

1. Introducere

Acest program a fost dezvoltat în cadrul Proiectului TEMPUS sJEP 112-19 la INPG-LEPMI Grenoble – Franța, de către Cezar Catrinescu - asistent universitar la Universitatea Tehnică "Gh. Asachi" Iași.

Aplicatia se adreseaza studentilor de la colegiile si facultatile de inginerie chimica in al caror programa de invatamant figureaza o disciplina de optimizari in industria chimica.

~n rezolvarea problemelor de optimizare a diverselor procese din chimie, se realizeaza, in primul rand, o modelare matematica a procesului care are ca rezultat obtinerea unei functii scop (care va descrie acel proces). Functia scop este dependenta de un numar de parametri care pot fi modificati si a caror variatie va influenta performantele acelui proces. ~ntr-o a doua etapa se realizeaza o optimizare a acestei functii scop, cand se gasesc valorile optime ale parametrilor ce influenteaza procesul. Rezolvarea acestei ultime etape constituie obiectivul aplicatiei dezvoltate sub MATLAB.

Astfel, obiectivul a fost de a dezvolta aplicatii sub MATLAB privind metode de optimizare in spatiul R^n pentru identificarea parametrilor a modelelor.

Cerinte :

Hard : procesor 386 si coprocesor 387 ; 15 Mo spatiu liber pe hard, 4 Mo RAM.

Soft :

WINDOWS 3.0 sau o versiune superioara

MATAB versiunea 4.0 sau o versiune superioara.

2. Baze teoretice

Optimizarea unei functii inseamna a gasi intr-o problema sau intr-o situatie de decizie solutia care, dintr-un anumit punct de vedere prestabilit, este cea mai buna dintre toate solutiile posibile. Optimizarea conduce la determinarea unei stari speciale a sistemului, care este cea mai favorabila dintr-un anumit punct de vedere. Astfel optimizarea unei probleme fara constrangeri se reduce la a gasi maximul sau minimul unei functii de n variabile.

Dintre aplicatiile tipice ale optimizarii citam:

- proiectarea unei instalatii, a unei linii tehnologice etc, astfel incat beneficiul sa fie maxim, investitia sa fe minima, randamentul in produs final sa fie maxim;
- stabilirea valorilor parametrilor tehnologici a unei instalatii existente astfel incat beneficiul sa fie maxim, sau randamentul sa fie maxim etc
- planificarea optima a experimentelor necesare pentru modelarea empirica a unui proces sau gasirea experimentală a optimului etc

Clasificarea metodelor de cautare directa a optimului unei functii scop de n variabile de decizie se poate face astfel:

1. Metode empirice, care nu implica decat cunoasterea functiei scop
2. Metode de gradient care implica si cunoasterea tuturor derivatelor partiale de ordin I
3. Metode de tip Newton care implica, pe langa cunoasterea functiei scop, si a derivatelor partiale de ordin I si II.
4. Metode geometrice (metoda Nelder si Mead)

De comun acord am ales urmatoarele metode de optimizare:

1. Metoda gradientului conjugat – tip Newton
2. Metoda Hooke si Jeeves – metoda euristica
3. Metoda Nedler si Mead – metoda geometrica
4. Medoda Newton generalizata – tip Newton
5. Metoda gradientului – tip gradient

Metoda Hooke-Jeeves

Metoda foloseste ca principiu de cautare *testarea* dupa directii paralele cu axele de coordonate si *deplasarea* (intr-un sens ce va fi prezentat mai jos).

Testarea

Se pleaca de la un punct initial arbitrar si se testeaza numai valoarea functiei scop pe directii paralele cu axele de coordonate iar deplasarile se fac in ambele sensuri cu un pas fix care poate fi acelasi sau diferit pentru fiecare axa de coordonate.

Primul punct in care se realizeaza un succes este luat ca noua baza temporara. Odata incheiata aceasta explorare pe toate directiile paralele cu axele de coordonate se face o miscare cu un pas diferit, dupa cum se va descrie in continuare.

Deplasarea

Avand doua baze succesive x_{k-1} si x_k deplasarea se face conform formulei:

$$x = x_k + (x_k - x_{k-1}) = 2x_k - x_{k-1}$$

Plecand de la acest punct se reincepte o testare locala. Daca la terminarea acestei faze am mai diminuat y atunci punctul obtinut este noua baza x_{k+1} . In caz contrar ne reintoarcem la x_k si facem o noua testare cu un pas redus la jumatate.

Metodele de tip gradient

Sunt metode de cautare numerica a optimului care folosesc pentru determinarea acestuia atat valorile functiei scop cat si ale derivatei ei partiale de ordinul I in diverse puncte. Denumirea provine din faptul ca ansamblul derivatelor partiale de ordinul I ale unei functii de mai multe variabile evaluate intr-un punct se mai numeste gradientul functiei in punctul respectiv.

Gradientul are urmatoarele proprietati esentiale :

1. este un vector si deci defineste o directie ;

2. gradientul intr-un punct X este normal la conturul lui $Y(X)$
3. directia gradientului corespunde directiei de cea mai rapida crestere a lui $Y(X)$ iar $-\text{grad } Y(X)$ arata directia celei mai rapide descresteri.

~n baza acestei proprietati directia gradientului apare ca cel mai eficient drum de urmat pentru gasirea extremului unei functii. Directia gradientului este inasa o proprietate locala, ea variind in general de la punct la punct, asa incat inaintarea spre optim pe directia gradientului ar trebui sa reprezinte riguros o succesiune de deplasari infinitesimale pe o curba. Metodele de gradient se limiteaza la urmarirea mai mult sau mai putin stransa a acestei curbe prin deplasari rectilinii spre a atinge optimul printr-un numar cat mai redus de pasi.

Foarte importanta este alegerea pasului cu care se face deplasarea de la un punct la cel urmator. Se poate proceda in doua moduri :

1. se alege pentru pas o valoarea fixa sau variabila, eventual de la un punct la altul ;
2. Se calculeaza valoarea pasului, adica acea valoare a acestuia care da cea mai mica valoare a functiei pe directia gradientului in punctul respectiv. Aceasta se realizeaza printr-o metoda de optimizare unidimensionala in care variabila optimizata este chiar valoarea pasului.

Odata calculate valorile gradientului in punctul respectiv si a pasului se trece la punctul urmator din traseul spre minim conform formulei :

$$x_i^{k+1} = x_i^k + s^k * \text{grad}_i^k$$

x_i^k, x_i^{k+1} unde sunt vectori in spatiul R^n ;

s^k – reprezinta pasul ;

grad_i^k reprezinta gradientul in punctul x_i^k

$$\text{grad } y(\bar{x}^{(k)}) = \begin{pmatrix} \frac{\partial y(\bar{x})}{\partial x_1} \\ \frac{\partial y(\bar{x})}{\partial x_2} \\ \vdots \\ \frac{\partial y(\bar{x})}{\partial x_n} \end{pmatrix}_{x=\bar{x}^{(k)}}$$

Metodele de gradient duc in general la oscilatii ale traiectoriei in cazul in care functia scop prezinta « vai » sau « dealuri » , deoarece directiile succesive de cea mai abrupta coborare pot sa nu treaca prin punctul optim.

~n schimb ele sunt eficiente daca functia scop are curbe de contur circulare . Evident, daca suprafata de raspuns are curbe de cuntur perfect circulare, optimul se atinge printr-o singura iteratie.

Metode de tip Newton

Sunt metodele care implica pentru cautarea optimului atat cunoasterea valorilor functiei scopcat si a tuturor derivatelor ei partiale de ordin I si II in diverse puncte.

Metoda pleaca de la faptul ca valoarea fntiei intr-un punct \bar{x}^{k+1} poate fi determinata cu o aproximatie satisfacatoare prin dezvoltare in serie Taylor a functiei scop $y(\bar{x})$ in punctul $\bar{x}^{(k)}$. Se obtine urmatoarea aproximare patratice:

in care :

$$s^{(k)} \bar{V}^{(k)} = \bar{x}^{(k+1)} - \bar{x}^{(k)}$$

$$\text{iar } H(\bar{x}^{(k)}) = \begin{vmatrix} \frac{\partial^2 y}{\partial x_1^2} & \frac{\partial^2 y}{\partial x_1 \partial x_2} & \dots & \frac{\partial^2 y}{\partial x_1 \partial x_n} \\ \frac{\partial^2 y}{\partial x_2 \partial x_1} & \frac{\partial^2 y}{\partial x_2^2} & \dots & \frac{\partial^2 y}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 y}{\partial x_n \partial x_1} & \frac{\partial^2 y}{\partial x_n \partial x_2} & \dots & \frac{\partial^2 y}{\partial x_n^2} \end{vmatrix}$$

este simbolul matricei Hessiene a functiei scop in punctul $\bar{x}^{(k)}$.

Vom avea deci :

$$\bar{x}^{(k+1)} = \bar{x}^{(k)} - [H(\bar{x}^{(k)})]^{-1} * \text{grad } y(\bar{x}^{(k)})$$

$$\bar{x}^{(k+1)} = \bar{x}^{(k)} - \frac{\left(\frac{\partial y}{\partial x} \right)_{\bar{x}=\bar{x}^{(k)}}}{\left(\frac{\partial^2 y}{\partial x^2} \right)_{\bar{x}=\bar{x}^{(k)}}}$$

Metode geometrice (Nelder si Mead)

Aceste metode tind sa elimine dezavantajele metodelor univariante. ~n esenta, apar doua tipuri de modificari fata de metodele univariante:

- in loc de cautare pe directii fixe se cauta pe directiile care dau cele mai bune rezultate;
- in loc sa se mearga pe o directie pana la suboptim se merge cu un pas fix.

Metoda Nelder si Mead ia ca baza de pornire o figura geometrica regulata, numita *simplex*, definita in spatiul n- dimensional ca o figura geometrica cu (n+1) varfuri, distanta dintre doua varfuri oarecare fiind aceeasi.

