Lucrarea 3

REZOLVAREA NUMERICĂ A SISTEMELOR LINIARE

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n = b_2 \\ - - - - - - - - - - - - - - - \\ a_{n1}x_1 + a_{n2}x_2 + ... + a_{nn}x_n = b_n \end{cases} \quad (3.1)$$

3.1.METODE DIRECTE

Aceste metode sunt caracterizate prin aceea că pot oferi soluţia exactă a sistemului de rezolvat, dacă sunt îndeplinite condiţiile de existenţă.

3.1.1. METODA LUI GAUSS DE ELIMINARE (ALGORITMUL)

```
real Gauss
(
   întreg n,                    // ordinul sistemului
   real A[][N],                 // matricea sistemului
   real B[],                    // vectorul termenilor liberi
   real X[]                     // vectorul soluţiilor
)
{                               // declaraţiile şi definiţiile variabilelor locale
       întreg i;                // indicele liniei
       întreg k;                // indice suplimentar al liniei
       întreg j;                // indice coloană
       real m;                  // multiplicator
       real d;                  // determinantul sistemului

   pentru i = 1,n-1 {           // for 1
     pentru k = i+1,n {
         m = A[k][i]/A[i][i];
         pentru j = i,n  A[k][j] = A[k][j] - m*A[i][j];
         B[k] = B[k] - m*B[i];
                     }
   }                            // for 1

   d=1;                         // iniţializarea cu 1 a valoarii determinantului
   dacă (A[n][n] = = 0)  returnează 0;
   pentru i =1,n   d = d*A[i][i];
                                // Rezolvă sistem superior triunghiular
   pentru i = n,1 {
         x[i] = B[i];
         pentru j = n,i+1 x[i] = x[i] - A[i][j]*x[j];
         x[i] = x[i] / A[i][i];
         }
   returnează d;                // returneaza la numele functiei valoarea determinantului
}
```

3.2. METODE INDIRECTE

3.2.1. METODA LUI JACOBI

Fie sistemul liniar (3.1). Se explicitează fiecare necunoscută din sistem de pe diagonala principală funcţie de celelalte necunoscute ale sistemului. Se alege o soluţie a sistemului oarecare $x_1^{(0)},x_2^{(0)},...,x_n^{(0)}$, ce va reprezenta soluţia de start, după care se continuă iteraţiile, până când două soluţii succesive diferă cu mai puţin de o eroare impusă (precizia impusă).

3.2.2 Algoritmul. Metoda lui Jacobi

```
întreg Jacobi
(
   întreg n,                  // ordinul sistemului;
   real A[][N],               // matricea sistemului;
   real B[],                  // vectorul termenilor liberi;
   real X[]                   // vectorul de start la intrare, soluţia la ieşire
)
{
                              // declaraţiile şi definiţiile variabilelor locale
întreg i, j;                  // indici;
real sum;
real Xn[N], Xn1[N];// soluţiile la pasul curent şi precedent
întreg sem;

                              // Verifică dacă matricea e dominant-diagonală – important!!!
   pentru  i= 1,n {
      sum = 0;
      pentru j= 1,n    sum = sum + fabs(A[i][j]);
      dacă (sum > 2*fabs(A[i][i])) returnează 1;
      }

                              // se salveaza valorile conţinute de vectorul de  start  X;
   pentru i= 1,n   Xn[i] = X[i];
   execută
   {
      sem = 0;
      pentru i= 1,n    Xn_1[i] = Xn[i];
      pentru i= 1,n
      {
         Xn[i] = B[i];
         pentru j= 1,n
               dacă (j != i)  Xn[i] = Xn[i] - A[i][j] * Xn_1[j];
         Xn[i] = Xn[i]/A[i][i];
      }
      pentru i= 1,n
         dacă (fabs(Xn[i]-Xn_1[i]) > eps )  sem = 1;
   } cât timp (sem = = 1);
   pentru i= 1,n    X[i] = Xn[i];       // se salveaza valorile solutiei in  vectorul de  X
   returnează 0;}
```

Lucrarea 4

DERIVAREA NUMERICĂ
INTEGRAREA NUMERICĂ
INTERPOLAREA (partea 1)

