Τεχνικές Βελτιστοποίησης

Ελαχιστοποίηση με χρήση παραγώγων

Θεόδωρος Κατζάλης
ΑΕΜ: 9282
`katzalis@auth.gr`

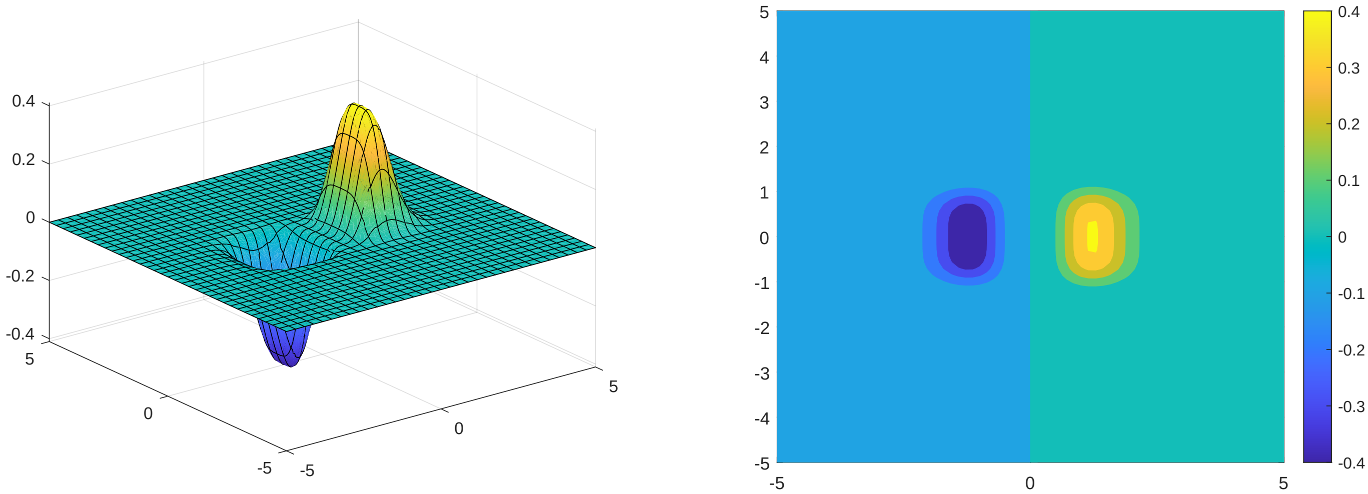
17/11/2021

Περιεχόμενα

Θέμα 1 - Εισαγωγή

Σκοπός αυτής της εργασίας είναι η μελέτη διάφορων μεθόδων ελαχιστοποίησης μιας δοσμένης συνάρτησης πολλών μεταβλητών $f : \mathbb{R}^n \to \mathbb{R}$ χωρίς περιορισμούς. Οι μέθοδοι που ακολουθούν είναι επαναληπτικοί. Για την αντικειμενική συνάρτηση έχουμε:

$$f(x, y) = x^3 e^{-x^2-y^4} \tag{1}$$



Εικόνα 1: Γραφική αναπαράσταση της f

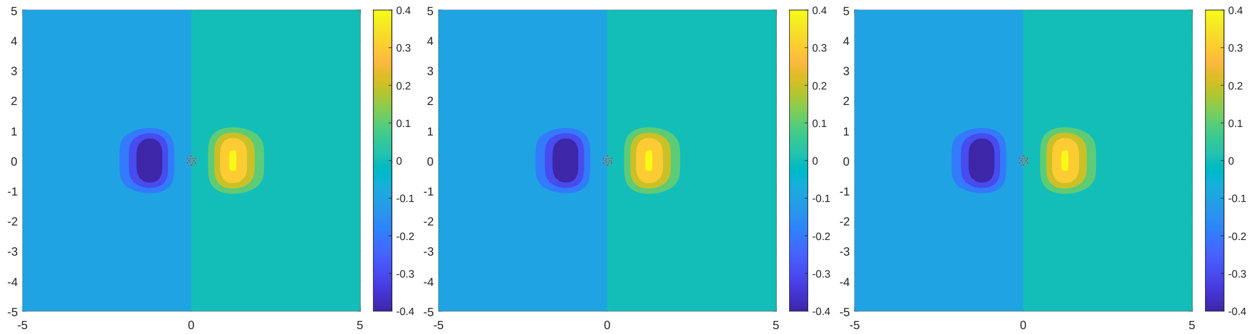
Τα αρχικά σημεία των αλγορίθμων είναι (0,0),(-1,-1) και (1,1).

