METODE ITERATIVE PENTRU REZOLVAREA SISTEMELOR

Rezolvarea sistemelor liniare

Metoda Jacobi

Un sistem de ecuații liniare este caracterizat prin aceea că toate necunoscutele ecuațiilor sistemului sunt la puterea întâi. Rezolvarea unui astfel de sistem, se poate realiza folosind funcția *[x_1, x_2, ..., x_n]=solve(ecuația_1, ecuația_2, ..., ecuația_n))*. De exemplu, sistemul $\begin{cases} 4\cdot x_1 - x_2 + x_3 = 7 \\ 4\cdot x_1 - 8\cdot x_2 + x_3 = -21 \\ -2\cdot x_1 + x_2 + 5\cdot x_3 = 15 \end{cases}$, admite soluțiile:

```
» syms x_1 x_2 x_3
» [x_1,x_2,x_3]=solve(4*x_1-x_2+x_3-7,4*x_1-8*x_2+x_3+21,...
                      -2*x_1+x_2+5*x_3-15)
x_1 =
       2
x_2 =
       4
x_3 =
       3
```

Dacă acest sistem se rescrie, sub forma $\begin{cases} x_1 = \dfrac{7 + x_2 - x_3}{4} \\ x_2 = \dfrac{21 + 4\cdot x_1 + x_3}{8} \\ x_3 = \dfrac{15 + 2\cdot x_1 - x_2}{5} \end{cases}$, această formulă devine formula de iterare Jacobi, scrisă sub o formă mai sugestivă:

$$\begin{cases} x_{1,k+1} = \dfrac{7 + x_{2,k} - x_{3,k}}{4} \\ x_{2,k+1} = \dfrac{21 + 4 \cdot x_{1,k} + x_{3,k}}{8} \\ x_{3,k+1} = \dfrac{15 + 2 \cdot x_{1,k} - x_{2,k}}{5} \end{cases}$$

. Pornind calculele iterative, de la soluția inițială $(x_{1,,0}; x_{2,0}; x_{3,0})$=(1,2,2), se obțin valorile: $(x_{1,,1}; x_{2,1}; x_{3,1})$=(1,75; 3,375; 3,00), apoi considerând valorile $(x_{1,,1}; x_{2,1}; x_{3,1})$, valori inițiale, se determină noile valori **k**, și, continuând, se găsesc, după un număr de pași, valori tot mai apropiate de valorile exacte, calculate cu funcția **solve()**. Folosind această formulare a iterării se poate admite ca util un fișier-funcție, care să realizeze iterarea numerică pentru calculul cu aproximație a soluției unui sistem de ecuații liniare:

```
function X=sist_jacobi(A,B,X0,toler,iter)

% Argumente de intrare
    %-A=matricea patratica a coeficientilor: size(A)=n
    %-B=matricea coloana a termenilor liberi: size(B)=(n,1)
    %-X0=matricea initiala
    %-toler=abaterea radacinilor
    %-iter=numarul de iteratii

A=input('Introduceti matricea patratica a coeficientilor:');
B=input('Introduceti matricea coloana a termenilor liberi:');
X0=input('Introduceti solutia initiala (matrice coloana) pentru startul iterarii:');
toler=input('Introduceti abaterea fata de valoarea corecta:');
iter=input('Dati numarul maxim de iteratii:');

% Argumente de iesire
    %-X=matricea coloana a solutiilor sistemului AX=B

    %NOTATII
    %-eps=eroarea de calcul implicita: eps=2.2204e-016

N=length(B);
it_car=num2str(iter);
num_de_car=length(it_car);
```

```
for k=1:iter
   for j=1:N
      X(j)=(B(j)-A(j,[1:j-1,j+1:N])*X0([1:j-1,j+1:N]))/A(j,j);
   end
   eroarea=abs(norm(X'-X0));
   eroarea_relativa=eroarea/(norm(X)+eps);
   X0=X';
   if(eroarea<toler)|(eroarea_relativa<toler)
      break
   end
end
X=X';
disp('|.......................................................|')
fprintf(1,'S O L U T I A   D U P A   ~~%g~~ I T E R A T I I
 E S T E:\n',iter)
disp('|.......................................................|')
X;
```

Rezolvarea sistemului anterior folosind metoda iterării a lui Jacobi, oferă următoarele rezultate, obținute aplicând fișierul funcție **sist_jacobi.m**. Au fost necesari 19 pași de iterare pentru a obține cea mai apropiată soluție a sistemului:

```
» sist_jacobi
Introduceti matricea patratica a coeficientilor:[4,-1,1;4,-8,1;-2,1,5]
Introduceti matricea coloana a termenilor liberi:[7;-21;15]
Introduceti solutia initiala (matrice coloana) pentru startul
iterarii:[1;2;2]
Introduceti abaterea fata de valoarea corecta:10^-10
Dati numarul maxim de iteratii:19
|...................................................................................|
S O L U T I A   D U P A   ~~19~~ I T E R A T I I   E S T E:
|...................................................................................|

ans =
   1.99999999930477
   3.99999999826193
   3.00000000000000
```