4.1. DERIVAREA NUMERICĂ.

4.1.1. DERIVATA NUMERICĂ PRIN DOUĂ PUNCTE

$$f'(x_0) \approx \frac{f\left(x_0 + \frac{h}{2}\right) - f\left(x_0 - \frac{h}{2}\right)}{h}$$

4.1.2.DERIVATA NUMERICĂ PRIN TREI PUNCTE

$$f'(x_0) \approx \frac{h_1^2 f(x_0 + h_2) + \left(h_2^2 - h_1^2\right) f(x_0) - h_2^2 f(x_0 - h_1)}{h_1 h_2 (h_1 + h_2)}$$

Pentru $h_1 = h_2 = h/2$ se obţine formula derivatei numerice *prin două puncte*.

4.1.3. DERIVATA NUMERICĂ PRIN CINCI PUNCTE

$$f'(x_0) = \frac{1}{12h}\left[f(x_0 - 2h) - 8f(x_0 - h) + 8f(x_0 + h) - f(x_0 + 2h)\right]$$

4.2. INTEGRAREA NUMERICĂ

4.2.1. METODA LUI RICHARDSON

Această metodă dă o precizie mai bună de calcul a integralei numerice decât metoda trapezului şi s-a obţinut prin modificarea metodei trapezului. Este o metodă în care apare *dubla-divizare* a intervalului de integrare.

Se pleacă de la eroarea de trunchiere a metodei trapezului, care pentru o diviziune de lăţime *h* este:

$$e_{T,h} = Ch^2, \quad h = (b-a)/n,$$

Pentru cealaltă diviziune, $k = (b-a)/m$, se obţine eroarea de trunchiere:

$$e_{T,k} = Ck^2$$

Pentru fiecare diviziune se poate scrie:

$$I = I_h + e_{T,h}$$

respectiv

$$I = I_k + e_{T,k}$$

Eliminând constanta *C* între relaţiile de mai sus, se obţine expresia:

$$I = I_h + \frac{I_h - I_k}{\left(\frac{k}{h}\right)^2 - 1}$$

expresie ce poartă denumirea de *formula lui Richardson*.

4.2.1.1. Metoda lui Richardson (algoritm)

real **I_Richardson**
(

```
   real ls,                    // limita stângă a intervalului de integrare
   real ld,                    // limita dreaptă a intervalului de integrare
   întreg n,                   // numărul de subintervale_1
   întreg m                    // numărul de subintervale_2
```

)

```
{
                                    // declararea şi definirea variabilelor locale
real h;                             // valoarea lungimii unui subinterval divizat  in n sub-diviziuni
real k;                             // valoarea lungimii unui subinterval divizat  im m sub-diviziuni
real sumh;                          // valoarea integralei cu diviziunea h
real sumk;                          // valoarea integralei cu diviziunea k
real sum;                           // valoarea integralei
întreg  i;                          // indice întreg pentru ciclul 'for()'

                                    // instrucţiunile de calcul
```

$h = \frac{ld - ls}{n}$; $k = \frac{ld - ls}{m}$;

```
    Sumh=(f(ls)+f(ld))/2*h;
    Sumk=(f(ls)+f(ld))/2*k;

pentru i= 1 ÷ n-1
         sumh = sumh + h*f(ls + i*h);
pentru i= 1 ÷ m-1
         sumk = sumk + k*f(ls + i*k);
sum=sumh + (sumh-sumk)/( (k/h)*(k/h)-1 );
returnează sum;
}
```

Această metodă determină punctele de divizare ale intervalului de integrare astfel ca eroarea de calcul a integralei să fie minimă.

4.2.2. METODA CUADRATURII GAUSSIENE CU DOUĂ PUNCTE (*tema*)

Această metodă reduce orice interval de integrare [a,b] la intervalul [-1,1] cu ajutorul formulei de *substituţie*:

$$y = \frac{2x - (b+a)}{b-a}$$

Pentru $x=1$ rezultă $y=-1$, iar pentru $x=b$ rezultă $y=1$. Din relaţia (5.11) se poate scrie:

$$x = \frac{1}{2}(b-a)y + \frac{1}{2}(b+a)$$

şi de aici:

$$dx = \frac{1}{2}(b-a)dy$$

Ca urmare, integrala iniţială în forma $I = \int_a^b f(x)dx$ se transformă în integrala:

$$I = \int_{-1}^{+1} f\left[\frac{1}{2}(b-a)y + \frac{1}{2}(b+a)\right]\frac{1}{2}(b-a)dy = \int_{-1}^{+1} \text{fi}(y)dy$$

în care noua variabilă este y.