Θέμα 2 - Μέθοδος Μέγιστης Καθόδου (Steepest Descent)

Η συγκεκριμένη μέθοδος περιγράφεται απο την σχέση: $x_{k+1} = x_k - \gamma_k \nabla f(x_k)$ και τερματίζεται όταν $|\nabla f(x_k)| < \epsilon$, όπου ϵ σταθερά της επιλογής μας. Στην συγκεκριμένη περίπτωση έχουμε $\epsilon = 10^{-6}$. Για να αποφύγουμε ατέρμονους βρόγχους σε περίπτωση που ο αλγόριθμος αποτύχει να εντοπίσει το επιθυμητό ελάχιστο σημείο, έχουμε μια επιπλέον συνθήκη για τον επιτρεπόμενο μέγιστο αριθμό επαναλήψεων (1000).

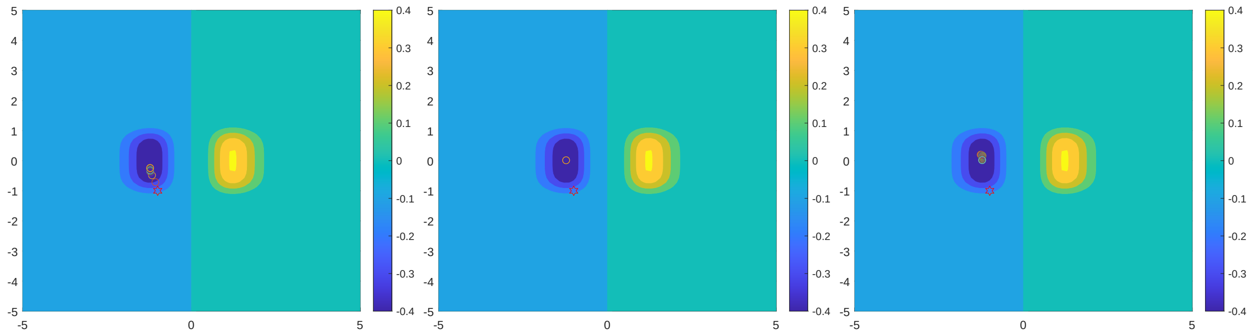
Η επιλογή της τιμής γ, καθορίζει το πόσο σύντομα ή μεγάλα θα είναι τα άλματα μεταξύ επαναλήψεων. Μικρή τιμή του γ σημαίνει πιο προσεκτική εξέλιξη των βημάτων άρα μπορούν να αποφευχθούν τυχόν αστάθειες και ταλαντώσεις γύρω απο το επιθυμητό σημείο. Ωστόσο, προκαλεί την αύξηση της πολυπλοκότητας του αλγορίθμου εξαιτίας των περισσότερων αριθμών επαναλήψεων. Απο την άλλη, μεγαλύτερο γ, προκαλεί μικρότερες επαναλήψεις αλλά υπάρχει η πιθανότητα αστάθειας.

Στη συνέχεια παραθέτουμε τρία διαφορετικά διαγράμματα για κάθε αρχικό σημείο (το αρχικό σημείο αναφοράς είναι το κόκκινο εξάγωνο). Αρχικά θα μελετήσουμε 3 διαφορετικές επιλογές του γ, α) σταθερό, β) τέτοιο ώστε ελαχιστοποιεί την τιμή $f(x_k + \gamma_k * (-\nabla f(x_k)))$ και γ) με τον κανόνα Armijo:



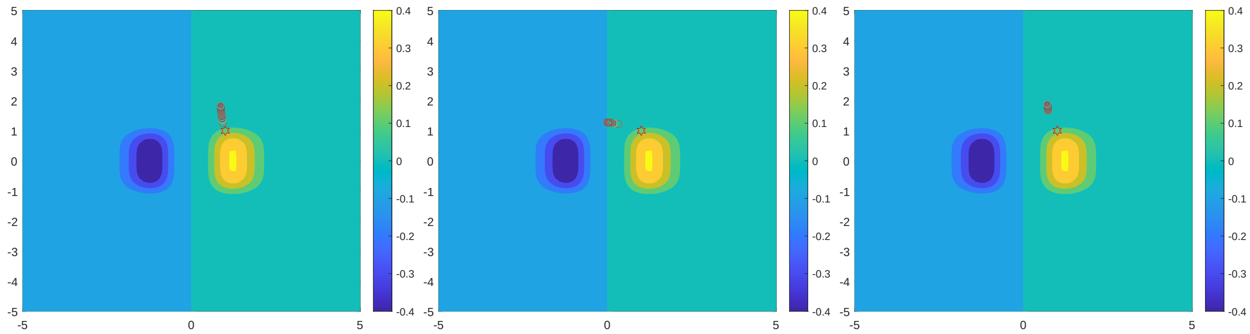
(a) k: 1000/1000, $\gamma = 0.5$ (b) k: 1000/1000, initialStep = 2 (c) k: 1000/1000

Εικόνα 2: (0,0)



(a) k: 10/1000, $\gamma = 0.5$ (b) k: 3/1000, initialStep = 2 (c) k: 1000/1000

Εικόνα 3: (-1,-1)



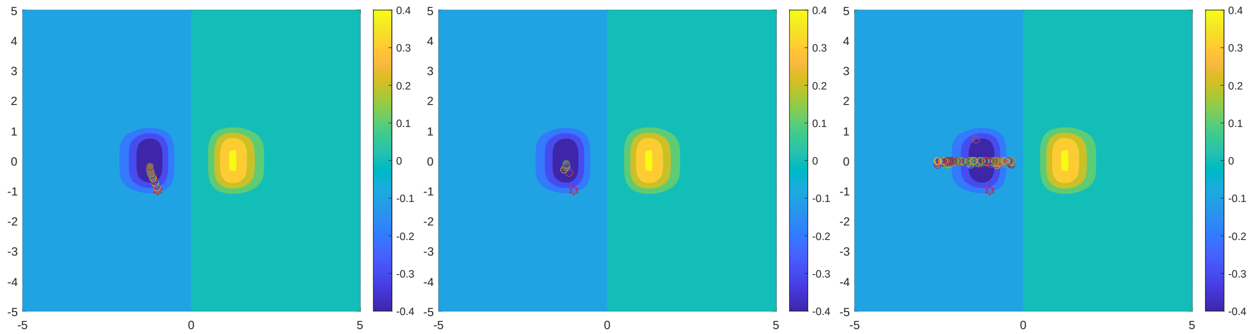
(a) k: 1000/1000, $\gamma = 0.5$ (b) k: 429/1000, initialStep = 2 (c) k: 1000/1000

Εικόνα 4: (1,1)

Παρατηρούμε ότι η επιλογή του αρχικού σημείου παίζει καθοριστικό ρόλο για την επιτυχία του αλγορίθμου. Αυτό είναι φυσιολογικό αφού η εξέλιξη των σημείων καθορίζεται απο το διάνυσμα $\nabla f(x)$ και την περιοχή στην οποία υπολογίζεται. Το σημείο (-1,-1) είναι το μόνο το οποίο θέτει τις κατάλληλες προδιαγραφές έτσι ώστε το διάνυσμα $\nabla f(x)$ να εντοπίσει το ολικό ελάχιστο.

Όσον αφορά το σημείο (-1,-1) θα συγκρίνουμε τις επιλογές του γ. Για την επιλογή του σταθερού σημείου γ παρατηρούμε ότι υπάρχει μια αργή αλλά σταδιακή εξέλιξη των τιμών. Για την ελαχιστοποίηση του $f(x_k + \gamma * (-\nabla f(x)))$ παρατηρούμε μια πιο αποτελεσματική προσέγγιση του ελάχιστης τιμής όπως και με τον κανόνα armijo, ο οποίος κάνει ένα μεγάλο άλμα αρχικά και μετά πολύ μικρά βήματα γύρω απο το επιθυμητό σημείο της ελάχιστης τιμής.

