Metoda Newton – Raphson pentru rezolvarea sistemelor neliniare

Noţiuni privind Jacobianul sistemelor matriciale

Prezentarea metodei Newton – Raphson necesită câteva preliminarii în ceea ce priveşte definirea Jacobianului asociat unui sistem de mai multe funcţii cu mai multe variabile.

Sistemele de funcţii cu **n** variabile implică definirea derivatei matriciale, adică a Jacobianului ca generalizare a derivatei parţiale a unei funcţii.

Dacă se consideră două funcţii, $\mathbf{f_1(x,y)}$ şi $\mathbf{f_2(x,y)}$, de două variabile independente, **x** şi **y**, atunci Jacobianul **J(x,y)** este definit de matricea: $\begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \end{bmatrix}$

respectiv, pentru trei funcţii de trei variabile independente $\mathbf{f_1(x,y,z)}$, $\mathbf{f_2(x,y,z)}$ şi $\mathbf{f_3(x,y,z)}$, atunci Jacobianul acestor funcţii este: $\begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial z} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial z} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} & \frac{\partial f_3}{\partial z} \end{bmatrix}$.

Pentru implementarea metodei lui Newton pentru sisteme neliniare se consideră sistemul: $\begin{cases} u = f_1(x,y) \\ v = f_2(x,y) \end{cases}$, care poate fi considerat transformarea de la planul (x,y) la planul (u,v). Este de urmărit comportarea transformării în vecinătatea punctului (x_0, y_0) a cărei imagine este punctul (u_0, v_0). Dacă cele două funcţii au derivatele parţiale continue, atunci se poate folosi diferenţiala pentru scrierea sistemului aproximaţiilor liniare valabile în vecinătatea punctului (x_0,y_0):

$$u - u_0 \approx \frac{\partial}{\partial x} f_1(x_0, y_0) \cdot (x - x_0) + \frac{\partial}{\partial y} f_1(x_0, y_0) \cdot (y - y_0)$$

$$v - v_0 \approx \frac{\partial}{\partial x} f_2(x_0, y_0) \cdot (x - x_0) + \frac{\partial}{\partial y} f_2(x_0, y_0) \cdot (y - y_0)$$

$$\begin{bmatrix} \frac{\partial}{\partial x} f_1(p_0,q_0) & \frac{\partial}{\partial y} f_1(p_0,q_0) \\ \frac{\partial}{\partial x} f_2(p_0,q_0) & \frac{\partial}{\partial y} f_2(p_0,q_0) \end{bmatrix} \cdot \begin{bmatrix} \Delta p \\ \Delta q \end{bmatrix} \approx - \begin{bmatrix} f_1(p_0,q_0) \\ f_2(p_0,q_0) \end{bmatrix}$$

In această, ultimă, relație, prima paranteză dreaptă reprezintă Jacobianul de ordinul doi. Dacă acesta este o matrice nesingulară, atunci se poate rezolva sistemul scris în ultima variantă, în funcție de necunoscutele Δp și Δq :

$$\Delta P = [\Delta p \quad \Delta q]' = [p \quad q]' - [p_0 \quad q_0]'$$

folosind formula:

$$\Delta P \approx -J(p_0,q_0)^{-1} \cdot F(p_0,q_0)$$

după care, următoarea aproximație P_1 **a soluției** $P = \begin{bmatrix} p \\ q \end{bmatrix}$ **este:**

$$P_1 = P_0 + \Delta P = P_0 - J(p_0,q_0)^{-1} \cdot F(p_0,q_0).$$

În rezumat, se pot releva etapele metodei lui Newton:

1) –Se evaluează funcția: $F(P_k) = \begin{bmatrix} f_1(p_k,q_k) \\ f_2(p_k,q_k) \end{bmatrix}$
2) Se evaluează Jacobianul $J(P_k) = \begin{bmatrix} \frac{\partial}{\partial x} f_1(p_k,q_k) & \frac{\partial}{\partial y} f_1(p_k,q_k) \\ \frac{\partial}{\partial x} f_2(p_k,q_k) & \frac{\partial}{\partial y} f_2(p_k,q_k) \end{bmatrix}$
3) –Se rezolvă sistemul liniar: $J(P_k) \cdot \Delta P = -F(P_k)$, cu necunoscutele ΔP .
4) –Se determină următorul punct: $P_{k+1} = P_k + \Delta P$
5) Se repetă procesul de calcul iterativ.

Un fișier funcție-M, dezvoltat pentru rezolvarea numerică a sistemelor liniare ar putea fi următorul:

Acest sistem reprezintă o transformare locală liniară, şi face legătura între variaţiile variabilelor independente şi variaţiile variabilelor dependente. Folosind Jacobianul, se poate scrie:

$$\begin{bmatrix} u - u_0 \\ v - v_0 \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x} f_1(x_0, y_0) \cdot (x - x_0) & \frac{\partial}{\partial y} f_1(x_0, y_0) \cdot (x - x_0) \\ \frac{\partial}{\partial x} f_2(x_0, y_0) \cdot (x - x_0) & \frac{\partial}{\partial y} f_2(x_0, y_0) \cdot (x - x_0) \end{bmatrix} \cdot \begin{bmatrix} x - x_0 \\ y - y_0 \end{bmatrix}.$$

Dacă sistemul iniţial considerat, se scrie ca o funcţie vectorială $\mathbf{V} = F(X)$, atunci Jacobianul J(x,y) este analog derivatei unei funcţii de două variabile, întrucât relaţia anterioară se poate scrie:

$$\Delta F \approx J(x_0, y_0) \cdot \Delta X,$$

relaţie din care derivă metoda lui Newton.