Integrala de mai sus este în final aproximată printr-o sumă, aşa cum s-a făcut de fiecare dată (din punct de vedere matematic integrala devine o sumă). Mai exact:

$$\int_{-1}^{+1} \text{fi}(y)dy = \sum_{i=1}^{n} k_i \cdot \text{fi}(y_i)$$

Precizia metodei este cea mai bună între metodele de integrare de *tip cuadratură* (pentru funcţii de o variabilă). Această putere de calcul este de fapt justificată, din moment ce metoda foloseşte în rezolvare *polinoamele Legendre*. Echivalentul matematic este acela de aproximare a funcţiei concrete de integrat printr-un polinom Legendre echivalent, pentru un anumit grad dorit de către utilizator. Cu cât gradul preluat în calcule este mai mare, cu atât precizia metodei evident creşte.

Polinoamele Legendre sunt definite pe intervalul [-1, 1]. De aceea se impune schimbarea limitelor de integrare iniţiale. Pe acest interval polinoamele prezintă un număr de rădăcini reale, distincte (gradul de multiplicitate al fiecărei soluţii este 1). Din cauză că aceste rădăcini nu sunt egal depărtate între ele, şi nici faţă de capetele intervalului de integrat, această metodă este clasificată drept metodă *cu divizare variabilă.*

Această precizie de calcul se poate impune în mod direct, prin stabilirea gradului polinomului *Legendre* dorit în calcul.

Există valori tabelate ale rădăcinilor acestor polinoame (notate aici cu Y_i), ca şi a ponderilor (K_i), valori ce intervin în relaţia de calcul de mai sus.

Pentru **N=3** aceste valori sunt: $k_1=5/9$, $k_2=8/9$, $k_3=5/9$ si
$y_1=0.7745$, $y_2=0$, $y_3= -0.7745$

4.2.3. CALCULUL NUMERIC AL INTEGRALELOR DUBLE

Pentru simplitate vom considera domeniul de integrare al funcţiei de două variabile un dreptunghi

$$\iint_D f(x,y)\,dxdy = \int_a^b \int_c^d f(x,y)\,dxdy$$

reprezintă integrala dublă dintr-o funcţie de două variabile. Pentru calculul valorii acestei integrale vom utiliza *formula de cubatură a trapezului* în continuare.

4.2.3.1. FORMULA DE CUBATURĂ A TRAPEZULUI

Intervalele de integrat [a,b] şi [c,d] se împart în subintervale de lungimi egale

$$h=\frac{b-a}{n} \quad \text{şi, respectiv} \quad k=\frac{d-c}{m}$$

şi se consideră dreptunghiul cu vârfurile $[x_i, y_i]$, $[x_{i+1}, y_i]$, $[x_{i+1}, y_{i+1}]$, $[x_i, y_{i+1}]$

$$I=\sum_{i=0}^{n-1}\sum_{j=0}^{m-1} I_{ij}=\frac{kh}{4}\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}\left[f(x_i,y_j)+f(x_i,y_{j+1})+f(x_{i+1},y_j)+f(x_{i+1},y_{j+1})\right]$$

expresie cunoscută sub numele de *formula de **cubatură** a trapezului*. Unde $x_i=a+i*h$, $y_j=c+j*k$;

4.2.3.2. Algoritmul. Metoda cubaturii trapezului

```
real I_Cubatura_Trapez
(
   real a,              // limita stângă a intervalului de integrare  pe axa Ox
   real b               // limita dreaptă a intervalului de integrare pe axa Ox
   real c,              // limita stângă a intervalului de integrare  pe axa Oy
   real d,              // limita dreaptă a intervalului de integrare  pe axa Oy
   întreg n,            // numărul de subintervale pe axa Ox
   întreg m             // numărul de subintervale pe axa Oy
)

{
                  // declararea şi definirea variabilelor locale
întreg i, j;
real h;              // valoarea lungimii unui subinterval diviziunea n
real k;              // valoarea lungimii unui subinterval diviziunea m
real sum;            // valoarea integralei
```

```
                    // instrucţiunile de calcul ale metodei
```
$h = \frac{b-a}{n}$; $k = \frac{d-c}{m}$;

```
sum = 0;
pentru i= 0 ÷  n-1
  pentru j= 0 ÷  m-1
   sum = sum+(h*k)/4*( f(a+i*h, c+j*k) + f(a+i*h, c+(j+1)*k) +
             +     f(a+(i+1)h,c+j*k) + f(a+(i+1)h, c+(j+1)*k) );
returnează   sum;
}
```

