n-1} & \beta_{n-1} & \gamma_{n-1} \\ & & & & \alpha_n & \beta_n \end{bmatrix} \cdot \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{n-1} \\ y_n \end{bmatrix} = \begin{bmatrix} D_1 - \alpha_1 \cdot y_0 \\ D_2 \\ D_3 \\ \vdots \\ D_{n-1} \\ D_n - \gamma_n \cdot y_L \end{bmatrix} \qquad \text{(I.4.48)}$$

Soluționarea ecuației diferențiale se transformă astfel în soluționarea unui sistem de tip tridiagonal, sistem a cărui soluționare nu ridică probleme.

I.4.2 Ecuații cu derivate parțiale.

In ingineria chimică, se întâlnesc frecvent ecuații diferențiale cu derivate parțiale de ordinul II. O ecuație cu derivate parțiale de ordinul II cu coeficienți constanți :

$$A \cdot \frac{\partial^2 y}{\partial x^2} + 2 \cdot B \cdot \frac{\partial^2 y}{\partial x \partial z} + C \cdot \frac{\partial^2 y}{\partial z^2} = D \qquad \text{(I.4.49)}$$

poate fi de tip eliptic, parabolic sau hiperbolic după cum urmează :

- $B^2 - 4AC = 0$ exemplu – legea a doua a difuziei :
  $\frac{\partial C}{\partial t} = D \frac{\partial^2 C}{\partial x^2}$ (parabolică) ;
- $B^2 - 4AC < 0$ exemplu – mişcarea valurilor: $\frac{\partial^2 u}{\partial t^2} = \rho \cdot \frac{\partial^2 u}{\partial x^2}$ (eliptică) ;
- $B^2 - 4AC > 0$ exemplu – ecuația Laplace a transmiterii căldurii prin conducție:

$$\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}=0 \quad \text{(hiperbolică).}$$

Intrucât în paragrafele ce vor urma vom întâlni cu precădere ecuații cu derivate parțiale de tip parabolic, vom prezenta în cele ce urmeaza o tehnică de soluționare utilizată frecvent în soluționarea numerică a acestor ecuații : metoda diferențelor finite.

NOTA : Toolbox-ul MATLAB « Partial differential equations » are la bază metoda elementului finit , utilizarea fişierelor MATLAB de tip « funcție » care compun acest pachet de programe necesită consultarea manualului de utilizare.

I.4.2.1 Scrierea derivatelor parțiale ca diferențe finite parțiale.

Fie funcția y = f(x,z) cu x∈ [a, b] şi z ∈ [c, d]

Cele două intervale sunt împărțite în n şi respectiv k segmente egale, de valoare Δx şi respectiv Δz.

Dacă ne propunem evaluarea derivatelor parțiale într-un nod i,j prin diferențe centrale, se poate scrie:

$$2\cdot\Delta x\cdot\left(\frac{\partial y}{\partial x}\right)_{i,j}=y_{i+1,j}-y_{i-1,j}+2\varepsilon_{1x} \qquad (I.4.50)$$

$$2\cdot\Delta z\cdot\left(\frac{\partial y}{\partial z}\right)_{i,j}=y_{i,j+1}-y_{i,j-1}+2\varepsilon_{1z} \qquad (I.4.51)$$

ε_{1x} şi ε_{1z} fiind erorile pe direcția x, respectiv z.

Derivatele parțiale de ordinul doi pot fi evaluate din relațiile:

$$\Delta x^2\cdot\left(\frac{\partial^2 y}{\partial x^2}\right)_{i,j}=y_{i+1,j}-2y_{i,j}+y_{i-1,j}+\varepsilon_{2x} \qquad (I..4.52)$$

$$\Delta z^2\cdot\left(\frac{\partial^2 y}{\partial z^2}\right)_{i,j}=y_{i,j+1}-2y_{i,j}+y_{i,j-1}+\varepsilon_{2z} \qquad (I.4.53)$$

$$4\cdot\Delta x\cdot\Delta z\cdot\left(\frac{\partial^2 y}{\partial x\partial z}\right)_{i,j}=\left(y_{i+1,j+1}-y_{i-1,j+1}\right)-\left(y_{i+1,j-1}-y_{i-1,j-1}\right)+2\varepsilon_{2x,y} \qquad (I.4.54)$$

ε_{2x}, ε_{2y}, ε_{2z} sunt erorile de aproximare.

