Τεχνικές Βελτιστοποίησης

Μέθοδος μέγιστης καθόδου με προβολή

Θεόδωρος Κατζάλης
ΑΕΜ: 9282
katzalis@auth.gr

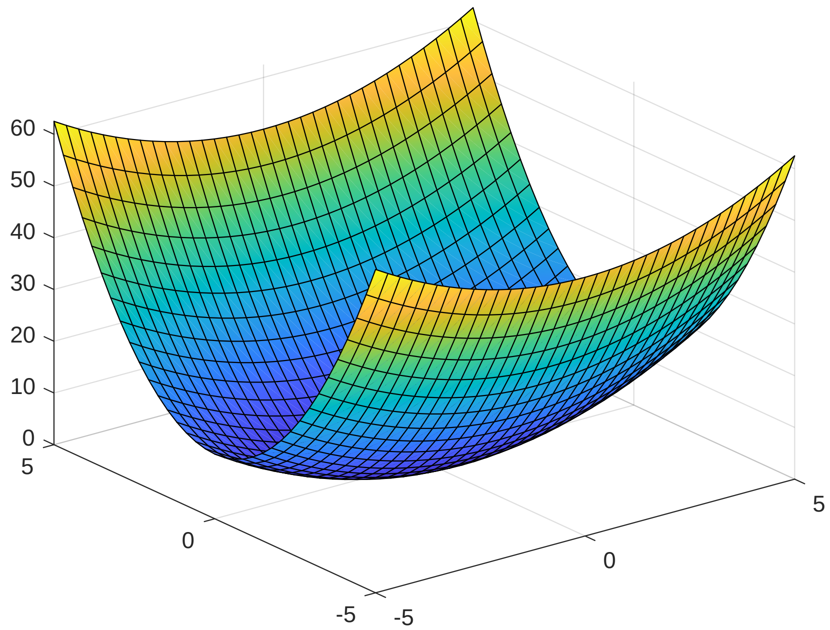
20/12/2021

Περιεχόμενα

Εισαγωγή

Σκοπός αυτής της εργασίας είναι η μελέτη της μεθόδου μέγιστης καθόδου, μιας τεχνικής βασισμένη στην χρήση παραγώγων για την επίλυση του προβλήματος βελτιστοποίησης της ελαχιστοποίησης μιας συνάρτησης. Η συνάρτηση που θα μελετήσουμε στη συνέχεια είναι η εξής:

$$f : \mathbb{R}^2 \to \mathbb{R}, \quad f(x_1, x_2) = \frac{1}{2}x_1^2 + 2x_2^2 \tag{1}$$



Εικόνα 1: Γραφική αναπαράσταση της f

Μελέτη της μεθόδου μέγιστης καθόδου

Η συγκεκριμένη μέθοδος είναι επαναληπτική και ο στόχος μας είναι να οδηγηθούμε στην εύρεση του σημείου που ελαχιστοποιεί την συνάρτηση. Για την μαθηματική ανάλυση, έχουμε:

$$x_{k+1} = x_k + \gamma_k d_k \tag{2}$$
$$d_k = -\nabla f(x_k) \tag{3}$$

όπου υπολογίζουμε ότι $\nabla f = \begin{bmatrix} x_{1k} \\ 4x_{2k} \end{bmatrix}$. Οπότε:

$$\begin{cases} x_{1k+1} = x_{1k} - \gamma_k x_k = x_{1k}(1 - \gamma_k) \\ x_{2k+1} = x_{2k} - \gamma_k 4x_{2k} = x_{2k}(1 - 4\gamma_k) \end{cases} \tag{4}$$