4.3.INTERPOLAREA (partea 1)

4.3.1. INTERPOLAREA POLINOMIALĂ LAGRANGE

Se consideră funcţia dată prin urmatorul tabel:

x	x_0	x_1	.	x_k	.	x_n
y	y_0	y_1	.	y_k	.	y_n

$$P_n(x)=\sum_{i=0}^{n}P_n(x_i)\frac{\Pi_i(x)}{\Pi_i(x_i)}=\sum_{i=0}^{n}y_i\frac{\prod_{j=0,j\neq i}^{n}(x-x_j)}{\prod_{j=0,j\neq i}^{n}(x_i-x_j)}=\sum_{i=0}^{n}y_i\prod_{j=0,j\neq i}^{n}\frac{x-x_j}{x_i-x_j}$$

Polinomul de gradul *n* care trece prin *n*+1 puncte date, numit şi *polinomul de interpolare al lui Lagrange*, are forma:

$$P_n(x)=\sum_{i=0}^{n}y_i\prod_{j=0,j\neq i}^{n}\frac{x-x_j}{x_i-x_j}$$

4.3.1.1. Algoritmul. Polinomul lui Lagrange

```
real Lagrange
(
   întreg n,          // gradul polinomului de interpolare
   real x[ ],         // vectorul absciselor punctelor cunoscute
   real y[ ],         // vectorul ordonatelor punctelor cunoscute
   real x̄ ,           // punctul în care se calculează interpolarea
)
{                     // declararea variabilelor locale functiei
întreg  i, j ;
real sum ;
real prod ;
                                  // corpul de instrucţiuni al funcţiei
   sum=0;
   pentru i= 0 ÷ n
   {
   prod=1;
   pentru j= 0 ÷ n    dacă ( j ≠ i )    prod = prod * (x̄ - x[j]) / (x[i] - x[j]);
   sum = sum + y[i] * prod ;
   }
   returnează sum; }          // valoarea în punctul cerut
```

Lucrarea 5

INTERPOLAREA (partea 2 a)

METODE DE OPTIMIZARE (partea 1 a)

5.1 POLINOMUL DE INTERPOLARE DE SPEŢA ÎNTÂI AL LUI NEWTON

Face parte din clasa metodelor ***cu pas constant*** între abscise. Aceasta se traduce prin a avea o anumită zonă preferată, în care precizia este cea mai bună pentru un interval acoperit prin punctele $x_0, .. , x_n$.

Acest polinom de interpolare se exprimă funcţie de ***diferenţele finite***. Fie $f:[a,b] \to \mathbf{R}$ şi reţeaua $x_0, x_1, x_2, ..., x_n$ cu pasul constant *h*.

Definiţia 5.1. Se numeşte *diferenţă finită* de *ordinul întâi* expresia:

$$\Delta f(x) = f(x+h) - f(x)$$

unde *h* este pasul constant.

Diferenţa finită de *ordinul n* se poate defini recursiv, folosindu-se de diferenţa finită de ordin *n-1*, şi are expresia de definiţie:

$$\Delta^n f(x) = \Delta\left(\Delta^{n-1} f(x)\right)$$

x_i	y_i	Δy_i	$\Delta^2 y_i$	$\Delta^3 y_i$	$\Delta^4 y_i$	$\Delta^5 y_i$	$\Delta^6 y_i$
x_0	y_0	Δy_0					
x_1	y_1	Δy_1	$\Delta^2 y_0$	$\Delta^3 y_0$			
x_2	y_2	Δy_2	$\Delta^2 y_1$	$\Delta^3 y_1$	$\Delta^4 y_0$	$\Delta^5 y_0$	
x_3	y_3	Δy_3	$\Delta^2 y_2$	$\Delta^3 y_2$	$\Delta^4 y_1$	$\Delta^5 y_1$	$\Delta^6 y_0$
x_4	y_4	Δy_4	$\Delta^2 y_3$	$\Delta^3 y_3$	$\Delta^4 y_2$		
x_5	y_5	Δy_5	$\Delta^2 y_4$				
x_6	y_6						

Definiţia. Se numeşte *putere generalizată de ordinul n a lui x* expresia:

$$x^{[n]} = x(x-h)(x-2h)...(x-(n-1)h)$$

Pentru h=0 puterea generalizată coincide cu puterea obişnuită.

