7. REZOLVAREA SISTEMELOR DE ECUAŢII NELINIARE. REZOLVAREA PROBLEMELOR DE OPTIMIZARE

Obiectivele lucrării:

- fixarea de cunoştinţe privitoare la rezolvarea sistemelor de ecuaţii neliniare, atât pe cale numerică, cât şi pe cale simbolică, folosind mediul de programare Matlab,
- fixarea de cunoştinţe privitoare la rezolvarea problemelor de optimizare folosind mediul de programare Matlab,

prin studierea unor exemple şi prin rezolvarea unor probleme.

Se recomandă parcurgerea anexei M7 înaintea studierii paragrafelor 7.1 şi 7.2.

7.1. Elemente despre rezolvarea sistemelor de ecuaţii neliniare în Matlab. Elemente despre rezolvarea problemelor de optimizare în Matlab

Rezolvarea sistemelor de ecuaţii neliniare

Fie sistemul de ecuaţii neliniare (vezi anexa M7):

$$\begin{cases} f_1(x_1,x_2,\ldots,x_n)=c_1 \\ f_2(x_1,x_2,\ldots,x_n)=c_2 \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ f_n(x_1,x_2,\ldots,x_n)=c_n \end{cases}$$

Pentru a rezolva acest sistem cu ajutorul mediului Matlab, el trebuie adus la ***forma canonică***:

$$\begin{cases} g_1(x_1,x_2,\ldots,x_n)=0 \\ g_2(x_1,x_2,\ldots,x_n)=0 \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ g_n(x_1,x_2,\ldots,x_n)=0 \end{cases}$$

unde $g_i=f_i-c_i,\ i=1,2,\ldots,n$.

Un sistem de ecuaţii neliniare poate fi rezolvat în Matlab pe cale numerică sau pe cale simbolică.

a. Rezolvarea numerică a sistemelor de ecuaţii neliniare

Pentru rezolvarea sitemelor de ecuații neliniare pe cale numerică se parcurg etapele de separare şi rezolvare aproximativă menționate în anexa M7.

În ceea ce priveşte etapa localizării, în cazul sistemelor de ecuații neliniare cu două necunoscute:

$$\begin{cases} g_1(x_1,x_2)=0 \\ g_2(x_1,x_2)=0 \end{cases}$$

se poate folosi metoda grafică. În acest scop se reprezintă grafic în plan mulțimea punctelor care satisfac ecuația $g_1(x_1,x_2)=0$ şi mulțimea punctelor care satisfac ecuația $g_2(x_1,x_2)=0$. Coordonatele punctelor de intersecție ale celor două grafice reprezintă soluțiile sistemului de ecuații. Citirea acestora de pe grafic se poate face cu funcția Matlab `ginput` prezentată în paragraful 6.1. Deoarece citirea de pe grafic se face cu o anumită eroare, coordonatele citite de pe grafic vor constitui valorile în vecinătatea cărora sunt localizate soluțiile şi vor fi utilizate în a doua etapă ca valori de pornire în determinarea soluției.

Cea de-a doua etapă are ca scop calculul soluției / soluțiilor din domeniul de interes cu o precizie apriori fixată.

Pentru calculul soluțiilor este necesară definirea în prealabil într-un fişier-funcție a funcțiilor g_i ca şi componente ale unei funcții vectoriale.

Pentru calculul fiecărei soluții a sistemului de ecuații neliniare, în vecinătatea unui punct rezultat prin separare, se foloseşte funcția Matlab `fsolve` din toolbox-ul de optimizare (*Optimization Toolbox*). Această funcție are mai multe sintaxe de apel, analog funcției Matlab `fzero` (vezi paragraful 6.1):

- dacă interesează doar determinarea soluției, se foloseşte una din sintaxele de apel:

```
x = fsolve(nume_fisier,x0)
x = fsolve(nume_fisier,x0,optiuni)
```

- dacă în afară de determinarea soluției interesează, din considerente de precizie, şi evaluarea funcțiilor g_i pentru soluția găsită, se apelează funcția cu una din sintaxele:

```
[x,fval]= fsolve(nume_fisier,x0)
[x,fval]= fsolve(nume_fisier,x0,optiuni)
```

- dacă interesează şi modul în care s-a ajuns la oprirea executării metodei numerice (rezolvare cu respectarea condițiilor impuse, oprire prin atingerea numărului maxim de iterații, etc), se foloseşte una din sintaxele de apel:

```
[x,fval,exitflag]= fsolve(nume_fisier,x0)
[x,fval,exitflag]= fsolve(nume_fisier,x0,optiuni)
```

- dacă interesează şi anumite date suplimentare, precum numărul de iterații efectuate, se apelează funcția cu una din sintaxele:

```
[x,fval,exitflag,output]= fsolve(nume_fisier,x0)
[x,fval,exitflag,output]= fsolve(nume_fisier,x0,optiuni)
```

unde:

- `nume_fisier` reprezintă un şir de caractere care conține numele fişierului-funcție în care au fost definite funcțiile g_i ca şi componente ale unei funcții vectoriale;
- `x0` reprezintă vectorul valorilor aproximative ale soluției căutate;
- `optiuni` reprezintă o structură care conține opțiuni de optimizare a calculării soluției; este un argument opțional; opțiunile de optimizare pot fi schimbate folosind funcția Matlab `optimset` (vezi paragraful 6.1);
- `x` reprezintă soluția calculată cu o precizie apriori fixată, sub forma unui vector;
- `fval` reprezintă valorile funcțiilor g_i pentru soluția calculată x;
- `exitflag` reprezintă o valoare de control, care precizează motivul opririi algoritmului; de exemplu, valoarea 1 are semnificația că algoritmul a ajuns la o soluție în condițiile impuse;
- `output` reprezintă o structură care conține mai multe informații, printre care şi numărul de iterații (*iterations*), numărul de evaluări (*funcCount*) efectuate şi algoritmii utilizați pentru determinarea soluției (*algorithm*).