όπου $\gamma_k = \gamma, \forall k$. Για να έχουμε σύγκλιση πρέπει $\frac{|x_{k+1}|}{|x_k|} < 1$. Οπότε:

$$\begin{cases} \frac{|x_{1k+1}|}{|x_{1k}|} < 1 \Rightarrow \frac{|x_{1k}||1-\gamma_k|}{|x_{1k}|} \Rightarrow |1 - \gamma_k| < 1 \\ \\ \frac{|x_{2k+1}|}{|x_{2k}|} < 1 \Rightarrow \frac{|x_{2k}||1-4\gamma_k|}{|x_{2k}|} \Rightarrow |1 - 4\gamma_k| < 1 \end{cases}$$

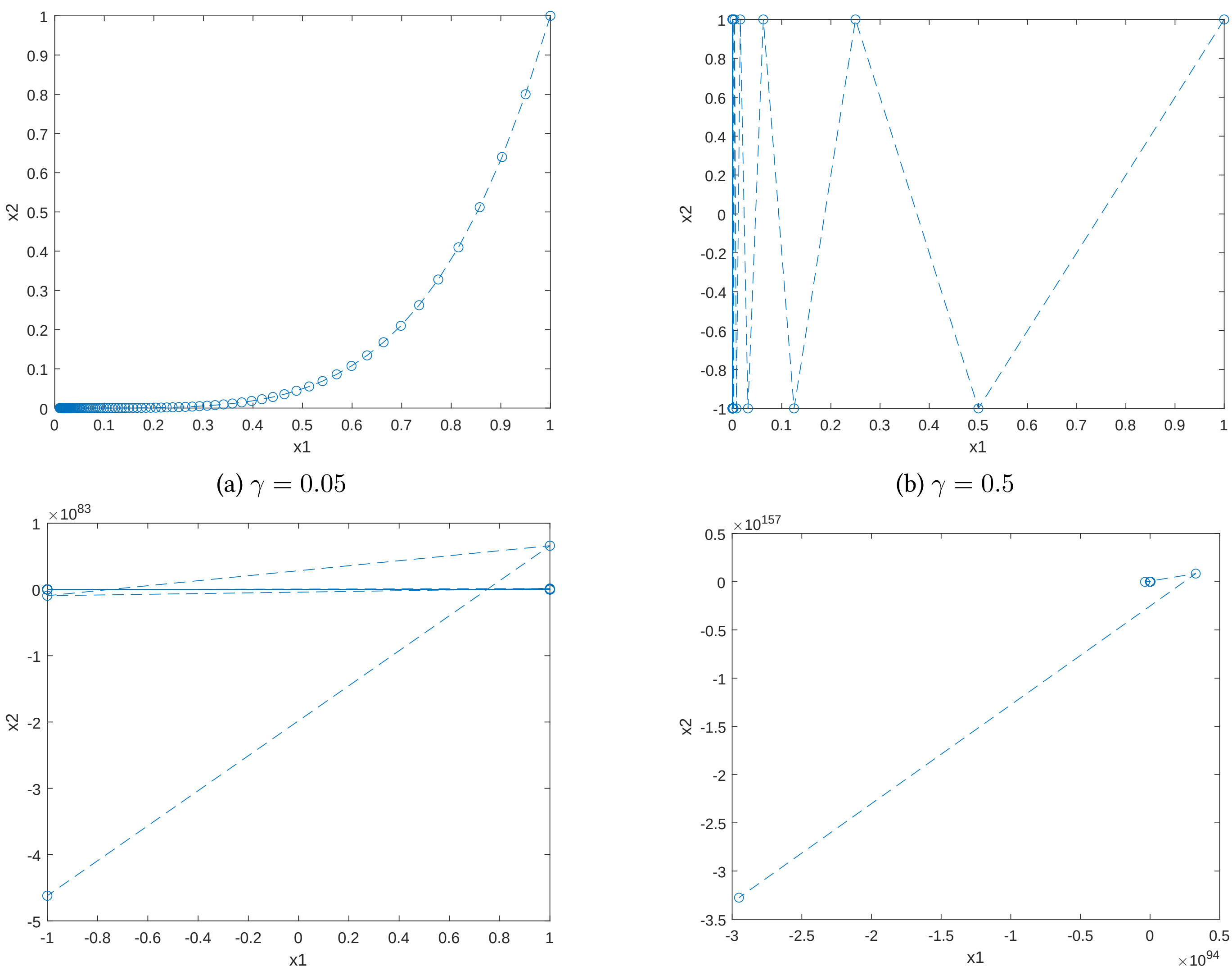
$$-1 < 1 - 4\gamma < 1 \Rightarrow 0 < \gamma < 0.5$$