1. *Diferenţa finită a puterii generalizate* este:

$$\Delta x^{[n]} = nhx^{[n-1]}$$

Fie funcţia tabelată dată în *Tabelul de mai sus*, unde reţeaua $x_0, x_1, x_2, x_3, ..., x_n$ este cu pasul constant *h*.

Prin cele *n+1* puncte trece un polinom de gradul *n* pe care îl căutăm de forma:

$$P_n(x) = C_0 + C_1(x-x_0)^{[1]} + C_2(x-x_0)^{[2]} + ... + C_n(x-x_0)^{[n]}$$

unde

$$(x-x_0)^{[i]} = (x-x_0)(x-x_1)...(x-x_{i-1}) \quad , \quad i = 1, 2, ...n$$

Ţinând cont de formulele de calcul ale coeficienţilor, *polinomul lui Newton de interpolare de speţa întâi* poate fi scris astfel:

$$P_n(x_0) = y_0 + \frac{\Delta y_0}{1!h}(x-x_0)^{[1]} + \frac{\Delta^2 y_0}{2!h^2}(x-x_0)^{[2]} + \cdots + \frac{\Delta^n y_0}{n!h^n}(x-x_0)^{[n]}$$

5.1.1 Algoritmul. Newton 1

```
real Newton_1
(
întreg n,                  // gradul polinomului de interpolare
real x0,                   // abscisa primului punct cunoscut
real h,                    // pasul constant între abscisele cunoscute
real y[ ],                 // vectorul ordonatelor punctelor cunoscute
real xp,                   // abscisa punctului în care se face interpolarea
)
{                          // declararea şi definirea variabilelor locale
real sum ;
real prod ;
integ i, j ;
                           // corpul de instrucţiuni al funcţiei
sum = y[0];                // pornesc cu suma de la y0
prod = 1 ;
 pentru i= 1… n
   {                       //calculul diferenţelor finite
   pentru j= 0… n-i    y[j] = y[j+1] – y[j];
```

$$prod = prod * \left(x_p - \left(x_0 + (i-1) * h\right)\right) * \frac{1}{h} * \frac{1}{i};$$

```
   sum = sum + y[0]*prod ;
   }
returnează sum ;                  // valoarea interpolată
}
```

5.2 POLINOMUL DE INTERPOLARE DE SPEŢA A DOUA AL LUI NEWTON

Pentru funcţia dată în *Tabelul 6.1* se caută un polinom de gradul *n* care trece prin cele *n+1* puncte, sub forma:

$$P_n(x) = C_0 + C_1(x - x_n) + C_2(x - x_n)(x - x_{n-1}) + ... + C_n(x - x_n)(x - x_{n-1})...(x - x_1) \quad)$$

Polinomul de gradul *n* poate fi scris sub forma:

$$P_n\left(x\ \right) = y_n + \frac{\Delta y_n}{1!h}\left(x - x_n\right)^{[1]} + \frac{\Delta^2 y_{n-1}}{2!h^2}\left(x - x_{n-1}\right)^{[2]} + \cdots + \frac{\Delta^n y_1}{n!h^n}\left(x - x_1\right)^{[n]}$$

5.3 POLINOMUL LUI NEWTON DE INTERPOLARE CU DIFERENŢE DIVIZATE

Fie funcţia $f(x)$ dată sub forma celei prezentate în tabelul (5.1) unde reţeaua $x_0, x_1, x_2, ..., x_n$ din domeniul de definiţie al funcţiei nu are pas constant. Fiind o metodă cu pas variabil între abscise, precizia sa nu este preferenţială.