I.4.2.2 Metoda diferențelor finite.

Diverse posibilitãti de aproximare a ecuațiilor diferențiale cu derivate parțiale prin ecuații cu diferențe dau posibilitatea construirii unei varietăți de scheme cu diferențe. In alegerea unei astfel de scheme pentru soluționarea unei ecuații diferențiale cu derivate parțiale, trebuie să se aibă în vedere o serie de factori:

- volumul de calcul (dependent de forma sistemului obținut prin discretizare şi de numãrul de puncte în care se face evaluarea);
- eroarea de calcul cât mai mică (metoda sã fie consistentã si stabilă);
- diferența dintre valorile soluției exacte şi a celei numerice să poată fi făcută oricât de mică prin alegerea paşilor rețelei.

Considerând o ecuație diferențiala de ordinul II cu derivate parțiale de forma:

$$\frac{\partial f(x,t)}{\partial t} = A(x)\cdot\frac{\partial^2 f(x,t)}{\partial x^2} + B(x)\cdot\frac{\partial f(x,t)}{\partial x} + C(x)\cdot f(x,t) + D(x) \qquad \text{(I.4.55)}$$

următoarele metode sunt utilizate mai frecvent pentru rezolvarea acestor ecuații :

- metode explicite ;
- metode implicite ;
- metode de tip Crank-Nicholson ;

a).Metode explicite**.**

Presupunem ca la momentul t cunoaştem valorile funcției f(x,t) pe tot intervalul de variatie a variabilei x. Se pot deci calcula derivatele de ordinul I şi II în raport cu variabila x. Dacă se utilizează pentru discretizare diferențe progresive pe axa timp şi diferențe centrale pe axa x se obține [9,14]:

$$\frac{f_{i,j+1} - f_{i,j}}{\Delta t} = A_i\cdot\frac{f_{i+1,j} - 2\cdot f_{i,j} + f_{i-1,j}}{\Delta x^2} + B_i\cdot\frac{f_{i+1,j} - f_{i-1,j}}{2\cdot\Delta x} + C_i\cdot f_{i,j} + D_i \qquad \text{(I.4.56)}$$

De unde :

$$f_{i,j+1} = f_{i,j} + \frac{A_i\cdot\Delta t}{\Delta x^2}\cdot\left(f_{i+1,j} - 2\cdot f_{i,j} + f_{i-1,j}\right) + \frac{B_i\cdot\Delta t}{2\cdot\Delta x}\cdot\left(f_{i+1,j} - f_{i-1,j}\right) +$$
$$+ C_i\cdot\Delta t\cdot f_{i,j} + D_i\cdot\Delta t$$

Deci :

$$f_{i,j+1} = \left[\frac{A_i \cdot \Delta t}{\Delta x^2} + \frac{B_i \cdot \Delta t}{2 \cdot \Delta x}\right] \cdot f_{i+1,j} + \left[1 - \frac{2 \cdot A_i \cdot \Delta t}{\Delta x^2} + C_i \cdot \Delta t\right] \cdot f_{i,j} + \\ + \left[\frac{A_i \cdot \Delta t}{\Delta x^2} - \frac{B_i \cdot \Delta t}{2 \cdot \Delta x}\right] \cdot f_{i-1,j} + D_i \cdot \Delta t \tag{I.4.57}$$

Se poate calcula deci f_i la momentul j+1 cu ajutorul valorilor lui f_{i-1}, f_i şi f_{i+1} la momentul j. Inconvenientul principal al metodei explicite este că ea necesită alegerea unui pas de timp suficient de mic, altfel soluția ecuației devenind instabilă :

$$\left(\frac{2 \cdot A_i}{\Delta x^2} - C_i\right) \cdot \Delta t \le 1 \tag{I.4.58}$$

b).Metode implicite.

