Metoda najstrmijeg spusta u optimizaciji nelinearnih viševeličinskih funkcija bez ograničenja

Kada tražimo minimum ili maksimum nelinearne viševeličinske funkcije metodom najstrmijeg spusta ili rasta imamo tri načina na koji biramo ili računamo koeficijent porasta/spusta alfa u jednadžbi :

$X_{novi} = X_{stari}$ – alfa * gradijent (X_{stari})

Prvi slučaj je kad je alfa stalna kroz iteracije. Drugi slučaj je kada se računa na temelju razvoja funkcije u X_{stari} u Taylorov red i aproksimirati je sa prva dva člana. Tada imamo :

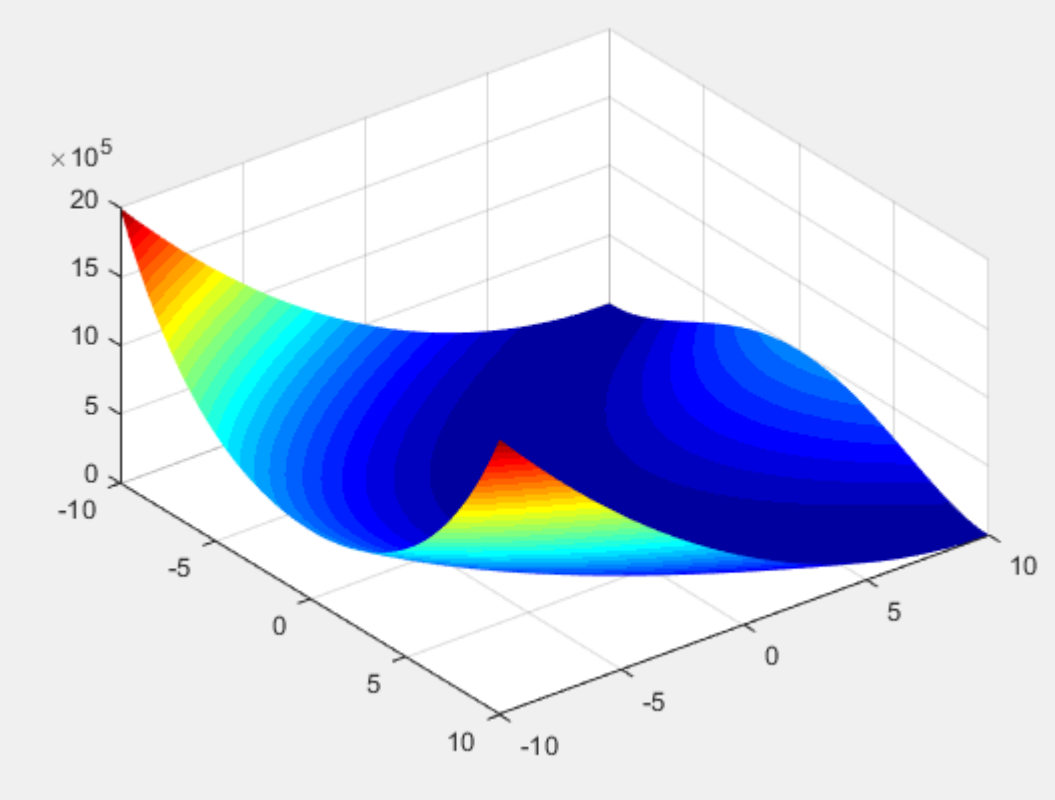
$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{g}_k^T \mathbf{H}_k \mathbf{g}_k} \mathbf{g}_k$$

Treći slučaj je kada, zbog nekog razloga, ne možemo računati Hessian pa računamo vrijednost funkcije u X_{stari} i X_{stari} – alfa (procijenjeni/stari) * gradijent (X_{stari}) ,pa član sa Hessianom ispada :

