ESE 415

Homework 3 Solutions

1. (a) Since α^k is a minimizer of

$$h(\alpha) = f(x^k + \alpha d^k),$$

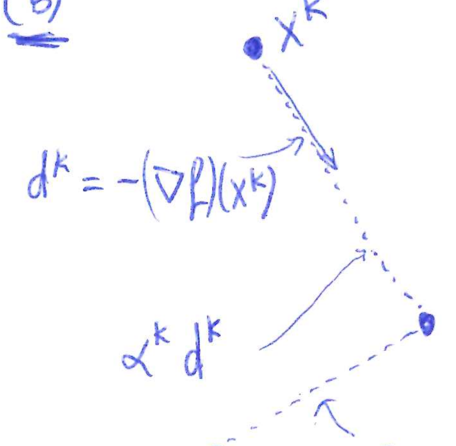
it necessarily follows from the necessary condition that α^k is a critical point of h, i.e. $h'(\alpha^k) = 0$

$$h'(\alpha) = (\nabla f)(x^k + \alpha d^k)^T d^k$$

so

$$\underbrace{(\nabla f)(x^k + \alpha^k d^k)^T d^k}_{= \langle d^k, (\nabla f)(x^k + \alpha^k d^k) \rangle} = 0$$

(b)



$$x^{k+1} = x^k + \alpha^k d^k \quad \text{with } d^k = -(\nabla f)(x^k)$$

$$(\nabla f)(x^{k+1}) = (\nabla f)(x^k + \alpha^k d^k)$$

Using the result from (a) with $d^k = -(\nabla f)(x^k)$ and $x^{k+1} = x^k + \alpha^k d^k$, the claim of $\langle (\nabla f)(x^k), (\nabla f)(x^{k+1}) \rangle = 0$ follows.

can drop negative sign via linearity argument

2(a)

Solution

Observe first that if we write $x = (x_1, x_2)^T$ then $f(x) = \frac{1}{2}x^T Qx - c^T x + a$ where

$$Q = \begin{bmatrix} 10 & 4 \\ 4 & 2 \end{bmatrix}, \qquad c = \begin{bmatrix} 14 \\ 6 \end{bmatrix}, \text{ and } a = 20$$

Note that $Q \succ 0$, and recall that $\nabla f(x) = Qx - c$. The steepest descent step is $x^{k+1} = x^k + \alpha^k d^k$ where $d^k = -\nabla f(x^k)$, hence the algorithm is:

Step 0: Initial guess x^0, threshold $\epsilon > 0$, counter $k = 0$.

Step 1: Choose direction $d^k = -\nabla f(x^k) = -Qx^k + c$. if $\|d^k\| < \epsilon$ stop.

Step 2: Compute step size

$$\alpha^k = \frac{(d^k)^T d^k}{(d^k)^T Q d^k}$$

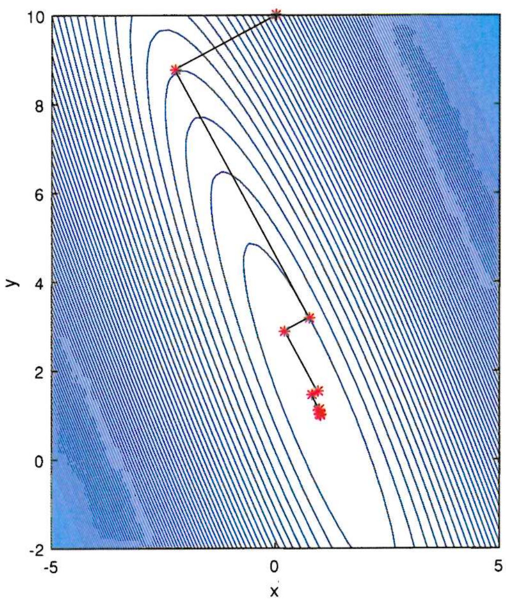
Step 3: Set $x^{k+1} = x^k + \alpha^k d^k$, and increment counter $k = k + 1$. Go to step 1.