~n spatiul tridimensional Simplexul este un tetraedru, iar in cel bidimensional un triunghi echilateral. ~n acest ultim caz, se alege un triunghi echilateral cu latura a, de obicei cu unul dintre varfuri in origine, si se calculeaza valorile functiei scop in varfuri. Varful care da rezultatul cel mai

nefavorabil este eliminat si se cauta un nou varf prin trei operatii principale : *reflexie*, *contractie* si *expansiune*. Se evalueaza valoarea functiei scop in acest nou punct si se elimina valoarea cea mai rea s.a.m.d.

Fie P_0, P_1, \dots, P_n cele $n+1$ puncte in spatiul n -dimensional care definesc Simplexul. Notam y_i ca valoarea functiei in punctul P_i ;

$$y_{\text{Max}} = \max(y_i) ;$$

$$y_{\text{min}} = \min(y_i)$$

P ca centroidul punctelor cu $i \neq h$;

$\langle P_i P_j \rangle$ distanta dintre P_i si P_j

Reflexia lui P_{max} , notata P^* este definita de relatia :

$$P^* = (1-\alpha)P - \alpha P_{\text{max}}$$

unde α este o constanta pozitiva, numita *coeficient de reflexie*.

Daca y^* este ca valoare intre y_i si y_{max} P_{min} este inlocuit de P^* , obtinandu-se astfel un nou Simplex.

Daca $y^* < y_{\text{min}}$, i.e prin reflexie s-a obtinut un punct mai mic decat minimul, se va face o *expansiune* spre P^{**} , conform relatiei:

$$P^{**} = \gamma P^* + (1-\gamma)P.$$

γ este numit coeficient de expansiune. Daca $y^{**} < y_{\text{min}}$ se inlocuieste P_{max} cu P^{**} si se continua procesul; daca insa $y^{**} > y_{\text{min}}$ avem o expansiune ratata si inlocuim P_{min} cu P^* , apoi se reincepe procesul. Daca prin reflexia lui P in P^* gasim ca $y^* > y_i$ pentru toti $i \in M$, vom alege ca nou punct ori P_{max} ori P^* , si anume pe cel care da valoarea lui y cea mai mica, apoi vom forma prin *contractie*:

$$P^{**} = \beta P_{\text{max}} + (1-\beta)P.$$

Coeficientul de contractie β este pozitiv subunitar.

Vom accepta P^{**} in locul lui P_{max} si vom continua: exceptie face situatia in care $y^{**} > \min(y_{\text{max}}, y^*)$, i.e punctul rezultat prin contractie este mai rau decat cel mai bun dintre P_{max} si P^* . ~n cazul unei astfel de situatii vom inlocui toti P_i cu $(P_i + P_{\text{min}})/2$ si vom porni din nou procesul.

3. Mod de lucru

1. In fereastra de comenzi MATLAB dati comanda " optim" pentru a lansa aplicatia. Aceasta operatie va deschide o fereastra grafica cu cinci butoane - fiecare corespunzatoare unei metode de optimizare (vezi figura 1)

2. Alegeti metoda de optimizarea prin apasarea butonului corespunzator> Aceasta actiune va deschide o noaua fereastra in care puteti introduce valorile pentru :

- punctul initial din care incepe cautarea minimului
- eroarea maxima a metodei
- numele fisierului functie de optimizat
- limitele intervalelor de reprezentare grafica

Utilizatorul poate introduce o functie care se afla in componenta programului sau poate crea o functie - obiecti proprie. In acest caz utilizatorul va trebui sa scrie functia intr-un fisier salvat ca " nume_functie.m ". Sa presupunem ca se doreste optimizarea functiei :

$$f(x) = 100 * (x_2 - x_1^2)^2 + (1 - x_1)^2$$

Fisierul matlab va avea forma urmatoare;

```
function svaluet=f(x)
value = 100*(x(2)-x(1)a2)a2 +(1-x(1))a2
```

Acest fisier trebuie salvat ca " nume_functie.m " in acelasi director ca si programul.

3. Apasati butonul "Start" pentru a lansa aplicatia. Valorile calculate vor fi afisate in aceeasi fereastra grafica.

In plus metoda Nedler si Mead afiseaza tipul de operatie efectuat .

4. Exemple

Exemplul 1

Sa se determine minimul functiei

$$f(x) = (x_1 + x_2 + 4)^2 + (x_1 - 2x_2 + 1)^2$$

1. Se lanseaza programul Matlab
2. Se creaza un fisier test1.m cu urmatorul continut:

```
function value=test1(xx)
value=(xx(1)+xx(2)+4)^2+(xx(1)-2*xx(2)+1)^2;
```

3. ~n fereastra de comenzi a programului se da comanda optim care va lansa programul de optimizare.
4. Se alege metoda de optimizare dorita. Vom alege pe rand fiecare metoda.

4.1 Metoda gradientilor conjugati

Se introduc :

- valorile punctului de plecare : x1 si x2 ;
- eroarea maxima admisa : err
- Numele fisierului in care s-a scris functia : Function (in cazul nostru test1)
- Valorile minime (x1min, respectiv x2m) si maxime(x1M respecti x2M) pentru reprezentarile grafice

Se apasa butonul start si se va obtine urmatoarea imagine. (figura1).

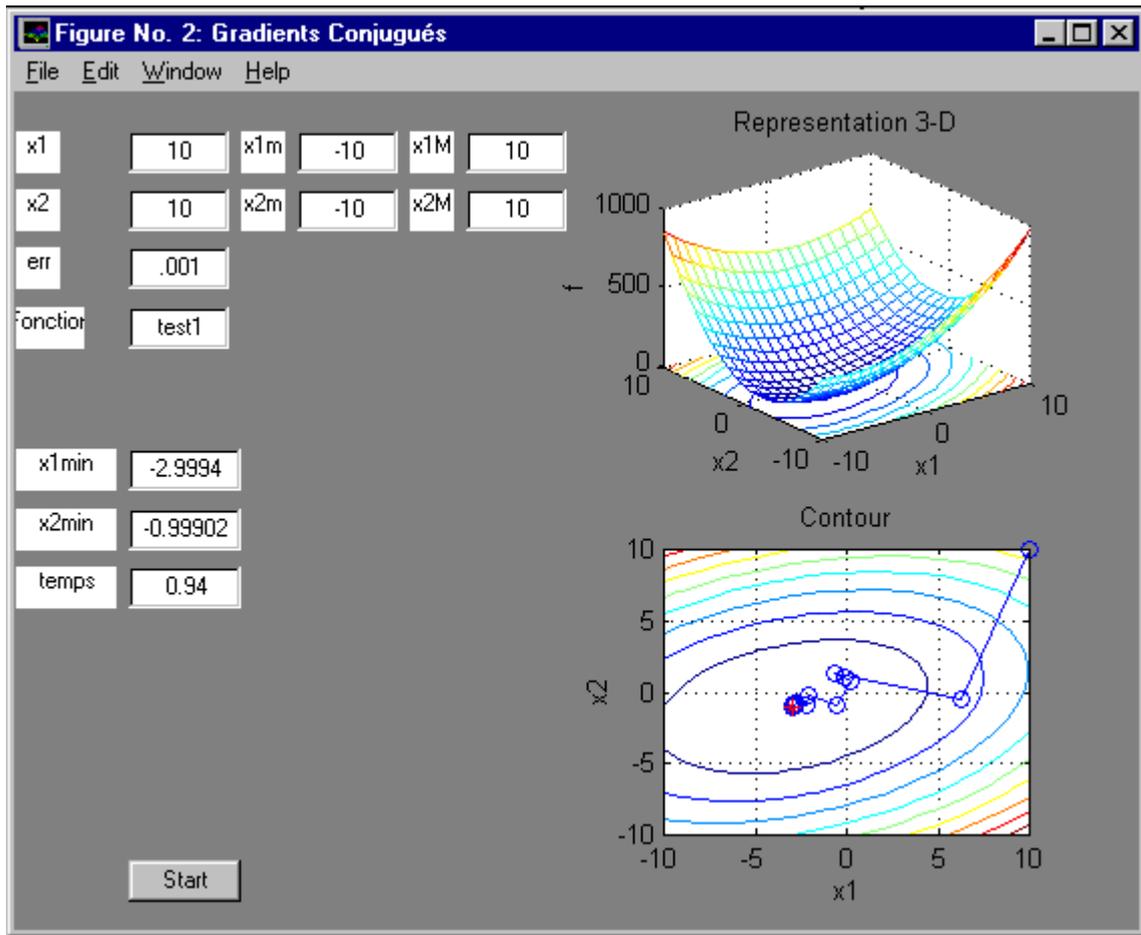


Figura 1. Metoda gradientului conjugat

- reprezentarea 3D a functiei in intervalul ales :

- reprezentarea contur cu traseul spre punctul minim :

Vor fi afisate si valorile variabilelor pentru care functia prezinta minim precum si timpul de calcul.

Metoda lui Newton generalizata

Procedand asemanator se vor obtine urmatoarele rezultate, conform figurii 2.