Pentru a dezvolta metoda lui Newton se consideră sistemul iniţial, $\begin{cases} u = f_1(x, y) \\ v = f_2(x, y) \end{cases}$, în care **u** şi **v** au valoarea zero: $\begin{cases} 0 = f_1(x, y) \\ 0 = f_2(x, y) \end{cases}$ şi se presupune că (**p**,**q**) este o soluţie a acestui sistem. Pentru variaţii mici ale funcţiilor în vecinătatea punctului ($\mathbf{p_0}$,$\mathbf{q_0}$), se poate scrie:

$$\Delta u = u - u_0, \qquad \Delta p = x - p_0.$$

$$\Delta v = v - v_0, \qquad \Delta q = y - q_0.$$

Dacă (**x**, **y**)=(**p**,**q**), în sistemul iniţial, şi folosind $\begin{cases} 0 = f_1(x, y) \\ 0 = f_2(x, y) \end{cases}$, se obţin vari

aţiile variabilelor dependente:

$$u - u_0 = f_1(p, q) - f_1(p_0, q_0) = 0 - f_1(p_0, q_0)$$

$$v - v_0 = f_2(p, q) - f_2(p_0, q_0) = 0 - f_2(p_0, q_0)$$

ceea ce permite să se scrie:

```
function [P,iter_max]=sist_nelin_newton(S,P,toler,iter)
%ACEST FISIER FUNCTIE RULEAZA PENTRU DOUA ECUATII NELINIARE
% ~~~Argumente de intrare~~~
%-S=sistemul neliniar
%-P=matricea initiala de aproximare
%-toler=limita abaterii radacinilor
%-iter=numarul de iteratii presupuse de utilizator
% ~~~Argumente de iesire~~~
%-Aprox=matricea coloana a solutiilor sistemului neliniar
%-iter_max=numarul de iteratii necesare aproximarii
%NOTATII
%-eps=eroarea de calcul implicita: eps=2.2204e-016
P=input('Introduceti solutia initiala (matrice coloana) pentru startul iterarii:');
toler=input('Introduceti abaterea fata de valoarea corecta:');
iter=input('Dati numarul maxim de iteratii:');
syms x y z
S=zeros(1,2);
s_1=input('Introduceti prima ecuatie (Sir de caractere):');
s_2=input('Introduceti a doua ecuatie (Sir de caractere):');
S=[s_1;s_2]
Y=subs(S,{x,y},P);
jacobian_nelin=jacobian(S,[x,y])
for k=1:iter
  J=subs(jacobian_nelin,{x,y},P);
  Q=P-(J\Y);
     A=subs(S,{x,y},Q);
   eroarea=norm(Q-P);
   eroarea_relativa=eroarea/(norm(Q)+eps);
   P=Q;
   Y=A;
   iter_max=k;

   if(eroarea<toler)|(eroarea_relativa<toler)
      break
  end
end
eroarea
eroarea_relativa
```

```
toler
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_max)
disp('|......................................................|')
```

Aplicând acest fişier pentru sistemul: $\begin{cases} 0 = f_1(x,y) = x^2 - y - 0.2 \\ 0 = f_2(x,y) = y^2 - x - 0.3 \end{cases}$, considerând ca punct de start punctul de coordonate (p_0,q_0)= (1,2;1,2), soluțiile se obțin după patru iterații:

```
[P,iter_max]=sist_nelin_newton
Introduceti solutia initiala (matrice coloana) pentru startul
iterarii:[1.2;1.2]
Introduceti abaterea fata de valoarea corecta:10^-10
Dati numarul maxim de iteratii:100
Introduceti prima ecuatie (Sir de caractere):x^2-y-0.2
Introduceti a doua ecuatie (Sir de caractere):y^2-x-0.3
S =
[  x^2-y-1/5]
[ y^2-x-3/10]
jacobian_nelin =
[ 2*x,  -1]
[  -1, 2*y]
eroarea =
   8.005932084973443e-016
eroarea_relativa =
   4.690010571570136e-016
toler =
   1.000000000000000e-010
|......................................................|
S O L U T I A   D U P A   ~~4~~ I T E R A T I I   E S T E:
|......................................................|
P =
   1.19230912514880
   1.22160104991311
iter_max =
                    4
```

şi apoi pentru punctul $(p_0,q_0)=(-0,2;-0,2)$, se obţin, după cinci iteraţii, rezultatele:

```
[P,iter_max]=sist_nelin_newton
Introduceti solutia initiala (matrice coloana) pentru startul iterarii:
[-0.2;-0.2]
Introduceti abaterea fata de valoarea corecta:10^-10
Dati numarul maxim de iteratii:100
Introduceti prima ecuatie (Sir de caractere):x^2-y-0.2
Introduceti a doua ecuatie (Sir de caractere):y^2-x-0.3
S =
[  x^2-y-1/5]
[ y^2-x-3/10]
jacobian_nelin =
[ 2*x,  -1]
[  -1, 2*y]
eroarea =
    0
eroarea_relativa =
    0
toler =
   1.000000000000000e-010
|.......................................................|
S O L U T I A   D U P A   ~~5~~ I T E R A T I I   E S T E:
|.......................................................|
P =
  -0.28603216362886
  -0.11818560136979
iter_max =
                     5
```