Συνεπώς θα πρέπει $\gamma \in (0, 0.5)$ για να συγκλίνει η μέθοδος.

Προφανώς, αναλυτικά για το ελάχιστο σημείο το οποίο αναζητάμε, έχουμε:

$$f(x_1, x_2) = \frac{1}{2}x_1^2 + 2x_2^2 = 0 \Rightarrow x = (0, 0)$$

Υλοποιώντας την μέθοδο σε `Matlab` (ο κώδικας μπορεί να βρεθεί στο τέλος της αναφοράς) χωρίς περιορισμούς, για διαφορετικές τιμές γ, ακρίβεια $\epsilon = 0.01$ και για αρχικό σημείο αναφοράς $(1, 1)$ έχουμε:



(a) $\gamma = 0.05$

(b) $\gamma = 0.5$

(c) $\gamma = 2$

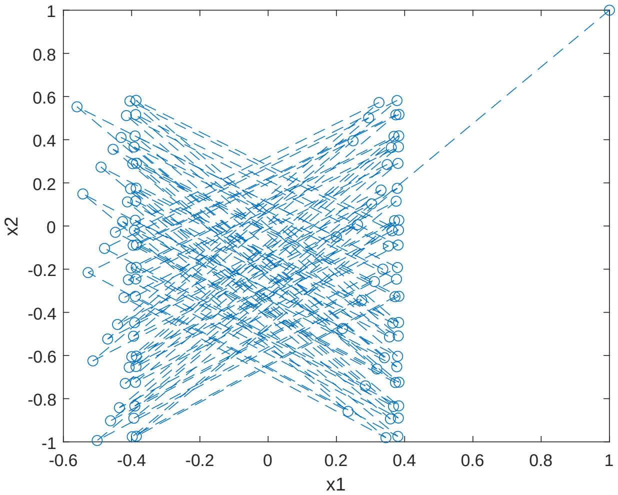
(d) $\gamma = 10$

Εικόνα 2

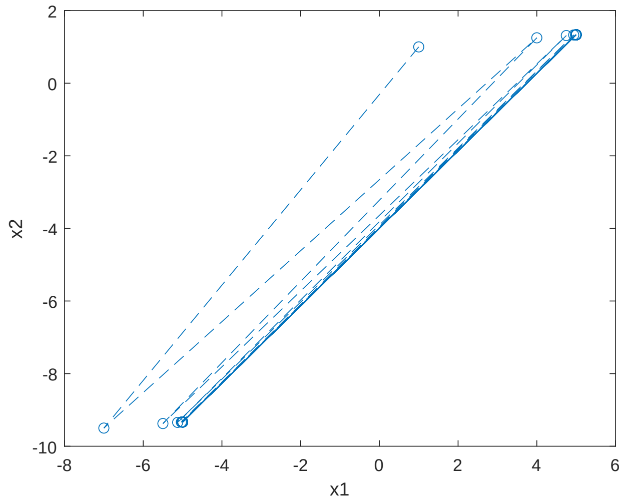
Επαληθεύοντας την θεωρία ο αλγόριθμος συγκλίνει για $\gamma = 0.05$ (< 0.5) σε 91 μόλις βήματα. Ενδιαφέρον παρουσιάζει η οριακή τιμή $\gamma = 0.5$, διότι βλέπουμε μια ταλάντωση γύρω απο το επιθυμητό σημείο $(0, 0)$. Ωστόσο, στις περιπτώσεις $\gamma = 2$ και $\gamma = 10$, έχουμε πλήρη αστάθεια και ο αλγόρθιμος για αύξηση της τιμής τείνει στο άπειρο.