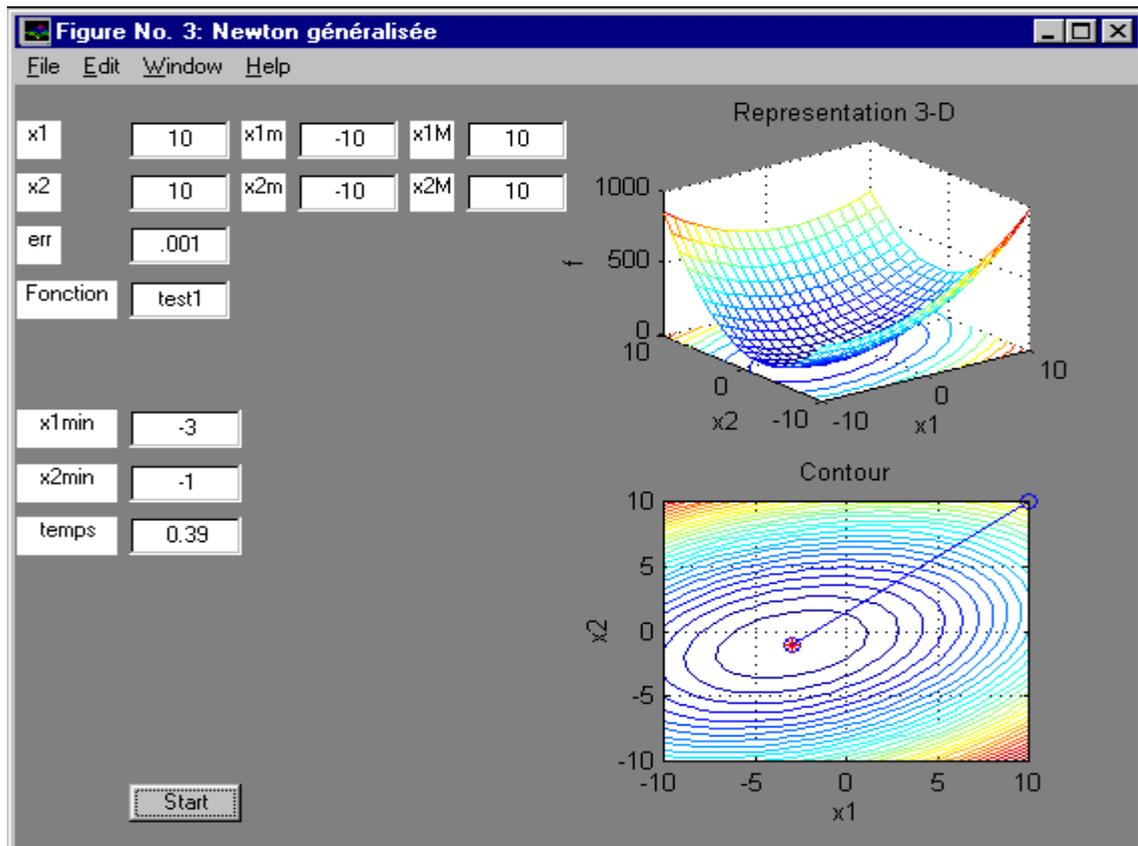


Figura 2. Metoda Newton generalizata

Metoda Nelder si

Mead

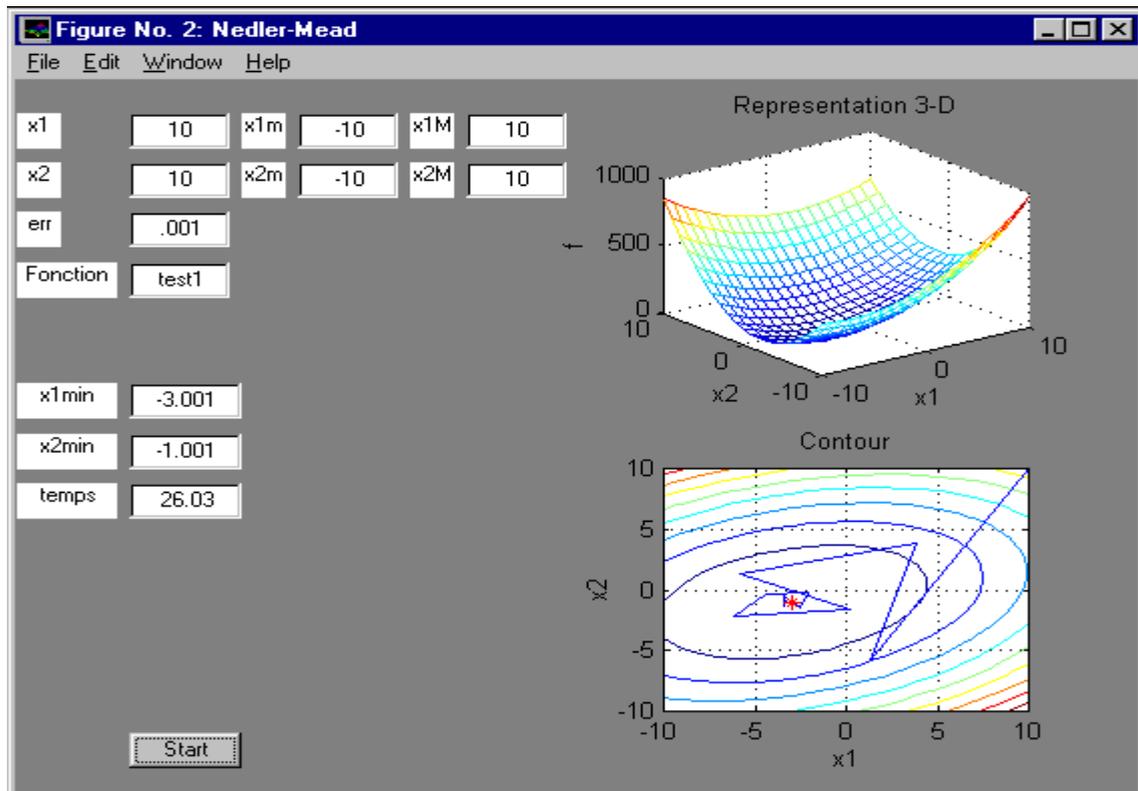


Figura 3. Metoda Nelder si Mead

Metoda Hooke si Jeeves

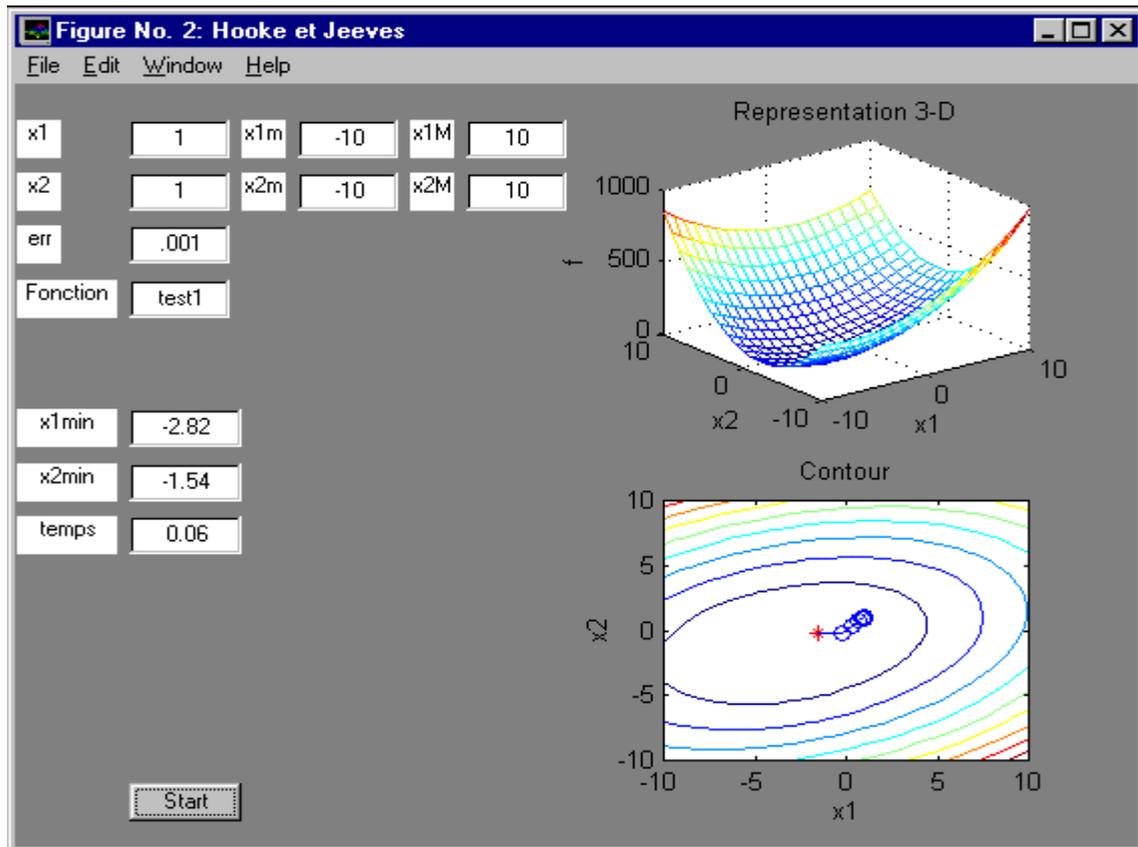


Figura 4. Metoda Hooke Jeeves

Metoda gradientului

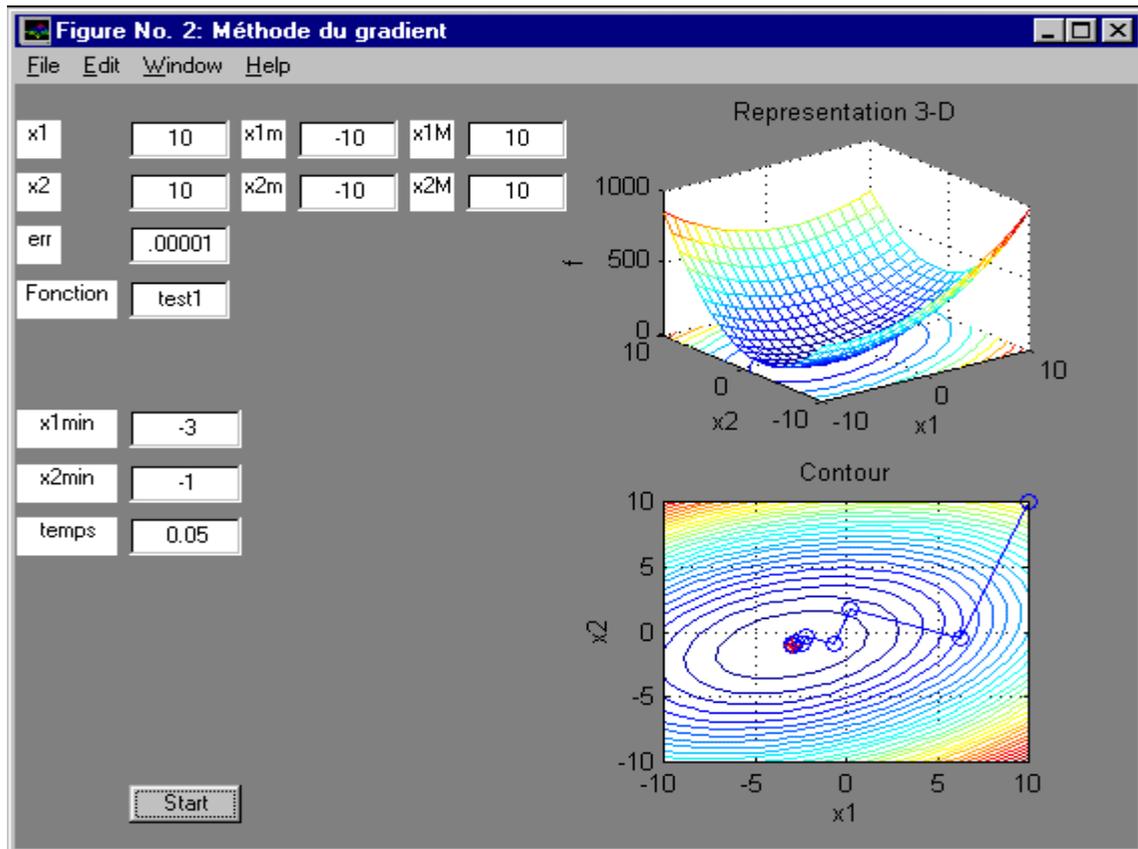


Figura 5. Metoda gradientului

~n functie de abilitatea de a gasi minimul si de timpul de calcul se poate evalua eficienta fiecarei metode.

5. Intretinerea surselor

Programul utilizeaza urmatoarele fisiere:

gc.m, hj.m, nm.m, nr.m, sd.m;

gc1.m, hj1.m, nm1.m, nr1.m, sd1.m;

grgc.m, grhj.m, grnm.m, grnr.m, grsd.m;

gradient.m, hessian.m, ini_brac.m, gss.m, opt_1_s.m, ch.m, lsfit.m

test1.m, test2.m, test3.m

Fiecare dintre metode se bazeaza pe o structura compusa din cel putin trei fisiere;

1. fisirul care contine algoritmul matematic

2. fisierul care creaza interfata grafica
3. fisierul care citeste interactiv datele introduse de utilizator, lanseaza programul principal si afiseaza rezultatele

Atunci cand gradientul si hessianul sunt necesari acestia sunt calculati numeric de fisierele functie : " gradient.m" si " hessian.m ". Aceste functii pot fi utilizate si separat (vezi help-ul pentru fiecare dintre ele). Dupa gasirea directiei de cautare - specifice fiecarei metode - programul gaseste pasul optim de deplasare utilizand metoda sectiunii de aur. Functia " opt_1_s.m " gaseste pasul optim prin crearea unui interval initial - in care se gaseste valoarea optima a pasului de cautare ("ini-brac.m ") - si apoi prin aplicarea metodei ' sectiunii de aur ' se calculeaza pasul optim de deplasare spre valoarea minima a functiei studiate.

In plus metoda lui Nedler si Mead afiseaza si tipul de operatiune efectat (vezi modul de utilizare).

Fisierul " lsfit.m " gaseste coeficientii unei functii polinomiale de doua variabile prin metoda celor mai mici patrate. Aceasta aplicatie nu are o interfata grafica ci afiseaza rezultatele - respectiv valorile coeficientilor - intr-o forma matriceala.

Fiecare dintre metode se afla intr-un fisier separat si poate fi utilizata intr-o maniera mai generala pentru functii cu mai multe variabile. In acest caz programul nu poate avea o interfata grafica ci afiseaza doar valorile parametrilor pentru care functia studiata are valoarea minima, timpul de calcul precum si valoarea functiei in punctele calculate.

Procedura generala, comuna acestor metode este urmatoarea:

1. Se alege un punct de plecare (start), respectiv un set de valori ale variabilelor de decizie
2. Se calculeaza valoarea functiei scop in punctul de plecare
3. Se alege prin una dintre metodele de cautare directa, respectiv dupa o schema sau un ciclu de iteratii, un al doilea punct de cautare.
4. Se calculeaza valoarea functiei scop in acest nou punct
5. Se compara cele doua valori ale functiei scop. Daca ultimul punct este mai bun decat primul se ia acesta ca nou punct de plecare si se continua ca mai sus.
6. Calculul se opreste de regula, atunci cand diferenta dintre valorile functiei scop in doua puncte succesive de cautare este mai mica decat o valoare prestabilita.

Conform algoritmului prezentat se vede ca metodele difera intre ele numai prin pasul 3. Mai exact ele se deosebesc prin alegerea directiei de cautare, adica a drumului pe care de merge de la un punct vechi la unul nou, si prin distanta dintre punctul vechi si cel nou pe directia de cautare.

Listing comentat

**** Fisiere generale****

****optim.m****

```
h=figure('Name','Méthodes de optimisation','color',s0.5 .5 .5t);
