
Lucrarea 3

REZOLVAREA NUMERICĂ A SISTEMELOR LINIARE

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

3.1.METODE DIRECTE

Aceste metode sunt caracterizate prin aceea că pot oferi soluția exactă a sistemului de rezolvat, dacă sunt îndeplinite condițiile de existență.

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 soluțiilor
)
{
    // declarațiile și definițiile 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 =  $\overline{1, n-1}$  { // for 1
        pentru k =  $\overline{i+1, n}$  {
            m = A[k][i]/A[i][i];
            pentru j =  $\overline{i, n}$  A[k][j] = A[k][j] - m*A[i][j];
            B[k] = B[k] - m*B[i];
        }
    } // for 1

    d=1;                    // inițializarea cu 1 a valorii determinantului
    dacă (A[n][n] == 0) returnează 0;
    pentru i =  $\overline{1, n}$  d = d*A[i][i]; // Rezolvă sistem superior triunghiular

    pentru i =  $\overline{n, 1}$  {
        x[i] = B[i];
        pentru j =  $\overline{n, i+1}$  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 ca funcție de celelalte necunoscute ale sistemului. Se alege o soluție a sistemului oarecare $x_1^{(0)}, x_2^{(0)}, \dots, x_n^{(0)}$, ce va reprezenta soluția de start, după care se continuă iterațiile, până când două soluții succesive diferă cu mai puțin de o eroare impusă (precizia impusă).

3.2.2 Algoritm. 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, soluția la ieșire
    real eps                 // eroarea de aproximare
)
{
    // declarațiile și definițiile variabilelor locale
    // indici;
    întreg i, j;
    real sum;
    real Xn[N], Xn_1[N];    // soluțiile la pasul curent și precedent
    întreg sem;

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

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

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

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

4.1.3. DERIVATA NUMERICĂ PRIN CINCI PUNCTE

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

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 și s-a obținut 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 lățime h este:

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

Pentru cealaltă diviziune, $k = (b-a)/m$, se obține 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 relațiile de mai sus, se obține 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 și definirea variabilelor locale
real h; // valoarea lungimii unui subinterval divizat in n sub-diviziuni
real k; // valoarea lungimii unui subinterval divizat in 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()'

// instrucțiunile de calcul


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

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 substituție:

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

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

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

și de aici:

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

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

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

în care noua variabilă este y .

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

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

Precizia metodei este cea mai bună între metodele de integrare de tip *cuadratură* (pentru funcții de o variabilă). Această putere de calcul este de fapt justificată, din moment ce metoda folosește în rezolvare *polinoamele Legendre*. Echivalentul matematic este acela de aproximare a funcției 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 crește.

Polinoamele Legendre sunt definite pe intervalul [-1, 1]. De aceea se impune schimbarea limitelor de integrare inițiale. Pe acest interval polinoamele prezintă un număr de rădăcini reale, distincte (gradul de multiplicitate al fiecărei soluții este 1). Din cauză că aceste rădăcini nu sunt egal depărtate între ele, și nici față 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 și a ponderilor (K_i), valori ce intervin în relația de calcul de mai sus.

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

4.2.3. CALCULUL NUMERIC AL INTEGRALELOR DUBLE

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

$$\square \int_D f(x,y) dx dy = \int_a^b \int_c^d f(x,y) dx dy$$

reprezintă integrala dublă dintr-o funcție 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 $[a,b]$ și $[c,d]$ se împart în subintervale de lungimi egale

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

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

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

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

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 și 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

```

```

// instrucțiunile de calcul ale metodei
h= (b-a) / n ; k= (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 funcția dată prin următorul 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 și *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 funcției
    întreg i, j ;
    real sum ;
    real prod ;

    // corpul de instrucțiuni al funcției
    sum=0;
    pentru i= 0 ÷ n
    {
        prod=1;

```

```

pentru j= 0 ÷ n   dacă ( j ≠ i )   prod = prod *  $\frac{\bar{x} - x[j]}{x[i] - x[j]}$ ;
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 SPEȚA 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, \dots, x_n .

Acest polinom de interpolare se exprimă funcție de *diferențele finite*. Fie $f: [a, b] \rightarrow \mathbf{R}$ și rețeaua $x_0, x_1, x_2, \dots, x_n$ cu pasul constant h .

