Aditya Vipradas
1209435588

Design optimization.

①

09/02/2016. MAE 598/494 Homework 2.

1. maximize $f = x_1 - x_2$
i.e. minimize $f = x_2 - x_1$
subject to: $g_1 = 2x_1 + 3x_2 - 10 \leq 0$;
$g_2 = -5x_1 - 2x_2 + 2 \leq 0$
$g_3 = -2x_1 + 7x_2 - 8 \leq 0$.

$\frac{\partial f}{\partial x_1} = -1$, $\frac{\partial f}{\partial x_2} = 1$, $\frac{\partial g_1}{\partial x_1} = 2$, $\frac{\partial g_1}{\partial x_2} = 3$

$\frac{\partial g_2}{\partial x_1} = -5$, $\frac{\partial g_2}{\partial x_2} = -2$, $\frac{\partial g_3}{\partial x_1} = -2$, $\frac{\partial g_3}{\partial x_2} = 7$

Creating the ~~moto~~ monotonicity principle 1 table,

	x_1	x_2
f	−	+
g_1	+	+
g_2	−	−
g_3	−	+

It is observed that, g_1 is an active constraint for x_1 and g_2 is an active constraint for x_2. g_3 is a redundant constraint.

As g_1 is active for x_1, from g_1,
$$2x_1 + 3x_2 - 10 = 0$$
$$\therefore 2x_1 = 10 - 3x_2$$
$$x_1 = \frac{10}{2} - \frac{3x_2}{2} = 5 - \frac{3}{2}x_2.$$
Substitute in f and g_2.

$$\therefore f = x_2 - 5 + \frac{3}{2} x_2$$

$$= \frac{5}{2} x_2 - 5.$$

$$g_2 = -5\left(5 - \frac{3}{2} x_2\right) - 2x_2 + 2 \leq 0.$$

$$\therefore g_2 = -25 + \frac{15}{2} x_2 - 2x_2 + 2 \leq 0$$

$$\therefore g_2 = \frac{11}{2} x_2 - 23 \leq 0.$$

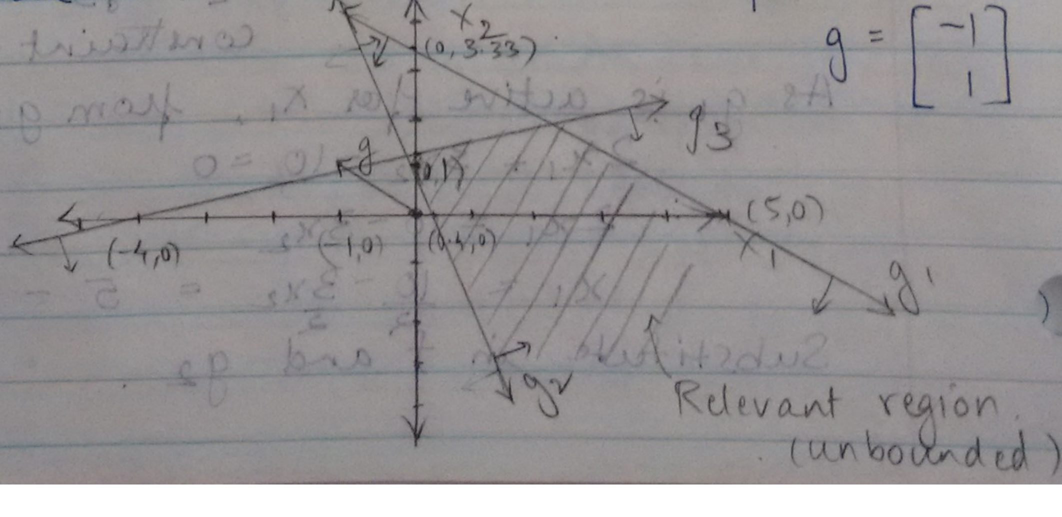
Apply monotonicity principle 1 again,

	x_2
f	+
g_2	+

$$\frac{\partial f}{\partial x_2} = \frac{5}{2}, \quad \frac{\partial g_2}{\partial x_2} = \frac{11}{2}$$

Thus, x_2 has no active constraint. Hence, there is no solution to this problem.

$$g = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$



As observed from the monotonicity principle 1, & the drawn figure, the optimization problem has no solution as it is not well constrained. The problem is thus not well constrained.

2) x_i, $i = 1, 2, 3$.

$\therefore$ $x_1 > 0$, $x_2 > 0$ & $x_3 > 0$

i.e. $-x_1 < 0$, $-x_2 < 0$ & $-x_3 < 0$.

maximize: x_1

i.e. minimize: $-x_1$

subject to: $g_1 = \exp(x_1) \leq x_2$; i.e. $e^{x_1} - x_2 \leq 0$,

$g_2 = \exp(x_2) \leq x_3$; i.e. $e^{x_2} - x_3 \leq 0$,

$g_3 = x_3 \leq 10$; i.e. $x_3 - 10 \leq 0$.

$g_4 = -x_1 < 0$

$g_5 = -x_2 < 0$

$g_6 = -x_3 < 0$.