hpb1=uicontrol('Style','Pushbutton','String','Gradients conjugués','Units',...
'normalized','Position',s.1 .7 .7 .09t,'Callback','grgc');
hpb2=uicontrol('Style','Pushbutton','String','Newton généralisée','Units',...
'normalized','Position',s.1 .6 .7 .09t,'Callback','grnr');
hpb3=uicontrol('Style','Pushbutton','String','Nedler et Mead','Units',...
'normalized','Position',s.1 .5 .7 .09t,'Callback','grnm');
hpb4=uicontrol('Style','Pushbutton','String','Hooke et Jeeves','Units',...
'normalized','Position',s.1 .4 .7 .09t,'Callback','grhj');
hpb5=uicontrol('Style','Pushbutton','String','Méthode du gradient','Units',...
'normalized','Position',s.1 .3 .7 .09t,'Callback','grsd');
htxt=uicontrol('Style','Text','FontSize',25,'String','TEMPUS sJEP 11219','Units',...
'normalized','Position',s.1 .83 .7 .09t);
%text(.1,.1,'FontSize',30,'String','TEMPUS sJEP 11219')
```

****gradient****

```
functionsgradt=gradient(fun,x0,err)
N=length(x0);
var=zeros(size(x0));
grad=x0';
for i=1:N,
var(i)=err;
```

```

avant=feval(fun,x0-var);
apres=feval(fun,x0+var);
grad(i)=(apres-avant)/(2*err);
var(i)=0;
end;
**hessian**
functionshest=hessian(fun,x0,err)
N=length(x0);
var=zeros(size(x0));
hes=zeros(length(x0),length(x0));
for i=1:N,
var(i)=err;
avant=gradient(fun,x0-var,err);
apres=gradient(fun,x0+var,err);
hes(:,i)=(apres-avant)/(2*err);
var(i)=0;
%hess=shes(1),hes(2)t;
end;
**gss.m**
functionalphat=gss(fun,v_x,v_d,init_a,init_b,err)
next_length=0.618034*(init_b-init_a);
xx=init_b-next_length; f_xx=feval(fun,v_x+xx*v_d);
yy=init_a+next_length; f_yy=feval(fun,v_x+yy*v_d);
err=0.618034*err;
while next_length>err,
next_length=next_length*0.618034;
if f_xx<f_yy
init_b=yy;
yy=xx; f_yy=f_xx;
xx=init_b-next_length; f_xx=feval(fun,v_x+xx*v_d);
else

```

```

init_a=xx;
xx=yy; f_xx=f_yy;
yy=init_a+next_length; f_yy=feval(fun,v_x+yy*v_d);
end;
end;
if f_xx>f_yy
f_xx=f_yy; xx=yy;
end;
alpha=(xx+yy)/2;
f_alpha=feval(fun,v_x+alpha*v_d);
if f_alpha>f_xx
alpha=xx;
end;
**ini_brac.m**
functionspr,sec,trt=ini_brac(fun,v_x,v_d,err)
alpha_ka=0;
alpha_k=0;
f_alpha_ka=feval(fun,v_x);
f_alpha_k=f_alpha_ka + 1;
step_length=err;
while f_alpha_ka < f_alpha_k,
    alpha_k=alpha_ka;
f_alpha_k=f_alpha_ka;
step_length=2*step_length;
alpha_ka=alpha_k+step_length;
f_alpha_ka=feval(fun,v_x+alpha_ka*v_d);
end;
alpha_try=(alpha_k+alpha_ka)/2;
f_alpha_try=feval(fun,v_x+alpha_try*v_d);
if f_alpha_try>f_alpha_k,
pr=alpha_k - step_length/2;

```

```

sec=alpha_k;
tr=alpha_try;
else
pr=alpha_k;
sec=alpha_try;
tr=alpha_ka;
end;
**opt_1_s.m**
function salphat=opt_1_s(fun,v_x,v_d,err)
sinit_a,init_b,init_ct=ini_brac(fun,v_x,v_d,err);
err=err/5;
alpha=gss(fun,v_x,v_d,init_a,init_c,err);

**Metoda gradientilor conjugati**
**grgc*
% Acest fisier construiește interfata grafica a aplicatiei
% Apasati butonul start pentru a lansa programul
% Fichier grgc.m
%
h=figure('Name','Gradients Conjugués','color',s0.5 0.5 .5t);
hed1=uicontrol('Style','Edit','String','10','Units',...
'normalized','Position',s.1 .9 .09 .05t,'BackgroundColor',s1,1,1t);
hed2=uicontrol('Style','Edit','String','10','Units',...
'normalized','Position',s.1 .83 .09 .05t,'BackgroundColor',s1,1,1t);
hed3=uicontrol('Style','Edit','String','.001','Units',...
'normalized','Position',s.1 .76 .09 .05t,'BackgroundColor',s1,1,1t);
hed4=uicontrol('Style','Edit','String','test1','Units',...
'normalized','Position',s.1 .69 .09 .05t,'BackgroundColor',s1,1,1t);
hed8=uicontrol('Style','Edit','String','-10','Units',...
'normalized','Position',s.25 .9 .09 .05t,'BackgroundColor',s1,1,1t);
hed9=uicontrol('Style','Edit','String','10','Units',...