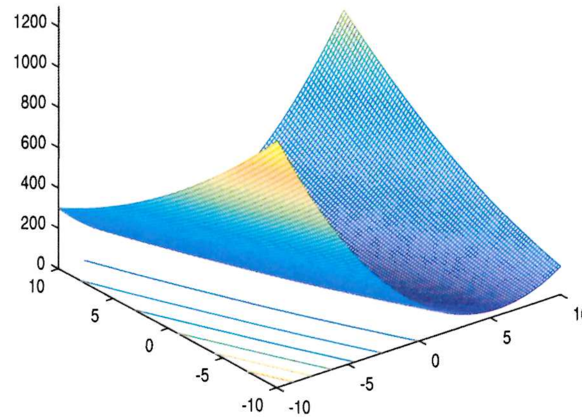
```
clc;clear all;close all
Q=[10 4; 4 2]; c=[14;6]; a=20; %define objective function
ep=10^(-6); %tolerance
X=[]; D=[]; A=[]; F=[]; %declare state, direction, step, function value

X(:,1)=[0;10]; %initialize state
D(:,1)=-Q*X(:,1)+c; %initial direction
A(1,1)=D(:,1)'*D(:,1)/(D(:,1)'*Q*D(:,1)); %initial step size
F(1,1)=X(:,1)'*Q*X(:,1)/2 - c'*X(:,1) + a;
for k=1:1000 %allow 1000 steps to converge
if(norm(D(:,k))<ep) break; end; %quit if tolerance reached
X(:,k+1)=X(:,k)+A(1,k)*D(:,k); %take the step
D(:,k+1)=-Q*X(:,k+1)+c; %descent direction
A(1,k+1)=D(:,k+1)'*D(:,k+1)/(D(:,k+1)'*Q*D(:,k+1)); %step size
F(1,k+1)=X(:,k+1)'*Q*X(:,k+1)/2 - c'*X(:,k+1) + a; %function value
end
%Contour plot
[xx,yy]=meshgrid(-10:.2:10);
zz=(Q(1,1)*xx.^2+(Q(2,1)+Q(1,2))*xx.*yy+Q(2,2)*yy.^2)/2-c(1)*xx-c(2)*yy+a;

figure
[C,h]=contour(xx,yy,zz,400); hold on, plot(X(1,:),X(2,:),'k',X(1,:),X(2,:),'r*')
hold off, xlabel('x'), ylabel('y')
% set(h,'ShowText','on','TextStep',get(h,'LevelStep')), colormap cool
set(h,'TextStep',get(h,'LevelStep')), colormap jet
axis([-5 5 -2 10])

figure
mesh(xx,yy,zz)
zlim([0 1200])
grid on
```





k	x_1^k	x_2^k	d_1^k	d_2^k	$\|d^k\|$	α^k	$f(x^k)$
1	0.000000	10.000000	-26.000000	-14.000000	29.529646	0.086645	60.000000
2	-2.252782	8.786963	1.379968	-2.562798	2.910712	2.180000	22.222576
3	0.755548	3.200064	-6.355739	-3.422321	7.218567	0.086645	12.987827
4	0.204852	2.903535	0.337335	-0.626480	0.711528	2.180000	10.730379
5	0.940243	1.537809	-1.553670	-0.836592	1.764590	0.086645	10.178542
6	0.805625	1.465322	0.082462	-0.153144	0.173934	2.180000	10.043645
7	0.985392	1.131468	-0.379797	-0.204506	0.431357	0.086645	10.010669
8	0.952485	1.113749	0.020158	-0.037436	0.042518	2.180000	10.002608
9	0.996429	1.032138	-0.092842	-0.049992	0.105446	0.086645	10.000638
10	0.988385	1.027806	0.004928	-0.009151	0.010394	2.180000	10.000156
11	0.999127	1.007856	-0.022695	-0.012221	0.025776	0.086645	10.000038
12	0.997161	1.006797	0.001205	-0.002237	0.002541	2.180000	10.000009
13	0.999787	1.001920	-0.005548	-0.002987	0.006301	0.086645	10.000002
14	0.999306	1.001662	0.000294	-0.000547	0.000621	2.180000	10.000001
15	0.999948	1.000469	-0.001356	-0.000730	0.001540	0.086645	10.000000
16	0.999830	1.000406	0.000072	-0.000134	0.000152	2.180000	10.000000
17	0.999987	1.000115	-0.000332	-0.000179	0.000377	0.086645	10.000000
18	0.999959	1.000099	0.000018	-0.000033	0.000037	2.180000	10.000000
19	0.999997	1.000028	-0.000081	-0.000044	0.000092	0.086645	10.000000
20	0.999990	1.000024	0.000004	-0.000008	0.000009	2.180000	10.000000
21	0.999999	1.000007	-0.000020	-0.000011	0.000023	0.086645	10.000000
22	0.999998	1.000006	0.000001	-0.000002	0.000002	2.180000	10.000000
23	1.000000	1.000002	-0.000005	-0.000003	0.000006	0.086645	10.000000
24	0.999999	1.000001	0.000000	0.000000	0.000001	2.180000	10.000000

(b) Solve (a) with MATLAB optimization solver "fminunc" by setting the same threshold, i.e., the stopping criterion ε. Print out the first 10 and the last 10 iterations.