Definiţia 5.3. Se numeşte *diferenţă divizată de ordinul k+i* a funcţiei f expresia :

$$f\left(x_{i-1}, x_i, x_{i+1}, \ldots, x_{i+k}\right) = \frac{f\left(x_i, x_{i+1}, \ldots, x_{i+k}\right) - f\left(x_{i-1}, x_i, \ldots, x_{i+k-1}\right)}{x_{i+k} - x_{i-1}}$$

Se determină polinomul de gradul *n,* de forma :

$$P_n(x) = C_0 + C_1(x - x_0) + C_2(x - x_0)(x - x_1) + ... + C_n(x - x_0)(x - x_1)...(x - x_{n-1})$$

cunoscând *n+1* puncte (x_i, y_i), i=0, ...,n , care verifică polinomul.

Se observă că $P_n(x_0) = y_0 = C_0$.

Se calculează *diferenţa divizată* de ordinul întâi pentru polinomul $P_n(x)$ şi se face $x = x_1$. Rezultă:

$$P_n(x_0, x_1) = C_1 .$$

Calculând în continuare, se determină diferenţa divizată de ordinul k şi luând pe $x = x_k$ se obţine valoarea coeficientului C_k :

$$C_k = P_n(x_0,x_1,x_2,...,x_k), \qquad k=0, 1, ...,n.$$

Polinomul se scrie acum sub forma:

$$P_n(x) = y_0 + P_n(x_0,x_1)(x-x_0) + P_n(x_0,x_1,x_2)(x-x_0)(x-x_1)+...$$
$$+ P_n(x_0,...,x_n)(x-x_0)(x-x_1)...(x-x_{n-1})$$

unde fiecare diferenţă divizată se calculează cu ajutorul formulei din definitia (5.3). Polinomul obţinut poartă numele de *polinomul lui Newton de interpolare cu diferenţe divizate.*

5.3.1 Algoritmul . Newton 3

```
real Newton_3
(
   întreg n,                  // numărul de puncte date ale funcţiei
   real x[ ],                 // vectorul absciselor punctelor date
   real y[ ],                 // vectorul ordonatelor punctelor date
   real x̄ ,                   // punctul în care se interpolează funcţia
)
{                             // declararea şi definirea variabilelor locale
întreg i, j ;
real sum ;
real prod ;
                              // corpul de instrucţiuni al funcţiei
   sum = y[0] ;
   prod = 1 ;
   pentru i= 1.. n
   {
                              // calculul diferenţelor divizate
         pentru j= 0.. n-i     y_j = (y_{j+1} - y_j)/(x_{j+i} - x_j);

         prod = prod*(x̄ - x_{i-1}) ;
         sum = sum + y[0]*prod ;
   }
         returnează  sum ;}
```

5.4. METODE DE OPTIMIZARE (partea 1)

PREZENTAREA TEORETICĂ

Metodele de optimizare se clasifică, în raport cu problema care se pune pentru funcţia ţintă sau scop, astfel:

1. Metode ce determină expresia analitică a funcţiei, care aproximează cel mai bine o funcţie tabelată dată prin puncte;
2. Metode ce determină diferiţi parametri ai funcţiei scop (ţintă), pentru a obţine un extrem al funcţiei.

5.4.1. METODA CELOR MAI MICI PĂTRATE

Presupunem că avem o funcţie definită printr-un tabel de valori. Se mai spune că funcţia este dată implicit, deci nu i se cunoaşte forma în mod direct. Problema care se pune este să determinăm funcţia analitică *care aproximează cel mai bine* datele din tabel sau curba care trece prin punctele tabelului.

5.4.1.1 REGRESIA LINIARĂ

Se consideră funcţia tabelată:

x	x_1	x_2	x_3	x_4	…..	x_m
y	y_1	y_2	y_3	y_4	…..	y_m

unde *m* reprezintă *numărul de măsurători* sau de valori ale funcţiei. Se cere să se determine funcţia liniară de forma generală

$$y = ax + b$$

care să aproximeze cel mai bine funcţia tabelată ***astfel ca eroarea să fie minimă***.