Definiția 5.1. Se numește *diferență finită* de *ordinul întâi* expresia:

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

unde h este pasul constant.

Diferența finită de *ordinul n* se poate defini recursiv, folosindu-se de diferența finită de ordin $n-1$, și are expresia de definiție:

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

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						

Definiția. Se numește *putere generalizată de ordinul n a lui x* expresia:

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

Pentru $h=0$ puterea generalizată coincide cu puterea obișnuită.

1. *Diferența finită a puterii generalizate* este:

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

Fie funcția tabelată dată în *Tabelul de mai sus*, unde rețeaua $x_0, x_1, x_2, x_3, \dots, x_n$ este cu pasul constant h .

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

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

unde

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

Ținând cont de formulele de calcul ale coeficienților, *polinomul lui Newton de interpolare de speța întâi* poate fi scris astfel:

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

5.1.1 Algoritm. 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 și definirea variabilelor locale
  real sum ;
  real prod ;
  înteg i, j ;

  // corpul de instrucțiuni al funcției
  // pornesc cu suma de la y0
  sum = y[0];
  prod = 1 ;
  pentru i= 1... n
  {
    //calculul diferențelor finite
    pentru j= 0... n-i   y[j] = y[j+1] - y[j];
    prod = prod*(xp - (x0 + (i-1)*h)) * 1/h * 1/i;
    sum = sum + y[0]*prod ;
  }
  returnează sum ;           // valoarea interpolată
}
    
```

5.2 POLINOMUL DE INTERPOLARE DE SPEȚA A DOUA AL LUI NEWTON

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

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

Polinomul de gradul n poate fi scris sub forma:

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

5.3 POLINOMUL LUI NEWTON DE INTERPOLARE CU DIFERENȚE DIVIZATE

Fie funcția $f(x)$ dată sub forma celei prezentate în tabelul (5.1) unde rețeaua $x_0, x_1, x_2, \dots, x_n$ din domeniul de definiție al funcției nu are pas constant. Fiind o metodă cu pas variabil între abscise, precizia sa nu este preferențială.

Definiția 5.3. Se numește *diferență divizată de ordinul $k+i$* a funcției f expresia :

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

Se determină polinomul de gradul n , de forma :

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

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

Se observă că $P_n(x_0) = y_0 = C_0$.

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

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

Calculând în continuare, se determină diferența divizată de ordinul k și luând pe $x = x_k$ se obține valoarea coeficientului C_k :

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

Polinomul se scrie acum sub forma:

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

unde fiecare diferență divizată se calculează cu ajutorul formulei din definiția (5.3). Polinomul obținut poartă numele de *polinomul lui Newton de interpolare cu diferențe divizate*.

5.3.1 Algoritm . Newton 3

```

real Newton_3
(
    întreg n,                // numărul de puncte date ale funcției
    real x[ ],              // vectorul absciselor punctelor date
    real y[ ],              // vectorul ordonatelor punctelor date
    real  $\bar{x}$ ,          // punctul în care se interpolează funcția
)
{
    // declararea și definirea variabilelor locale
    întreg i, j ;
    real sum ;
    real prod ;

    // corpul de instrucțiuni al funcției
    sum = y[0] ;
    prod = 1 ;
    pentru i= 1.. n
    {
        // calculul diferențelor divizate
        pentru j= 0.. n-i     $y_j = \frac{y_{j+1} - y_j}{x_{j+i} - x_j}$ ;

        prod = prod*(  $\bar{x} - x_{i-1}$  ) ;
        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 funcția țintă sau scop, astfel:

1. Metode ce determină expresia analitică a funcției, care aproximează cel mai bine o funcție tabelată dată prin puncte;
2. Metode ce determină diferiți parametri ai funcției scop (țintă), pentru a obține un extrem al funcției.

