UNIVERSITY OF TEHRAN

COLLEGE OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

CONVEX OPTIMIZATION

BONUS PROJECT

MOHAMMAD HEYDARI

810197494

UNDER SUPERVISION OF:

DR. HADI AMIRI

ASSISTANT PROFESSOR

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

UNIVERSITY OF TEHRAN

Jan. 2022

1 CONTENTS

2 QUESTION #1

2.1 THEORETICAL APPROACH FOR QUESTION #3

In this part we intend to calculate the response based on theoretical approach:

Q3)

الف) برای بررسی محدب بودن می‌بایست مشتقات جزئی را بدست آوریم و بررسی نماییم.

$$f(x,y) = x^4 + (y-1)^2 \longrightarrow \text{مشتق مرتبه اول} \longrightarrow \begin{cases} f'_x = \dfrac{\partial f}{\partial x} = 4x^3 \\ f'_y = \dfrac{\partial f}{\partial y} = 2(y-1) \end{cases}$$

$$\xrightarrow[\text{مشتق}]{\text{مرتبه دوم}} \begin{cases} f''_x = \dfrac{\partial f}{\partial x^2} = 12x^2 \\ f''_y = \dfrac{\partial f}{\partial y^2} = 2 \end{cases} \longrightarrow \begin{cases} f''_{xy} = 0 \\ f''_{yx} = 0 \end{cases}$$

از معادلات بالا نتیجه خواهیم گرفت که ← نقطه‌ی $(0,1)$

ریشه‌های مشتق مرتبه‌ی اول نقاط مورد بحث به عنوان مینیمم تابع خواهند بود. [با توجه به مثبت بودن مشتقات مرتبه‌ی دوم تابع محدب می‌باشد]

حال بررسی می‌کنیم که آیا $(0,1)$ دقیقا مینیمم نسبی است یا مینیمم مطلق است داریم ←

$$D(x,y) = f''_{xx}(x,y) \times f''_{yy}(x,y) - [f''_{xx}(x,y)]^2 = 0 - 0 = 0$$

پس نقطه‌ی $p(0,1)$ مینیمم تابع خواهد بود.

جهت نزول

ب) در این قسمت روش گرادیان را بکار می‌گیریم ← ماتریس مشتقات جزئی بصورت روبرو است.

$$d_k = -\nabla f(x^k, y^k) = -\begin{pmatrix} 4x^3 \\ 2(y-1) \end{pmatrix} = \begin{pmatrix} -4 \\ 0 \end{pmatrix}$$

برای نقطه‌ی اولیه $(1,1) = (x_0, y_0)$ ←

$$x_{new} = x_{old} + a\, d_k$$

نقطه‌ی شروع — ضریب آپدیت — جهت نزول

در این قسمت می‌بایست برای هر مرحله a (یا ضریب یادگیری) را بدست آوریم ←

بدست آوردن a $\longrightarrow \min_a h(a) := f(x^k + a d^k)$

$$\longrightarrow h(a) = f(x^0 + a d^0) = (1-4a)^4 + (1 - a \times 0 - 1)^2 = (1-4a)^4$$

$$\longrightarrow \frac{dh(a)}{da} = 0 \longrightarrow 4(1-4a)^3 \times -4 = 0 \longrightarrow (1-4a)^3 = 0$$

$$\longrightarrow \boxed{a_0 = 0{,}25}$$

آپدیت مرحله 1

$$\longrightarrow (x_1, y_1) = (x_0, y_0) + 0{,}25(-4, 0)$$

$$= (1,1) + (-1,0) = (0,1) \longrightarrow$$

همانطور که مشاهده می‌کنیم به دلیل بالا رفتن روش دقیق الگوریتم در همان مرحله‌ی اول ← جواب بهینه همگرا شد در نتیجه شرط برقرار است

$$\longrightarrow f_{estimated}(0,1) = 0 \quad , \quad f_{actual} = 0 \longrightarrow |y - \hat{y}| < \varepsilon = 10^{-4}$$