Creating the monotonicity principle 1 table,

	x_1	x_2	x_3
f	$-$		
g_1	$+$	$-$	
g_2		$+$	$-$
g_3			$+$
g_4	$-$		
g_5		$-$	
g_6			$-$

As seen, g_1 is the active constraint for x_1:

$e^{x_1} - x_2 = 0$

$\therefore e^{x_1} = x_2$ — (1)

$x_1 = \ln x_2$

According to monotonicity principle 2,
x_2 is bounded by g_1 (below) and g_2 (above)
and x_3 is bounded by g_2 (below) and g_3 (above).

$\therefore$ i.e. $\min f: -\ln x_2$ (express ~~x_1~~ f in x_2 terms)

$g_2 = e^{x_2} - x_3 \leq 0$

	x_2
f	$-$
g_2	$+$

g_2 is active constraint for x_2.

$\therefore e^{x_2} - x_3 = 0$

$\therefore e^{x_2} = x_3$ — ②

$\therefore x_2 = \ln x_3$ (express f in x_3 terms)

i.e. $\min f = -\ln(\ln x_3)$. $\quad \frac{\partial f}{\partial x_3} = -\frac{1}{\ln x_3} \times \frac{1}{x_3}$.

	x_3
f	$-$
g_3	$+$

$\therefore$

g_3 is active constraint for x_3.

$\therefore x_3 = 10$. — ③

Back substitute in ② & ①,

$\therefore e^{x_2} = 10 \quad \therefore x_2 = 2.303$.

& $e^{x_1} = x_2 \quad \therefore e^{x_1} = 2.303$

$\therefore x_1 = 0.834$

$\therefore$ The solution is $\underline{\{x_1, x_2, x_3\} = \{0.834, 2.303, 10\}}$

3) $f = 2x_1^2 - 4x_1x_2 + 1.5x_2^2 + x_2$

$$g = \begin{bmatrix} \partial f/\partial x_1 \\ \partial f/\partial x_2 \end{bmatrix} = \begin{bmatrix} 4x_1 - 4x_2 \\ -4x_1 + 3x_2 + 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \begin{array}{l} -① \\ -② \end{array}$$

Solve ① & ②.

$\therefore$ the stationary point $[x_1^*, x_2^*] = [1, 1]$

$$H = \begin{bmatrix} \partial^2 f/\partial x_1^2 & \partial^2 f/\partial x_1 \partial x_2 \\ \partial^2 f/\partial x_2 \partial x_1 & \partial^2 f/\partial x_2^2 \end{bmatrix}$$

$$= \begin{bmatrix} 4 & -4 \\ -4 & 3 \end{bmatrix}$$

$|H| = 12 - 16 = -4$ $\qquad |H| < 0$.

$|H| = \lambda_1 \lambda_2$ (λ_1, λ_2 are eigenvalues of H)

$(\lambda_1 \lambda_2) < 0$.

$\therefore \lambda_1 > 0$ & $\lambda_2 < 0$.

Therefore, the function has positive curvature (for λ_1) in one direction and a negative curvature (for λ_2) in another direction. Hence, the stationary point (1,1) is a saddle.

Using Taylor's expansion,

$$f(x) = f(x^*) + g^T_{x^*}\,\partial x + \frac{1}{2}\,\partial x^T H_{x^*}\,\partial x$$

For $x = x^*$,

$$f(x) - f(x^*) = g^T_{x^*}\,\partial x + \frac{1}{2}\,\partial x^T H_{x^*}\,\partial x = 0.$$

At x^*, $g^T = [0 \;\; 0]$.

$\therefore \quad \partial x^T H_{x^*}\,\partial x = 0.$ $\quad \cancel{\partial x} =$

$$\therefore [\partial x_1 \;\; \partial x_2] \begin{bmatrix} 4 & -4 \\ -4 & 3 \end{bmatrix} \begin{bmatrix} \partial x_1 \\ \partial x_2 \end{bmatrix} = 0.$$

$$\therefore \begin{bmatrix} 4\partial x_1 - 4\partial x_2 & -4\partial x_1 + 3\partial x_2 \end{bmatrix} \begin{bmatrix} \partial x_1 \\ \partial x_2 \end{bmatrix} = 0$$

$$\therefore 4\partial x_1^2 - 4\partial x_2\partial x_1 - 4\partial x_1\partial x_2 + 3\partial x_2^2 = 0$$

$$\therefore 4\partial x_1^2 - 8\partial x_1\partial x_2 + 3\partial x_2^2 = 0.$$

$$4\partial x_1^2 - 6\partial x_1\partial x_2 - 2\partial x_1\partial x_2 + 3\partial x_2^2 = 0$$

$$\therefore 2\partial x_1(2\partial x_1 - 3\partial x_2) - \partial x_2(2\partial x_1 - 3\partial x_2) = 0$$