Solution: Here is the results obtained by starting from $x_0 = \begin{bmatrix} 0 \\ 10 \end{bmatrix}$.

First 10 iterations:

Iteration	Func-count	$f(x)$	Step-size	First-order optimality
0	3	60		26
1	6	33.9053	0.0384615	13.8
2	12	22.0956	0.1	3.57
3	18	20.7477	0.176204	3.55
4	24	19.55	0.158109	3.17
5	30	18.4858	0.176204	3.16
6	36	17.5401	0.158109	2.82
7	42	16.6999	0.176204	2.8
8	48	15.9533	0.158109	2.5
9	54	15.2899	0.176204	2.49
10	60	14.7004	0.158109	2.22

Last 10 iterations:

Iteration	Func-count	$f(x)$	Step-size	First-order optimality
192	1152	10	0.15485	4.65e-05
193	1158	10	0.181212	4.74e-05
194	1164	10	0.155063	4.14e-05
195	1170	10	0.181965	4.23e-05
196	1176	10	0.15361	3.65-05
197	1182	10	0.183124	3.75e-05
198	1188	10	0.152404	3.23e-05
199	1194	10	0.18358	3.34e-05
200	1200	10	0.15122	2.86e-05
201	1206	10	0.183269	2.97e-05
202	1212	10	0.152357	2.56e-05

Code for reference:

```
clear
close all
opt = optimoptions(@fminunc,'Display','iter-detailed','Algorithm',...
    'quasi-newton','HessUpdate','steepdesc','MaxFunctionEvaluations',2000);
fun = @(x) 5*x(1)^2 + x(2)^2 + 4*x(1)*x(2) - 14*x(1) -6*x(2) + 20;
x0 = [0 10]';
[x,fval,exitflag,output] = fminunc(fun,x0,opt);
```

3

(a) $$f(x_1, x_2) = \frac{x_1^4}{4} - x_1^2 + 2x_1 + \overbrace{(x_2-1)^2}^{x_2^2 - 2x_2 + 1}$$

Newton's Method: $x^{k+1} = x^k - (Hf)(x^k)^{-1} (\nabla f)(x^k)$

So one step: $x^1 = x^0 - (Hf)(x^0)^{-1} (\nabla f)(x^0)$

$$(\nabla f)(x) = \begin{pmatrix} x_1^3 - 2x_1 + 2 \\ 2x_2 - 2 \end{pmatrix}$$

$$(Hf)(x) = \begin{pmatrix} 3x_1^2 - 2 , & 0 \\ 0 , & 2 \end{pmatrix}$$

$(Hf)(x)$ not invertible when $3x_1^2 - 2 = 0 \Leftrightarrow x_1 = \pm\sqrt{\frac{2}{3}}$

$\rightarrow$ Newton's Method breaks down when $x^0 = \begin{pmatrix} \pm\sqrt{\frac{2}{3}} \\ y \end{pmatrix}$, $y \in \mathbb{R}$.

3(b)

```
function newtons_method()
    % initialize
    x0 = [-1; 0];
    tolerance = 1e-6;
    max_iter = 10000;
    iter = 0;
    x = x0;

    path = x';
    fprintf('%5s %10s %10s %10s %10s %10s\n', 'Iter(k)', 'x1', 'x2', 'd1', 'd2', 'f(x)');

    while iter < max_iter
        H = hessian_f(x);
        g = gradient_f(x);

        if rank(H) < 2
            error('The hessian matrix is singular.');
        end

        d = -H \ g;
        x = x + d;

        path = [path; x'];
        f_value = f(x);

        fprintf('%5d %10.4f %10.4f %10.4f %10.4f %10.4f\n', iter, x(1), x(2), d(1), d(2), f_value);

        if norm(d) < tolerance
            break;
        end

        iter = iter + 1;
    end

    [X1, X2] = meshgrid(linspace(-5, 2, 100), linspace(-2, 2, 100));
    Z = (X1.^4)/4 - X1.^2 + 2*X1 + (X2 - 1).^2;
    contour(X1, X2, Z, 50);
    hold on;
    plot(path(:,1), path(:,2), 'r-o', 'LineWidth', 1, 'MarkerSize', 5);
    title('Newton Method Path');
    xlabel('x_1');
    ylabel('x_2');
    hold off;
end

function f_val = f(x)
    f_val = (x(1)^4) / 4 - x(1)^2 + 2*x(1) + (x(2) - 1)^2;
end

function g = gradient_f(x)
    g = [x(1)^3 - 2*x(1) + 2; 2*(x(2) - 1)];
end

function H = hessian_f(x)
    H = [3*x(1)^2 - 2, 0; 0, 2];
end
```