Ecuația implicită se obține scriind membrul drept al ecuației inițiale la momentul j+1 când soluția nu este cunoscută [9,14]:

$$\frac{f_{i,j+1} - f_{i,j}}{\Delta t} = A_i \cdot \frac{f_{i+1,j+1} - 2 \cdot f_{i,j+1} + f_{i-1,j+1}}{\Delta x^2} + B_i \cdot \frac{f_{i+1,j+1} - f_{i-1,j+1}}{2 \cdot \Delta x} + C_i \cdot f_{i,j+1} + D_i \tag{I.4.59}$$

sau :

$$f_{i,j+1} = f_{i,j} + \frac{A_i \cdot \Delta t}{\Delta x^2} \cdot \left(f_{i+1,j+1} - 2 \cdot f_{i,j+1} + f_{i-1,j+1}\right) + \frac{B_i \cdot \Delta t}{2 \cdot \Delta x} \cdot \left(f_{i+1,j+1} - f_{i-1,j+1}\right) \\ + C_i \cdot \Delta t \cdot f_{i,j+1} + D_i \cdot \Delta t \tag{I.4.60}$$

Cu alte cuvinte :

$$\left[\frac{A_i \cdot \Delta t}{\Delta x^2} + \frac{B_i \cdot \Delta t}{2 \cdot \Delta x}\right] \cdot f_{i+1,j+1} - \left[1 + \frac{2 \cdot A_i \cdot \Delta t}{\Delta x^2} - C_i \cdot \Delta t\right] \cdot f_{i,j+1} + \\ + \left[\frac{A_i \cdot \Delta t}{\Delta x^2} - \frac{B_i \cdot \Delta t}{2 \cdot \Delta x}\right] \cdot f_{i-1,j+1} = -f_{i,j} - D_i \cdot \Delta t \tag{I.4.61}$$

Se obține deci f_i cunoscut la momentul j ca o combinatie a lui f_{i-1}, f_i şi f_{i+1} necunoscute la momentul j+1. Ecuația anterioară se transformă într-un sistem de

ecuații de tip tridiagonal care poate fi rezolvat uşor utilizând oricare dintre tehnicile numerice de rezolvare a unui sistem de ecuații liniare. Se obişnuieşte să se aleagă ca şi soluție de start la fiecare pas pe axa timpului soluția obținută la pasul anterior.

Avantajul esențial al acestei metode este că ea este universal stabilă.

c). Metode de tip Crank-Nicholson.

In acest caz, metoda constă în a scrie al doilea membru al ecuației principale la momentul j+1/2, exprimându-l ca semi-suma termenilor corespunzători metodelor implicită şi explicită[9,14] :

$$\frac{f_{i,j+1}-f_{i,j}}{\Delta t}=\frac{1}{2}\cdot\left[A_i\cdot\frac{f_{i+1,j}-2\cdot f_{i,j}+f_{i-1,j}}{\Delta x^2}+B_i\cdot\frac{f_{i+1,j}-f_{i-1,j}}{2\cdot\Delta x}+C_i\cdot f_{i,j}+D_i\right]+$$
$$+\frac{1}{2}\cdot\left[A_i\cdot\frac{f_{i+1,j+1}-2\cdot f_{i,j+1}+f_{i-1,j+1}}{\Delta x^2}+B_i\cdot\frac{f_{i+1,j+1}-f_{i-1,j+1}}{2\cdot\Delta x}+C_i\cdot f_{i,j+1}+D_i\right] \qquad \text{(I.4.62)}$$

Rezolvarea ecuației se realizează ca şi în cazul metodei implicite prin transformarea acesteia într-un sistem de ecuații liniare.

Avantajul principal al acestei metode este că pentru o valoare dată a lui Δx, eroarea de trunchiere comisă asupra termenului în Δt este mult mai mică decât în cazul metodelor implicită sau explicită.

d). Exemplul I.4.4..

Fie o bară metalică cu secțiunea unitară, izolată lateral şi la una din extremități. În momentul inițial (τ=0), temperatura este aceeaşi în toată bara: 0°C. Se cere să se determine evoluția temperaturii în lungul barei şi în timp dacă la capătul liber se aplică şi se menține constantă o temperatură de 100°C.