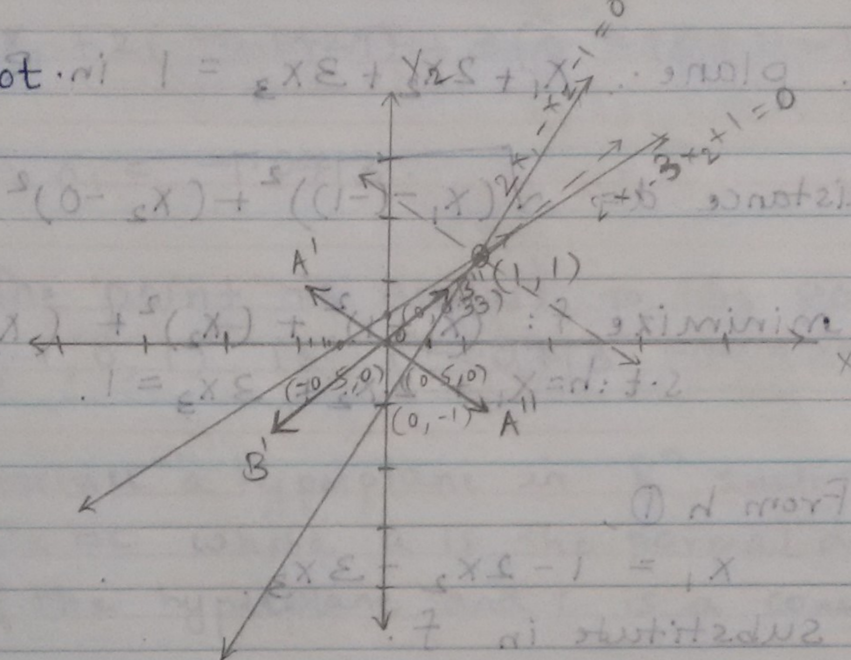
$$\therefore 2\partial x_1 - 3\partial x_2 = 0 \qquad \partial x_1 = x_1 - 1$$

$$2\partial x_1 - \partial x_2 = 0. \qquad \partial x_2 = x_2 - 1.$$

$$\therefore 2(x_1 - 1) - 3(x_2 - 1) = 0. \qquad \therefore \underline{2x_1 - 3x_2 + 1 = 0}$$

$$2(x_1 - 1) - (x_2 - 1) = 0 \qquad \therefore \underline{2x_1 - x_2 - 1 = 0}$$

Plot



Finding eigenvalues and eigen vectors of

$$H = \begin{bmatrix} 4 & -4 \\ -4 & 3 \end{bmatrix}$$

$$\lambda_1 = 7.5311, \quad \begin{bmatrix} x_1^1 \\ x_2^1 \end{bmatrix} = \begin{bmatrix} -0.7497 \\ 0.6618 \end{bmatrix}.$$

$$\lambda_2 = -0.5311, \quad \begin{bmatrix} x_1^2 \\ x_2^2 \end{bmatrix} = \begin{bmatrix} -0.6618 \\ -0.7497 \end{bmatrix}.$$

The eigenvector B'OB'' represents the direction that reduces f as the curvature along this vector is negative (-0.5311). Thus, the function downslopes along B'OB'' direction i.e. $[-0.6618 \ \ -0.7497]$ eigenvector away from the saddle $[1, 1]$.

(4) a). plane : $x_1 + 2x_2 + 3x_3 = 1$ in $\mathbb{R}^3$.

distance $d = \sqrt{(x_1-(-1))^2 + (x_2-0)^2 + (x_3-1)^2}$

minimize $f: (x_1+1)^2 + (x_2)^2 + (x_3-1)^2$.

s.t: $h = x_1 + 2x_2 + 3x_3 = 1$. — ①

From h ①,

$x_1 = 1 - 2x_2 - 3x_3$.

substitute in f.

$\therefore$ min $f: [1 - 2x_2 - 3x_3 + 1]^2 + x_2^2 + (x_3-1)^2$.

$\therefore f: (2 - 2x_2 - 3x_3)^2 + x_2^2 + (x_3-1)^2$.

Thus, the problem is converted to an unconstrained problem.

$$g = \begin{bmatrix} \partial f/\partial x_2 \\ \partial f/\partial x_3 \end{bmatrix} = \begin{bmatrix} 2(2-2x_2-3x_3)(-2) + 2x_2 \\ 2(2-2x_2-3x_3)(-3) + 2(x_3-1) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$\therefore -8 + 8x_2 + 12x_3 + 2x_2 = 0$

$\therefore 10x_2 + 12x_3 = 8$ — ①.

$-12 + 12x_2 + 18x_3 + 2x_3 - 2 = 0$

$\therefore 12x_2 + \cancel{18}\,20\,x_3 = 14$. — ②

$\therefore x_2 = -0.1429$, $x_3 = 0.7857$.

substitute in plane equation

$x_1 + 2(-0.1429) + 3(0.7857) = 1$.

$\therefore x_1 = -1.0713$.

$\therefore$ The point nearest to the point $(-1, 0, 1)^T$ is $\underline{(-1.0713, -0.1429, 0.7857)}$.

5). Consider a hyperplane in R^n such that $a^T x = c$ where a is the normal direction of the hyperplane and c is a constant.