5.4.1. METODA CELOR MAI MICI PĂTRATE

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

5.4.1.1 REGRESIA LINIARĂ

Se consideră funcția 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 funcției. Se cere să se determine funcția liniară de forma generală

$$y = ax + b$$

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

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

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

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

5.4.1.2. Algoritm. Regresia liniară

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

    // corpul de instrucțiuni al funcției
    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 =  $\frac{m \cdot s_{xy} - s_x \cdot s_y}{m \cdot s_{xx} - s_x \cdot s_x}$  ;    *pB =  $\frac{s_{xx} \cdot s_y - s_x \cdot s_{xy}}{m \cdot s_{xx} - s_x \cdot s_x}$  ;
    // se calculeaza eroarea
    pentru i= 1 ÷ m
        *E=*E+(y[i]-(*pa)*x[i]-(*pb))2
}
```

5.4.2.1. REGRESIA HIPERBOLICĂ

Fie funcția numerică dată în tabel. Cautăm determinării funcției hiperbolice: $y = \frac{1}{ax+b}$ care aproximează cel mai bine funcția numerică. Valorile lui a și b sunt:

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

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

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

Pentru a liniariza rezolvarea sistemului s-a lucrat cu inversele valorilor lui y , adică $1/y$, motiv pentru care sumele s_y și s_{xy} conțin valorile inversate pentru y .

5.4.2.2. Algoritm. Regresia hiperbolică

```
void Reg_Hip
(
    întreg m,                // numărul experiențelor
    real x[ ],              // vectorul absciselor funcției numerice
    real y[ ],              // vectorul ordonatelor funcției numerice
    real *pA                // adresa coeficientului lui a (pointer)
    real *pB                // adresa coeficientului lui b (pointer)
    real *E                 // adresa erorii (pointer)
)
{
    // declararea și definirea variabilelor locale
    real s_x , s_y , s_xx , s_xy ;
    întreg i ;

    // corpul de instrucțiuni al funcției
    s_x = 0 ; s_y = 0 ; s_xy = 0 ; s_xx = 0 ;
    pentru i= 1÷m
    {
        s_x = s_x + x[i] ;
        s_y = s_y +  $\frac{1}{y[i]}$  ;
        s_xy = s_xy +  $\frac{x[i]}{y[i]}$  ;
        s_xx = s_xx + x[i]*x[i] ;
    }
    *pA =  $\frac{m \cdot s_{xy} - s_x \cdot s_y}{m \cdot s_{xx} - s_x \cdot s_x}$  ; *pB =  $\frac{\cdot s_{xx} \cdot s_y - s_{xy} \cdot s_x}{m \cdot s_{xx} - s_x \cdot s_x}$  ;
    // se calculeaza eroarea
}
```

```

    pentru i= 1÷ m
        *E=*E+( y[i]-1/(( *pa)*x[i]+(*pb)) )^2 ;
    }

```

Lucrarea 6

METODE DE OPTIMIZARE (partea 2 a) REZOLVAREA NUMERICĂ A ECUAȚIILOR DIFERENȚIALE DE ORDINUL I

6.1. REGRESIA EXPONENȚIALĂ

Dacă se constată că funcția numerică din tabelul 8.1, reprezentată grafic, se aseamănă foarte mult cu o exponențială, atunci se consideră funcția exponențială, de forma generală:

$$y = a \cdot b^x$$

în care a și b sunt constante pozitive.

Pentru un calcul comod al constantelor a și b se logaritmează funcția:

$$\ln y = \ln a + x \ln b$$

și se determină constantele astfel încât eroarea pătratică de ansamblu să fie *minimă*.

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

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

$$E = \sum_{i=1}^m (y_i - ab^{x_i})^2$$

6.1.1 Algoritm. Regresia exponențială

```

void Reg_Exp
(
    întreg m,           // numărul absciselor funcției
    real x[ ],         // vectorul absciselor funcției numerice
    real y[ ],         // vectorul ordonatelor funcției numerice
    real *pA,          // adresa coef lui a al fct exponentiale (pointer)
    real *pB           // adresa coef lui b al funcției exponentiale (pointer)
    real *E            // adresa erorii (pointer)
)
{
    // declararea și definirea variabilelor locale
    real s_x ;
    real s_xx ;
}

```

