

Problem 1 (60%)

Minimize the function

$$f = x_1^2 + 2x_1x_2 + 4x_1x_3 + 3x_2^2 + 2x_2x_3 + 5x_3^2 \quad (1)$$

using (a) the simple steepest descent direction method (25%), and (b) the Newton's method (25%). Perform a few iterations with each method starting from the point $x_0 = (1, 1, 1)^T$. (c) Confirm results analytically. (10%)

Objective_function_hw2_1.m :

```
function [f,df,H]=objective_function_hw2_1(x)
f=x(1).^2+2.*x(1).*x(2)+4.*x(1).*x(3)+3.*x(2).^2+2.*x(2).*x(3)+5.*x(3).^2;
df=zeros(3,1);
df(1)=2.*x(1)+2.*x(2)+4.*x(3);
df(2)=2.*x(1)+6.*x(2)+2.*x(3);
df(3)=4.*x(1)+2.*x(2)+10.*x(3);
H=[2 2 4; 2 6 2; 4 2 10];
```

(a)

hw2_1_steepest_descent_direction_method.m :

```
x0=[1;1;1];
max_step=100;
iteration_x=zeros(3,max_step);
iteration_f=zeros(1,max_step);
iteration_x(:,1)=x0;
for ii=1:max_step
    [iteration_f(ii),df,H]=objective_function_hw2_1(iteration_x(:,ii));
    if ii>1 && abs(iteration_f(ii)-iteration_f(ii-1))<1e-4 || ii==max_step
        f=iteration_f(ii);
        x=iteration_x(:,ii);
        fprintf("steps = %d \n", ii-1)
        fprintf("f = %s \n",f)
        fprintf("x1 = %s , x2 = %s , x3 = %s \n",x(1),x(2),x(3))
        break
    end
    iteration_x(:,ii+1)=iteration_x(:,ii)-df;
end
```

result :

```
Command Window

>> hw2_1_steepest_descent_direction_method
steps = 99
f = 4.487314e+212
x1 = -3.240912e+105 , x2 = -3.095144e+105 , x3 = -7.115327e+105
fx>>
```

Without the coefficient α , we can't find the optima in this objective function. Since the gradient is too big to make the step correct.

(b)

hw2_1_Newtons_method.m :

```
x0=[1;1;1];
max_step=100;
iteration_x=zeros(3,max_step);
iteration_f=zeros(1,max_step);
iteration_x(:,1)=x0;
for ii=1:max_step
    [iteration_f(ii),df,H]=objective_function_hw2_1(iteration_x(:,ii));
    if ii>1 && abs(iteration_f(ii)-iteration_f(ii-1))<1e-4 || ii==max_step
        f=iteration_f(ii);
        x=iteration_x(:,ii);
        fprintf("steps = %d \n", ii-1)
        fprintf("f = %s \n",f)
        fprintf("x1 = %s , x2 = %s , x3 = %s \n",x(1),x(2),x(3))
        break
    end
    iteration_x(:,ii+1)=iteration_x(:,ii)-df;
end
```

result :

```
Command Window

>> hw2_1_Newtons_method
steps = 2
f = 2.066041e-58
x1 = 6.310887e-30 , x2 = 1.577722e-30 , x3 = 3.155444e-30
fx>>
```

Since the objective function is quadratic, Newton's method use only two steps to find the optima.

(c)

$$f = x_1^2 + 2x_1x_2 + 4x_1x_3 + 3x_2^2 + 2x_2x_3 + 5x_3^2$$

$$\begin{aligned} \Rightarrow \frac{\partial f}{\partial x_1} &= 2x_1 + 2x_2 + 4x_3 \\ \Rightarrow \frac{\partial f}{\partial x_2} &= 2x_1 + 6x_2 + 2x_3 \\ \Rightarrow \frac{\partial f}{\partial x_3} &= 4x_1 + 2x_2 + 10x_3 \end{aligned} \quad \Rightarrow H = \begin{bmatrix} 2 & 2 & 4 \\ 2 & 6 & 2 \\ 4 & 2 & 10 \end{bmatrix}$$