```

```

'normalized','Position',s.4 .9 .09 .05t,'BackgroundColor',s1,1,1t);
hed10=uicontrol('Style','Edit','String','-10','Units',...
'normalized','Position',s.25 .83 .09 .05t,'BackgroundColor',s1,1,1t);
hed11=uicontrol('Style','Edit','String','10','Units',...
'normalized','Position',s.4 .83 .09 .05t,'BackgroundColor',s1,1,1t);
hpb1=uicontrol('Style','Pushbutton','String','Start','Units',...
'normalized','Position',s.1 .03 .1 .05t,'Callback','gc1');
htxt1=uicontrol('style','text',...
'units','normalized',...
        'position',s.001 .9 .04 .05t,...
        'string','x1','BackgroundColor',s1,1,1t);
htxt2=uicontrol('style','text',...
'units','normalized',...
        'position',s.001 .83 .04 .05t,...
        'string','x2','BackgroundColor',s1,1,1t);
htxt3=uicontrol('style','text',...
'units','normalized',...
        'position',s.001 .76 .04 .05t,...
        'string','err','BackgroundColor',s1,1,1t);
htxt4=uicontrol('style','text',...
'units','normalized',...
        'position',s.001 .69 .06 .05t,...
        'string','Fonction','BackgroundColor',s1,1,1t);
htxt10=uicontrol('style','text',...
'units','normalized',...
        'position',s.2 .9 .04 .05t,...
        'string','x1m','BackgroundColor',s1,1,1t);
htxt11=uicontrol('style','text',...
'units','normalized',...
        'position',s.35 .9 .04 .05t,...
        'string','x1M','BackgroundColor',s1,1,1t);

```

```

htxt12=uicontrol('style','text',...
'units','normalized',...
    'position',s.2 .83 .04 .05t,...
    'string','x2m','BackgroundColor',s1,1,1t);
htxt13=uicontrol('style','text',...
'units','normalized',...
    'position',s.35 .83 .04 .05t,...
    'string','x2M','BackgroundColor',s1,1,1t);

**gc1**

% Fisier care citeste interactiv datele introduse
% lanseaza programul principal si afiseaza rezultatele
% Ce fisier appelle la fonction gc.m qui contient l'algorithmme mathématique
cla reset

val1=get(hed1,'String');
val2=get(hed2,'String');
e=get(hed8,'String');
f=get(hed9,'String');
g=get(hed10,'String');
h=get(hed11,'String');
er=get(hed3,'String');
ev=sstr2num(e):1:str2num(f)t;
fv=sstr2num(g):1:str2num(h)t;
err=str2num(er);
v_x=sstr2num(val1) str2num(val2)t;
fun=get(hed4,'String')
ssolution,n_x,tempst=gc(fun,v_x,err,ev,fv);
hed5=uicontrol('Style','Edit','String',num2str(n_x(1)), 'Units',...
'normalized','Position',s.1 .52 .1 .05t,'BackgroundColor',s1,1,1t);
hed6=uicontrol('Style','Edit','String',num2str(n_x(2)), 'Units',...
'normalized','Position',s.1 .45 .1 .05t,'BackgroundColor',s1,1,1t);

```

```
hed7=icontrol('Style','Edit','String',num2str(temps),'Units',...
'normalized','Position',s.1 .38 .1 .05t,'BackgroundColor',s1,1,1t);
```

```
htxt5=icontrol('style','text',...
'units','normalized',...
'position',s.001 .52 .09 .05t,...
'string','x1min','BackgroundColor',s1,1,1t);
```

```
htxt6=icontrol('style','text',...
'units','normalized',...
'position',s.001 .45 .09 .05t,...
'string','x2min','BackgroundColor',s1,1,1t);
```

```
htxt7=icontrol('style','text',...
'units','normalized',...
'position',s.001 .38 .09 .05t,...
'string','temps','BackgroundColor',s1,1,1t);
```

```
**gc**
```

```
function ssolution, n_x,tempst = fr(fun, v_x, err,ev,fv)
% ssoluti, new_xt = fr(fun_name, old_x, error)
% Cette fonction trouve la valeur optimale en utilisant
% la méthode de gradient conjugue..
% ' solution ' est la solution optimale trouvée par la méthode
% ' de n_x ' est le x-vecteur optimal
% 'temps' este le temps de calcul (CPU time)
% ' fun ' est la fonction objective
% ' v_x ' est le point de départ
% ' err ' est l'exactitude exigée
% si le nombre d'itérations est plus de 500 la fonction est terminé
%
% du fichier fr.m

% Initialisation
```

```

x_init=v_x;
n_x = v_x;
g_n_x = gradient(fun, n_x, err);
v_d = zeros(size(v_x));
k = 0;
iter=0;
tic
while ((norm(n_x - v_x) > err) | (k < 1)) & (k < 500),

    k = k+1;
iter=iter+1;
    g_v_x = g_n_x;
    g_n_x = gradient(fun, n_x, err);

    beta = (g_n_x*g_n_x')/(g_v_x*g_v_x');

        v_x = n_x;
v_d = -g_n_x'+ v_d*beta;

alpha = opt_1_s(fun, v_x, v_d, err);

n_x = v_x+alpha*v_d;
T(k,:)=n_x;
end;
temps=toc;
if length(v_x)<3,
T;
a1=T(:,1);
b1=T(:,2);
a=sx_init(1); a1t;

```

```

b=sx_init(2); b1t;
l=length(a);
p=(fv(length(fv))-fv(1))/(ev(length(ev))-ev(1));
fv=fv(1):p:fv(length(fv));
sA,Bt=meshgrid(ev,fv);
%sD,Et=meshgrid(a,b);%aici
for i=1:length(a),
c(i)=feval(fun,sa(i),b(i)t);
end
%for i=1:length(a)
%for j=1:length(b)
%F(i,j)=feval(fun,sD(i,j),E(i,j)t);
%end
%end
for i=1:length(ev),
    for j=1:length(fv)
C(i,j)=feval(fun,sA(i,j),B(i,j)t);
end
end
%for i=1:length(a),
%    for j=1:length(a)
%C(i,j)=feval(fun,sA(i,j),B(i,j)t);
%end
%end
subplot(2,2,2);grid
%mesh(C)
cla reset
meshc(A,B,C);
title('Representation 3-D');
xlabel('x1');ylabel('x2');zlabel('f');
subplot(2,2,4);

```

```

contour(A,B, C);
title('Contour');
xlabel('x1');ylabel('x2');

hold on
for i=1:(length(a)-1),
plot(a(i),b(i),'o')
pause(.5)
hold on
end
plot(a(1),b(1),'*r');
hold on
plot(a,b,'-');
grid;
%subplot(2,2,4);
%plot3(a,b,c,'o')
end
grid;
solution = feval(fun, n_x)

n_x
iter

**Metoda lui Newton generalizata**

**grnr**

% Ce fichier construit l'interface graphique
% Appuyez le bouton "Start" pour lancer l'application
% Ce fichier appelle le fichier nr1.m
% Fichier grnr.m
%

h=figure('Name','Newton généralisée','color',s0.5 0.5 .5t);
hed1=uicontrol('Style','Edit','String','10','Units',...
'normalized','Position',s.1 .9 .09 .05t,'BackgroundColor',s1,1,1t);
hed2=uicontrol('Style','Edit','String','10','Units',...

```

```

'normalized','Position',s.1 .83 .09 .05t,'BackgroundColor',s1,1,1,t);
hed3=uicontrol('Style','Edit','String','.001','Units',...
'normalized','Position',s.1 .76 .09 .05t,'BackgroundColor',s1,1,1,t);
hed4=uicontrol('Style','Edit','String','test1','Units',...
'normalized','Position',s.1 .69 .09 .05t,'BackgroundColor',s1,1,1,t);
hed8=uicontrol('Style','Edit','String','-10','Units',...
'normalized','Position',s.25 .9 .09 .05t,'BackgroundColor',s1,1,1,t);
hed9=uicontrol('Style','Edit','String','10','Units',...
'normalized','Position',s.4 .9 .09 .05t,'BackgroundColor',s1,1,1,t);
hed10=uicontrol('Style','Edit','String','-10','Units',...
'normalized','Position',s.25 .83 .09 .05t,'BackgroundColor',s1,1,1,t);
hed11=uicontrol('Style','Edit','String','10','Units',...
'normalized','Position',s.4 .83 .09 .05t,'BackgroundColor',s1,1,1,t);
hpb1=uicontrol('Style','Pushbutton','String','Start','Units',...
'normalized','Position',s.1 .03 .1 .05t,'Callback','nr1');
htxt1=uicontrol('style','text',...
'units','normalized',...
        'position',s.001 .9 .04 .05t,...
        'string','x1','BackgroundColor',s1,1,1,t);
htxt2=uicontrol('style','text',...
'units','normalized',...
        'position',s.001 .83 .04 .05t,...
        'string','x2','BackgroundColor',s1,1,1,t);
htxt3=uicontrol('style','text',...
'units','normalized',...
        'position',s.001 .76 .04 .05t,...
        'string','err','BackgroundColor',s1,1,1,t);
htxt4=uicontrol('style','text',...
'units','normalized',...
        'position',s.001 .69 .09 .05t,...
        'string','Fonction','BackgroundColor',s1,1,1,t);