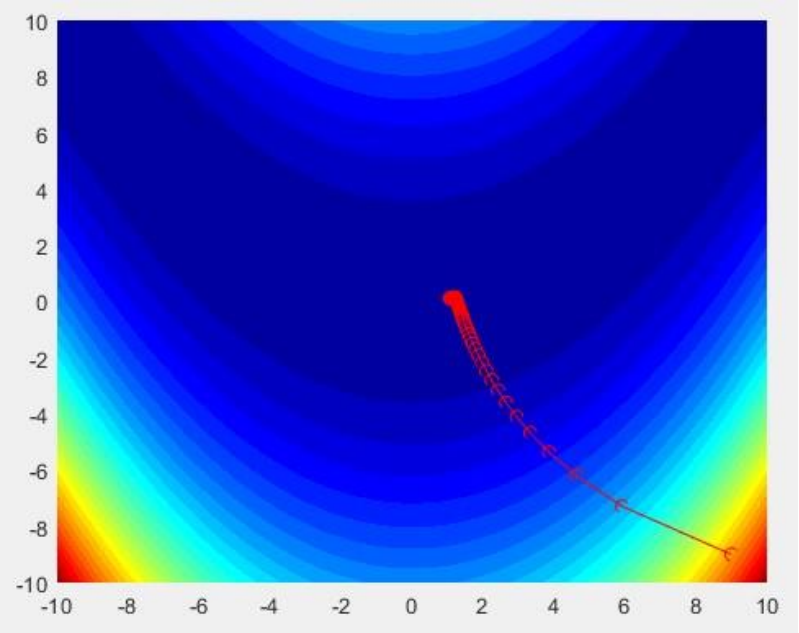
$$\mathbf{g}_k^T \mathbf{H}_k \mathbf{g}_k \approx \frac{2(\hat{f} - f_k + \hat{\alpha} \mathbf{g}_k^T \mathbf{g}_k)}{\hat{\alpha}^2}$$

Kao testna funkcija uzeta je Rosenbrockova funkcija u obliku :