$$\text{Let } \frac{\partial f}{\partial x_1} = \frac{\partial f}{\partial x_2} = \frac{\partial f}{\partial x_3} = 0$$

$$\begin{bmatrix} 2 & 2 & 4 \\ 2 & 6 & 2 \\ 4 & 2 & 10 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \Rightarrow \begin{bmatrix} 2 & 2 & 4 \\ 0 & 4 & -2 \\ 0 & -2 & 2 \end{bmatrix} \Rightarrow \begin{bmatrix} 2 & 2 & 4 \\ 0 & 4 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} 2 & 2 & 4 \\ 0 & 4 & -2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \Rightarrow \begin{cases} x_1 = 0 \\ x_2 = 0 \\ x_3 = 0 \end{cases} \Rightarrow f(0,0,0) = 0 \quad \text{optima}$$

Problem 2 (40%)

For the function

$$f = 2x_1^2 - 3x_1x_2 + 8x_2^2 + x_1 - x_2 \quad (2)$$

1. find and proof the minimum analytically. (5%)
2. using the Newton's method to obtain the solution with the starting point $x_0 = (1, 1)^T$. (5%)
3. find the value α_k that is the solution to an exact line search $x_{k+1} = x_k + \alpha_k s_k$. Perform iterations using s_k from the gradient methods, starting from $x_0 = (1, 1)^T$. (15%)
4. compare your analytical result with Newton's method and gradient with line search. Write down your observations. (5%)

objective_function_hw2_2.m :

```
function [f,df,H]=objective_function_hw2_2(x)
f=2*x(1)^2-3*x(1)*x(2)+8*x(2)^2+x(1)-x(2);
df=zeros(2,1);
df(1)=4*x(1)-3*x(2)+1;
df(2)=-3*x(1)+16*x(2)-1;
H=[4 -3; -3 16];
```

(a)

$$\begin{aligned} f &= 2x_1^2 - 3x_1x_2 + 8x_2^2 + x_1 - x_2 \\ \Rightarrow \frac{\partial f}{\partial x_1} &= 4x_1 - 3x_2 + 1 \\ \Rightarrow \frac{\partial f}{\partial x_2} &= -3x_1 + 16x_2 - 1 \end{aligned} \quad \Rightarrow H = \begin{bmatrix} 4 & -3 \\ -3 & 16 \end{bmatrix}$$
$$\text{Let } \frac{\partial f}{\partial x_1} = \frac{\partial f}{\partial x_2} = 0 \Rightarrow \begin{cases} 4x_1 - 3x_2 = -1 \\ -3x_1 + 16x_2 = 1 \end{cases}$$
$$\begin{bmatrix} 4 & -3 \\ -3 & 16 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \Rightarrow \left[\begin{array}{cc|c} 4 & -3 & -1 \\ 0 & \frac{55}{4} & \frac{1}{4} \end{array} \right] \Rightarrow \begin{bmatrix} 4 & -3 \\ 0 & \frac{55}{4} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -1 \\ \frac{1}{4} \end{bmatrix}$$
$$\Rightarrow \begin{cases} x_1 = \frac{-13}{55} \\ x_2 = \frac{1}{55} \end{cases} \Rightarrow f\left(\frac{-13}{55}, \frac{1}{55}\right) = -0.1273 \quad \# \quad \text{optimal}$$

(b)

hw2_2_Newtons_method.m :

```
x0=[1;1];
max_step=100;
iteration_x=zeros(2,max_step);
iteration_f=zeros(1,max_step);
iteration_x(:,1)=x0;
f=0;
x=zeros(3,1);
for ii=1:max_step
    [iteration_f(ii),df,H]=objective_function_hw2_2(iteration_x(:,ii));
    if ii>1 && abs(iteration_f(ii)-iteration_f(ii-1))<1e-9 || ii==max_step
```

```

        f=iteration_f(ii);
        x=iteration_x(:,ii);
        fprintf("steps = %d \n", ii-1)
        fprintf("f = %s \n",f)
        fprintf("x1 = %s , x2 = %s \n",x(1),x(2))
        break
    end
    iteration_x(:,ii+1)=iteration_x(:,ii)-H^-1*df;
end

```

result :

Command Window

```

>> hw2_2_Newtons_method
steps = 2
f = -1.272727e-01
x1 = -2.363636e-01 , x2 = 1.818182e-02
fx>>

```

Since the objective function is quadratic, Newton's method use only two steps to find the optima.