Πραγματοποιώντας ακριβώς το ίδιο με την προηγούμενη φορά, χρησιμοποιώντας τον περιορισμό:

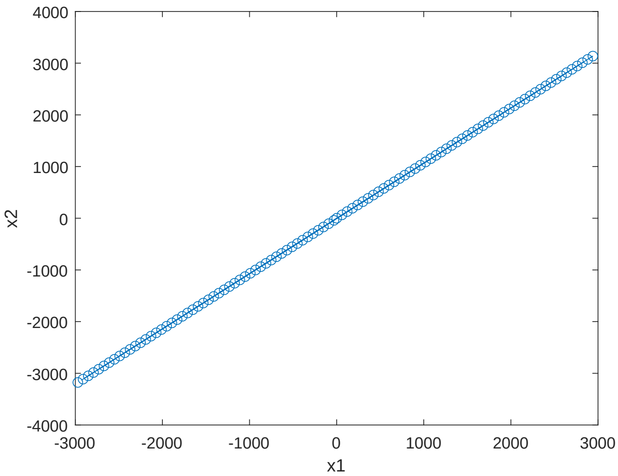
$$-15 <= x_1 <= 15$$
$$-20 <= x_2 <= 12$$



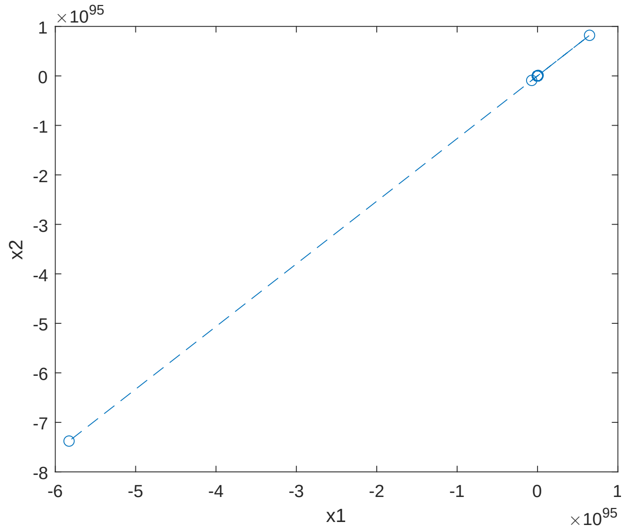
(a) $\gamma = 0.05$



(b) $\gamma = 0.5$



(c) $\gamma = 2$



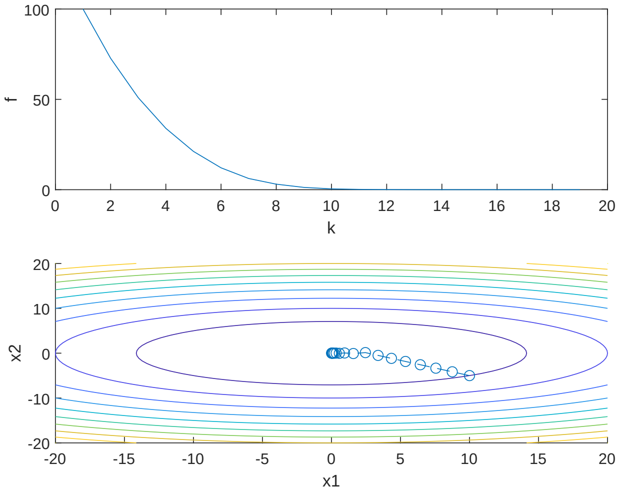
(d) $\gamma = 10$

Εικόνα 3

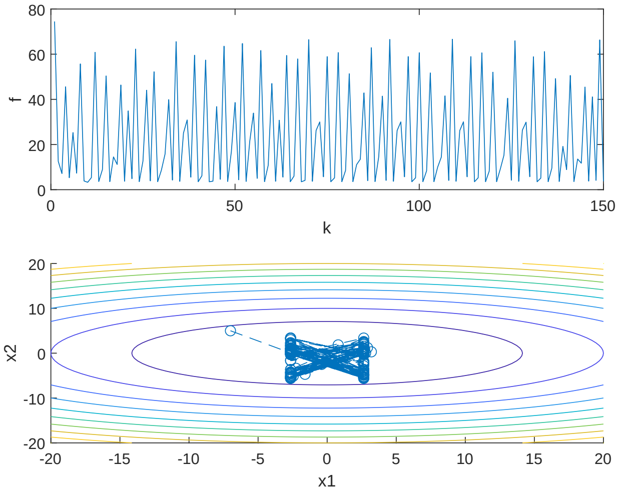
Για μια ακόμη φορά, μπροούμε να παρατηρήσουμε την αστάθεια για τιμές του γ εκτός των επιτρεπτών ορίων σύγκλισης καθώς και μια ταλάντωση γύρω απο μια συγκεκριμένη περιοχή για την τιμή $\gamma = 0.05$. Αξίζει ακόμη να σημειωθεί ότι μέχρι στιγμής προκειμένου να τερματίσει ο αλγόριθμος σε περίπτωση μη σύγκλισης χρησιμοποιήσαμε ως μέγιστο αριθμό επιτρεπτών επαναλήψεων το 100.

Μέθοδος με προβολή

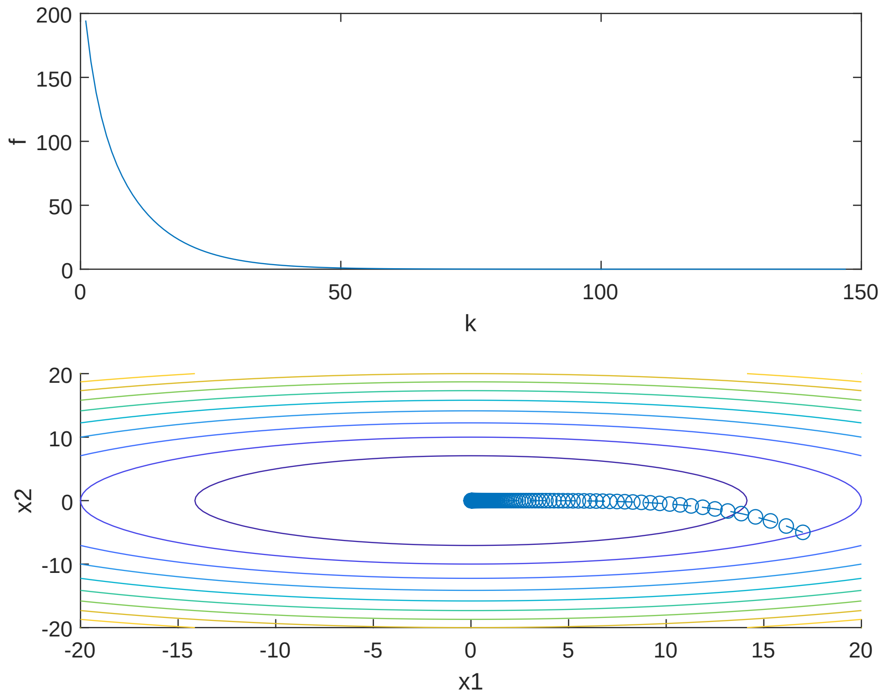
Στη συνέχεια, θα μελετήσουμε την μέθοδο μέγιστης καθόδου ενσωματώνοντας την έννοια της προβολή για διαφορετικές τιμές παραμέτρων. Για την μέθοδο της προβολής έχουμε για κάθε δοκιμή 2 διαγράμματα για να βλέπουμε την τιμή της συνάρτησης f συναρτήσει των αριθμών των επαναλήψεων k αλλά και την εξέλιξη των συνεταγμένων (x_1, x_2):