Στη συνέχεια για το σημείο (-1,-1) και για σταθερό γ, μπορούμε να δούμε πως διαφορετικές τιμές του γ επηρεάζουν τον αλγόριθμο:

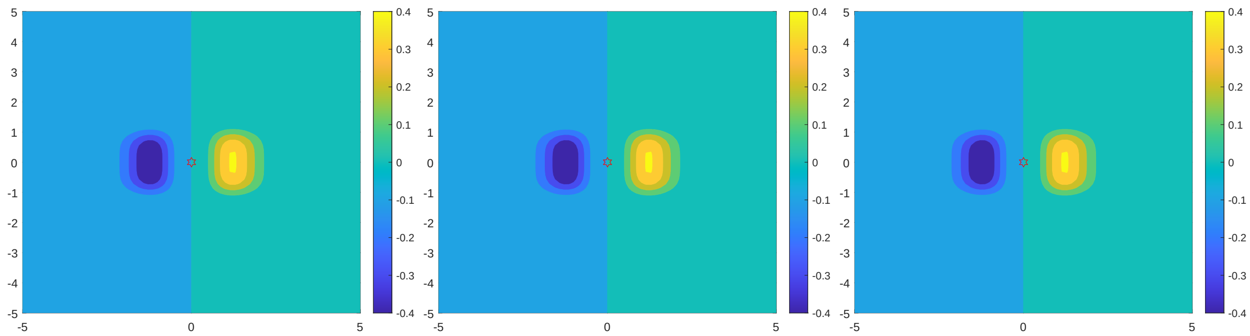


(a) k: 77/1000, $\gamma = 0.1$ (b) k: 29/1000, $\gamma = 1$ (c) k: 1000/1000, $gamma = 3$

Εικόνα 5: (1,1)

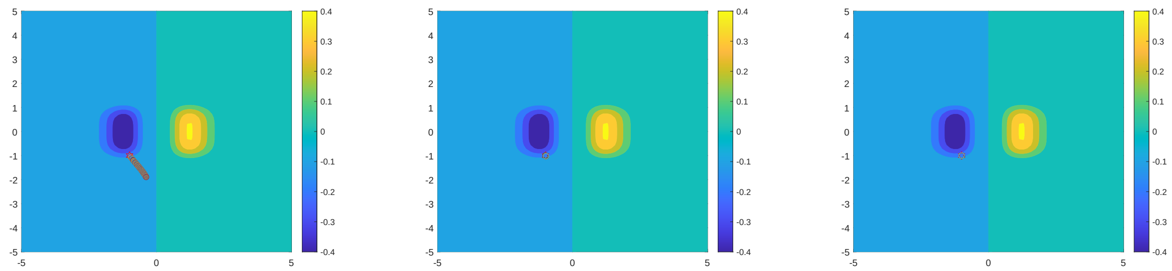
Θέμα 3 - Μέθοδος Newton

Για την μέθοδο Newton έχουμε:



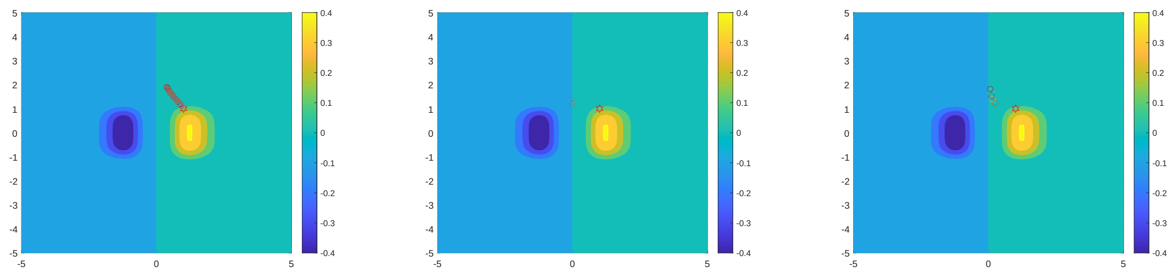
(a) k: 2/1000, $\gamma = 0.5$ (b) k: 2/1000, initialStep = 2 (c) k: 2/1000