پ) حال الگوریتم گرادیان را برای روش back-tracking یا اصطلاحاً روش آرمیجو بکار خواهیم برد ←

در این روش نیز ابتدا جهت نزول را می‌یابیم ← $g = \nabla f(x^k, y^k) = \begin{pmatrix}4x^3\\2(y-1)\end{pmatrix} \overset{\text{نقطه صفر}}{=} \begin{pmatrix}4\\0\end{pmatrix}$

پارامترهای داده شده در این روش $(a, B) = (0.5, 0.5)$, $\epsilon = 10^{-4}$

روش الگوریتم بصورت روبرو است ← if $f(x_k + ad_k) \leq f(x_k) + a\epsilon \Delta f(x_k)' d_k$ stop

on return alpha as our result

Else update new value $a = Ba_0$ → آلفا را کوچک می‌کنیم!

$x^{k+1} = x^k + a \times d^k$

نقطه‌ی شروع (۱و۱)

از نقطه‌ی اولیه شرط بالا را چک می‌کنیم ← $f(x_k + ad_k) = f(1-4a_0, 1) \leq f(1,1) + 0.5\times 10^{-4} \times [4\ 0]\begin{bmatrix}-4\\0\end{bmatrix}$

$a = 0.5 \rightarrow f(-1,1) \leq 1 + 0.5 \times 10^{-4} \times -16 \longrightarrow 1 \nleq 1 + -8\times 10^{-4}$

← آلفا را کوچک‌تر می‌کنیم ←

$a_{new} = Ba_0 = 0.5 \times 0.5 = 0.25$

دوباره شرط را چک می‌کنیم ←

دوباره شرط را چک می‌کنیم ←

$f(1-4a_{new}, 1) \leq 1 + \frac{1}{4}\times 10^{-4} \times -16$ ✓

$\rightarrow f(0,1) \leq 1 - 4\times 10^{-4} \longrightarrow 0 \leq 1 - 4\times 10^{-4}$

شرط برقرار است پس آلفا را انتخاب و x را به‌عنوان جواب گزارش می‌کنیم

$\rightarrow x_{optimum} = (0,1) \longrightarrow f_{\text{متناظر}} = 0$

ت) جهت امتیازی ← الگوریتم به روش نیوتن ←

$$x_{k+1} \leftarrow x_k - H^{-1} g$$

H: ماتریس هسیان — g: گرادیان

در روش نیوتن داریم ←

$$g_k = \nabla f(x^0, y^0) = \begin{pmatrix} 4x^3 \\ 2(y-1) \end{pmatrix} \quad @\ x = (1,1) = \begin{pmatrix} 4 \\ 0 \end{pmatrix} = g_0$$

مشتقات جزئی را نیز برای تشکیل ماتریس هسیان در قسمت الف بدست آوردیم ←

$$f_{xx} = 12x^2 \quad @\ (1,1) \longrightarrow f_{xx} = 12$$

$$f_{yy} = 2$$

$$f_{xy} = 0 \longrightarrow H_0 = \nabla^2 f(x^k, y^k) = \begin{bmatrix} 12 & 0 \\ 0 & 2 \end{bmatrix}$$

$$f_{yx} = 0 \longrightarrow H_0^{-1} = \frac{1}{24} \begin{bmatrix} 2 & 0 \\ 0 & 12 \end{bmatrix} = \begin{bmatrix} \frac{1}{12} & 0 \\ 0 & \frac{1}{2} \end{bmatrix}$$

$$\longrightarrow x_1 \leftarrow x_0 - H_0^{-1} g \longrightarrow x_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix} - \begin{bmatrix} \frac{1}{12} & 0 \\ 0 & \frac{1}{12} \end{bmatrix} \begin{bmatrix} 4 \\ 0 \end{bmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} - \begin{pmatrix} \frac{1}{3} \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{2}{3} \\ 1 \end{pmatrix}$$

$$x_{new} = \begin{pmatrix} \frac{2}{3} \\ 1 \end{pmatrix} \quad g_{new} = \begin{pmatrix} 4x^3 \\ 2(y-1) \end{pmatrix} = \begin{pmatrix} \frac{32}{27} \\ 0 \end{pmatrix} \quad H_1 = \begin{bmatrix} 5.3 & 0 \\ 0 & 2 \end{bmatrix}$$