```

real sy ;
real sxy ;
întreg i ;
// corpul de instrucțiuni al funcției
sx = 0 ; sy = 0 ; sxy = 0 ; sxx = 0 ;
pentru i= 1 ÷ m
{
    sx = sx + x[i] ;
    sxx = sxx + x[i]*x[i] ;
    sy = sy+log( y[i]) ;
    sxy = sxy+ x[i] * log( y[i]) ;
}
*pA = exp( ( sy · sxx - sx · sxy ) / ( m · sxx - sx · sx ) ) ;      *pB = exp( ( m sxy - sx sy ) / ( m · sxx - sx · sx ) ) ;
// se calculeaza eroarea
pentru i= 1 ÷ m
    *E = *E + ( y[i] - (*pA) * (*pB)x[i] )2
}

```

6.2. REGRESIA GEOMETRICĂ

Dacă funcția numerică din tabel se aseamănă cu o funcție de tip geometric, vom căuta să determinăm funcția de forma:

$$y = ax^b$$

care să aproximeze cel mai bine funcția numerică. Adică aceeași condiție de eroare pătratică, de ansamblu, minimă.

Și în această situație se recurge la o logaritmare a relației, pentru a aduce rezolvarea în domeniul liniar. Avem:

$$\ln y = \ln(a) + b \cdot \ln(x)$$

Se obțin următoarele valori pentru a și b :

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

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

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

6.2.1 Algoritmul. Regresia geometrică

```

void Reg_Geo
(
    întreg m,                // numărul de abscise
    real x[ ],              // vectorul absciselor funcției numerice
    real y[ ],              // vectorul ordonatelor funcției numerice
    real *pA,                // adresa coef lui a al fct geometrice (pointer)

```

Îndrumar de laborator pentru Metode Numerice

```

real *pB // adresa coef lui a al fct geometrice (pointer)
real *E // adresa erorii (pointer)

)
{ // declararea și definirea variabilelor locale
real s_x ;
real s_y ;
real s_xy ;
real s_xx ;
întreg i ;

// corpul de instrucțiuni al funcției
s_x = 0 ; s_y = 0 ; s_xx = 0 ; s_xy = 0 ;
pentru i= 1÷ m
{
s_x = s_x + log(x[i]) ;
s_xx = s_xx + log(x[i])*log(x[i]) ;
s_y = s_y + log(y[i]) ;
s_xy = s_xy + log(x[i])* log(y[i]) ;
}

*pA = exp( (s_y * s_xx - s_x * s_xy) / (m * s_xx - s_x * s_x) ); *pB = (m * s_xy - s_x * s_y) / (m * s_xx - s_x * s_x) ;
// se calculeaza eroarea
pentru i= 1÷ m
*E=*E+( y[i] - (*pa) *x[i] (*pb) )^2
}

```

6.3. REGRESIA TRIGONOMETRICĂ

Se consideră funcția numerică dată în tabelul 8.1 și funcția trigonometrică de forma:

$$y = a + b \cdot \cos(\omega \cdot x)$$

care aproximează cel mai bine funcția numerică.

Valorile lui a și b sunt:

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

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

$$E = \sum_{i=1}^m (y_i - a - b * \cos(\omega x_i))^2$$

$\omega = 2 \cdot \pi \cdot f = \frac{2 \cdot \pi}{T}$ unde f - frecvența, ω - pulsația, iar T - perioada.

ω se stabilește funcție de periodicitatea funcției numerice date.

6.3.1 Algoritm. Regresia trigonometrică