Εικόνα 6: (0,0)



(a) k: 119/1000, $\gamma = 0.5$ (b) k: 1000/1000, initialStep = 2 (c) k: 1000/1000

Εικόνα 7: (-1,-1)



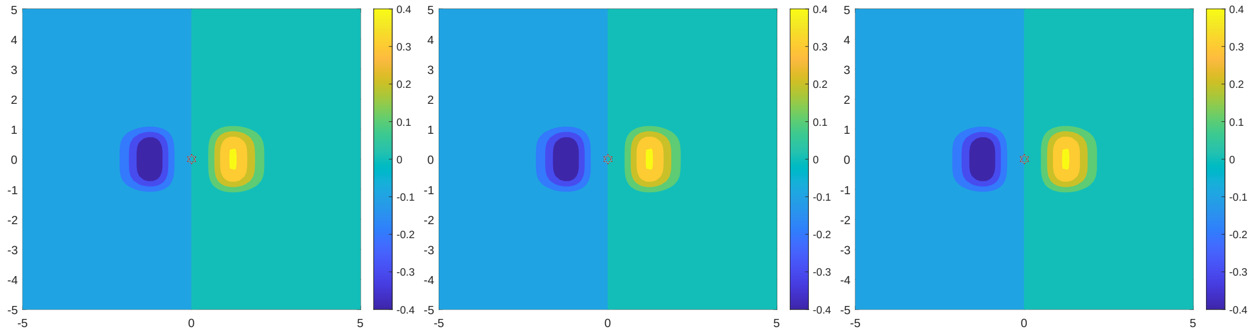
(a) k: 119/1000, $\gamma = 0.5$ (b) k: 1000/1000, initialStep = 2 (c) k: 7/1000

Εικόνα 8: (1,1)

Για την μέθοδο Newton παρατηρούμε ότι αποτυγχάνει να συγκλίνει προς την επιθυμητή τιμή.

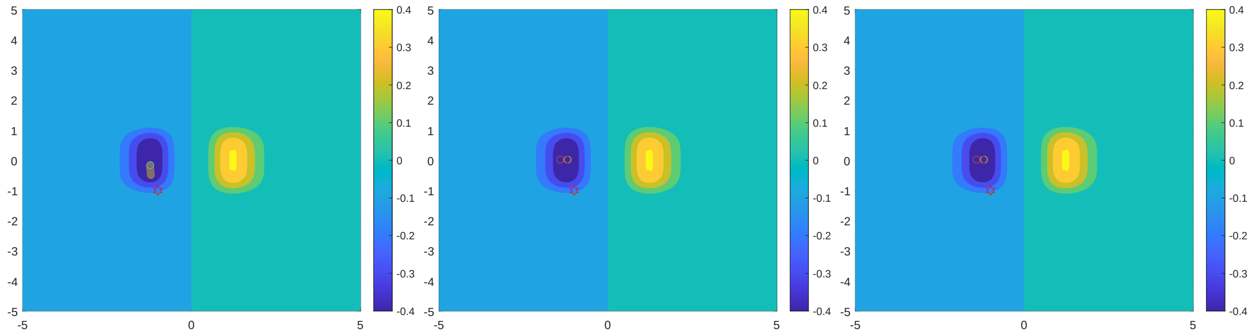
Θέμα 4 - Μέθοδος Levenberg-Marquardt

Για την μέθοδο Levenberg-Marquardt έχουμε:



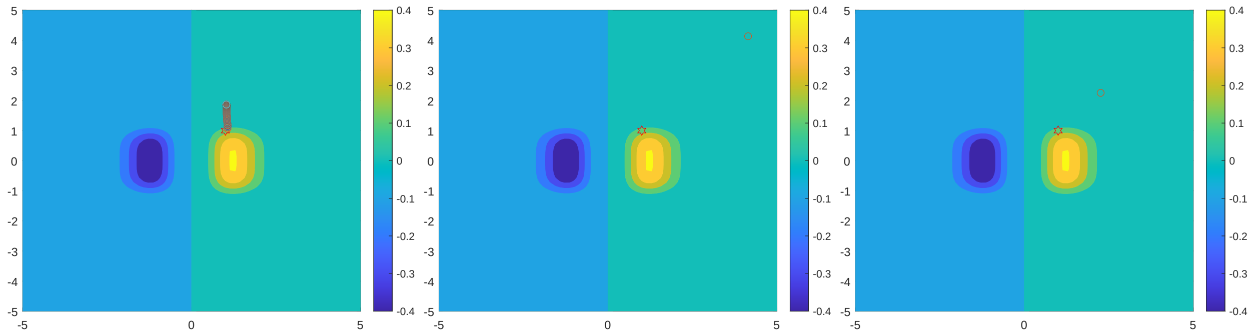
(a) k: 1000/1000, $\gamma = 0.5$ (b) k: 1000/1000, initialStep = 2 (c) k: 1000/1000

Εικόνα 9: (0,0)



(a) k: 209/1000, $\gamma = 0.5$ (b) k: 1000/1000, initialStep = 2 (c) k: 7/1000

Εικόνα 10: (-1,-1)



(a) k: 1000/1000, $\gamma = 0.5$ (b) k: 2/1000, initialStep = 2 (c) k: 2/1000

Εικόνα 11: (1,1)

Για την μέθοδο Levenberg-Marquardt παρατηρούμε ότι για αρχικό σημείο (-1,-1) ο αλγόριθμος συγκλίνει προς την επιθυμητή τιμή και μάλιστα με τον κανόνα armijo αυτό συμβαίνει σε μόλις 7 βήματα.