b. Rezolvarea simbolică a sistemelor de ecuaţii neliniare

Pentru rezolvarea pe cale simbolică a sistemelor de ecuații neliniare, se utilizează funcția Matlab `solve` din *Symbolic Math Toolbox*, având sintaxele:

```
s=solve(eq_1,eq_2,...,eq_n)
s=solve(eq_1,eq_2,...,eq_n,var_1,var_2,...,var_n)
```

unde:

- `eq_i` reprezintă expresia membrului stâng al ecuației i a sistemului, scrisă între apostrofuri;
- `var_1,var_2,...,var_n` reprezintă necunoscutele sistemului, scrise între apostrofuri;
- `s` este o structură, având ca membri vectori de valori; fiecare vector corespunde unei necunoscute, având aceeaşi denumire cu aceasta, şi conține valorile obținute pentru acea necunoscută în cazul fiecărei soluții găsite.

Avantajul rezolvării unui sistem de ecuații neliniare pe cale simbolică constă în faptul că nu trebuie indicat nici un set de valori de pornire în calculul soluției, iar valoarea soluției este determinată cu exactitate. Dezavantajul constă în faptul că puține sisteme pot fi rezolvate pe această cale.

Rezolvarea problemelor de optimizare

Deoarece orice problemă de maximizare poate fi transformată într-o problemă de minimizare prin schimarea semnului funcției obiectiv, se va considera în continuare doar problema minimizării unei funcții reale de una sau mai multe variabile reale.

Matlab pune la dispoziția utilizatorului mai multe funcții de rezolvare a problemelor de optimizare, grupate în toolbox-ul de optimizare (*Optimization Toolbox*). Aceste funcții implementează diverse metode numerice. Astfel:

- funcția Matlab `fminbnd` foloseşte o combinație a metodelor secțiunii de aur şi interpolare pătratică;
- funcția Matlab `fminunc` are implementată o combinație de mai multe metode, printre care metoda regiunii de încredere în model, metode cvasi-Newton, metode de interpolare pătratică şi metode de interpolare cubică;
- funcția Matlab `fminsearch` implementează metoda de căutare simplex.

Funcția Matlab `fminbnd` caută minimul local al unei funcții reale de o variabilă reală, situat într-un interval deschis. Sintaxele de apel ale funcției `fminbnd` sunt:

```
x = fminbnd(nume_fisier,a,b)
x = fminbnd(nume_fisier,a,b,optiuni)
[x,fval]= fminbnd(nume_fisier,a,b)
[x,fval]= fminbnd(nume_fisier,a,b,optiuni)
[x,fval,exitflag]= fminbnd(nume_fisier,a,b)
[x,fval,exitflag]= fminbnd(nume_fisier,a,b,optiuni)
[x,fval,exitflag,output]= fminbnd(nume_fisier,a,b)
[x,fval,exitflag,output]= fminbnd(nume_fisier,a,b,optiuni)
```

unde:

- `nume_fisier` reprezintă un şir de caractere care conține numele fişierului-funcție în care a fost definită expresia funcției obiectiv;
- `a, b` reprezintă capetele intervalului deschis în care se caută minimul local;
- `optiuni` reprezintă o structură care conține opțiuni de optimizare a calculării soluției / soluțiilor; este un argument opțional; opțiunile de optimizare pot fi schimbate folosind funcția Matlab `optimset` (a se vedea paragraful 6.1.);
- `x` reprezintă punctul de minim local;
- `fval` reprezintă valoarea funcției în punctul de minim local (minimul);
- `exitflag` reprezintă o valoare de control, care este 1 în caz că s-a ajuns la minimul local;
- `output` reprezintă o structură care conține următoarele informații: numărul de iterații (*iterations*), numărul de evaluări (*funcCount*) efectuate, algoritmii utilizați pentru determinarea soluției (*algorithm*) şi un mesaj de terminare (*message*).

Funcțiile Matlab `fminunc` şi `fminsearch` sunt folosite pentru rezolvarea problemelor de minimizare fără restricții. Sintaxele comune de apel ale ale celor două funcții sunt:

```
x = functie_Matlab(nume_fisier,x0)
x = functie_Matlab(nume_fisier,x0,optiuni)
[x,fval]= functie_Matlab(nume_fisier,x0)
[x,fval]= functie_Matlab(nume_fisier,x0,optiuni)
[x,fval,exitflag]= functie_Matlab(nume_fisier,x0)
[x,fval,exitflag]= functie_Matlab(nume_fisier,x0,optiuni)
[x,fval,exitflag,output]= functie_Matlab(nume_fisier,x0)
[x,fval,exitflag,output]= functie_Matlab(nume_fisier,x0,optiuni)
```

unde:

- `functie_Matlab` este una din funcțiile `fminunc`, `fminsearch`;
- `nume_fisier` reprezintă un şir de caractere care conține numele fişierului-funcție în care a fost definită expresia funcției obiectiv (funcție de una sau mai multe variabile);
- `x0` reprezintă valoarea inițială / vectorul valorilor inițiale, de la care se porneşte căutarea minimului;
- `optiuni` reprezintă o structură care conține opțiuni de optimizare a calculării soluției / soluțiilor; este un argument opțional; opțiunile de optimizare pot fi schimbate folosind funcția Matlab `optimset` (a se vedea paragraful 6.1.);
- `x` reprezintă punctul (vector) de minim local;
- `fval` reprezintă valoarea funcției în punctul de minim local (minimul);
- `exitflag` reprezintă o valoare de control care precizează motivul opririi algoritmului; de exemplu, în cazul funcției `fminsearch` valoarea exitflag=1 arată că algoritmul a ajuns la minimul local;
- `output` reprezintă o structură care conține următoarele informații: numărul de iterații (*iterations*), numărul de evaluări (*funcCount*) efectuate, precum şi algoritmii utilizați pentru determinarea soluției (*algorithm*).

7.2. Exemple de studiat

Exemplul 7.1: Să se determine soluția sistemului de ecuații neliniare:

$$\begin{cases} x \cdot y + z = -3 \\ \dfrac{x}{z} - y = -2 \\ y \cdot z + x = 6 \end{cases},$$

situată în vecinătatea punctului $x_0 = 1$, $y_0 = 0$, $z_0 = -1$.

Soluție: Deoarece se specifică valorile cu care se porneşte căutarea soluției, nu mai este necesar să se parcurgă etapa de localizare a soluției; se trece direct la etapa de calcul a soluției. În acest scop, se aduce mai întâi sistemul de ecuații neliniare la forma canonică:

$$\begin{cases} x \cdot y + z + 3 = 0 \\ \dfrac{x}{z} - y + 2 = 0 \\ y \cdot z + x - 6 = 0 \end{cases}$$

Apoi, se defineşte membrul stâng al sistemului sub forma canonică într-un fişier funcție (de exemplu, `sistnelin1.m`):

```
function g=sistnelin1(v)
% x,y,z sunt reprezentate de v(1),v(2),v(3)
x=v(1); y=v(2); z=v(3);
% membrul stang al ecuatiei i este reprezentat de g(i)
g(1)=x*y+z+3;
g(2)=x/z-y+2;
g(3)=y*z+x-6;
```

După definirea membrului stâng, pentru calculul soluției se execută următoarea secvență de program Matlab (de exemplu, fişier script):

```
% vectorul valorilor de aproximare initiala a solutiei
v0=[1; 0; -1];
% rezolvarea sistemului
sol=fsolve('sistnelin1',v0)
```

În urma execuției acestei secvențe se obține:

```
Optimization terminated: first-order optimality is less than
options.TolFun.
sol =    6.0000   -0.0000   -3.0000
```

Rezultă că soluția este x=6, y=0, z=-3.

Exemplul 7.2: Să se determine, pornind de la valorile (1,2,3,4), cvadruplul (a,b,c,d) care satisface simultan condițiile:

i) a, b, c, d sunt în progresie aritmetică,

ii) a-2, b-6, c-7, d-8 sunt în progresie geometrică,

cu precizia 10^{-30} atât pentru soluție, cât şi pentru funcțiile din membrul stâng al sistemului scris sub forma canonică.

Soluție: Condițiile i) şi ii) sunt echivalente cu următorul sistem de ecuații neliniare:

$$\begin{cases} a + c = 2 \cdot b \\ b + d = 2 \cdot c \\ (a-2) \cdot (c-7) = (b-6)^2 \\ (b-6) \cdot (d-8) = (c-7)^2 \end{cases}$$

a cărui formă canonică este:

$$\begin{cases} a+c-2\cdot b=0 \\ b+d-2\cdot c=0 \\ (a-2)\cdot(c-7)-(b-6)^2=0 \\ (b-6)\cdot(d-8)-(c-7)^2=0 \end{cases}$$

Având valorile de pornire în determinarea soluției, se trece direct la a doua etapă:

Se reprezintă membrul stâng al sistemului de ecuații neliniare într-un fişier funcție:

```
function g=sistnelin2(v)
% a,b,c,d sunt reprezentate de v(1),v(2),v(3),v(4)
% membrul stang al sistemului, scris sub forma de functie
% vectoriala
g=[v(1)+v(3)-2*v(2);
    v(2)+v(4)-2*v(3);
    (v(1)-2)*(v(3)-7)-(v(2)-6)^2;
    (v(2)-6)*(v(4)-8)-(v(3)-7)^2];
```

Pentru rezolvare se execută următoarea secvență Matlab (de exemplu, fişier script):

```
% vectorul valorilor de aproximare initiala a solutiei
v0=[1; 2; 3; 4];
% setarea proprietatilor cerute
optiuni=optimset('TolX',10^(-30),'TolFun',10^(-30));
% rezolvarea sistemului
[sol,gval]=fsolve('sistnelin2',v0,optiuni)
```

şi se obțin rezultatele:

```
Optimization terminated: relative function value changing by less
than max(options.TolFun^2,eps) and sum-of-squares of function values
is less than sqrt(options.TolFun).
sol =    4.9999    6.0000    7.0000    8.0000
gval =  1.0e-008 *
   -0.0001    0.0000   -0.1595   -0.1599
```

Prin urmare s-a obținut soluția: a=4.9999 (prin verificare rezultă valoarea exactă a=5), b=6, c=7, d=8.