Now, consider two points in hyperplane x_1 & x_2. They lie in that hyperplane.
Then, $\alpha x_2 + (1-\alpha) x_1$ is any point on the line joining x_1 & x_2 such that $0 \leq \alpha \leq 1$.

$\therefore a^T x_1 = c$ & $a^T x_2 = c$

$$a^T\left[\alpha x_2 + (1-\alpha) x_1\right] = \alpha a^T x_2 + (1-\alpha) a^T x_1$$
$$= \alpha c + (1-\alpha) c$$
$$= \underline{c}$$

$\therefore$ All the points on the line joining x_1 & x_2 lie in the hyperplane $a^T x = c$ as they fulfill the criteria.
According to the definition of convex set, a set $S \subseteq R^n$ is convex if, for every point x_1, x_2 in S, the point $x(\lambda) = \lambda x^2 + (1-\lambda) x_1$, $0 \leq \lambda \leq 1$ belongs to the set.

Therefore, ~~it cont~~ a hyperplane $a^T x = c$ is a convex set.

Problem 4(b)

Aditya Vipradas
ASU ID: 1209435588

February 9, 2016

The gradient descent and Newton's methods are employed to find the minimum of the given convex function. Both the methods are complemented with the Armijo line search with t = 0.1, b = 0.55. The methods are tested for two guess values. Let us assume the first guess value to be [-1, 2]. The gradient descent with Armijo line search takes 64 iterations to arrive at the minimum for x_2 and x_3 as (-0.14289, 0.78573) whereas the Newton's method completes the search in 1 iteration as shown below and gives the minimum at (-0.14286, 0.78571). The epsilon i.e. gradient norm criterion is kept at 1e-4.

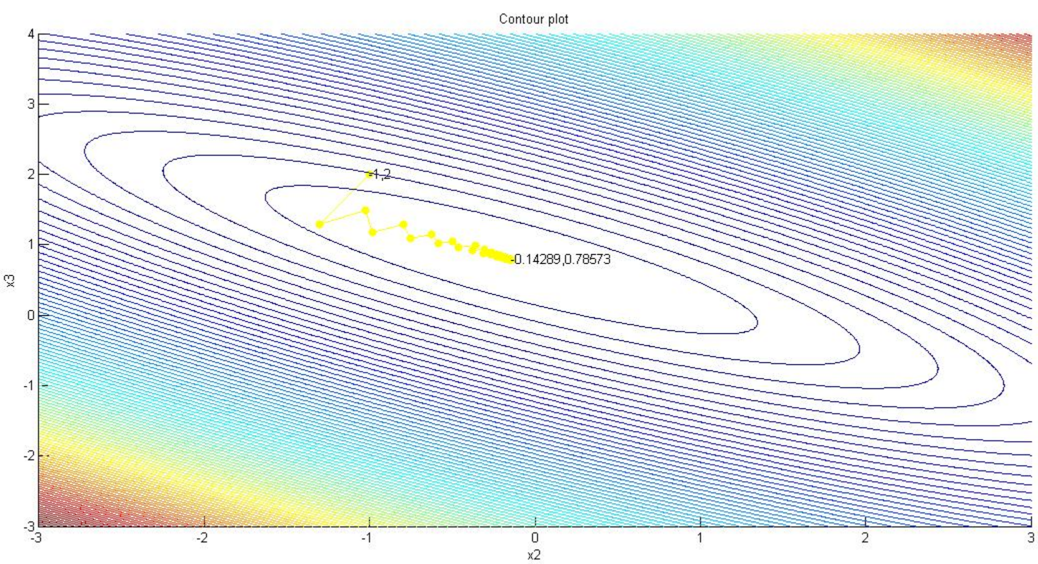


Figure 1: Gradient descent with Armijo search for guess [-1, 2]

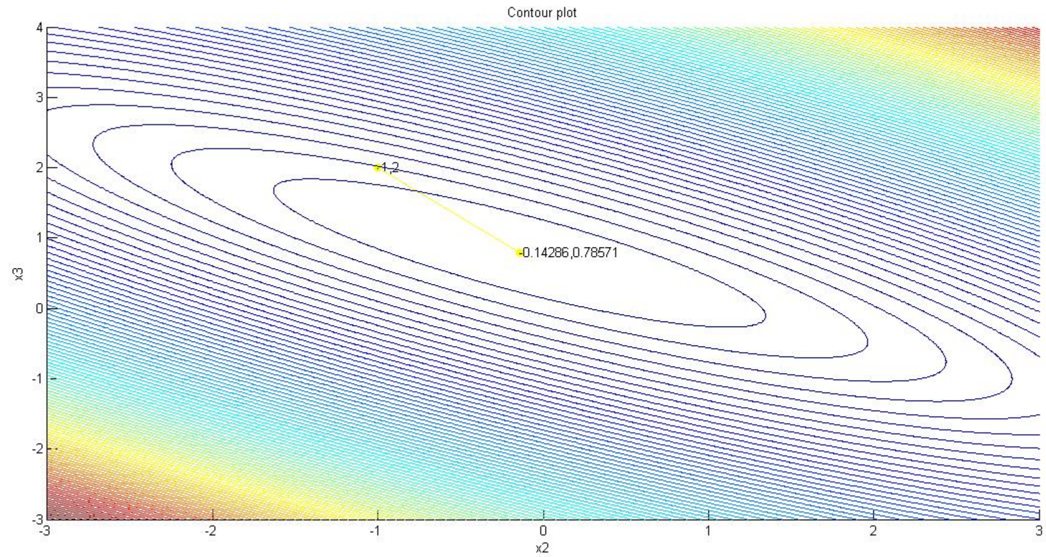


Figure 2: Newton's method with Armijo search for guess [-1, 2]