$F(x1,x2) = -(10 - X1)^2 + 50 * (10 * X2 - X1^2)^2$

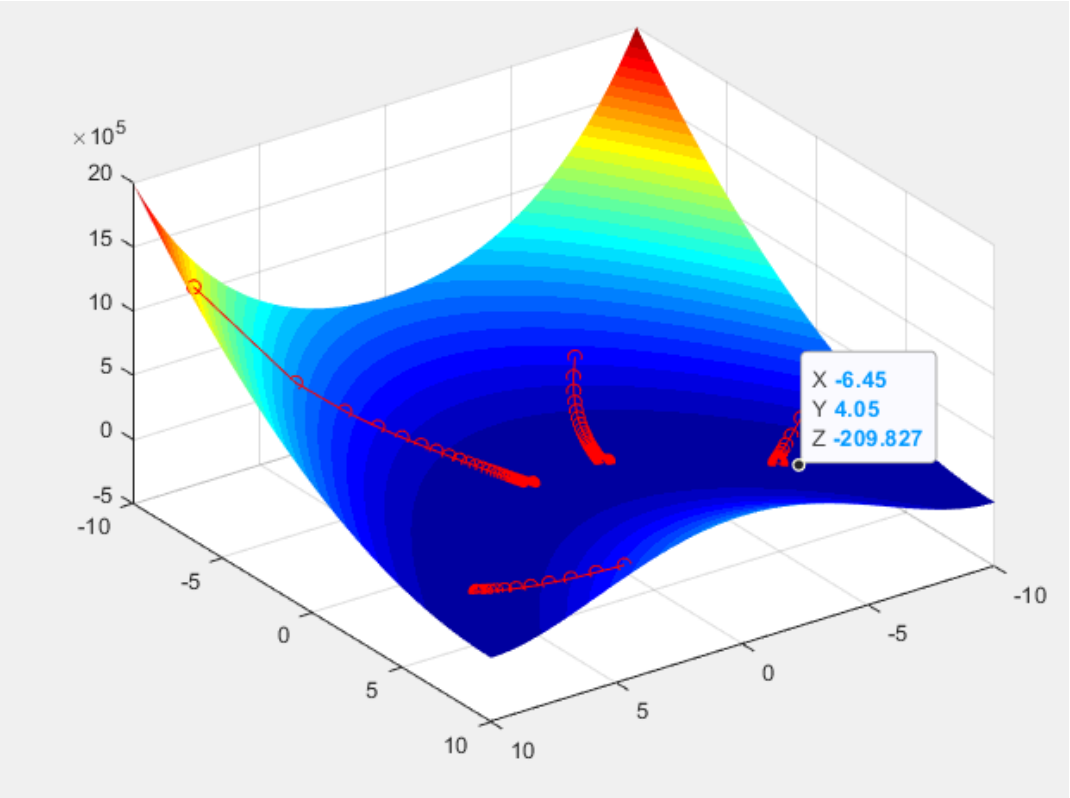


Fiksni alfa:



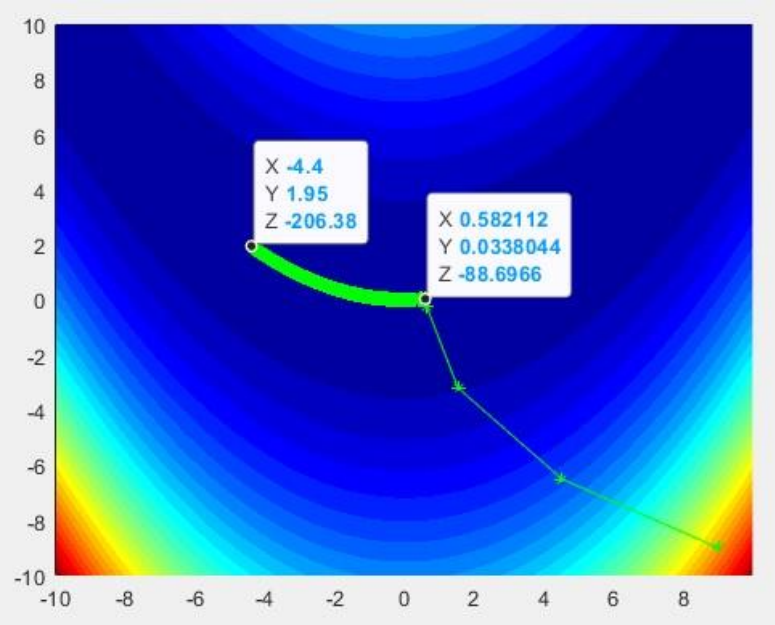
Kriteriji zaustavljanja su broj iteracija od 1000 , minimalna Euklidova ,L2, norma vektora [X_{novi}-X_{stari}] i vektora gradijenta. Norma vektora X predstavlja najmanju moguću udaljenost u prostoru između stare i nove točke. A norma vektora gradijenta je mjera promjene kuta između starog i novog pravca po kojemu pomak ide.

Za vrijednost alfa je eksperimentalno uzeta vrijednost $1*10^{-5}$. Na slici su prikazane iteracije iz različitih početnih točaka :