Matlab κώδικας

Για όλα τα παραπάνω παρατίθεται ο κώδικας Matlab.

```matlab
%%% MINIMIZATION USING DERIVATIVES
clear
clc

%% PLOT OBJECTIVE FUNCTION

f = @(x,y) x^3 * exp(-x^2-y^4);
%figure;fsurf(f)
%exportgraphics(gcf,'surf.pdf','ContentType','Vector')
%figure;fcontour(f,'Fill','On'); colorbar
%exportgraphics(gcf,'contour.pdf','ContentType','Vector')

initialConditions = [0,0 ; -1,-1 ; 1,1];
epsilon = 1e-6;
maxIterations = 1e3;

%% STEEPEST DECENT
[fig,k] = ...
    steepestDescent(maxIterations,epsilon,initialConditions(3,:),f,'stable',0.5);
exportgraphics(gcf,'steepestDescent_(1,1)_stable.pdf','ContentType','Vector')
[fig,k] = ...
    steepestDescent(maxIterations,epsilon,initialConditions(3,:),f,'min',0.5);
exportgraphics(gcf,'steepestDescent_(1,1)_min.pdf','ContentType','Vector')
[fig,k] = ...
    steepestDescent(maxIterations,epsilon,initialConditions(3,:),f,'armijo',0.5);
exportgraphics(gcf,'steepestDescent_(1,1)_armijo.pdf','ContentType','Vector')

%% NEWTON
[fig,k] = newton(maxIterations,epsilon,initialConditions(3,:),f,'stable',0.1);
exportgraphics(gcf,'newton_(1,1)_stable.pdf','ContentType','Vector')
[fig,k] = newton(maxIterations,epsilon,initialConditions(3,:),f,'min',0.1);
exportgraphics(gcf,'newton_(1,1)_min.pdf','ContentType','Vector')
[fig,k] = newton(maxIterations,epsilon,initialConditions(3,:),f,'armijo',0.1);
exportgraphics(gcf,'newton_(1,1)_armijo.pdf','ContentType','Vector')

%% Levenberg Marquardt
[fig,k] = levenMarq(maxIterations,epsilon,initialConditions(3,:),f,'stable',0.1);
exportgraphics(gcf,'leven_(1,1)_stable.pdf','ContentType','Vector')
[fig,k] = levenMarq(maxIterations,epsilon,initialConditions(3,:),f,'min',0.1);
exportgraphics(gcf,'leven_(1,1)_min.pdf','ContentType','Vector')
[fig,k] = levenMarq(maxIterations,epsilon,initialConditions(3,:),f,'armijo',0.1);
exportgraphics(gcf,'leven_(1,1)_armijo.pdf','ContentType','Vector')


% FUNCTIONS
function [fig,k] = ...
    steepestDescent(maxIterations,epsilon,x0,f,gammaFlag,gammaStable)
    fig = figure;fcontour(f,'Fill','on'); colorbar
    hold on

    x = sym('x');
    y = sym('y');
    fx = matlabFunction(diff(sym(f),x));
    fy = matlabFunction(diff(sym(f),y));

    xk = x0(1);
    yk = x0(2);
```

```matlab
    plot(xk,yk,'h','color','red','MarkerSize',8)
    %text(xk,yk,"x0",'Color','red','FontSize',14)
    xkNext = @(gamma,xk,yk) xk - gamma * fx(xk,yk);
    ykNext = @(gamma,xk,yk) yk - gamma * fy(xk,yk);
    k = 1;
    while norm(fx(xk,yk),fy(xk,yk)) > epsilon && k<maxIterations
        if gammaFlag == "stable"
            gamma = gammaStable;
        elseif gammaFlag == "min"
            fMin = @(gamma) f(xkNext(gamma,xk,yk),ykNext(gamma,xk,yk));
            gamma = fminbnd(fMin,0,5);
        elseif gammaFlag == "armijo"
            initStep = 2;
            [gamma,mk] = ...
                armijo(initStep,f,fx,fy,@(x,y)-fx(x,y),@(x,y)-fy(x,y),xk,yk,xkNext,ykNext)
        else
            disp("Invalid gamma flag")
            exit
        end
        %fprintf("%0.2f,%0.2f,gamma=%d\n",xk,yk,gamma)
        xk = xkNext(gamma,xk,yk);
        yk = ykNext(gamma,xk,yk);
        k  = k + 1;
        plot(xk,yk,'--o')
    end
    fprintf("Iters: %d/%d\n",k,maxIterations)
end

function [fig,k] = newton(maxIterations,epsilon,x0,f,gammaFlag,gammaStable)
    fig = figure;fcontour(f,'Fill','on'); colorbar
    hold on

    x = sym('x');
    y = sym('y');
    fx  = diff(sym(f),x);
    fy  = diff(sym(f),y);
    fxx = diff(sym(fx),x);
    fxy = diff(sym(fx),y);
    fyx = diff(sym(fy),x);
    fyy = diff(sym(fy),y);
    J = [fxx fxy ; fyx fyy];
    dk  = -inv(J) * [fx ; fy];
    dkX = matlabFunction(dk(1));
    dkY = matlabFunction(dk(2));
    fx  = matlabFunction(sym(fx));
    fy  = matlabFunction(sym(fy));
    xkNext = @(gamma,xk,yk) xk + gamma * dkX(xk,yk);
    ykNext = @(gamma,xk,yk) yk + gamma * dkY(xk,yk);


    xk = x0(1);
    yk = x0(2);
    plot(xk,yk,'h','color','red','MarkerSize',8)
    %text(xk,yk,"x0",'Color','red','FontSize',14)
    k = 1;
    while norm(fx(xk,yk),fy(xk,yk)) > epsilon && k<maxIterations
        if gammaFlag == "stable"
            gamma = gammaStable;
        elseif gammaFlag == "min"
            fMin = @(gamma) f(xkNext(gamma,xk,yk),ykNext(gamma,xk,yk));
            gamma = fminbnd(fMin,0,5);
        elseif gammaFlag == "armijo"
            initStep = 2;
```

```
                [gamma,mk] = armijo(initStep,f,fx,fy,dkX,dkY,xk,yk,xkNext,ykNext);
            else
                disp("Invalid gamma flag")
                exit
            end
            %fprintf("%0.2f,%0.2f,gamma=%d\n",xk,yk,gamma)
            xk = xkNext(gamma,xk,yk);
            yk = ykNext(gamma,xk,yk);
            k  = k + 1;
            plot(xk,yk,'--o')
        end
        fprintf("Iters: %d/%d\n",k,maxIterations)
end

function [fig,k] = levenMarq(maxIterations,epsilon,x0,f,gammaFlag,gammaStable)
        fig = figure;fcontour(f,'Fill','on'); colorbar
        hold on

        x = sym('x');
        y = sym('y');
        fx  = diff(sym(f),x);
        fy  = diff(sym(f),y);
        fxx = diff(sym(fx),x);
        fxy = diff(sym(fx),y);
        fyx = diff(sym(fy),x);
        fyy = diff(sym(fy),y);
        J = [fxx fxy ; fyx fyy];
        J = matlabFunction(sym(J));
        fx = matlabFunction(diff(sym(f),x));
        fy = matlabFunction(diff(sym(f),y));

        xk = x0(1);
        yk = x0(2);
        plot(xk,yk,'h','color','red','MarkerSize',8)
        %text(xk,yk,"x0",'Color','red','FontSize',14)
        k = 1;
        while norm(fx(xk,yk),fy(xk,yk)) > epsilon && k<maxIterations
            hessian = J(xk,yk);
            eigen = eig(hessian);
            m = abs(max(eigen))+0.1;
            d = - inv(hessian+m*eye(2)) * [fx(xk,yk) ; fy(xk,yk)];

            if gammaFlag == "stable"
                gamma = gammaStable;
            elseif gammaFlag == "min"
                fMin = @(gamma) f(xk+gamma*d(1),yk+gamma*d(2));
                gamma = fminbnd(fMin,0,5);
            elseif gammaFlag == "armijo"
                initStep = 2;
                [gamma,mk] = armijo(initStep,f,fx,fy,J,xk,yk);
            else
                disp("Invalid gamma flag")
                exit
            end

            %fprintf("%0.2f,%0.2f,gamma=%d\n",xk,yk,gamma)
            xk = xk + gamma * d(1);
            yk = yk + gamma * d(2);
            k  = k + 1;
            plot(xk,yk,'--o')
        end
        fprintf("Iters: %d/%d\n",k,maxIterations)