یک دور دیگر نیز الگوریتم را تکرار می‌کنیم.

$$x_2 = \begin{pmatrix} \frac{2}{3} \\ 1 \end{pmatrix} - \begin{bmatrix} 5.3 & 0 \\ 0 & 2 \end{bmatrix}^{-1} \begin{bmatrix} 1.1852 \\ 0 \end{bmatrix} = \begin{pmatrix} 0/44 \\ 1 \end{pmatrix}$$

$$x_{new} = \begin{pmatrix} 0/44 \\ 1 \end{pmatrix} \quad g_{new} = \begin{pmatrix} 4x^3 \\ 2(y-1) \end{pmatrix} = \begin{pmatrix} 0/3512 \\ 0 \end{pmatrix}, \quad H_2 = \begin{bmatrix} 2.3704 & 0 \\ 0 & 2 \end{bmatrix}$$

$$x_{new} = \begin{pmatrix} 0/29 \\ 1 \end{pmatrix} \quad g_{new} = \begin{pmatrix} 0/104 \\ 0 \end{pmatrix}, \quad H_3 = \begin{bmatrix} 1.0535 & 0 \\ 0 & 2 \end{bmatrix}$$

$$x_{new} = \begin{pmatrix} 0/197 \\ 1 \end{pmatrix} \xrightarrow{\text{متناظر } f} f(x,y) = 0/0015 \quad g_{new} = \begin{pmatrix} 0/0308 \\ 0 \end{pmatrix}, \quad H_4 = \begin{bmatrix} 0/4682 & 0 \\ 0 & 2 \end{bmatrix}$$

$$\longrightarrow \sqrt{|0 - 0/0015|} \not< 10^{-4} \longrightarrow$$ ادامه می‌دهیم

$$x_{new} = \begin{pmatrix} 0/1317 \\ 1 \end{pmatrix} \quad g_{new} = \begin{pmatrix} 0/0091 \\ 0 \end{pmatrix} \quad H_5 = \begin{bmatrix} 0/2081 & 0 \\ 0 & 2 \end{bmatrix}$$

$$x_{new} = \begin{pmatrix} 0/0878 \\ 1 \end{pmatrix} \xrightarrow{\text{متناظر } f} f(x,y) = 5.94 \times 10^{-5}$$

$$\longrightarrow \mathrm{norm}(f - f^*) < 10^{-4} \longrightarrow \checkmark$$

شرط همگرایی ارضا خواهد شد!

3 QUESTION #2

3.1 FINDING GLOBAL MINIMUM

In this part I have written a function which let us to investigate is there any global minimum or not.

As we know from the theoretical part we suppose to find a global minimum at point (0,1).