```

void Reg_Tri
(
întreg m, // numărul absciselor funcției numerice

```

```

real x[ ], // vectorul absciselor funcției numerice
real y[ ], // vectorul ordonatelor funcției numerice
real *pA, // adresa coef a din expresia fct trig. (pointer)
real *pB // adresa coef b din expresia fct trig. (pointer)
real *E // adresa erorii (pointer)
)
{ // declararea și definirea variabilelor locale
real ω; // pulsația (w=2*PI*T)
real s_y;
real s_cos_2x;
real s_cos_x;
real s_ycosx;
întreg i;
s_y = 0; s_cos_2x = 0; s_cos_x = 0; s_ycosx = 0; // corpul de instrucțiuni al funcției
pentru i= 1÷ m
{
s_y = s_y + y[i];
s_cos_2x = s_cos_2x + cos(ω x_i) * cos(ω x_i);
s_cos_x = s_cos_x + cos(ω x_i);
s_ycosx = s_ycosx + y_i * cos(ω x_i);
}
*pA =  $\frac{s_y \cdot s_{\cos 2x} - s_{\cos x} \cdot s_{y\cos x}}{m \cdot s_{\cos 2x} - s_{\cos x} \cdot s_{\cos x}}$ ; *pB =  $\frac{m \cdot s_{y\cos x} - s_{\cos x} \cdot s_y}{m \cdot s_{\cos 2x} - s_{\cos x} \cdot s_{\cos x}}$ ;
// se calculează eroarea
pentru i= 1÷ m
*E=*E+( y[i] - (*pa) - (*pb) * cos(w*x[i]) )^2
}

```

REZOLVAREA NUMERICĂ A ECUAȚIILOR DIFERENȚIALE DE ORDINUL 1

6.4. METODA LUI EULER ÎMBUNĂTĂȚITĂ (media pantelor)

Această metodă face parte din grupul de metode de tip *Runge-Kutta de ordin 2*. deoarece dă aceleași soluții ca și metoda lui Taylor până la h^2 .

Utilizează media pantelor din punctele (x_m, y_m) și (x_{m+1}, y_{m+1}) unde $x_{m+1} = x_m + h$ iar $y_{m+1} = y_m + hy'_m$ este ordonata obținută din ecuația tangentei în punctul M la curbă cu panta $y'_m = f(x_m, y_m)$. Grafic metoda este prezentată în figura 6.1. În acest punct H, de coordonate $(x_{m+1}, y_m + hy'_m)$ se calculează panta la curbă, anume:

$$\overline{y}_{m+1} = f(x_m + h, y_m + hy'_m)$$

Se face media dintre panta dreptei prin punctul M și panta dreptei prin punctul H, și se notează cu:

$$z_m = \frac{1}{2} (f(x_m, y_m) + f(x_m + h, y_m + hy'_m))$$

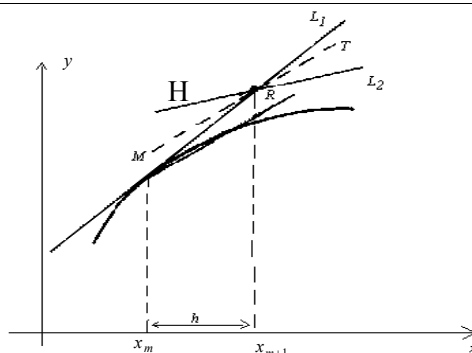


Fig. 6.1. - Calculul grafic al soluției prin metoda lui Euler îmbunătățită.

Dreapta de pantă z_m este HT. Cu această pantă z_m se trasează coarda prin M (echivalent cu a spune că se translatează dreapta HT în punctul M) și se intersectează cu dreapta (verticală) din punctul $x = x_{m+1}$, rezultând punctul R, de coordonate (x_{m+1}, y_{m+1}) considerat al doilea punct de pe curba soluție, primul fiind considerat $M(x_m, y_m)$. Ecuația coardei prin M este:

$$y = y_m + 0.5 \cdot \left(f(x_m, y_m) + f(x_m + h, y_m + h y'_m) \right) (x - x_m)$$

Punctul R, care reprezintă soluția numerică a ecuației diferențiale, are ordonata:

$$y_{m+1} = y_m + 0.5 \cdot h \cdot \left(f(x_m, y_m) + f(x_m + h, y_m + h y'_m) \right)$$

unde $y'_m = f(x_m, y_m)$.

Această formulă (de mai sus) reprezintă formula de calcul a soluțiilor numerice pentru o ecuație diferențială ordinară de ordinul întâi.

6.4.1. Algoritmul. Metoda lui Euler îmbunătățită

```

void Euler_Imb
(
  real x0,           //abscisa pct inițial prin care trece graficul soluției
  real y0,           //ordonata pct inițial prin care trece graficul soluției
  real h,            //pasul între abscisele punctelor de calcul
  real y[],          //vectorul ordonatelor (ce vor fi calculate)
  real x[],          //vectorul absciselor (ce vor fi calculate)
  întreg n           //numarul de esantioane ale solutiei ec diferențiale
)
{
  //declarațiile și definițiile variabilelor locale
  întreg i;

  y[0]= y0;  x[0]=x0;
  pentru i= 1÷ n
  {
  x[i]= x[0]+i*h;
  y[i]=y[i-1]+0.5*h*( f(x[i-1], y[i-1])+f(x[i], y[i-1]+h*f(x[i-1], y[i-1])) );
  }
  }
    