Ecuația cu derivate parțiale care exprimă modificarea temperaturii în timp şi în lungul axei este:

$$\frac{\partial T}{\partial \tau}=\frac{\lambda}{\rho\cdot c}\cdot\frac{\partial^2 T}{\partial x^2} \qquad \text{(I .4.63)}$$

unde: λ = conductivitatea termică;

ρ = densitatea barei;

c = căldura specifică;

şi cu condițiile la limită:

$T(x=0, \tau)=T_0$ $\qquad \left.\frac{\partial T}{\partial x}\right|_{x=1} = 0$ (I.4.64)

Deoarece cele mai pronunțate variații ale temperaturii au loc la capătul liber al barei, precizia soluției numerice se poate îmbunătății printr-o schimbare a dimensiunii secțiunilor, raportul lungimii a două secțiuni consecutive fiind constant[11]:

$$\frac{\Delta x_{i+1}}{\Delta x_i} = R$$

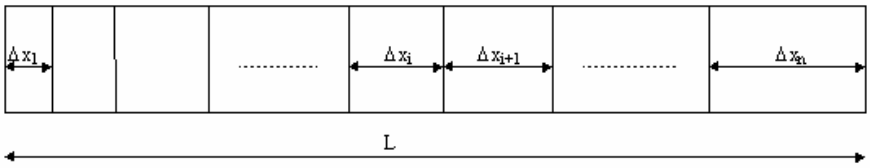

Figura I.4.7 Bară metalică : schimbarea dimensiunilor secțiunilor

Dacă lungimea primei secțiuni este dL, lungimea totală a barei se poate calcula cu relația:

$$L = \sum_{i=1}^{n} dL \cdot R^{i-1} \qquad \text{şi deci: } \Delta x_1 = dL = \frac{L}{\sum_{i=1}^{n} R^{i-1}}$$

Metodă explicită bazată pe calculul temperaturilor în centrul fiecărei secțiuni.

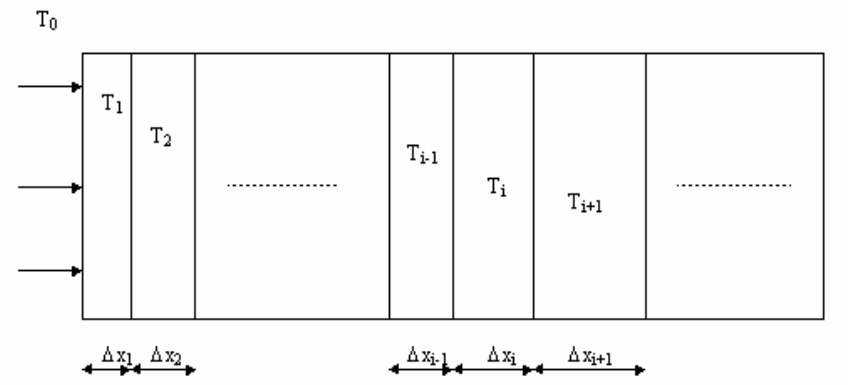

Figura I.4.8 Bară metalică, metoda explicită : discretizare pe baza temperaturilor plasate in centrul secțiunilor.

Pentru o secțiune i, bilanțul termic este (A= suprafața secțiunii transversale):

$$\rho \cdot c \cdot A \cdot \Delta x_i \cdot \frac{dT_i}{d\tau} = \lambda \cdot A \cdot \frac{T_{i-1} - T_i}{0.5 \cdot \Delta x_{i-1} + 0.5 \cdot \Delta x_i} - \lambda \cdot A \cdot \frac{T_i - T_{i+1}}{0.5 \cdot \Delta x_i + 0.5 \cdot \Delta x_{i+1}} \quad \text{(I.4.65)}$$

şi deci:

$$\frac{dT_i}{d\tau} = \frac{2 \cdot \lambda}{\rho \cdot c \cdot (\Delta x_i)^2} \left[\frac{T_{i-1} - T_i}{1 + \frac{1}{R}} - \frac{T_i - T_{i+1}}{1 + R} \right] \quad \text{(I.4.66)}$$