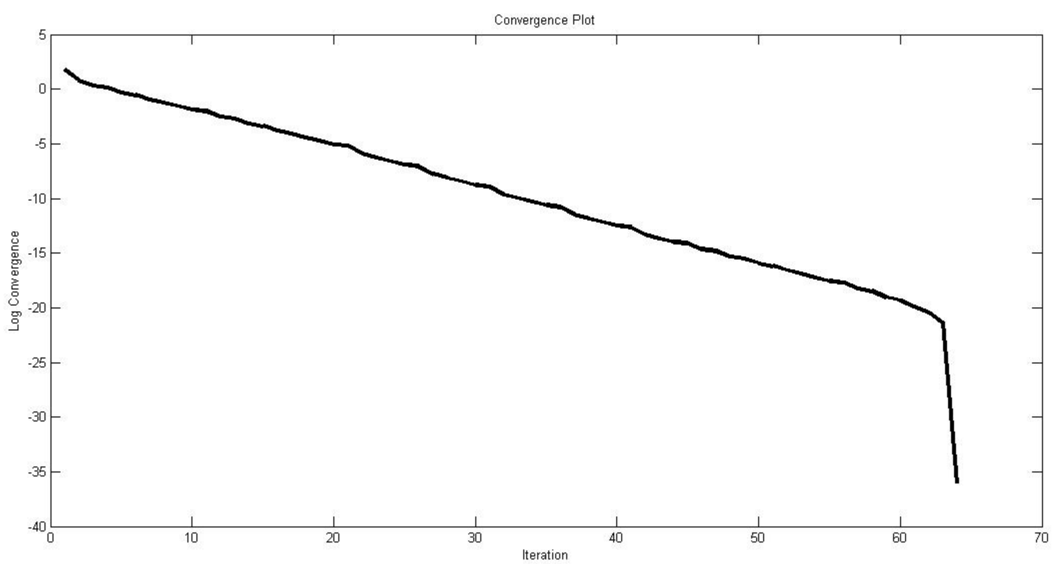


Figure 3: Gradient descent convergence for guess [-1, 2]

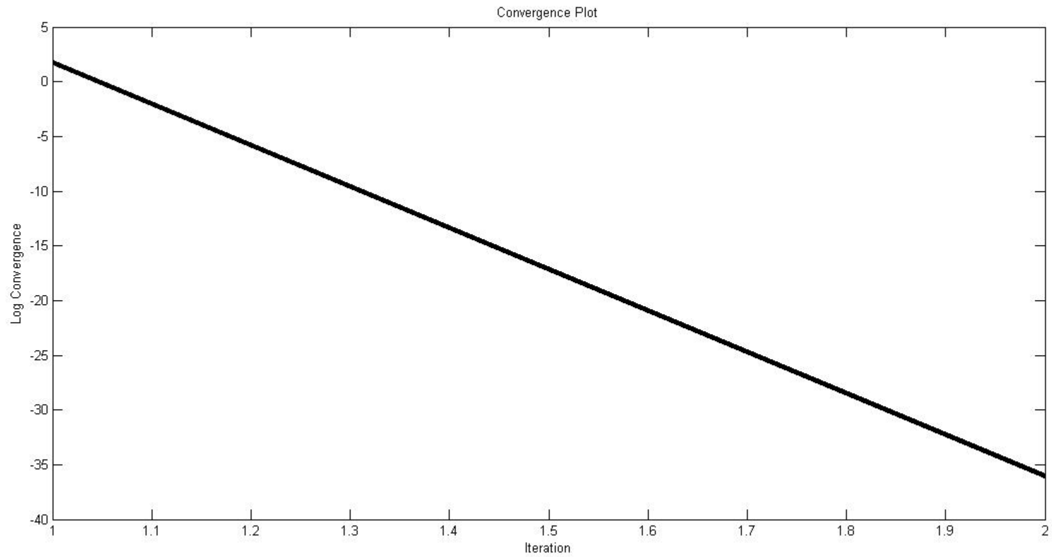


Figure 4: Newton's method convergence for guess [-1, 2]

The second guess value considered is [2, 3]. The gradient descent with Armijo line search takes 59 iterations to arrive at the minimum for x_2 and x_3 as (-0.14283, 0.7857) whereas the Newton's method completes the search in 1 iteration as shown below and gives the minimum at (-0.14286, 0.78571). The epsilon i.e. gradient norm criterion is kept at 1e-4.