```

6.5. METODA LUI EULER MODIFICATĂ (media punctelor)

În cadrul acestei metode nu se mediază pantele, ci se evaluează panta în punctul mediu.

Considerăm curba din figura 6.2 în care se dă punctul inițial M prin care să treacă curba soluție a ecuației diferențiale și dorim să determinăm cel de al doilea punct al soluției. Prin punctul (x_m, y_m) dat,

cunoscut sau determinat, se duce tangenta prin M și se determină pe această tangentă punctul H de coordonate (x_{m+1}, y_{m+1}) . Până în acest punct nu există deosebire cu metoda Euler îmbunătățită.

Tangenta prin punctul M are ecuația

$$y = y_m + f(x_m, y_m)(x - x_m)$$

Prin intersecția cu dreapta (verticala) în punctul $x = x_m + h$ rezultă coordonatele punctului H:

$$H(x_{m+1} = x_m + h, y_{m+1} = y_m + hf(x_m, y_m)), \text{ prezentat în figura 6.2.}$$

Metoda face media coordonatelor punctelor M și H și determină punctul:

$$P\left(x_p = x_m + \frac{h}{2}, y_p = y_m + \frac{h}{2}f(x_m, y_m)\right)$$

Se calculează panta soluției y în acest punct P și rezultă:

$$z_m = f\left(x_m + \frac{h}{2}, y_m + \frac{h}{2}f(x_m, y_m)\right)$$

Se scrie ecuația dreptei care trece prin P, de pantă z_m , și se intersectează cu verticala în punctul următor pe abscisă:

$$x_{m+1} = x_m + h$$

Dreapta MN are ecuația:

$$y = y_m + f\left(x_m + \frac{h}{2}, y_m + \frac{h}{2}f(x_m, y_m)\right)(x - x_m)$$

Prin intersecția dreptei (8.11) cu verticala în punctul $x = x_{m+1} = x_m + h$ se determină ordonata R:

$$y_{m+1} = y_m + hf\left(x_m + \frac{h}{2}, y_m + \frac{h}{2}f(x_m, y_m)\right)$$

care reprezintă formula de calcul a ordonatelor soluției ecuației diferențiale.

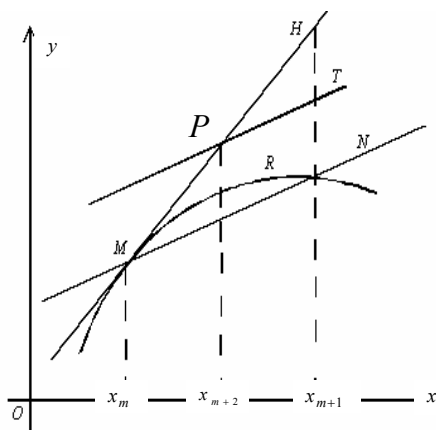


Fig.6.2. - Calculul grafic al soluției ecuației diferențiale prin metoda lui Euler modificată.

6.5.1. Algoritm. Metoda lui Euler modificată

```
void Eul_Mod
(
  real x0,           //abscisa pct inițial prin care trece graficul soluției
  real y0,           //ordonata pct inițial prin care trece graficul soluției
  real h,            //pasul între abscisele punctelor de calcul
  real y[],          // vectorul ordonatelor (ce vor fi calculate)
  real x[],          // vectorul absciselor (ce vor fi calculate)
  intreg n           //numarul de esntioane ale solutiei ec diferențiale
)
```

Îndrumar de laborator pentru Metode Numerice

```
{                                     // declarațiile și definițiile variabilelor locale
întreg i;

    x[0]= x0;  y[0]= y0;
    pentru i= 1÷n
    {                                     // se calculează abscisele și ordonatele soluției
    x[i]= x[0] + i*h;
    y[i]=y[i-1]+h*( f(x[i-1]+0.5*h, y[i-1]+0.5*h*f(x[i-1],y[i-1])) );
    }
}
```