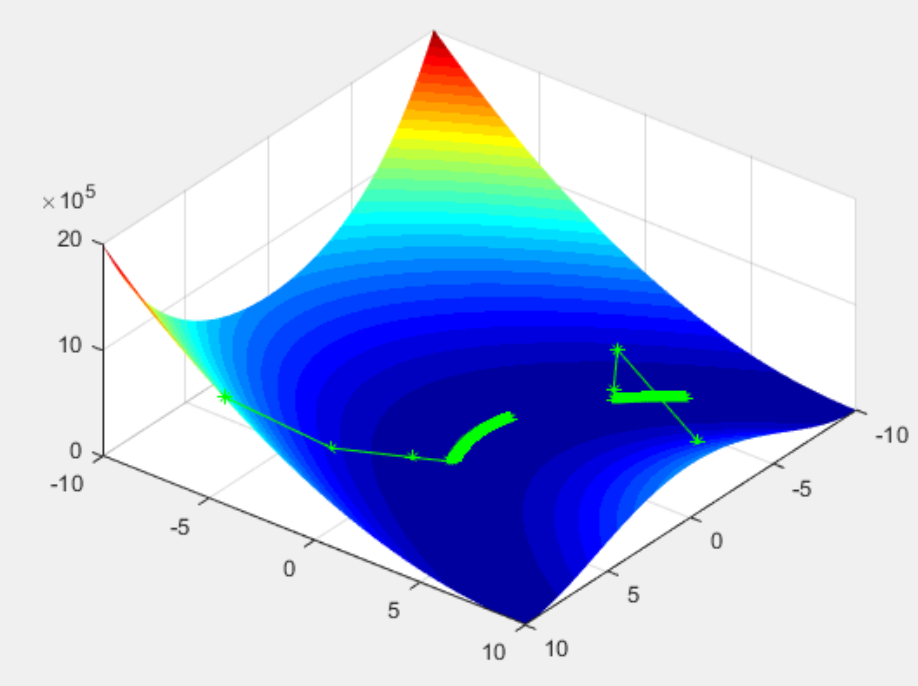


Stvarni minimum ove funkcije je negdje oko označene točke. Možemo reći da ova osnovna metoda daje stabilno napredovanje ali kada se pojavi ovako velika dolina tada broj iteracija raste, a nema velikog pomaka u novoj točki. Sve ove iteracije su završene na maksimalnom broju 'maxiter' u programu (1000). Vrlo je osjetljiva na početne uvjete i sporo konvergira. Vrijednost norme pomaka i vrijednost norme gradijenta za prekid je postavljena na $5*10^{-10}$.

Izračunati alfa:



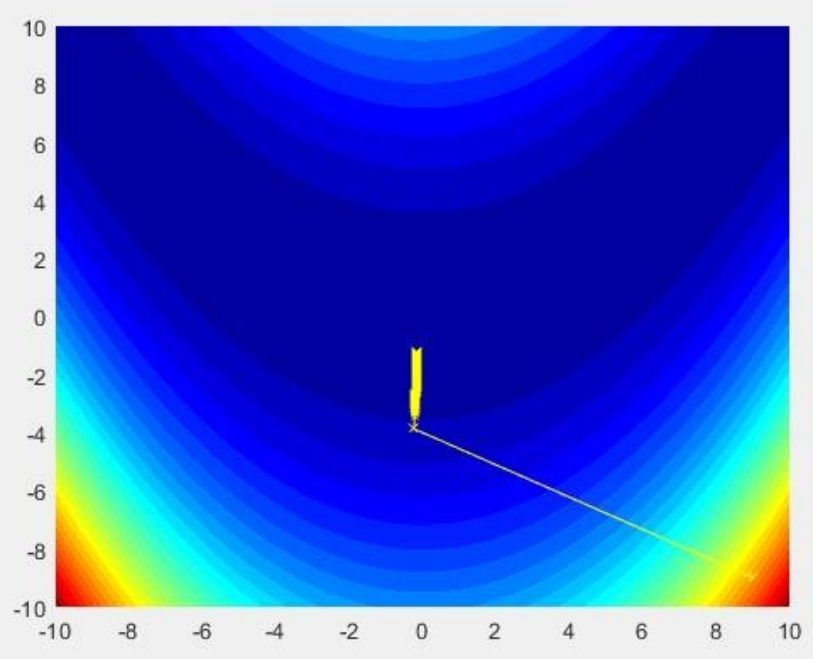
Kada alfu računamo iz aproksimacije funkcije Taylorovim redom sa dva člana dobije se kontinuirano primicanje minimumu . Iteracije su zaustavljene u 1000 , a kriteriji zaustavljanja su postavljeni također na $5*10^{-10}$.