(a) $s_k = 8, \gamma_k = 0.05, x_0 = (10, -5), \epsilon = 0.01$



(b) $s_k = 10, \gamma_k = 0.3, x_0 = (-7, 5), \epsilon = 0.02$



(c) $s_k = 0.5, \gamma_k = 0.1, x_0 = (17, -5), \epsilon = 0.01$

Εικόνα 4

Στην πρώτη περίπτωση (α) ο αλγόριθμος συγκλίνει σε 19 επαναλήψεις και στην τρίτη (γ) για 147 σε αντίθεση με την δεύτερη (β) όπου παρατηρείται αστάθεια.

Η αστάθεια της δεύτερης προκύπτει απο τις υψηλές τιμές του s_k και του γ καθώς $s_k * \gamma > 0.5$. Ωστόσο παρόλο που το βήμα είναι αρκετά μεγάλο για να συγκλίνει, ο αλγόριθμος εγκλώβιζεται στους περιορισμούς που έχουμε θέσει λόγω προβολής. Στην 3η περίπτωση είναι η μοναδική δοκιμή όπου το σημείο αναφοράς βρίσκεται εκτός του συνόλου X που προσδιορίζεται απο τους περιορισμούς που θέσαμε προηγουμένως. Εξαιτίας της μεθόδου της προβολής αυτό το σημείο στην πρώτη επαναλήψη, θα οδηγηθεί εντός του X και επειδή το ολικό ελάχιστο ανήκει εντός του X αλλά και υπάρχει αρκετά μικρό βήμα, τότε γνωρίζουμε ότι ο αλγόριθμος θα συγκλίνει. Βέβαια αυτό συμβαίνει για ένα μεγαλύτερο αριθμό επαναλήψεων σε σχέση με την 1η περίπτωση αφού ξεκινάμε απο ένα πιο μακρινό σημείο αναφοράς. Αξίζει να σημειωθεί ότι σε αυτές τις δοκιμές θέσαμε ως μέγιστο αριθμό επαναλήψεων το 150.