Exemplul 7.3: Să se determine pe cale simbolică toate tripletele (a,b,c) care îndeplinesc simultan condițiile:

i) a, b, c sunt în progresie aritmetică,

ii) suma lor este 30,

iii) a-5, b-4, c sunt în progresie geometrică,

Soluție: Condițiilor i), ii), iii) le corespunde următorul sistem de ecuații neliniare scris sub forma canonică:

$$\begin{cases} a+c-2\cdot b=0 \\ a+b+c-30=0 \\ (a-5)\cdot c-(b-4)^2=0 \end{cases}.$$

Este uşor de observat că, primele două ecuații fiind de gradul I, iar a treia ecuație fiind de gradul II numai în raport cu *b*, sistemul are cel mult două soluții.

Pentru rezolvarea pe cale simbolică a acestui sistem se apelează funcția Matlab *solve* care primeşte ca argumente expresiile funcțiilor din membrul stâng al sistemului şi variabilele scrise între apostrofuri. Se execută următoarea secvență de program Matlab (de exemplu, fişier script):

```
sol=solve('a+c-2*b','a+b+c-30',...
    '(a-5)*c-(b-4)^2','a','b','c')
% afisarea solutiilor
for i=1:length(sol.a)
    disp([sol.a(i) sol.b(i) sol.c(i)])
end
```

În urma executării secvenței de mai sus se obține:

```
sol =
    a: [2x1 sym]
    b: [2x1 sym]
    c: [2x1 sym]
[ 17, 10,  3]
[  8, 10, 12]
```

S-au obținut două soluții: tripletele (*a*,*b*,*c*)=(17,10,3) şi (*a*,*b*,*c*)=(8,10,12).

Exemplul 7.4: Să se rezolve pe cale numerică sistemul de ecuații neliniare:

$$\begin{cases} x^2+y^2=2 \\ x^2-y^2=1 \end{cases}$$

Soluție: Se observă că prima ecuație a sistemului reprezintă ecuația implicită a cercului cu centrul în originea sistemului de coordonate, (0,0), şi de rază $\sqrt{2}$, iar a doua ecuație reprezintă ecuația implicită a unei hiperbole echilatere de semiaxe 1.

1. Pentru a localiza soluțiile sistemului de ecuații neliniare se aplică metoda grafică (a se vedea paragraful 7.1): se reprezintă grafic cele două figuri geometrice plane şi se „citesc” cu mouse-ul coordonatele punctelor de intersecție. În acest scop se execută următoarea secvență Matlab (de exemplu, fişier script):

```
% cerc de centru (0,0) si raza sqrt(2)
theta=0:pi/60:2*pi; r=sqrt(2);
x=r*cos(theta); y=r*sin(theta);
plot(x,y,'r--')
hold on
% hiperbola echilatera de centru (0,0) si semiaxe 1
% y^2=x^2-1 => x^2>=1, deci x<=-1 sau x>=1,
```

```
% si y1=sqrt(x^2-1), y2=-sqrt(x^2-1)=-y1
x1=-3:0.1:-1; y11=sqrt(x1.^2-1); y12=-y11;
x2=sort(-x1); y21=sqrt(x2.^2-1); y22=-y21;
plot(x1,y11,'b',x1,y12,'b',x2,y21,'b',x2,y22,'b')
axis equal, grid, hold off
[xcoord,ycoord]=ginput;
```

Graficul este redat în figura 7.1.

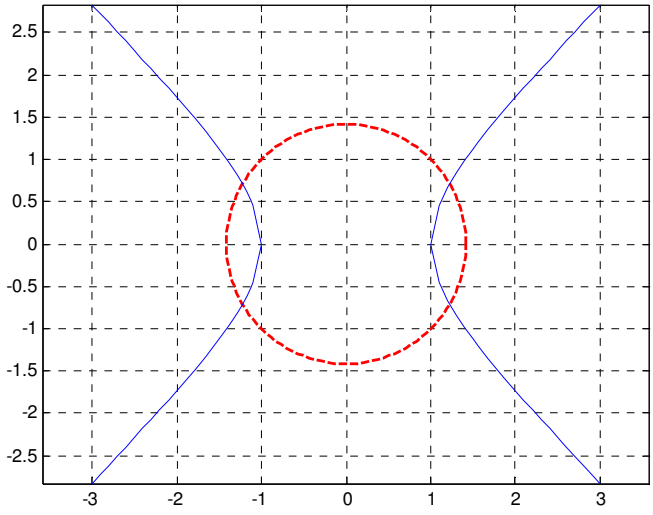

Fig.7.1. *Graficul figurilor geometrice din exemplul 7.4.*

Din grafic se observă că există patru puncte de intersecție, deci sistemul are 4 soluții.

2. Etapa a doua – de calcul a soluțiilor – începe cu aducerea sistemului la forma canonică şi definirea membrului stâng al sistemului într-un fişier funcție:

```
function y=sistNelin(v)
x=v(1); y=v(2);
y=[x.^2+y.^2-2; x.^2-y.^2-1];
```

Pentru a determina soluțiile, se apelează funcția Matlab *fsolve* de atâtea ori câte puncte de intersecție au fost găsite, fiecare apel având ca valori de pornire câte un set de coordonate „citite”. În acest scop se completează codul sursă de la punctul 1 cu următoarele instrucțiuni:

```
disp('solutiile sistemului')
for i=1:length(xcoord)
    sol=fsolve('sistNelin',[xcoord(i),ycoord(i)])
    if i<length(xcoord) pause, end
end
```

Soluțiile sunt:

```
solutiile sistemului
Optimization terminated: first-order optimality is less than
options.TolFun.
sol =   -1.2247    0.7071
Optimization terminated: first-order optimality is less than
options.TolFun.
```

```
sol =    1.2247    0.7071
Optimization terminated: first-order optimality is less than
options.TolFun.
sol =   -1.2247   -0.7071
Optimization terminated: first-order optimality is less than
options.TolFun.
sol =    1.2247   -0.7071
```

Exemplul 7.5: Să se determine punctul de minim situat în vecinătatea punctului *x0*=2 şi minimul local al funcției:

$f(x)=x-\sin(x\cdot\pi)-3,\ x\in \boldsymbol{R}$.