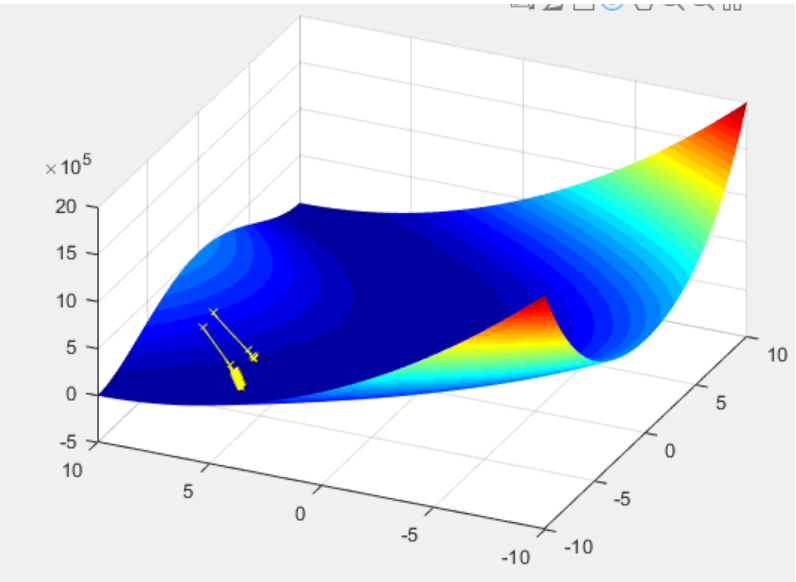
Naravno, ako smanjimo te granice algoritam će se prije zaustaviti i biti ćemo dalje od minimuma. Također postoji velika osjetljivost na početne uvjete, ali opet manja nego kod fiksne alfe.

Treći slučaj:

Početni alfa je postavljen na 3e-5 (inače se postavlja početni alfa na 1) ,jer veće vrijednosti uzrokuju nestabilnost algoritma, pa niti ne dolazimo blizu rješenja. Problem kod procijenjene vrijednosti alfa je da se ona stalno smanjuje sa svakom sljedećom iteracijom ,pa je napredovanje ka minimumu sve sporije. Brzo se izlazi iz petlje zbog male norme u vektoru X ,a dobije se rješenje daleko od minimuma, ali u 17 iteracija. Prednost je ta da se brzo dođe do 'doline' za razliku od prve dvije metode. U jednom koraku za razliku od računate alfe ,gdje imamo 4 koraka do doline.



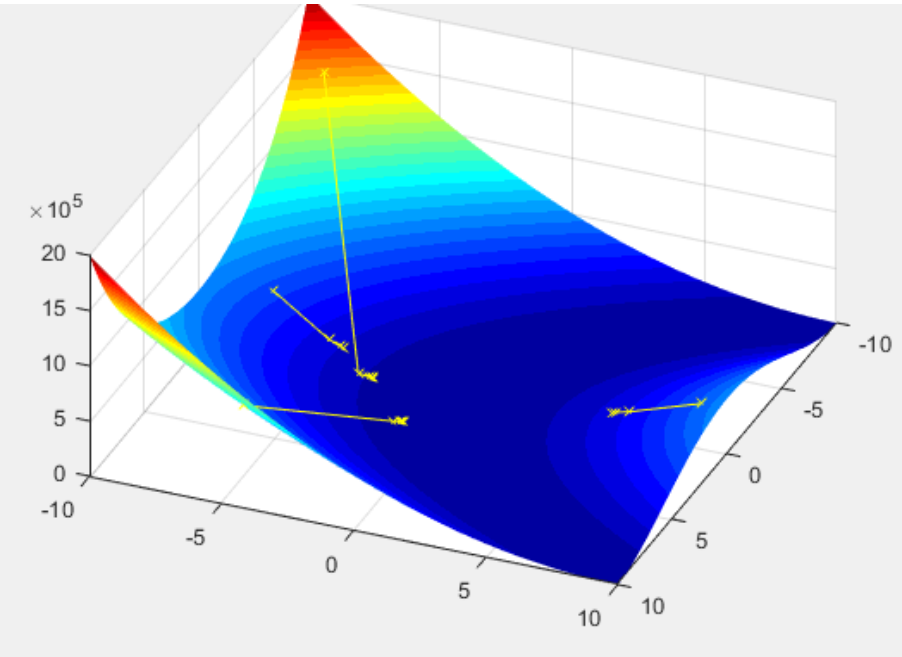
Moguće je prijeći na fiksni alfa, ako ona padne ispod neke vrijednosti npr. 1e-7.