Matlab κώδικας

Για όλα τα παραπάνω παρατίθεται ο κώδικας Matlab.

```
%%% STEEPEST DESCENT WITH PROJECTION
%clc
clear

syms x y
f(x,y) = 0.5 * x^2 + 2* y^2;
gradf = gradient(f,[x,y]);
gradf = matlabFunction(gradf);
gradfvect = @(x) gradf(x(1),x(2));
f = matlabFunction(f);
fvect = @(x) f(x(1),x(2));

%figure; fsurf(f);
%exportgraphics(gcf,'surf.pdf','ContentType','Vector')

% goal: minimum of f, which is 0
xMin = [0,0];
maxIter = 150;

%% EXERCISE 1
e = 0.01;
x0 = [1;1];
%gammas = [0.05,0.5,2,10];
gammas = [0.05];

for i=1:numel(gammas)
    figure
    xk = steepestDescent(maxIter,e,x0,gradfvect,gammas(i));
    plot(xk(:,1),xk(:,2),'--o')
    xlabel('x1')
    ylabel('x2')
    %exportgraphics(gcf,['surf_' int2str(gammas(i)) ...
        '.pdf'],'ContentType','Vector')
end

for i=1:numel(gammas)
    figure
    xk = steepestDescentCondition(maxIter,e,x0,gradfvect,gammas(i));
    plot(xk(:,1),xk(:,2),'--o')
    xlabel('x1')
    ylabel('x2')
    %exportgraphics(gcf,['surf_constraint_' int2str(gammas(i)) ...
        '.pdf'],'ContentType','Vector')
end

% %% EXERCISE 2
e = 0.01;
gamma = 0.05;
sk = 8;
x0 = [10;-5];
figure
xk = steepestDescentProjection(maxIter,e,sk,x0,gradfvect,gamma);
myPlot(xk,f,fvect)
%exportgraphics(gcf,'exe2.pdf','ContentType','Vector')

% %% EXERCISE 3
e = 0.02;
```

```
gamma = 0.3;
sk = 10;
x0 = [-7;5];
figure
xk = steepestDescentProjection(maxIter,e,sk,x0,gradfvect,gamma);
myPlot(xk,f,fvect)
%exportgraphics(gcf,'exe3.pdf','ContentType','Vector')


% %% EXERCISE 4
e = 0.01;
gamma = 0.1;
sk = 0.5;
x0 = [17;-5];
figure
xk = steepestDescentProjection(maxIter,e,sk,x0,gradfvect,gamma);
myPlot(xk,f,fvect)
exportgraphics(gcf,'exe4.pdf','ContentType','Vector')

function myPlot(xk,f,fvect)
    [k,dims] = size(xk);
    for i=1:k
        fvalues(i) = fvect(xk(i,:));
    end
    subplot(2,1,1)
    plot(1:k,fvalues)
    xlabel('k')
    ylabel('f')
    %subplot(2,2,2)
    %plot(xk(:,1),xk(:,2),'--o')
    subplot(2,1,2)
    hold on
    fcontour(f,[-20 20])
    plot(xk(:,1),xk(:,2),'--o')
    xlabel('x1')
    ylabel('x2')
end

function x = steepestDescent(maxIter,e,x0,gradfvect,gamma)
    k = 1;
    xk = x0;
    x(k,:) = xk;

    while norm(gradfvect(xk)) > e && k<maxIter
        xk = xk - gamma .* gradfvect(xk);
        k  = k + 1;
        x(k,:) = xk;
    end
    fprintf("Iters: %d/%d\n",k,maxIter)
end

function x = steepestDescentCondition(maxIter,e,x0,gradfvect,gamma)
    k = 1;
    xk = x0;
    x(k,:) = xk;
    while norm(gradfvect(xk)) > e && k<maxIter
        options = optimoptions('fmincon','Display','off');
        constraintPoint = fmincon(@(x)gradfvect(xk)' * ...
            ([x(1)-xk(1);x(2)-xk(2)]),[1,1],[],[],[],[],[-15,-20],[15,12],[],options);
        constraintPoint = constraintPoint';
        xk = xk + gamma .* (constraintPoint - xk);
        k = k + 1;
        x(k,:) = xk;
```

```
    end
    fprintf("Iters: %d/%d\n",k,maxIter)
end

function x = steepestDescentProjection(maxIter,e,sk,x0,gradfvect,gamma)
    k = 1;
    xk = x0;
    x(k,:) = xk;
    %while ¬isCritical(gradfvect,sk,xk) && k<maxIter
    while norm(gradfvect(xk))>e && k<maxIter
        xProjection = project(xk,sk,gradfvect,[-15;15],[-20;12]);
        xk = xk + gamma .* (xProjection - xk);
        k = k + 1;
        x(k,:) = xk;
    end
    fprintf("Iters: %d/%d\n",k,maxIter)

    function out = isCritical(gradfvect,s,x)
        xProjection = x - s * gradfvect(x);
        out = isequal(xProjection,x);
    end

    function xProjection = project(xk,sk,gradfvect,x1bounds,x2bounds)
        x1min = x1bounds(1);
        x1max = x1bounds(2);
        x2min = x2bounds(1);
        x2max = x2bounds(2);
        xProjection = xk - sk * gradfvect(xk);
        if xProjection(1) ≤ x1min
            xProjection(1) = x1min;
        end
        if xProjection(1) ≥ x1max
            xProjection(1) = x1max;
        end
        if xProjection(2) ≤ x2min
            xProjection(2) = x2min;
        end
        if xProjection(2) ≥ x2max
            xProjection(2) = x2max;
        end
    end
end
```