Pentru prima secțiune fluxul de intrare este $\frac{T_0 - T_1}{0.5 \cdot \Delta x_1}$ iar pentru ultima secțiune nu există flux de ieşire. Ecuațiile sunt:

$$\frac{dT_1}{d\tau} = \frac{2 \cdot \lambda}{\rho \cdot c \cdot (\Delta x_1)^2} \cdot \left[T_0 - T_1 - \frac{T_1 - T_2}{1 + R} \right] \quad \text{(I.4.67)}$$

$$\frac{dT_N}{d\tau} = \frac{2 \cdot \lambda}{\rho \cdot c \cdot (\Delta x_n)^2} \cdot \left[\frac{T_{N-1} - T_N}{1 + \frac{1}{R}} \right] \quad \text{(I.4.68)}$$

Ecuațiile (2) - (4) se pot integra pe axa timpului utilizând una din metodele cunoscute (Euler, Runge-Kutta, etc.).

1. Metodă explicită bazată pe calculul temperaturii în nodurile T_0 .. T_N ale rețelei.

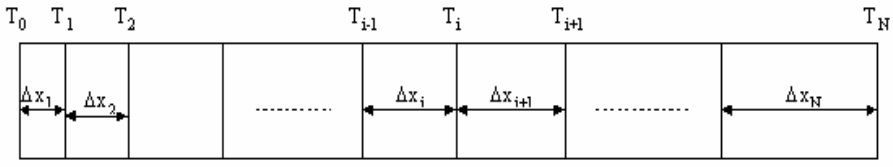

Figura I.4.8 Bară metalică, metoda explicită : discretizare pe baza temperaturilor plasate la finele fiecărei secțiuni.

În cazul în care se utilizează pentru axa x diferențe centrale iar pentru axa timp diferențe progresive, discretizarea ecuației (1) duce, succesiv, la relațiile (k - secvența pe axa timpului, i - secvența pe axa x):

$$\frac{\partial T_i}{\partial \tau} = \frac{\lambda}{\rho \cdot c} \cdot \frac{\partial}{\partial x}\left(\frac{T_{i+0.5}^k - T_{i-0.5}^k}{0.5 \cdot (\Delta x_i + \Delta x_{i+1})} \right) \tag{I.4.69}$$

$$\frac{\partial T_i}{\partial \tau} = \frac{\lambda}{\rho \cdot c} \cdot \frac{\partial}{\partial x}\left[\left(\frac{T_{i+0.5}^k}{0.5 \cdot \Delta x_i \cdot (1+R)} \right) - \frac{\partial}{\partial x}\left(\frac{T_{i-0.5}^k}{0.5 \cdot \Delta x_i \cdot (1+R)} \right) \right] \tag{I.4.70}$$

$$\frac{dT_i}{d\tau} = \frac{\lambda}{\rho \cdot c} \cdot \left[\frac{T_{i+1}^k - T_i^k}{0.5 \cdot \Delta x_i \cdot (1+R) \cdot \Delta x_{i+1}} - \frac{T_i^k - T_{i-1}^k}{0.5 \cdot \Delta x_i \cdot (1+R) \cdot \Delta x_i} \right] \tag{I.4.71}$$

$$\frac{dT_i}{d\tau} = \frac{2 \cdot \lambda}{\rho \cdot c \cdot (\Delta x_i)^2} \cdot \left[\frac{T_{i+1}^k - T_i^k}{R(1+R)} - \frac{T_i^k - T_{i-1}^k}{1+R} \right] \tag{I.4.72}$$

Pentru prima secțiune T_{i-1} devine T_0=100°C, iar pentru ultima secțiune, scriind condiția la limită în forma:

$$\frac{T_{N+1}^k - T_{N-1}^k}{\Delta x_n + \Delta x_{n+1}} = 0 \tag{I.4.73}$$

rezultă că: $T_{N+1}^k = T_{N-1}^k$ şi deci:

$$\frac{dT_n}{d\tau} = \frac{2 \cdot \lambda}{\rho \cdot c \cdot (\Delta x_i)^2} \cdot \left[\frac{T_{N-1}^k - T_N^k}{R \cdot (1+R)} - \frac{T_N^k - T_{N-1}^k}{1+R} \right] \tag{I.4.74}$$