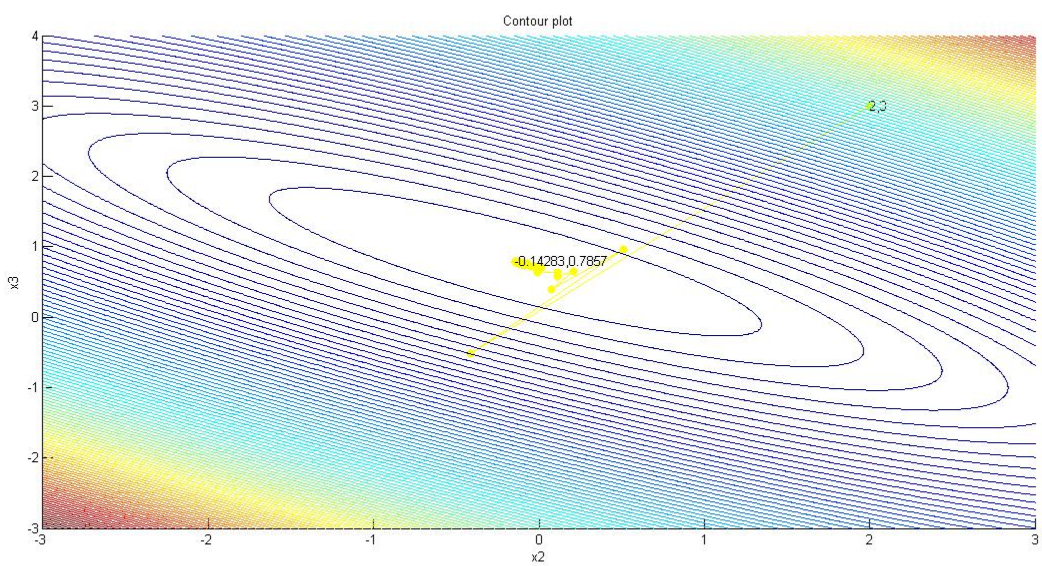


Figure 5: Gradient descent with Armijo search for guess [2, 3]

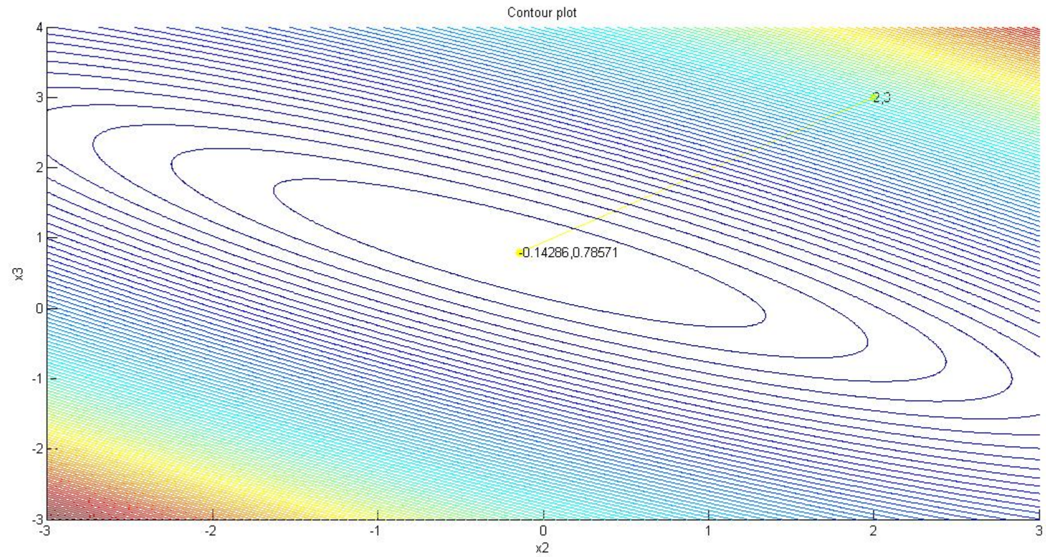


Figure 6: Newton's method with Armijo search for guess [2, 3]