Soluție: Se parcurg următorii paşi:

Pasul 1. Se defineşte expresia funcției obiectiv într-un fişier funcție (de exemplu, `f.m`):

```
function y=f(x)
y=x-sin(pi*x)-3;
```

Pasul 2. Se caută minimul folosind una din funcțiile Matlab *fminunc* sau *fminsearch*. În cazul acestui exemplu s-a utilizat a doua funcție. Se execută următoarea secvență de instrucțiuni Matlab (de exemplu, fişier script):

```
[xmin,fmin,ct_terminare,detalii]=fminsearch('f',2)
disp(detalii.message)
```

Pasul 3. În urma execuției secvenței de la pasul 2. se obține:

```
xmin =    2.3969
fmin =   -1.5511
ct_terminare =     1
detalii =
    iterations: 13
     funcCount: 26
     algorithm: 'Nelder-Mead simplex direct search'
       message: [1x196 char]
Optimization terminated:
 the current x satisfies the termination criteria using OPTIONS.TolX
of 1.000000e-004 and F(X) satisfies the convergence criteria using
OPTIONS.TolFun of 1.000000e-004
```

S-a obținut punctul de minim 2.3969, minimul funcției fiind -1.5511. Căutarea minimului s-a terminat cu succes (constanta de terminare fiind 1). Au fost executate 13 iterații, funcția a fost apelată de 26 de ori, s-a utilizat metoda căutării directe simplex în varianta Nelder-Mead. Precizia este de 10^{-4} atât pentru punctul de minim, cât şi pentru minim (valoarea funcției în punctul de minim).

Exemplul 7.6: Fie funcția $f:\boldsymbol{R}\to\boldsymbol{R}$, $f(x)=\left(\frac{x}{4}\right)^2-sin(x)-0.5$. Să se determine:

i) un punct de minim local din intervalul (-8,10), precum şi minimul corespunzător al funcției;

ii) toate punctele de extrem local din intervalul (-8,10), precum şi valorile funcției în aceste puncte, precizând totodată şi tipul de extrem pentru fiecare punct în parte.

Soluție: i) Pasul 1. Se defineşte expresia funcției obiectiv într-un fişier funcție (de exemplu, `fct.m`):

```
function y=fct(x)
y= (x/4).^2-sin(x)-0.5;
```

Pasul 2. Deoarece căutarea punctului minim se face într-un interval, se va utiliza funcția Matlab *fminbnd*.

Se execută următoarea secvență de instrucțiuni Matlab (de exemplu, fişier script):

```
[xmin,fmin,ct_terminare,detalii]=fminbnd('fct',-8,10)
disp(detalii.message)
```

Pasul 3. În urma execuției secvenței de la pasul 2. se obține:

```
xmin =    1.3955
fmin =   -1.3630
ct_terminare =     1
detalii =
    iterations: 9
     funcCount: 10
     algorithm: 'golden section search, parabolic interpolation'
       message: [1x112 char]
Optimization terminated:
 the current x satisfies the termination criteria using OPTIONS.TolX
of 1.000000e-004
```

S-a obținut punctul de minim local 1.3955, minimul funcției fiind -1.3630. Căutarea minimului s-a terminat cu succes (constanta de terminare fiind 1). Au fost executate 9 iterații, funcția a fost apelată de 10 de ori. Au fost utilizate metoda secțiunii de aur şi metoda de interpolare pătratică. Punctul de minim a fost determinat cu precizia 10^{-4}.

ii) Pasul 1 este identic cu primul pas de la punctul i).

Pasul 2. Pentru a afla câte puncte de extrem local se află în intervalul (-8,10), funcția obiectiv se reprezintă grafic pe acest interval. Pe baza graficului care se obține, se alege pentru fiecare punct de extrem un interval de vecinătate, în care să nu se afle nici un alt punct de extrem. În final, se apelează funcția Matlab *fminbnd* pentru fiecare punct de extrem în parte, căutarea fiecărui punct făcându-se în intervalul de vecinătate în care acesta se situează.

Se execută următoarea secvență de instrucțiuni Matlab (de exemplu, fişier script):

```
clear, clc
% graficul
x=-8:0.1:10; plot(x,fct(x))

% functia -f pentru determinarea maximelor locale
g=@(x) -fct(x);

% determinarea punctelor de extrem si a minimelor si maximelor
% locale
[xmin1,fmin1]=fminbnd('fct',-6,-3);
fprintf('punct de minim: %g, minimul: %g\n',xmin1,fmin1)
[xmax1,fmax1]=fminbnd(g,-3,-1);
fprintf('punct de maxim: %g, maximul: %g\n',xmax1,-fmax1)
[xmin2,fmin2]=fminbnd('fct',0,3);
fprintf('punct de minim: %g, minimul: %g\n',xmin2,fmin2)
[xmax2,fmax2]=fminbnd(g,4,6);
fprintf('punct de maxim: %g, maximul: %g\n',xmax2,-fmax2)
[xmin3,fmin3]=fminbnd('fct',6,8);
fprintf('punct de minim: %g, minimul: %g\n',xmin3,fmin3)
```