```

```

htxt10=uicontrol('style','text',...
'units','normalized',...
    'position',s.2 .9 .04 .05t,...
    'string','x1m','BackgroundColor',s1,1,1t);
htxt11=uicontrol('style','text',...
'units','normalized',...
    'position',s.35 .9 .04 .05t,...
    'string','x1M','BackgroundColor',s1,1,1t);
htxt12=uicontrol('style','text',...
'units','normalized',...
    'position',s.2 .83 .04 .05t,...
    'string','x2m','BackgroundColor',s1,1,1t);
htxt13=uicontrol('style','text',...
'units','normalized',...
    'position',s.35 .83 .04 .05t,...
    'string','x2M','BackgroundColor',s1,1,1t);
**nr1**
cla reset
val1=get(hed1,'String');
val2=get(hed2,'String');
er=get(hed3,'String');
err=str2num(er);
%x1=str2num(val1);
e=get(hed8,'String');
f=get(hed9,'String');
g=get(hed10,'String');
h=get(hed11,'String');
er=get(hed3,'String');
ev=sstr2num(e):1:str2num(f)t;
fv=sstr2num(g):1:str2num(h)t;
%x2=str2num(val2);

```

```

v_x=sstr2num(val1) str2num(val2)t;
fun=get(hed4,'String')
ssolution,n_x,tempst=nr(fun,v_x,err,ev,fv);
hed5=uicontrol('Style','Edit','String',num2str(n_x(1)), 'Units',...
'normalized','Position',s.1 .52 .1 .05t,'BackgroundColor',s1,1,1t);
hed6=uicontrol('Style','Edit','String',num2str(n_x(2)), 'Units',...
'normalized','Position',s.1 .45 .1 .05t,'BackgroundColor',s1,1,1t);
hed7=uicontrol('Style','Edit','String',num2str(temps), 'Units',...
'normalized','Position',s.1 .38 .1 .05t,'BackgroundColor',s1,1,1t);
htxt5=uicontrol('style','text',...
'units','normalized',...
'position',s.001 .52 .09 .05t,...
'string','x1min','BackgroundColor',s1,1,1t);
htxt6=uicontrol('style','text',...
'units','normalized',...
'position',s.001 .45 .09 .05t,...
'string','x2min','BackgroundColor',s1,1,1t);
htxt7=uicontrol('style','text',...
'units','normalized',...
'position',s.001 .38 .09 .05t,...
'string','temps','BackgroundColor',s1,1,1t);
**nr**
function ssolution, n_x,tempst = nr(fun, v_x, err,ev,fv)
%
n_x = v_x;
x_init=v_x;
k = 0;
iter=0;
tic;
while ((norm(n_x - v_x) > err) | (k < 1)) & (k < 500),
k = k+1;

```

```

iter=iter+1;
v_x = n_x;
v_d = inv(hessian(fun, v_x, err));
v_d = (gradient(fun, v_x, err))*(-v_d);
alpha = opt_1_s(fun, v_x, v_d, err);
n_x = v_x+alpha*v_d;
T(k,:)=n_x;
end;
temps=toc;
if length(v_x)<3,
T;
a1=T(:,1);
b1=T(:,2);
a=sx_init(1); a1t;
b=sx_init(2); b1t;
l=length(a);
p=(fv(length(fv))-fv(1))/(ev(length(ev))-ev(1));
fv=fv(1):p:fv(length(fv));
sA,Bt=meshgrid(ev,fv);
% sD,Et=meshgrid(a,b);% aici
for i=1:length(a),
c(i)=feval(fun,sa(i),b(i)t);
end
%for i=1:length(a)
%for j=1:length(b)
%F(i,j)=feval(fun,sD(i,j),E(i,j)t);
%end
%end
for i=1:length(ev),
    for j=1:length(fv)
C(i,j)=feval(fun,sA(i,j),B(i,j)t);

```

```

end
end
%for i=1:length(a),
%    for j=1:length(a)
%C(i,j)=feval(fun,sA(i,j),B(i,j)t);
%end
%end
subplot(2,2,2);grid
%mesh(C)
cla reset
meshc(A,B,C);
title('Representation 3-D');
xlabel('x1');ylabel('x2');zlabel('f');
subplot(2,2,4);
contour(A,B, C,30);
title('Contour');
xlabel('x1');ylabel('x2');
hold on
for i=1:(length(a)-1),
plot(a(i),b(i),'o')
pause(.5)
hold on
end
plot(a(l),b(l),'*r');
hold on
plot(a,b,'-');
grid;
%subplot(2,2,4);
%plot3(a,b,c,'o')
end
grid;

```

```

solution = feval(fun, n_x)

n_x

iter

**Metoda Nelder si Mead**

**grmm**

h=figure('Name','Nedler-Mead','color',s0.5 0.5 .5t);
hed1=uicontrol('Style','Edit','String','10','Units',...
'normalized','Position',s.1 .9 .09 .05t,'BackgroundColor',s1,1,1t);
hed2=uicontrol('Style','Edit','String','10','Units',...
'normalized','Position',s.1 .83 .09 .05t,'BackgroundColor',s1,1,1t);
hed3=uicontrol('Style','Edit','String','.001','Units',...
'normalized','Position',s.1 .76 .09 .05t,'BackgroundColor',s1,1,1t);
hed4=uicontrol('Style','Edit','String','test1','Units',...
'normalized','Position',s.1 .69 .09 .05t,'BackgroundColor',s1,1,1t);
hed8=uicontrol('Style','Edit','String','-10','Units',...
'normalized','Position',s.25 .9 .09 .05t,'BackgroundColor',s1,1,1t);
hed9=uicontrol('Style','Edit','String','10','Units',...
'normalized','Position',s.4 .9 .09 .05t,'BackgroundColor',s1,1,1t);
hed10=uicontrol('Style','Edit','String','-10','Units',...
'normalized','Position',s.25 .83 .09 .05t,'BackgroundColor',s1,1,1t);
hed11=uicontrol('Style','Edit','String','10','Units',...
'normalized','Position',s.4 .83 .09 .05t,'BackgroundColor',s1,1,1t);
hpb1=uicontrol('Style','Pushbutton','String','Start','Units',...
'normalized','Position',s.1 .03 .1 .05t,'Callback','nm1');
htxt1=uicontrol('style','text',...
'units','normalized',...
'position',s.001 .9 .04 .05t,...
'string','x1','BackgroundColor',s1,1,1t);
htxt2=uicontrol('style','text',...
'units','normalized',...
'position',s.001 .83 .04 .05t,...

```

```

        'string','x2','BackgroundColor',s1,1,1t);
htxt3=icontrol('style','text',...
'units','normalized',...
        'position',s.001 .76 .04 .05t,...
        'string','err','BackgroundColor',s1,1,1t);
htxt4=icontrol('style','text',...
'units','normalized',...
        'position',s.001 .69 .09 .05t,...
        'string','Fonction','BackgroundColor',s1,1,1t);
htxt10=icontrol('style','text',...
'units','normalized',...
        'position',s.2 .9 .04 .05t,...
        'string','x1m','BackgroundColor',s1,1,1t);
htxt11=icontrol('style','text',...
'units','normalized',...
        'position',s.35 .9 .04 .05t,...
        'string','x1M','BackgroundColor',s1,1,1t);
htxt12=icontrol('style','text',...
'units','normalized',...
        'position',s.2 .83 .04 .05t,...
        'string','x2m','BackgroundColor',s1,1,1t);
htxt13=icontrol('style','text',...
'units','normalized',...
        'position',s.35 .83 .04 .05t,...
        'string','x2M','BackgroundColor',s1,1,1t);
**nm1**
cla reset
val1=get(hed1,'String');
val2=get(hed2,'String');
e=get(hed8,'String');
f=get(hed9,'String');
```

```

g=get(hed10,'String');
h=get(hed11,'String');
%er=get(hed3,'String');
ev=sstr2num(e):1:str2num(f)t;
fv=sstr2num(g):1:str2num(h)t;
%er=get(hed3,'String');
%err=str2num(er);
%x1=str2num(val1);
%x2=str2num(val2);
x=sstr2num(val1) str2num(val2)t;
fun=get(hed4,'String')
sx,fmax,nf,tempst=nm(fun,x,ev,fv);
hed5=uicontrol('Style','Edit','String',num2str(x(1)), 'Units',...
'normalized','Position',s.1 .52 .1 .05t,'BackgroundColor',s1,1,1t);
hed6=uicontrol('Style','Edit','String',num2str(x(2)), 'Units',...
'normalized','Position',s.1 .45 .1 .05t,'BackgroundColor',s1,1,1t);
hed7=uicontrol('Style','Edit','String',num2str(temps), 'Units',...
'normalized','Position',s.1 .38 .1 .05t,'BackgroundColor',s1,1,1t);

htxt5=uicontrol('style','text',...
'units','normalized',...
    'position',s.001 .52 .09 .05t,...
    'string','x1min','BackgroundColor',s1,1,1t);
htxt6=uicontrol('style','text',...
'units','normalized',...
    'position',s.001 .45 .09 .05t,...
    'string','x2min','BackgroundColor',s1,1,1t);
htxt7=uicontrol('style','text',...
'units','normalized',...
    'position',s.001 .38 .09 .05t,...
    'string','temps','BackgroundColor',s1,1,1t);

```

```

**nm**

function sx, fmax, nf,tempst = nmsmax(fun, x, ev,fv,stopit,savit)
x_init=x;
n = prod(size(x));
x0 = x(:);
if nargin < 5, stopit(1) = 1e-3; end
tol = stopit(1);
if max(size(stopit)) == 1, stopit(2) = inf; end
if max(size(stopit)) == 2, stopit(3) = inf; end
if max(size(stopit)) == 3, stopit(4) = 0; end
if max(size(stopit)) == 4, stopit(5) = 1; end
trace = stopit(5);
if nargin < 6, savit = st; end
V = szeros(n,1) eye(n);
f = zeros(n+1,1);
V(:,1) = x0; x(:) = x0; f(1) = - feval(fun,x);
fmax_old = f(1);
if trace, fprintf('f(x0) = %9.4ean', f(1)), end
k = 0; m = 0;
tic;
%aici Set up initial simplex.
scale = max(norm(x0,inf),1);
if stopit(4) == 0
    % aici Regular simplex - all edges have same length.
    alpha = scale / (n*sqrt(2)) * s sqrt(n+1)-1+n sqrt(n+1)-1 t;
    V(:,2:n+1) = (x0 + alpha(2)*ones(n,1)) * ones(1,n);
    for j=2:n+1
        V(j-1,j) = x0(j-1) + alpha(1);
        x(:) = V(:,j); f(j) = -feval(fun,x);
    end
else

```

```

%aici Right-angled simplex based on co-ordinate axes.
alpha = scale*ones(n+1,1);
for j=2:n+1
    V(:,j) = x0 + alpha(j)*V(:,j);
    x(:) = V(:,j); f(j) = -feval(fun,x);
end
end
nf = n+1;
how = 'initial ';
stemp,jt = sort(f);
j = j(n+1:-1:1);
f = f(j); V = V(:,j);
alpha = 1; beta = 1/2; gamma = 2;
while 1 %%%%%% aici Outer (and only) loop.
k = k+1;
fmax = f(1);
if fmax > fmax_old
    if ~isempty(savit)
        x(:) = V(:,1); eval(s'save ' savit ' x fmax nft)
    end
    if trace
        fprintf('Iter. %2.0f,', k)
        fprintf(s' how = ' how ' t);
        % ' size = %2.0f,t;
        fprintf('nf = %3.0f, f = %9.4e (%2.1f%%)an', nf, fmax, ...
            100*(fmax-fmax_old)/(abs(fmax_old)+eps))
    end
end
end
fmax_old = fmax;
%%%%% aici Three stopping tests from nm.M
% Stopping Test 1 - f reached target value?