```

```
    % TODO: Refactoring. Concistency among armijo functions
    function [gamma,mk] = armijo(s,f,fx,fy,J,xk,yk)
        alpha = 1e-1;
        beta  = 0.1;
        mk = 0;
        gamma  = s*(beta^mk);

        hessian = J(xk,yk);
        eigen = eig(hessian);
        m = abs(max(eigen))+0.1;
        d = - inv(hessian+m*eye(2)) * [fx(xk,yk) ; fy(xk,yk)];
        xkN = xk + gamma * d(1);
        ykN = yk + gamma * d(2);

        D = fx(xk,yk)*d(1) + fy(xk,yk)*d(2);
        while (f(xk,yk) - f(xkN,ykN)) < -alpha*gamma*D
            mk = mk + 1;
            gamma = s*(beta^mk);

            hessian = J(xk,yk);
            eigen = eig(hessian);
            m = abs(max(eigen))+0.1;
            d = - inv(hessian+m*eye(2)) * [fx(xk,yk) ; fy(xk,yk)];
            xkN = xk + gamma * d(1);
            ykN = yk + gamma * d(2);
        end
    end
end

function [gamma,mk] = armijo(s,f,fx,fy,dkX,dkY,xk,yk,xkNext,ykNext)
    alpha = 1e-1;
    beta  = 0.1;
    mk = 0;
    gamma  = s*(beta^mk);
    xkN = xkNext(gamma,xk,yk);
    ykN = ykNext(gamma,xk,yk);
    D = fx(xk,yk)*dkX(xk,yk) + fy(xk,yk)*dkY(xk,yk);
    while (f(xk,yk) - f(xkN,ykN)) < -alpha*gamma*D
        mk = mk + 1;
        gamma = s*(beta^mk);
        xkN = xkNext(gamma,xk,yk);
        ykN = ykNext(gamma,xk,yk);
    end
end
```