Pasul 3. În urma execuției secvenței de la pasul 2. se obțin cele 5 puncte de extrem (graficul funcției este redat în figura 7.2):

```
punct de minim: -4.16483, minimul: -0.269685
punct de maxim: -1.79742, maximul: 0.67635
punct de minim: 1.39547, minimul: -1.36296
punct de maxim: 5.46431, maximul: 2.09655
punct de minim: 6.83067, minimul: 1.89559
```

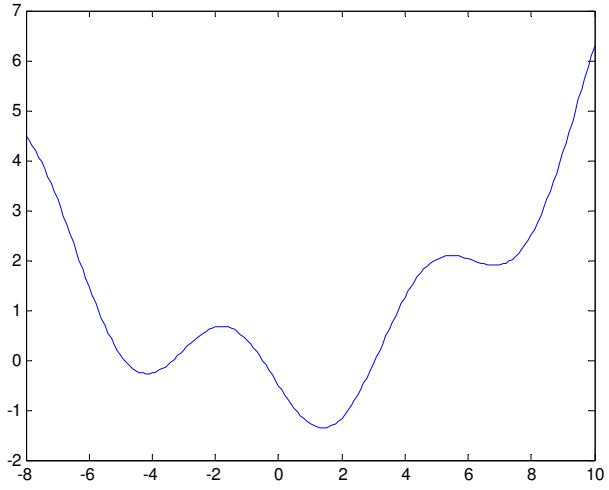

Fig.7.2. *Graficul funcției obiectiv din exemplul 7.6.*

Exemplul 7.7: Folosind funcția Matlab bazată pe metode cvasi-Newton să se rezolve următoarea problemă de optimizare fără restricții:

$$(\hat{x},\hat{y},\hat{z}) = arg \min_{(x,y,z)} f(x,y,z), \qquad f(x,y,z) = \frac{x \cdot y \cdot z}{(x-y)^2 + (y-z)^2 + 1},\ (x,y,z) \in \mathbf{R^3},$$

pornind de la aproximarea (1,0,-1).

Soluţie: Pasul 1. Se defineşte expresia funcţiei obiectiv într-un fişier funcţie (de exemplu, `fct2.m`):

```
function y=fct2(v)
y=v(1)*v(2)*v(3)/((v(1)-v(2))^2+(v(2)-v(3))^2+1);
```

Pasul 2. Se caută minimul folosind funcţia Matlab *fminunc*. Se execută următoarea secvenţă de instrucţiuni Matlab (de exemplu, fişier script):

```
clear, clc
v0=[1;0;-1];
options=optimset('LargeScale','off');
[vmin,fmin,exitflag,output]=fminunc('fct2',v0,options);
fprintf('Punctul de minim [%g %g %g]\n',vmin)
fprintf('Minimul: %g\n',fmin)
fprintf('Metoda de optimizare: %s\n',output.algorithm)
```

Pasul 3. În urma execuţiei secvenţei de la pasul 2. se obţine:

```
Optimization terminated: relative infinity-norm of gradient less
than options.TolFun.
Punctul de minim [4.53616e+015 4.56755e+015 -2.25904e+015]
Minimul: -1.00434e+015
Metoda de optimizare: medium-scale: Quasi-Newton line search
```

Exemplul 7.8: Să se rezolve următoarea problemă de optimizare fără restricţii, folosind funcţia Matlab bazată pe metoda de căutare simplex:

$$(\hat{x},\hat{y}) = arg \min_{(x,y)} f(x,y), \qquad f(x,y) = x \cdot e^{-x^2-y^2}, \ (x,y) \in \boldsymbol{R^2}$$

pornind de la aproximarea (0,0).

Soluţie: Pasul 1. Se defineşte expresia funcţiei obiectiv într-un fişier funcţie (de exemplu, `fvect.m`):

```
function y=fvect(v)
y=v(1)*exp(-v(1)^2-v(2)^2);
```

Pasul 2. Se caută minimul folosind funcţia Matlab *fminsearch*. Se execută următoarea secvenţă de instrucţiuni Matlab (de exemplu, fişier script):

```
clear, clc
[vmin,fmin,exitflag,output]=fminsearch('fvect',[0 0]);
fprintf('Punctul de minim [%g %f]\n',vmin)
fprintf('Minimul: %g\n',fmin)
fprintf('Metoda de optimizare: %s\n',output.algorithm)
```

Pasul 3. În urma execuţiei secvenţei de la pasul 2. se obţine:

```
Punctul de minim [-0.707127 0.000042]
Minimul: -0.428882
Metoda de optimizare: Nelder-Mead simplex direct search
```

7.3. Probleme de rezolvat

P7.1. Să se determine o soluție a sistemului de ecuații neliniare:

$$\begin{cases} \sin(x+y) - 1.1 \cdot x = 0.2 \\ 1.1 \cdot x^2 + 2 \cdot y^2 = 1 \end{cases}$$

pornind de la aproximația inițială x_0=1, y_0=1.

P7.2. Să se determine pe cale numerică toate soluțiile sistemului de ecuații neliniare:

$$\begin{cases} x^2 + y^2 = 4 \\ x - y = 1 \end{cases}$$

P7.3. Să se rezolve pe cale simbolică sistemul de ecuații neliniare:

$$\begin{cases} m + p - 2 \cdot n = 0 \\ n \cdot q - p^2 = 0 \\ m + q - 37 = 0 \\ n + p - 36 = 0 \end{cases}$$

P7.4. Să se rezolve sistemul subdeterminat:

$$\begin{cases} x^2 + y \cdot a - z = 0 \\ \frac{x}{z} = a \end{cases}$$

în necunoscutele *x*, *y*, *z*.