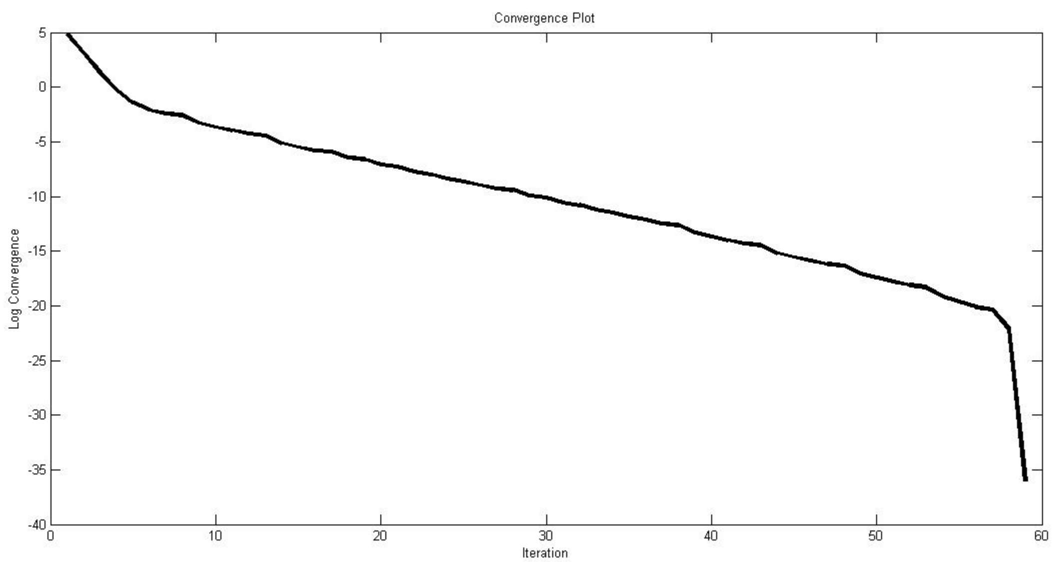


Figure 7: Gradient descent convergence for guess [2, 3]

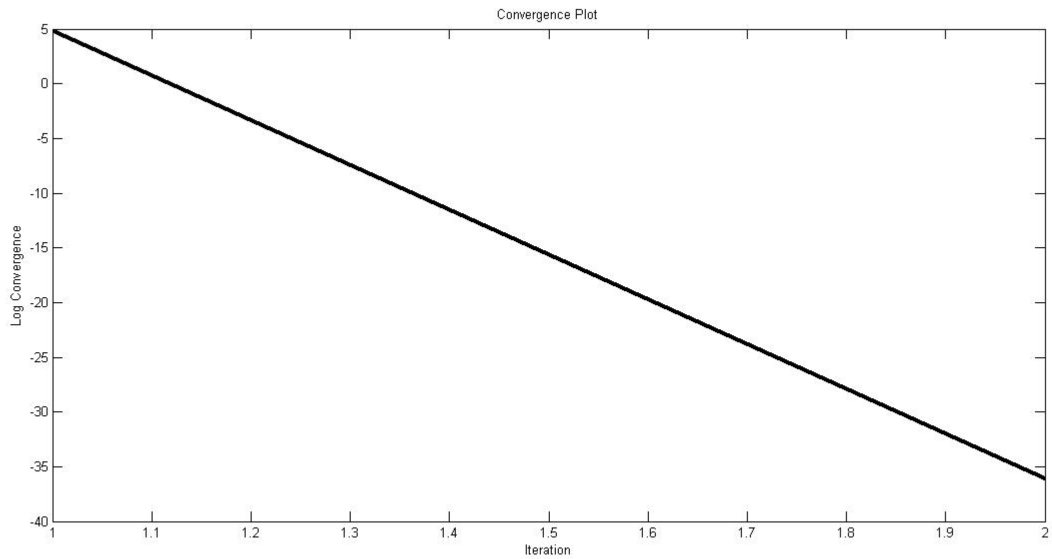


Figure 8: Newton's method convergence for guess [2, 3]

As observed from all the convergence plots, a log convergence between -35 and -40 is obtained which is acceptable. The plots for gradient descent is non-linear whereas it is linear for the Newton's method.