```

```

if fmax >= stopit(3)
    msg = s'Exceeded target...quittingan't;
    break % Quit.
end
% aici Stopping Test 2 - too many f-evals?
if nf >= stopit(2)
    msg = s'Max no. of function evaluations exceeded...quittingan't;
    break % Quit.
end
% aici Stopping Test 3 - converged? This is test (4.3) in s1t.
v1 = V(:,1);
size_simplex = norm(V(:,2:n+1)-v1(:,ones(1,n)),1) / max(1, norm(v1,1));
if size_simplex <= tol
    msg = sprintf('Simplex size %9.4e <= %9.4e...quittingan', ...
        size_simplex, tol);
    break % Quit.
end
% aici One step of the Nelder-Mead simplex algorithm

vbar = (sum(V(:,1:n))/n); % Mean value
vr = (1 + alpha)*vbar - alpha*V(:,n+1); x(:) = vr; fr = -feval(fun,x);
nf = nf + 1;
vk = vr; fk = fr; how = 'reflect, ';
if fr > f(n)
    if fr > f(1)
        ve = gamma*vr + (1-gamma)*vbar; x(:) = ve; fe = -feval(fun,x);
        nf = nf + 1;
        if fe > f(1)
            vk = ve; fk = fe;
            how = 'expand, ';
        end
    end
end

```

```

    end
else
    vt = V(:,n+1); ft = f(n+1);
    if fr > ft
        vt = vr; ft = fr;
    end
    vc = beta*vt + (1-beta)*vbar; x(:) = vc; fc = -feval(fun,x);
    nf = nf + 1;
    if fc > f(n)
        vk = vc; fk = fc;
        how = 'contract, ';
    else
        for j = 2:n
            V(:,j) = (V(:,1) + V(:,j))/2;
            x(:) = V(:,j); f(j) = -feval(fun,x);
        end
        nf = nf + n-1;
        vk = (V(:,1) + V(:,n+1))/2; x(:) = vk; fk = -feval(fun,x);
        nf = nf + 1;
        how = 'shrink, ';
    end
end

end

V(:,n+1) = vk;
f(n+1) = fk;
stemp,jt = sort(f);
j = j(n+1:-1:1);
f = f(j); V = V(:,j);

a(:,k)=V(:,1);
b(:,k)=V(:,2);

```

```

p=(fv(length(fv))-fv(1))/(ev(length(ev))-ev(1));
fv=fv(1):p:fv(length(fv));
sA,Bt=meshgrid(ev,fv);
%a=sx_init(1); a1t;
%b=sx_init(2); b1t;
l=length(a);
%for i=1:length(a),
%c(i)=feval(fun,sa(i),b(i)t);
%end
for i=1:length(ev),
    for j=1:length(fv)
C(i,j)=feval(fun,sA(i,j),B(i,j)t);
end
end
subplot(2,2,2);grid
mesh(A,B,C);
title('Representation 3-D');
xlabel('x1');ylabel('x2');zlabel('f');
%aici
subplot(2,2,4);
contour(A,B,C);
title('Contour');
xlabel('x1');ylabel('x2');
hold on
%for i=1:length(a),
%plot(a(1),a(2),'o')
%hold on
%plot(b(1),b(2),'r-')
%pause(.1)
end
grid

```

```

end %%%%%%%%% aici End of outer (and only) loop.

temps=toc;

%hold on

%plot(a,b,'-');

% Finished.

if trace, fprintf(msg), end

%iar aici

x(:) = V(:,1);

%a1=a(1,:);

%a2=a(2,:);

%plot(a1,a2);

%hold on

b1=b(1,:);

b2=b(2,:);

l=length(b1);

%for p=1:(l-1)

%plot(b1(p),b2(p),'o');

%pause(.5)

%hold on

%end

plot(b1,b2,'-')

plot(b1(l),b2(l),'r*');

grid;

**Metoda Hooke si Jeeves**

**grhj**

h=figure('Name','Hooke et Jeeves','color',s0.5 0.5 .5t);

hed1=uicontrol('Style','Edit','String','0','Units',...

'normalized','Position',s.1 .9 .09 .05t,'BackgroundColor',s1,1,1t);

hed2=uicontrol('Style','Edit','String','0','Units',...

'normalized','Position',s.1 .83 .09 .05t,'BackgroundColor',s1,1,1t);

hed3=uicontrol('Style','Edit','String','.001','Units',...

```

```

'normalized','Position',s.1 .76 .09 .05t,'BackgroundColor',s1,1,1t);
hed4=uicontrol('Style','Edit','String','test1','Units',...
'normalized','Position',s.1 .69 .09 .05t,'BackgroundColor',s1,1,1t);
hed8=uicontrol('Style','Edit','String','-10','Units',...
'normalized','Position',s.25 .9 .09 .05t,'BackgroundColor',s1,1,1t);
hed9=uicontrol('Style','Edit','String','10','Units',...
'normalized','Position',s.4 .9 .09 .05t,'BackgroundColor',s1,1,1t);
hed10=uicontrol('Style','Edit','String','-10','Units',...
'normalized','Position',s.25 .83 .09 .05t,'BackgroundColor',s1,1,1t);
hed11=uicontrol('Style','Edit','String','10','Units',...
'normalized','Position',s.4 .83 .09 .05t,'BackgroundColor',s1,1,1t);
hpb1=uicontrol('Style','Pushbutton','String','Start','Units',...
'normalized','Position',s.1 .03 .1 .05t,'Callback','hj1');
htxt1=uicontrol('style','text',...
'units','normalized',...
        'position',s.001 .9 .04 .05t,...
        'string','x1','BackgroundColor',s1,1,1t);
htxt2=uicontrol('style','text',...
'units','normalized',...
        'position',s.001 .83 .04 .05t,...
        'string','x2','BackgroundColor',s1,1,1t);
htxt3=uicontrol('style','text',...
'units','normalized',...
        'position',s.001 .76 .04 .05t,...
        'string','err','BackgroundColor',s1,1,1t);
htxt4=uicontrol('style','text',...
'units','normalized',...
        'position',s.001 .69 .09 .05t,...
        'string','Fonction','BackgroundColor',s1,1,1t);
htxt10=uicontrol('style','text',...
'units','normalized',...

```

```

        'position',s.2 .9 .04 .05t,...
        'string','x1m','BackgroundColor',s1,1,1t);
htxt11=icontrol('style','text',...
'units','normalized',...
        'position',s.35 .9 .04 .05t,...
        'string','x1M','BackgroundColor',s1,1,1t);
htxt12=icontrol('style','text',...
'units','normalized',...
        'position',s.2 .83 .04 .05t,...
        'string','x2m','BackgroundColor',s1,1,1t);
htxt13=icontrol('style','text',...
'units','normalized',...
        'position',s.35 .83 .04 .05t,...
        'string','x2M','BackgroundColor',s1,1,1t);
**hj1**
cla reset
val1=get(hed1,'String');
val2=get(hed2,'String');
er=get(hed3,'String');
e=get(hed8,'String');
f=get(hed9,'String');
g=get(hed10,'String');
h=get(hed11,'String');
ev=sstr2num(e):1:str2num(f)t;
fv=sstr2num(g):1:str2num(h)t;
err=str2num(er);
dist=s.01 .01t;
v_x=sstr2num(val1) str2num(val2)t;
fun=get(hed4,'String');
hj;
hed5=icontrol('Style','Edit','String',num2str(n_x(1)),'Units',...