So let's have an accurate investigation by plotting the 3D representation of our function:

```
% PART A
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
X = -15:0.1:15;
Y = -15:0.1:15;
[x,y] = meshgrid(X,Y);
f_a = x.^4 +(y-1).^2;
figure(1)
surf(x,y,f_a)
xlabel('x'); ylabel('y'); zlabel('f(x,y)');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Figure 1: implementing function 3D represent in MATLAB

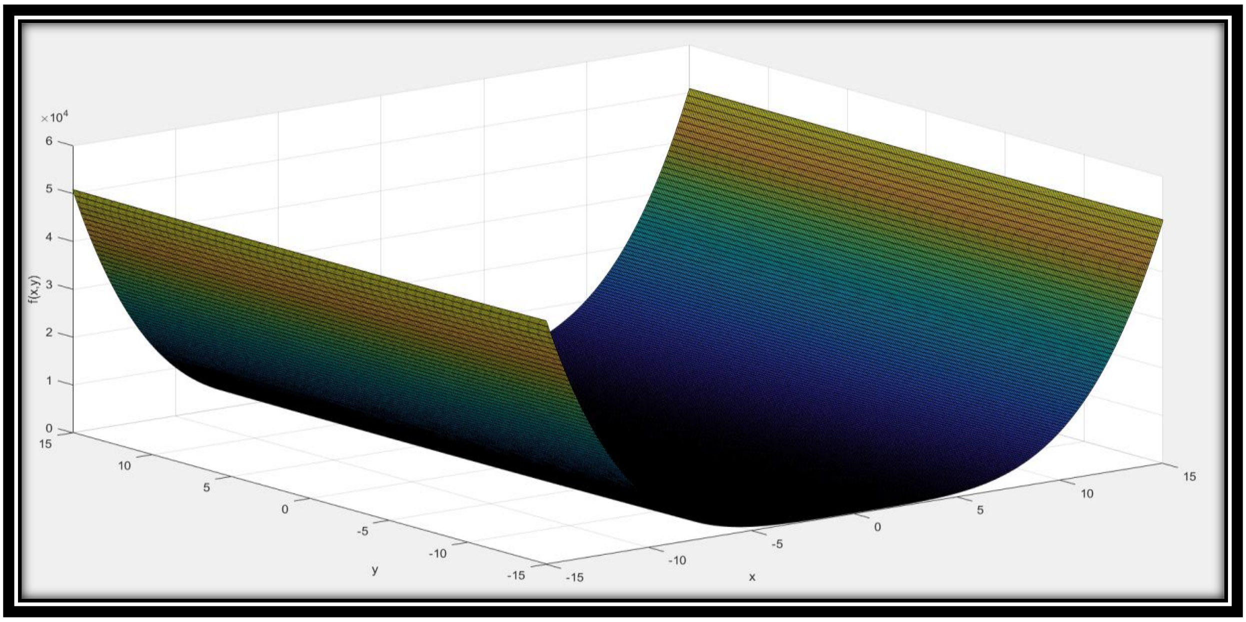

Figure 2: 3D representation

As you can see obviously we face with a global minimum is point (0,1) which matches with our previous knowledge as well as we supposed.

4 QUESTION #3

4.1 IMPLEMENTING GRADIENT DESCENT USING MATLAB

In this part I have implemented gradient descent algorithm in MATLAB using materials which discussed in class sessions to get us an easier approach to obtain our desired result:

Note that I have make three different function to satisfy all of aspects of given question and in this part I have written gradient_descent_func() to provide us an exact solution to stochastic gradient descent problem:

```
function [Target_xvalue,yvalue_optimal,iterations_Num,gnorm] = gradient_descent_func(x0,Epsilon,alpha)
gnorm = inf;
x = x0;
f = @(x,y) x.^4 + (y-1).^2;
figure(2);
clf;
fcontour(f,[-15 15 -15 15]);
axis equal;
hold on
f2 = @(x) f(x(1),x(2));
iterations_Num=0;
while (gnorm>=Epsilon)
g = [4*(x(1)^3) ; 2*(x(2)-1)];
gnorm = norm(g)
direction=-g;
xnew = x + alpha*direction;
plot([x(1) xnew(1)],[x(2) xnew(2)],'ko-')
refresh
iterations_Num = iterations_Num + 1;
dx = norm(xnew-x);
x = xnew;
iterations_Num=iterations_Num+1;
end
Target_xvalue = x;
yvalue_optimal = f2(Target_xvalue);
end
```

Afterward, let's see the outputs of implementation and investigate whether it matches or not ;)