P7.5. Să se determine toate numerele *a,b,c* care îndeplinesc simultan următoarele condiții:

a) *a*, *b*, *c* sunt în progresie geometrică;

b) *a*, *b*+4, *c* sunt în progresie aritmetică;

c) *a*, *b*+4, *c*+32 sunt în progresie geometrică.

P7.6. Să se determine punctul de minim situat în vecinătatea punctului *x0*=0.5 şi minimul local al funcției: $f(x)=\ln\left(1 - x + \frac{x^2}{3}\right)$, $x \in \boldsymbol{R}$.

P7.7. Fie funcția $f:\boldsymbol{R}\to\boldsymbol{R}$, $f(x)=sin(x)+\sqrt{|x|}$. Să se determine toate punctele de extrem local din intervalul (-6,6), precum şi valorile funcției în aceste puncte, precizând totodată şi tipul de extrem pentru fiecare punct în parte.

P7.8. Să se rezolve următoarea problemă de optimizare fără restricții, folosind funcția Matlab bazată pe metode cvasi-Newton:

$$(\hat{x},\hat{y}) = arg \min_{(x,y)} f(x,y), \qquad f(x,y) = x^2 - y^2, \ (x,y) \in \boldsymbol{R^2}$$

pornind de la aproximarea (1,0).

P7.9. Să se rezolve următoarea problemă de optimizare fără restricții, folosind funcția Matlab bazată pe metoda de căutare simplex:

$$(\hat{x},\hat{y},\hat{z}) = arg \min_{(x,y,z)} f(x,y,z), \qquad f(x,y,z) = \frac{x}{x^2+y^2+z^2+1}, \ (x,y,z) \in \boldsymbol{R^3}$$

pornind de la aproximarea (0.6,-0.2,-0.1).

7.4. Întrebări recapitulative

Î7.1. Definiți noțiunea de „sistem de ecuații neliniare”.

Î7.2. Câte soluții poate avea un sistem de ecuații neliniare?

Î7.3. Precizați paşii de rezolvare a unui sistem de ecuații neliniare în Matlab, pe cale numerică, atunci când se cunoaşte numărul de soluții şi unde sunt acestea localizate.

Î7.4. Precizați forma canonică a unui sistem de *n* ecuații neliniare cu *n* necunoscute.

Î7.5. Enumerați funcțiile Matlab destinate rezolvării sistemelor de ecuații neliniare pe care le cunoaşteți (denumire, din ce toolbox-uri fac parte).

Î7.6. Numerele reale *q*, *u* şi *w* sunt în progresie aritmetică. Precizați care este relația dintre cele 3 numere.

Î7.7. Numerele reale *h*, *k* şi *p* sunt în progresie geometrică. Precizați care este relația dintre cele 3 numere.

Î7.8. Cum se poate transforma o problemă de maximizare într-o problemă de minimizare şi cum se determină în acest caz maximul?

Î7.9. Precizați pentru fiecare din funcțiile Matlab *fminbnd*, *fminunc* şi *fminsearch*, tipul de extrem (maxim sau minim) pe care îl determină.

Î7.10. Ce metode stau la baza funcției Matlab *fminbnd*?

Î7.11. Ce metode stau la baza funcției Matlab *fminunc*?

Î7.12. Ce metode stau la baza funcției Matlab *fminsearch*?

Î7.13. Precizați care din funcțiile Matlab *fminbnd*, *fminunc* şi *fminsearch*, se poate folosi pentru minimizarea unei funcții reale de o variabilă reală.

Î7.14. Precizați care din funcțiile Matlab *fminbnd*, *fminunc* şi *fminsearch*, se poate folosi pentru minimizarea unei funcții reale de cinci variabile reale.

ANEXA M7. ELEMENTE DESPRE REZOLVAREA SISTEMELOR DE ECUAŢII NELINIARE. ELEMENTE DESPRE REZOLVAREA PROBLEMELOR DE OPTIMIZARE

M7.1. Sisteme de ecuaţii neliniare

Se numeşte ***sistem de n ecuații neliniare cu n necunoscute*** orice sistem care poate fi adus la forma:

$$\begin{cases} f_1(x_1, x_2, \ldots, x_n) = c_1 \\ f_2(x_1, x_2, \ldots, x_n) = c_2 \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ f_n(x_1, x_2, \ldots, x_n) = c_n \end{cases}$$

unde:

- $c_1, c_2, \ldots, c_n$, sunt constante reale;
- $f_1, f_2, \ldots, f_n$, sunt pe domeniul de interes funcții continue de variabilele reale x_1, x_2, …, x_n, şi nu conțin termeni constanți;
- cel puțin una din funcții, f_i, este neliniară în raport cu cel puțin o necunoscută.

Comentariu: Funcția f_i este liniară dacă $\forall$ x, y vectori n-dimensionali din domeniul de interes şi $\forall$ $a,b \in \boldsymbol{R}$, este îndeplinită relația:

$f_i(a \cdot x + b \cdot y) = a \cdot f_i(x) + b \cdot f_i(y)$.

Un sistem de ecuații neliniare poate avea un număr finit de soluții (compatibil determinat), o infinitate de soluții (compatibil nedeterminat) sau nici o soluție (incompatibil).