$$a = \frac{m\sum_{i=1}^{m} x_i y_i - \sum_{i=1}^{m} x_i \sum_{i=1}^{m} y_i}{m\sum_{i=1}^{m} x_i^2 - \left(\sum_{i=1}^{m} x_i\right)^2}$$

$$b = \frac{\sum_{i=1}^{m} x_i^2 \sum_{i=1}^{m} y_i - \sum_{i=1}^{m} x_i \sum_{i=1}^{m} x_i y_i}{m\sum_{i=1}^{m} x_i^2 - \left(\sum_{i=1}^{m} x_i\right)^2}$$

$$E = \sum_{i=1}^{nm} (y_i - ax_i - b)^2$$

5.4.1.2. Algoritm. Regresia liniară

```
void Reg_Lin
(
   întreg m,            // numărul experienţelor
   real x[ ],           // vectorul absciselor funcţiei
   real y[ ],           // vectorul ordonatelor funcţiei
   real *pA,            // adresa coeficientului lui x (pointer)
   real *pB             // adresa termenului liber (pointer)
   real *E              // adresa erorii (pointer)
)
{                       // declararea şi definirea variabilelor locale
   real sx, sy, sxx, sxy ;
   întreg i ;

                        // corpul de instrucţiuni al funcţiei
   sx= 0 ; sy = 0 ; sxy = 0 ; sxx = 0;
   pentru i= 1 ÷ m
   {
         sx = sx + x[i] ;
         sy = sy + y[i] ;
         sxy = sxy + x[i]*y[i] ;
         sxx = sxx + x[i]*x[i] ;
   }
         *pA = (m·sxy-sx·sy) / (m·sxx-sx·sx) ;    *pB = (·sxx·sy-sx·sxy) / (m·sxx-sx·sx) ;
                        // se calculeaza eroarea
   pentru i= 1 ÷ m
         *E=*E+(y[i]-(*pa)*x[i]-(*pb))2
  }
```

5.4.2.1.REGRESIA HIPERBOLICĂ

Fie funcţia numerică dată în tabel. Cautam determinării funcţiei hiperbolice: $y = \frac{1}{ax+b}$ care aproximează cel mai bine funcţia numerică. Valorile lui *a* şi *b* sunt:

$$a = \frac{(\sum_{i=1}^{m} x_i)(\sum_{i=1}^{m} \frac{1}{y_i}) - m\sum_{i=1}^{m} \frac{x_i}{y_i}}{\left(\sum_{i=1}^{m} x_i\right)^2 - m\sum_{i=1}^{m} x_i^2}$$

$$b = \frac{(\sum_{i=1}^{m} \frac{1}{y_i})(\sum_{i=1}^{m} x_i^2) - (\sum_{i=1}^{m} \frac{x_i}{y_i})(\sum_{i=1}^{m} x_i)}{m\sum_{i=1}^{m} x_i^2 - \left(\sum_{i=1}^{m} x_i\right)^2}$$

$$E = \sum_{i=1}^{m} (\frac{1}{y_i} - ax_i - b)^2$$

Pentru a liniariza rezolvarea sistemului s-a lucrat cu inversele valorilor lui *y*, adică *1/y*, motiv pentru care sumele *sy* şi *sxy* conţin valorile inversate pentru *y*.

5.4.2.2. Algoritm. Regresia hiperbolică

```
void Reg_Hip
(
întreg m,                  // numărul experienţelor
real x[ ],                 // vectorul absciselor funcţiei numerice
real y[ ],                 // vectorul ordonatelor funcţiei numerice
real *pA                   // adresa coeficientului lui a (pointer)
real *pB                   // adresa coeficientului lui b (pointer)
real *E                    // adresa erorii (pointer)
)
{                          // declararea şi definirea variabilelor locale
real sx , sy , sxx , sxy ;
întreg i ;
                           // corpul de instrucţiuni al funcţiei
   sx = 0 ; sy = 0 ; sxy = 0 ; sxx = 0 ;
   pentru i= 1÷m
   {
         sx = sx + x[i] ;
         sy = sy+ 1/y[i] ;
         sxy= sxy+ x[i]/y[i] ;
         sxx = sxx + x[i]*x[i] ;
   }
   *pA = (m·sxy − sx·sy)/(m·sxx − sx·sx) ; *pB = (·sxx·sy-sxy·sx)/(m·sxx-sx·sx) ;
                           // se calculeaza eroarea
   pentru i= 1÷ m
         *E=*E+( y[i]-1/((*pa)*x[i]+(*pb)) )2;

}
```