3. Metoda implicită

În cazul metodei implicite, discretizarea pentru secțiunea i este:

$$\frac{T_i^{k+1} - T_i^k}{\Delta \tau} = \frac{2 \cdot \lambda}{\rho \cdot c \cdot (\Delta x_i)^2 \cdot (1+R)} \cdot \left[\frac{T_{i+1}^{k+1} - T_i^{k+1}}{R} - \left(T_i^{k+1} - T_{i-1}^{k+1} \right) \right] \tag{I.4.75}$$

$$T_i^{k+1} - T_i^k = A(i) \cdot \left(\frac{T_{i+1}^{k+1}}{R} - \frac{T_i^{k+1}}{R} - T_i^{k+1} + T_{i-1}^{k+1} \right) \tag{I.4.76}$$

unde: $$A(i)=\frac{2\cdot\lambda\cdot\Delta\tau}{\rho\cdot c\cdot\left(\Delta x_i\right)^2\cdot(1+R)}$$

sau, altfel:

$$-\frac{A_i}{R}\cdot T_{i+1}^{k+1}+\left[1+A_i\cdot\left(1+\frac{1}{R}\right)\right]\cdot T_i^{k+1}-A_i\cdot T_{i-1}^{k+1}=T_i^k \qquad \text{(I.4.77)}$$

Fie: $\gamma_i=-\frac{A_i}{R}$ $\beta_i=1+A_i\cdot\left(1+\frac{1}{R}\right)$ $\alpha_i=-A_i$

Pentru i=1, 2, ..., N-1 avem:

$\alpha_1\cdot T_0+\beta_1\cdot T_1^{k+1}+\gamma_1\cdot T_2^{k+1}=T_1^k$

.....

$\alpha_i\cdot T_{i-1}^{k+1}+\beta_i\cdot T_i^{k+1}+\gamma_i\cdot T_{i+1}^{k+1}=T_i^k$

.....

$\alpha_{N-1}\cdot T_{N-2}^{k+1}+\beta_{N-1}\cdot T_{N-1}^{k+1}+\gamma_{N-1}\cdot T_N^{k+1}=T_{N-1}^k$

Pentru secțiunea N, deoarece $T_{N-1}=T_{N+1}$ avem:

$$T_N^{k+1}-T_N^k=\frac{2\cdot\lambda\cdot\Delta\tau}{\rho\cdot c\cdot\left(\Delta x_N\right)^2\cdot(1+R)}\cdot\left(\frac{T_{N-1}^{k+1}-T_N^{k+1}}{R}-T_N^{k+1}+T_{N-1}^{k+1}\right) \qquad \text{(I.4.78)}$$

$$T_N^{k+1}-T_N^k=\frac{A_N}{R}\cdot T_{N-1}^{k+1}-\frac{A_N}{R}\cdot T_N^{k+1}-A_N\cdot T_N^{k+1}+A_N\cdot T_{N-1}^{k+1} \qquad \text{(I.4.79)}$$

$$-A_N\cdot\left(1+\frac{1}{R}\right)\cdot T_{N-1}^{k+1}+\left[1+A_N\cdot\left(1+\frac{1}{R}\right)\right]\cdot T_N^{k+1}=T_N^k \qquad \text{(I.4.80)}$$

Notând: $\alpha_N=-A_N\cdot\left(1+\frac{1}{R}\right)$ $\beta_N=1+A_N\cdot\left(1+\frac{1}{R}\right)$

ultima ecuație are forma:

$$\alpha_N \cdot T_{N-1}^{k+1} + \beta_N \cdot T_N^{k+1} = T_N^k$$

Deci, la fiecare pas pe axa timpului, trebuie rezolvat sistemul de ecuații:

$$\begin{bmatrix} \beta_1 & \gamma_1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \alpha_2 & \beta_2 & \gamma_2 & 0 & \cdots & 0 & 0 & 0 \\ 0 & \alpha_3 & \beta_3 & \gamma_3 & \cdots & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & \alpha_i & \beta_i & \gamma_i & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \cdots & \alpha_{N-1} & \beta_{N-1} & \gamma_{N-1} \\ 0 & 0 & 0 & 0 & \cdots & 0 & \alpha_N & \beta_N \end{bmatrix} \cdot \begin{bmatrix} T_1^{k+1} \\ T_2^{k+1} \\ T_3^{k+1} \\ \vdots \\ T_i^{k+1} \\ \vdots \\ T_{N-1}^{k+1} \\ T_N^{k+1} \end{bmatrix} = \begin{bmatrix} T_1^k - \alpha_1 \cdot T_0 \\ T_2^k \\ T_3^k \\ \vdots \\ T_i^k \\ \vdots \\ T_{N-1}^k \\ T_N^k \end{bmatrix} \quad \text{(I.4.81)}$$

Diagonalizarea duce la un sistem de forma:

$$\begin{bmatrix} 1 & \delta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \delta_2 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \delta_3 & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} T_1^{k+1} \\ \vdots \\ \vdots \\ \vdots \\ T_N^{k+1} \end{bmatrix} = \begin{bmatrix} \theta_1 \\ \vdots \\ \vdots \\ \vdots \\ \theta_N \end{bmatrix} \quad \text{(I.4.82)}$$

unde: $\delta_1 = \dfrac{\gamma_1}{\beta_1}$ $\theta_1 = \dfrac{T_1^k - \alpha_1 \cdot T_0}{\beta_1}$

$$\delta_i = \frac{\gamma_i}{\beta_i - \alpha_i \cdot \delta_{i-1}} \qquad \theta_i = \frac{T_i^k - \alpha_i \cdot \theta_{i-1}}{\beta_i - \alpha_i \cdot \delta_{i-1}} \qquad \text{pentru i=2, 3, ..., N-1}$$

pentru i=N avem $\theta_N = \dfrac{T_N^k - \alpha_N \cdot \theta_{N-1}}{\beta_N - \alpha_N \cdot \delta_{n-1}}$

Rezolvarea acestui sistem este imediată.

BIBLIOGRAFIE

Capitolul I.1 :

1. Eykhoff, P., *Identificarea sistemelor*, Editura Tehnică, Bucureşti, **1977**.
2. Himelblau, D.M., Bishof, K.B., *Process Analysis and Simulation : Deterministic Systems*, John Wiley&Sons, **1968**.
3. Luyben, W. , *Process Modeling, Simulation and Control for Chemical Engineers*, McGraw – Hill, **1986**.
4. Mihail R. , *Modelarea reactoarelor chimice*, Editura Tehnică, Bucureşti, 1976.
5. Todinca, T., Delia Perju, Suta, M., *Optimizări în industria chimică*, Editura Mirton, Timişoara, **1993**.

Capitolul I.2 :

6. *** *Using MATLAB - Version 5* , The Mathworks Inc., **1998**.
7. *** *Using MATLAB Graphics – Version 5*, The Mathworks Inc., **1996**.
8. Ghinea, M., Fireteanu, V., *MATLAB :Calcul numeric. Grafica. Aplicaţii.*, Editura Teora, Bucureşti, **1997**.

Capitolele I.3 si I.4 :

9. Davis, M., *Numerical Methods and Modeling for Chemical Engineers*, John Wiley&Sons, **1984**.
10. Dorn, W., McCracken, D., *Metode numerice cu programe în Fortran* , Editura Tehnică, Bucureşti, **1972**.
11. Franks, R., *Modelarea şi simularea în Ingineria Chimica*, Editura Tehnică, Bucureşti, **1979**.
12. Marinoiu, V., Stratulă, C., *Metode numerice aplicate în ingineria chimica*, Editura Tehnică, Bucureşti, **1986**.
13. Nougier, J.P., *Methodes de calcul numerique*, Masson, **1991**.
14. Penny, J., Lindfield, G., *Numerical Methods Using MATLAB*, Ellis Horwood, **1995**.
15. Rice, R., Do, D., *Applied Mathematics and Modeling for Chemical Engineers*, John Wiley&Sons, **1995**.