(c)

hw2_2_optimize_apha_method.m :

```

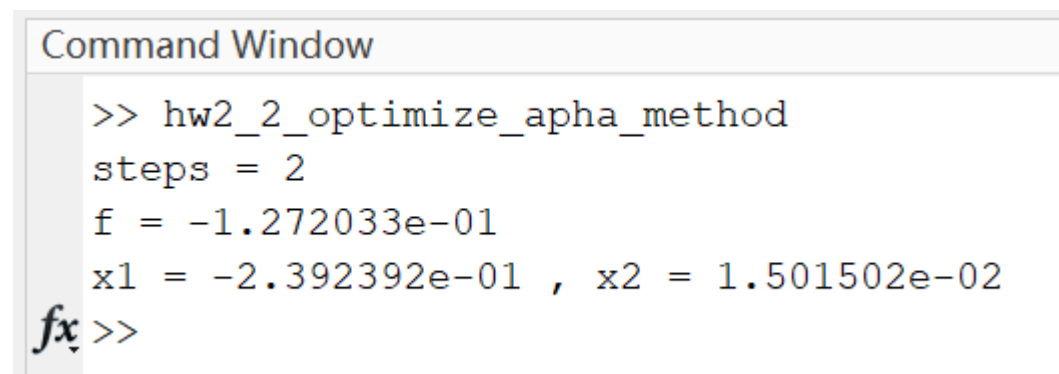
x0=[1;1];
max_step=100;
iteration_x=zeros(2,max_step);
iteration_f=zeros(1,max_step);
iteration_apha=zeros(2,max_step);
iteration_x(:,1)=x0;
a1=linspace(0,1,1000);
a2=linspace(0,1,1000);
n=size(a1,2);
try_apha=cell(n,n);
for aa=1:n
    for bb=1:n
        try_apha{aa,bb}=[a1(aa); a2(bb)];
    end
end
end

```

```

for ii=1:max_step
    [iteration_f(ii),df,H]=objective_function_hw2_2(iteration_x(:,ii));
    if ii>1 && abs(iteration_f(ii)-iteration_f(ii-1))<1e-15 || ii==max_step
        f=iteration_f(ii);
        x=iteration_x(:,ii);
        fprintf("steps = %d \n", ii-1)
        fprintf("f = %s \n",f)
        fprintf("x1 = %s , x2 = %s \n",x(1),x(2))
        break
    end
    for jj=1:n*n
        try_x=iteration_x(ii)-try_apha{jj}.*df;
        try_f=objective_function_hw2_2(try_x);
        if jj==1
            step_f=try_f;
            iteration_apha(:,ii)=try_apha{jj};
        elseif jj>1 && try_f<step_f
            step_f=try_f;
            iteration_apha(:,ii)=try_apha{jj};
        end
    end
    iteration_x(:,ii+1)=iteration_x(:,ii)-iteration_apha(:,ii).*df;
end
result :

```



```

Command Window

>> hw2_2_optimize_apha_method
steps = 2
f = -1.272033e-01
x1 = -2.392392e-01 , x2 = 1.501502e-02
fx>>

```

The starting point is really close to the optima point, so it didn't iteration to many times but try to figure out the correct α to step forward.

(d)

Although initial point $[1,1]^T$ is relatively close to the optima point, the gradient at that point is still too big to get to the lower position. In other words, if the α_k is

[1 1], the first steps will make the algorithm exceed the optima and go to a higher point. Thus, the α_k must be smaller than 1 in the first step.