Rezolvarea sistemelor de ecuații neliniare cu ajutorul metodelor numerice presupune – la fel ca şi rezolvarea ecuațiilor transcendente - parcurgerea următoarelor două etape:

I. ***separarea (localizarea)*** soluțiilor, adică descompunerea domeniului de interes într-o partiție de subdomenii, astfel încât fiecare subdomeniu să conțină cel mult o soluție;

II. ***calculul soluțiilor cu o precizie apriori fixată***, de obicei pornind de la seturi de valori aproximative ale acestora.

Progresii aritmetice. Progresii geometrice

Progresii aritmetice

Se numeşte ***progresie aritmetică*** un şir de numere în care, fiecare termen, începând cu al doilea termen, se obține din cel precedent prin adăugarea la acesta a unui număr real constant, numit ***raţia progresiei aritmetice***.

Suma primilor n termeni ai unei progresii aritmetice a_1, a_2, ..., a_k, ... este $S_n = \frac{(a_1 + a_n)\cdot n}{2}$.

Condiția necesară şi suficientă pentru ca trei numere reale a, b şi c să formeze o progresie aritmetică (cu termenul din mijloc b) este ca ele să satisfacă relația $2\cdot b = a + c$ (adică, termenul din mijloc să fie media aritmetică a celorlalți doi termeni).

Progresii geometrice

Se numeşte ***progresie geometrică*** un şir de numere în care, fiecare termen, începând cu al doilea termen, se obține din cel precedent prin înmulțirea acestuia cu un număr real nenul constant, numit ***raţia progresiei geometrice***.

Suma primilor n termeni ai unei progresii geometrice b_1, b_2, ..., b_k, ... de raţie q neunitară este $S_n = b_1 \cdot \frac{1-q^n}{1-q}$.

Condiția necesară şi suficientă pentru ca trei numere reale a, b şi c să formeze o progresie geometrică (cu termenul din mijloc b) este ca ele să satisfacă relația $b^2 = a \cdot c$ (adică, termenul din mijloc în valoare absolută să fie media geometrică a celorlalți doi termeni).

M7.2. Probleme de optimizare

a. Problema de optimizare

Fiind dată o funcție f: $D \rightarrow \boldsymbol{R}$, unde D este o mulțime oarecare, de obicei, $D \subseteq \boldsymbol{R^n}$, $\boldsymbol{R^n}$ fiind spațiul euclidian, se cere determinarea unui element $x_0 \in D$, astfel încât $f(x_0) \leq f(x)$, $\forall x \in D$, în cazul criteriului de minimizare, respectiv, $f(x_0) \geq f(x)$, $\forall x \in D$, în cazul criteriului de maximizare.

Un element oarecare x din D se numeşte ***soluție admisibilă***. f se numeşte ***funcție-obiectiv***. O soluție admisibilă pentru care se obține optimul funcției obiectiv (minimul sau maximul, în funcție de criteriul dorit) se numeşte ***soluție optimală*** (se mai folosesc şi denumirile ***punct de minim global***, în cazul unei probleme de minimizare, respectiv, ***punct de maxim global***, în cazul unei probleme de maximizare). D se numeşte ***spațiul soluțiilor admisibile***.

Dacă $D=\boldsymbol{R^n}$, problema de optimizare este ***o problemă fără restricții***.

Un element $x_l \in D$, cu proprietatea că există o vecinătate $V \subset D$ a sa astfel încât $f(x_l) \leq f(x)$, $\forall x \in V$, în cazul criteriului de minimizare, respectiv, $f(x_l) \geq f(x)$, $\forall x \in V$, în cazul criteriului de maximizare, se numeşte ***punct de optim local*** (***punct de minim local***, respectiv ***punct de maxim local***), iar valoarea funcției f în x_l poartă denumirea de ***optim local***.

Rezolvarea unei probleme de optimizare presupune utilizarea unei ***metode de optimizare*** pentru determinarea soluției optimale.

Se poate observa că o problemă de maximizare poate fi transformată într-o problemă de minimizare, şi invers, prin înlocuirea funcției obiectiv *f* cu opusa acesteia, *-f*.

În continuare se consideră doar probleme de minimizare fără restricții.

b. Metode de optimizare în cazul problemelor de optimizare fără restricţii

Metodele clasice de optimizare în cazul problemelor de optimizare fără restricții pot fi grupate în următoarele categorii:

- ***metode neiterative*** (într-un pas), care se bazează pe anumite proprietăți ale funcției obiectiv, cum ar fi derivabilitatea, convexitatea etc.;
- ***metode iterative*** (în mai mulți paşi), care pornesc cu o soluție inițială şi determină la fiecare pas o nouă soluție; cele mai cunoscute metode din această categorie sunt metodele de descreştere (de relaxare, de coborâre), caracterizate prin faptul că valoarea funcției obiectiv scade (descreşte, coboară) la fiecare pas, care pot fi încadrate în următoarele subcategorii:
  - metode de ordinul zero – necesită doar calculul valorilor funcției obiectiv (de exemplu, algoritmii genetici, metoda secțiunii de aur);
  - metode de ordinul I – necesită atât calculul valorilor funcției obiectiv, cât şi al valorilor gradientului acesteia (de exemplu, metoda gradientului, metode de gradient conjugat);
  - metode de ordinul II – necesită atât calculul valorilor funcției obiectiv, cât şi a gradientului şi a hessianului acesteia (de exemplu, metoda clasică a lui Newton).