```

```

'normalized','Position',s.1 .52 .1 .05t,'BackgroundColor',s1,1,1t);
hed6=uicontrol('Style','Edit','String',num2str(n_x(2)),'Units',...
'normalized','Position',s.1 .45 .1 .05t,'BackgroundColor',s1,1,1t);
hed7=uicontrol('Style','Edit','String',num2str(temps),'Units',...
'normalized','Position',s.1 .38 .1 .05t,'BackgroundColor',s1,1,1t);
htxt5=uicontrol('style','text',...
'units','normalized',...
        'position',s.001 .52 .09 .05t,...
        'string','x1min','BackgroundColor',s1,1,1t);
htxt6=uicontrol('style','text',...
'units','normalized',...
        'position',s.001 .45 .09 .05t,...
        'string','x2min','BackgroundColor',s1,1,1t);
htxt7=uicontrol('style','text',...
'units','normalized',...
        'position',s.001 .38 .09 .05t,...
        'string','temps','BackgroundColor',s1,1,1t);
**hj**
% hj
% Cette fonction trouve la valeur optimale en utilisant
% la méthode de Hooke et jeeves.
% ' solution ' est la solution optimale trouvée par la méthode
% ' de n_x ' est le x-vecteur optimal
% 'temps' este le temps de calcul (CPU time)
% ' fun ' est la fonction objective
% ' v_x ' est le point de départ
% ' err ' est l'exactitude exigée
% si le nombre d'itérations est plus de 500 la fonction est terminé
%
% Fichier hj.m

```

```

k=0;
n_x = v_x;
x_init=v_x;
tic
while ((norm(n_x - v_x) > err) | (k < 1)) & (k < 1000),
k = k+1;
v_x=n_x;
sn_f,v_f,n_xt=ch(fun,v_x,dist);
if feval(fun,n_x) < feval(fun,v_x),
x_try=2*n_x-v_x;
if feval(fun,x_try) < feval(fun,n_x),
v_x=n_x;
n_x=x_try;
dist=2*dist;
T(k,:)=n_x;
end
else
%disp('bla')
dist=dist/20;
end
end
T;
temps=toc;
a1=T(:,1);
b1=T(:,2);
a=sx_init(1); a1t;
b=sx_init(2); b1t;
l=length(a);
p=(fv(length(fv))-fv(1))/(ev(length(ev))-ev(1));
fv=fv(1):p:fv(length(fv));

```

```

sA,Bt=meshgrid(ev,fv);
%sD,Et=meshgrid(a,b);%aici
for i=1:length(a),
c(i)=feval(fun,sa(i),b(i)t);
end
%for i=1:length(a)
%for j=1:length(b)
%F(i,j)=feval(fun,sD(i,j),E(i,j)t);
%end
%end
for i=1:length(ev),
    for j=1:length(fv)
C(i,j)=feval(fun,sA(i,j),B(i,j)t);
end
end
%for i=1:length(a),
%    for j=1:length(a)
%C(i,j)=feval(fun,sA(i,j),B(i,j)t);
%end
%end
subplot(2,2,2);grid
mesh(A,B,C)
% meshc(A,B,C);
title('Representation 3-D');
xlabel('x1');ylabel('x2');zlabel('f');
subplot(2,2,4);
contour(A,B,C);
title('Contour');
xlabel('x1');ylabel('x2');
hold on
for i=1:(length(a)-1),

```

```

plot(a(i),b(i),'o')
pause(.5)
hold on
end
plot(a(l),b(l),'*r');
hold on
plot(a,b,'-');
grid;
%subplot(2,2,4);
%plot3(a,b,c,'o')
solution = feval(fun, n_x);
n_x
**Metoda gradientului**
**grsd**
h=figure('Name','Méthode du gradient','color',s0.5 0.5 .5t)
hed1=uicontrol('Style','Edit','String','10','Units',...
'normalized','Position',s.1 .9 .09 .05t,'BackgroundColor',s1,1,1t);
hed2=uicontrol('Style','Edit','String','10','Units',...
'normalized','Position',s.1 .83 .09 .05t,'BackgroundColor',s1,1,1t);
hed3=uicontrol('Style','Edit','String','.00001','Units',...
'normalized','Position',s.1 .76 .09 .05t,'BackgroundColor',s1,1,1t);
hed4=uicontrol('Style','Edit','String','test1','Units',...
'normalized','Position',s.1 .69 .09 .05t,'BackgroundColor',s1,1,1t);
hed8=uicontrol('Style','Edit','String','-10','Units',...
'normalized','Position',s.25 .9 .09 .05t,'BackgroundColor',s1,1,1t);
hed9=uicontrol('Style','Edit','String','10','Units',...
'normalized','Position',s.4 .9 .09 .05t,'BackgroundColor',s1,1,1t);
hed10=uicontrol('Style','Edit','String','-10','Units',...
'normalized','Position',s.25 .83 .09 .05t,'BackgroundColor',s1,1,1t);
hed11=uicontrol('Style','Edit','String','10','Units',...
'normalized','Position',s.4 .83 .09 .05t,'BackgroundColor',s1,1,1t);

```

```
hpb1=icontrol('Style','Pushbutton','String','Start','Units',...
```

```
'normalized','Position',s.1 .03 .1 .05t,'Callback','sd1');
```

```
htxt1=icontrol('style','text',...
```

```
'units','normalized',...
```

```
    'position',s.001 .9 .04 .05t,...
```

```
    'string','x1','BackgroundColor',s1,1,1t);
```

```
htxt2=icontrol('style','text',...
```

```
'units','normalized',...
```

```
    'position',s.001 .83 .04 .05t,...
```

```
    'string','x2','BackgroundColor',s1,1,1t);
```

```
htxt3=icontrol('style','text',...
```

```
'units','normalized',...
```

```
    'position',s.001 .76 .04 .05t,...
```

```
    'string','err','BackgroundColor',s1,1,1t);
```

```
htxt4=icontrol('style','text',...
```

```
'units','normalized',...
```

```
    'position',s.001 .69 .09 .05t,...
```

```
    'string','Fonction','BackgroundColor',s1,1,1t);
```

```
htxt10=icontrol('style','text',...
```

```
'units','normalized',...
```

```
    'position',s.2 .9 .04 .05t,...
```

```
    'string','x1m','BackgroundColor',s1,1,1t);
```

```
htxt11=icontrol('style','text',...
```

```
'units','normalized',...
```

```
    'position',s.35 .9 .04 .05t,...
```

```
    'string','x1M','BackgroundColor',s1,1,1t);
```

```
htxt12=icontrol('style','text',...
```

```
'units','normalized',...
```

```
    'position',s.2 .83 .04 .05t,...
```

```
    'string','x2m','BackgroundColor',s1,1,1t);
```

```
htxt13=icontrol('style','text',...
```

```

'units','normalized',...
        'position',s.35 .83 .04 .05t,...
        'string','x2M','BackgroundColor',s1,1,1t);
**sd1**
cla reset
val1=get(hed1,'String');
val2=get(hed2,'String');
er=get(hed3,'String');
err=str2num(er);
%x1=str2num(val1);
e=get(hed8,'String');
f=get(hed9,'String');
g=get(hed10,'String');
h=get(hed11,'String');
er=get(hed3,'String');
ev=sstr2num(e):1:str2num(f)t;
fv=sstr2num(g):1:str2num(h)t;
%x2=str2num(val2);
v_x=sstr2num(val1) str2num(val2)t;
fun=get(hed4,'String')
ssolution,n_x,tempst=sd(fun,v_x,err,ev,fv);
hed5=uicontrol('Style','Edit','String',num2str(n_x(1)),'Units',...
'normalized','Position',s.1 .52 .1 .05t,'BackgroundColor',s1,1,1t);
hed6=uicontrol('Style','Edit','String',num2str(n_x(2)),'Units',...
'normalized','Position',s.1 .45 .1 .05t,'BackgroundColor',s1,1,1t);
hed7=uicontrol('Style','Edit','String',num2str(temps),'Units',...
'normalized','Position',s.1 .38 .1 .05t,'BackgroundColor',s1,1,1t);
htxt5=uicontrol('style','text',...
'units','normalized',...
        'position',s.001 .52 .09 .05t,...
        'string','x1min','BackgroundColor',s1,1,1t);

```

```

htxt6=uicontrol('style','text',...
'units','normalized',...
    'position',s.001 .45 .09 .05t,...
    'string','x2min','BackgroundColor',s1,1,1t);
htxt7=uicontrol('style','text',...
'units','normalized',...
    'position',s.001 .38 .09 .05t,...
    'string','temps','BackgroundColor',s1,1,1t);
**sd**
function ssolution, n_x,tempst = sd(fun, v_x, err,ev,fv)
% ssolution, new_xt = sd(fun_name, old_x, error)
% This function finds the optimal value using Steepest Descent method.
% 'solution' is the optimal solution found by the method
% 'new_x' is the optimal x-vector
% 'fun_name' is the objective function
% 'old_x' is the starting point
% 'error'is the demanded accuracy
% remark if the number of iterations is more than 500 the function is terminated
%
% File sd.m
n_x = v_x;
x_init=v_x;
k = 0;
iter=0;
while ((norm(n_x - v_x) > err) | (k < 1)) & (k < 500),
tic
iter=iter+1;
k = k+1;
    v_x = n_x;
    v_d = -gradient(fun, v_x, err);
    alpha = opt_1_s(fun, v_x, v_d, err);

```

```

n_x = v_x+alpha*v_d;
T(k,:)=n_x;
end;
temps=toc;
if length(v_x)<3,
T;
a1=T(:,1);
b1=T(:,2);
a=sx_init(1); a1t;
b=sx_init(2); b1t;
l=length(a);
p=(fv(length(fv))-fv(1))/(ev(length(ev))-ev(1));
fv=fv(1):p:fv(length(fv));
sA,Bt=meshgrid(ev,fv);
% sD,Et=meshgrid(a,b);% aici
for i=1:length(a),
c(i)=feval(fun,sa(i),b(i)t);
end
%for i=1:length(a)
%for j=1:length(b)
%F(i,j)=feval(fun,sD(i,j),E(i,j)t);
%end
%end
for i=1:length(ev),
    for j=1:length(fv)
C(i,j)=feval(fun,sA(i,j),B(i,j)t);
end
end
%for i=1:length(a),
%    for j=1:length(a)
%C(i,j)=feval(fun,sA(i,j),B(i,j)t);

```

```
%end
%end
subplot(2,2,2);grid
%mesh(C)
cla reset
meshc(A,B,C);
title('Representation 3-D');
xlabel('x1');ylabel('x2');zlabel('f');
subplot(2,2,4);
contour(A,B, C,30);
title('Contour');
xlabel('x1');ylabel('x2');
hold on
for i=1:(length(a)-1),
plot(a(i),b(i),'o')
pause(.5)
hold on
end
plot(a(1),b(1),'*r');
hold on
plot(a,b,'-');
grid;
%subplot(2,2,4);
%plot3(a,b,c,'o')
end
grid;
solution = feval(fun, n_x)
n_x
iter
```