As seen from all the plots, the gradient descent method takes more number of iterations than the Newton's method to reach the minimum. The Armijo search method is employed in both the methods. Therefore, for the given problem, the Newton's method with Armijo line search is better because it tracks the global minimum in just one iteration. This is because the given objective function is quadratic and Newton's method approximates a given function as quadratic through Taylor's expansion. Hence, for a given quadratic convex function, the Newton's method can track the global minimum in a single step. (Find all the MATLAB codes in the following slides).

```matlab
%%%%%%%%%%%%%% Main Entrance %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%% By Max Yi Ren and Emrah Bayrak %%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Instruction: Please read through the code and fill in blanks
% (marked by ***). Note that you need to do so for every involved
% function, i.e., m files.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Optional overhead
clear; % Clear the workspace
% Note: for debugging purpose, do not use "clear all"
close all; % Close all windows
clc; %Clear screen

%% Optimization settings
% Here we specify the objective function by giving the function handle to a
% variable, for example:
f = @(x)distance(x); % replace rosenbrock with your objective function
% In the same way, we also provide the gradient and the Hessian of the
% objective:
g = @(x)distanceg(x); % replace accordingly
H = @(x)distanceH(x); % replace accordingly
% Note that explicit gradient and Hessian information is only optional.
% However, providing these information to the search algorithm will save
% computational cost from finite difference calculations for them.

% Specify algorithm
opt.alg = 'gradient';
%opt.alg = 'newton';

% Turn on or off line search. You could turn on line search once other
% parts of the program are debugged.
opt.linesearch = true; % or true

% Set the tolerance to be used as a termination criterion:
opt.eps = 1e-4; % this should be a small number like 1e-3

% Set the initial guess:
x0 = [2; 3]; % this should be a p-dim vector where p is the size of the
% problem

%% Run optimization
% Run your implementation of the gradient descent and Newton's method. See
% gradient.m and newton.m.
if strcmp(opt.alg,'gradient')
    solution = gradient(f,g,H,x0,opt);
elseif strcmp(opt.alg,'newton')
    solution = newton(f,g,H,x0,opt);
end

%% Report
% Implement report.m to generate a report.
report(solution,f);
```

```
function y = distance(x)
    y = (2-2*x(1)-3*x(2))^2 + x(1)^2 + (x(2)-1)^2;
```

```
function g = distanceg(x)
    g = [-4*(2-2*x(1)-3*x(2)) + 2*x(1); -6*(2-2*x(1)-3*x(2)) + 2*(x(2)-1)];
```

```
function H = distanceH(x)
    H = [10 12;12 20];
```

```
%%%%%%%%%%%%%% Gradient Descent Implementation %%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%% By Max Yi Ren and Emrah Bayrak %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function solution = gradient(f,g,H,x0,opt)
    % Set initial conditions
     x = x0; % Set current solution to the initial guess
     iter = 0; % Set iteration counter to 0

    % Initialize a structure to record search process
     solution = [];

    % Calculate the norm of the gradient
     gnorm = norm(g(x),2); % this needs to be a scalar

    % Set the termination criterion:
    while gnorm>opt.eps % if not terminated
        iter = iter + 1

        % save current step
        solution.x([1,2],iter) = x;
        % solution.x is an array of solutions, i.e., a matrix
        % opt.linesearch switches line search on or off.
        % You can first set the variable "a" to different constant values and see how↙
it
        % affects the convergence.
        if opt.linesearch
            a = lineSearch1(f,g,H,x,opt);
        else
            a = 0.001;
        end


        % Gradient descent:
        d = -1*g(x);
        x = x + a*d; % update x based on gradient info
        disp(x);
        % Update termination criterion:
        gnorm = norm(g(x),2); % update the norm of gradient
    end
```

```
%%%%%%%%%%%%%% Newton's Method Implementation %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%% By Max Yi Ren and Emrah Bayrak %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function solution = newton(f,g,H,x0,opt)
    % Set initial conditions
     x = x0; % Set current solution to the initial guess
     iter = 0; % Set iteration counter to 0
     solution.x([1,2],1) = x;
    % Calculate the norm of the gradient
     gnorm = norm(g(x),2);

    while gnorm>opt.eps % if not terminated
        iter = iter + 1;

        % opt.linesearch switches line search on or off.
        % You can first set the variable "a" to different constant values and see how it
        % affects the convergence.
        if opt.linesearch
            a = lineSearch1(f,g,H,x,opt);
        else
            a = 0.001;
        end

        % Newton's method:
         x = x - a*inv(H(x))*g(x);

        % save current step
         solution.x([1,2],iter+1) = x;

        % Update termination criterion:
         gnorm = norm(g(x),2);
    end
    disp(x);
    disp(iter);
```

```
% Armijo line search
function a = lineSearch1(f,g,H,x,opt)
    t = 0.1; % scale factor on current gradient: [0.01, 0.3]
    b = 0.55; % scale factor on backtracking: [0.1, 0.8]
    a = 1; % maximum step length
    G = feval(g,x);

    % Calculate the descent direction D for gradient or newton
    if strcmp(opt.alg,'gradient')
         D = -1*G;
    elseif strcmp(opt.alg,'newton')
         D = -1*inv(H(x))*G;
    end

    % terminate if line search takes too long
    count = 0;
    while f(x+a*D) > f(x)+t*a*G'*D
        % stop if condition satisfied
        % implement Armijo's criterion here
        % perform backtracking
        a = b*a
        count = count + 1;
    end
    disp(a);
    disp(count);
end
```

```matlab
%%%%%%%%%%%%%% Generate Report  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%% By Max Yi Ren and Emrah Bayrak %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function report(solution,f)
    figure; % Open an empty figure window
    hold on; % Hold on to the current figure

    % Draw a 2D contour plot for the objective function
    % You can edit drawing parameters within the file: drawContour.m
    drawContour(f);

    % Plot the search path
    x = solution.x;
    iter = size(x,2);
    plot(x(1,1),x(2,1),'.y','markerSize',20);
    str1 = [num2str(x(1,1)),',',num2str(x(2,1))];
    text(x(1,1),x(2,1),str1);
    for i = 2:iter
        % Draw lines. Type "help line" to see more drawing options.
        line([x(1,i-1),x(1,i)],[x(2,i-1),x(2,i)],'Color','y');
        plot(x(1,i),x(2,i),'.y','markerSize',20);
    end
    title('Contour plot');
    xlabel('x2');
    ylabel('x3');
    str2 = [num2str(x(1,i)),',',num2str(x(2,i))];
    text(x(1,i),x(2,i),str2);
    % Plot the convergence
    F = zeros(iter,1);
    for i = 1:iter
        F(i) = feval(f,x(:,i));
    end
    figure;
    plot(1:iter, log(F-F(end)+eps),'k','lineWidth',3);
    title('Convergence Plot');
    xlabel('Iteration');
    ylabel('Log Convergence');
```