Metoda Gauss- Seidel

Metoda Gauss-Seidel pentru rezolvarea numerică prin iterare a sistemelor de ecuații liniare, implică anumite considerente de procedură mai eficiente, în ceea ce privește obținerea soluției celei mai convenabile pentru un sistem. Astfel, dacă valoarea $x_{1,k+1}$, de exemplu, poate fi considerată o aproximare mai bună pentru soluția $x_{1,k}$, atunci, este indicat ca în calculul soluției $x_{2,k+1}$ să se folosească valoarea $x_{1,k+1}$. În acest sens, se poate scrie noua exprimare analitică a metodei, prin extrapolarea la celelalte necunoscute:

$$\begin{cases} x_{1,k+1} = \dfrac{7 + x_{2,k} - x_{3,k}}{4} \\ x_{2,k+1} = \dfrac{21 + 4 \cdot x_{1,k+1} + x_{3,k}}{8} \\ x_{3,k+1} = \dfrac{15 + 2 \cdot x_{1,k+1} - x_{2,k+1}}{5} \end{cases}.$$

Pentru aplicarea acestei formule de calcul iterativ un fișier funcție ar putea avea următoarea construcție:

```
function X=sist_gauss_seidel(A,B,X0,toler,iter)

% Argumente de intrare
    %-A=matricea patratica a coeficientilor: size(A)=n
    %-B=matricea coloana a termenilor liberi: size(B)=(n,1)
    %-X0=matricea initiala
    %-toler=abaterea radacinilor
    %-iter=numarul de iteratii

A=input('Introduceti matricea patratica a coeficientilor:');
B=input('Introduceti matricea coloana a termenilor liberi:');
X0=input('Introduceti solutia initiala (matrice coloana) pentru startul
iterarii:');
toler=input('Introduceti abaterea fata de valoarea corecta:');
iter=input('Dati numarul maxim de iteratii:');

% Argumente de iesire
    %-X=matricea coloana a solutiilor sistemului AX=B

    %NOTATII
```

```
%-eps=eroarea de calcul implicita: eps=2.2204e-016

N=length(B);
it_car=num2str(iter);
num_de_car=length(it_car);
for k=1:iter
   for j=1:N
      if j==1
         X(1)=(B(1)-A(1,2:N)*X0(2:N))/A(1,1);
      elseif j==N
         X(N)=(B(N)-A(N,1:N-1)*(X(1:N-1))')/A(N,N);
      else
         X(j)=(B(j)-A(j,1:j-1)*X(1:j-1)'-A(j,j+1:N)* X0(j+1:N))/A(j,j);
      end
   end

   eroarea=abs(norm(X'-X0));
   eroarea_relativa=eroarea/(norm(X)+eps);
   X0=X';
   if(eroarea<toler)|(eroarea_relativa<toler)
      break
   end
end
X=X';
disp('|.........................................................|')
fprintf(1,'S O L U T I A   D U P A   ~~%g~~ I T E R A T I I   E S T
E:\n',iter)
disp('|.........................................................|')
X;
```

Rezolvarea sistemului anterior, folosind fişierul funcție **sist_gauss_seidel.m**, oferă următoarele rezultate:

```
sist_gauss_seidel
Introduceti matricea patratica a coeficientilor:[4,-1,1;4,-8,1;-2,1,5]
Introduceti matricea coloana a termenilor liberi:[7;-21;15]
Introduceti solutia initiala (matrice coloana) pentru startul
iterarii:[1;2;2]
```

Introduceti abaterea fata de valoarea corecta:10^-10
Dati numarul maxim de iteratii:1000
|..|
S O L U T I A   D U P A   ~~1000~~   I T E R A T I I   E S T E:
|..|

ans =
1.99999999996036
3.99999999997090
2.99999999998997

Rezolvarea sistemelor neliniare

Sistemele de ecuații neliniare conțin, în exprimarea analitică a ecuațiilor, necunoscute la puteri diferite de unitate. Aceste reprezentări sunt ecuații ale curbelor de diferite ordine.

Rezolvarea sistemelor de ecuații neliniare este posibilă atât exact (folosind, în MATLAB, facilitățile toolbox-ului Symbolic Math Toolbox, prin funcția *[x,y]=solve(ecuația_1, ecuația_2, ..., ecuația_n)*), cât și folosind diferite metode iterative, pornind de la o soluție inițială de aproximare (metoda Seidel, metoda Newton-Raphson).

Rezolvarea sistemelor neliniare prin utilizarea funcției **solve()** este deosebit de avantajoasă, prin faptul că oferă soluții exacte și rapide, în cazul sistemelor cu număr mare de ecuații. Dezavantajul major al folosirii acestei funcții constă în faptul că, în multe cazuri sunt oferite soluții redundante, provenind din situații de simetrie a soluțiilor. De exemplu, pentru sistemul de ecuații $\begin{cases} a^3 \cdot b^2 = 0 \\ a - 2 \cdot b = p \end{cases}$ se obțin, datorită primei ecuații, soluții redundante $a_1=a_2=a_3=0$, conform succesiunii:

```
» syms a b p
» [a b]=solve('a^3*b^2=0','a-2*b=p',a,b)
a =
      [ 0]
      [ 0]
      [ 0]
      [ p]
      [ p]
b =
      [ -1/2*p]
```