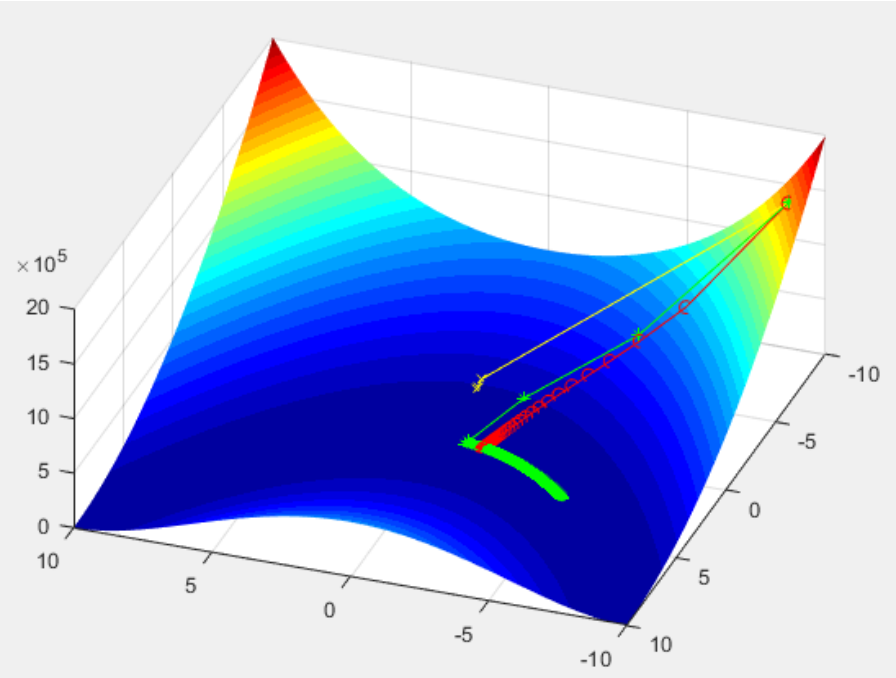
Na gornjoj slici su prikazane iteracije iz početne točke X=[-3;8] bez ograničenja alfe (kraća putanja 20 iteracija) i X=[-4;8] sa ograničenjem alfe na 1e-7 (duža putanja 1000 iteracija).

Procijenjena alfa bez ograničenja za različite početne točke:

Početni alfa = 3e-5, minimum norme vektora X err= 0.5e-9 i minimum norme gradijenta errg= 0.5e-3.



Prikaz sve tri metoda sa iste početne toče za iste uvjete zaustavljanja:



Prva metoda i druga metoda (najbolje rješenje) 1000 iteracija , a treća 16 iteracija i najgore rješenje. Možemo zaključiti da prva metoda ima stabilnost, druga točnost a treća brzinu u pronalaženju minimuma ove funkcije.

```
%x0 = [15 15]'; % Zadavanje poèetne toèke
x0 = [-9 -9]';% Za Rosenbrockovu funkciju
f = @(a,b) -(10-a).^2+50*(10*b-a.^2).^2 ;
%f = @(x1,x2) x1.^2 + x1.*x2 + 3*x2.^2; % Tipièna kvadratna funkcija
% redefine objective function syntax for use with optimization:
f2 = @(x) f(x(1),x(2));

err = 0.5e-9;
errg = 0.5e-3;
maxiter=1000;
xnorm = inf;
gnorm = inf;
x = x0;
niter = 0;
figure(1);
a = -10:0.05:10; % Za Rosenbrockovu funkciju
b = -10:0.05:10; % Za Rosenbrockovu funkciju
%a = -20:0.05:20;
%b = -20:0.05:20;
[X,Y] = meshgrid(a,b);
Z = -(10-X).^2+50*((10*Y-(X.^2)).^2) ;
%Z = X.^2+X.*Y+3*Y.^2;
surf(X,Y,Z)
colormap(jet(31));
shading interp;
hold on

alfa = input('Za unaprijd zadani alfa unesi 1,\nza raèunanje alfa pomoæu
hesijana 2,\na za raèunanje alfa pomoæu procjenjenih vrijednosti 3');

switch alfa
    case 0
        %alfa=0.01;
        alfa=0.00001; % Za Rosenbrockovu funkciju
        flag = 0;
    case 1
        flag = 1;
    case 2
        fold= f2(x0);
        %alfa=0.3;
        alfa=3e-5; % Za Rosenbrockovu funkciju
        flag = 2;
    otherwise
        disp('Niste unijeli 0,1 ili 2.')
        error('Alfa nije definiran.')
end

while and(gnorm >= errg, and(niter <= maxiter, xnorm >= err))
    gnormp=gnorm;
    g = grad(x);
    if (flag == 1)
        h = hessian(x);
        alfa = (g'*g)/(g'*h*g);
    end

    % take step:
    xnew = x - alfa*g;
    if (flag == 2)
        fnew = f2(xnew);
        alfa = -((g)'*g*alfa^2)/(2*(fnew-fold-alfa*(g)'*g));
```

```
        % if alfa < 1e-7  % Nužno kod Rosenbrockove funkcije za
%         približavanje minimumu
        %     alfa = 1e-7;
        % end
        fold = fnew;

%          if (gnorm-gnormp)<0.01 %Ubrzava za kvadratnu funkciju
%             flag=3;
%             alfa=0.25;
%             err = 0.5e-3;
%          end

    end
    xnorm = norm(x-xnew);
    gnorm = norm(g);
    % update termination metrics
    niter = niter + 1;

    % check step
    if ~isfinite(xnew)
        display(['Number of iterations: ' num2str(niter)])
        error('x is inf or NaN')
    end
    % plot current point
    if (flag == 0)
    plot3([x(1) xnew(1)],[x(2) xnew(2)],[f(x(1),x(2))
f(xnew(1),xnew(2))],'ro-')
    refresh
    end
    if(flag == 1)
    plot3([x(1) xnew(1)],[x(2) xnew(2)],[f(x(1),x(2))
f(xnew(1),xnew(2))],'g*-')
    end
    if or(flag == 2 , flag == 3)
    plot3([x(1) xnew(1)],[x(2) xnew(2)],[f(x(1),x(2))
f(xnew(1),xnew(2))],'yx-')
    end
  x = xnew;
end
xopt = x;
fopt = f2(xopt);
niter = niter - 1;
fprintf('Minimum je nadjen na poziciji x = %.4f, y = %.4f',xopt(1),xopt(2))
fprintf('\nVrijednost funkcije u minimumu je %.4f', fopt)
fprintf('\nBroj iteracija %d',niter)
function g = grad(x)
g = [2*(10-x(1))-200*(10*x(2)*x(1)-x(1)^3) ; 1000*(10*x(2)-x(1)^2)];
%g = [2*x(1) + x(2);x(1) + 6*x(2)];
end
function h = hessian(x)
h=[600*x(1)^2-2000*x(2)-2 -2000*x(1);-2000*x(1) 10000];
%h=[2, 1;1 6];
end
```