```
>> main
gnorm =

     4


gnorm =

     0


Target_xvalue =

     0
     1


yvalue_optimal =

     0


iterations_Num =

     4


gnorm =

     0
```

As you can see cause of choosing an appropriate initial point and also using an efficient way to finding the initial alpha value the convergence rate is so nice and considerable and in other word we meet the optimum value just with making four iterations!

The optimum point is (0,1) and the correspond value is equal to zero!

Further you can find function recalling in the main body of my implementation:

```
% PART B
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x=1;
y=1;
x0=[x ;y];
alpha =0.25;
Epsilon = 1e-4;
[Target_xvalue,yvalue_optimal,iterations_Num,gnorm] = gradient_descent_func(x0,Epsilon,alpha)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

<u>**For running this part please have an accurate look on related directory and There you can easily find all of you need and also I would appreciate it if you would consider them ☺**</u>

5 QUESTION #4

5.1 IMPLEMENTING BACK TRACKING METHOD USING MATLAB

In this part of my report I made piece of codes to give us deeper insight during implementing Back-Tracking Method Using MATLAB.

Further you can see all of my implementation in MATLAB:

```
function [xnew,f,alpha_armijo,Num_iteration] = back_tracking_func(alpha,Beta,Epsilon,x)
f_z = @(x,y) x.^4 + (y-1).^2;
f0 = @(x) f_z(x(1),x(2));
g = [4*(x(1)^3) ; 2*(x(2)-1)];
d=-g;
loop= 1;
Num_iteration=0;
while (loop>0)
Num_iteration=Num_iteration+1;
x_new = x+alpha.*d;
if (f0(x_new)<=f0(x)+Epsilon*alpha*g'*d)
loop = 0;
alpha_armijo = alpha;
else
alpha = alpha*Beta;
end
end
f=f0(x_new);
xnew=x_new;
end
```

```
:new =

     0
     1


: =

     0


ılpha_armijo =

    0.2500


Ium_iteration =

     2


ı =

     4
     0
```

```
% PART C
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
alpha=0.5;
x=1;
y=1;
x=[x ;y];
Beta = 0.5;
Epsilon = 1e-4;
[xnew,f,alpha_armijo,Num_iteration] =back_tracking_func(alpha,Beta,Epsilon,x)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

As you can see the results matches with theoretical approach as well as we supposed.

<u>For running this part please have an accurate look on related directory and There you can easily find all of you need and also I would appreciate it if you would consider them ☺</u>

6 QUESTION#5

6.1 IMPLEMENTING NEWTON ALGORITHM USING MATLAB

In this part of my report I made piece of codes to give us deeper insight during implementing Newton Method Using MATLAB.

Further you can see all of my implementation in MATLAB:

```
function [xnew,f,Num_iteration] = newton_algorithm_func(Epsilon,x)
f_z = @(x,y) x.^4 + (y-1).^2;
f0 = @(x) f_z(x(1),x(2));
loop= 1;
Num_iteration=0;
while (loop>0)
Num_iteration=Num_iteration+1;
g = [4*(x(1)^3) ; 2*(x(2)-1)]
d=-g;
H_matrix=[12*(x(1)^2) 0 ; 0  2]
x_new = x+inv(H_matrix)*d
if ((f0(x_new)-f0(x))^2<Epsilon)
loop = 0;
else
x = x_new;
end
end
f=f0(x_new);
xnew=x_new;
end
```

```
H_matrix =

    0.2081         0
         0    2.0000


x_new =

    0.0878
    1.0000


xnew =

    0.0878
    1.0000


f =

   5.9403e-05


Num_iteration =

     6
```

```
% PART D
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x=1;
y=1;
x=[x ;y];
Epsilon = 1e-4;
[xnew,f,Num_iteration] = newton_algorithm_func(Epsilon,x)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

As you can see the results matches with theoretical approach as well as we supposed.