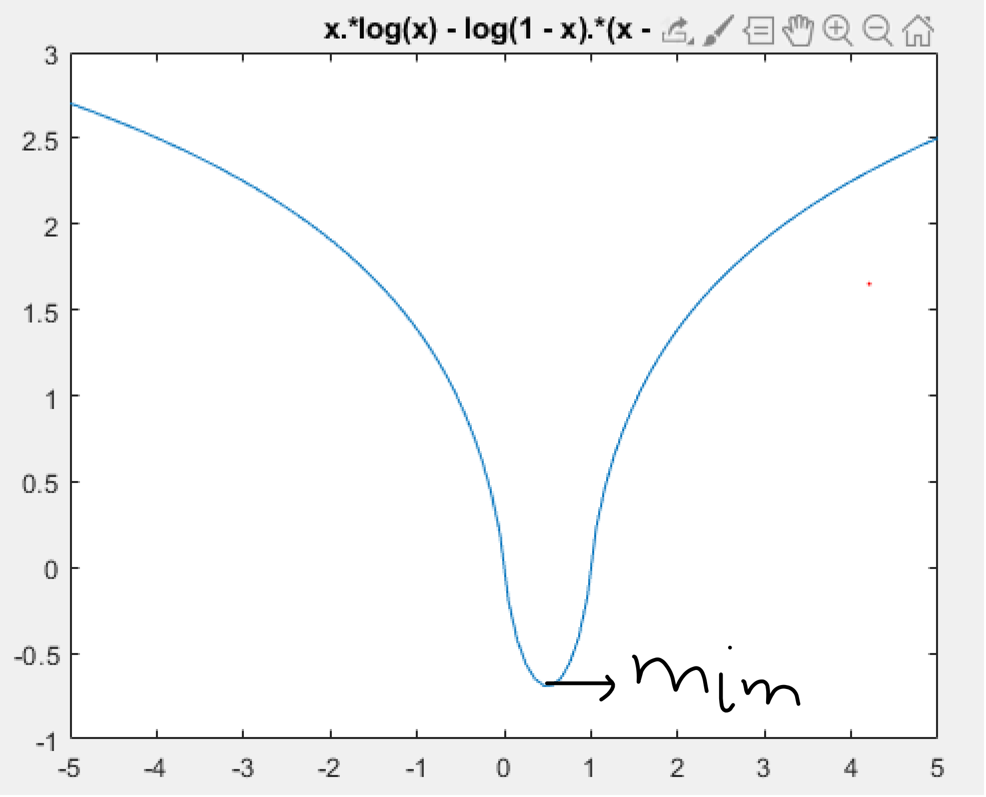
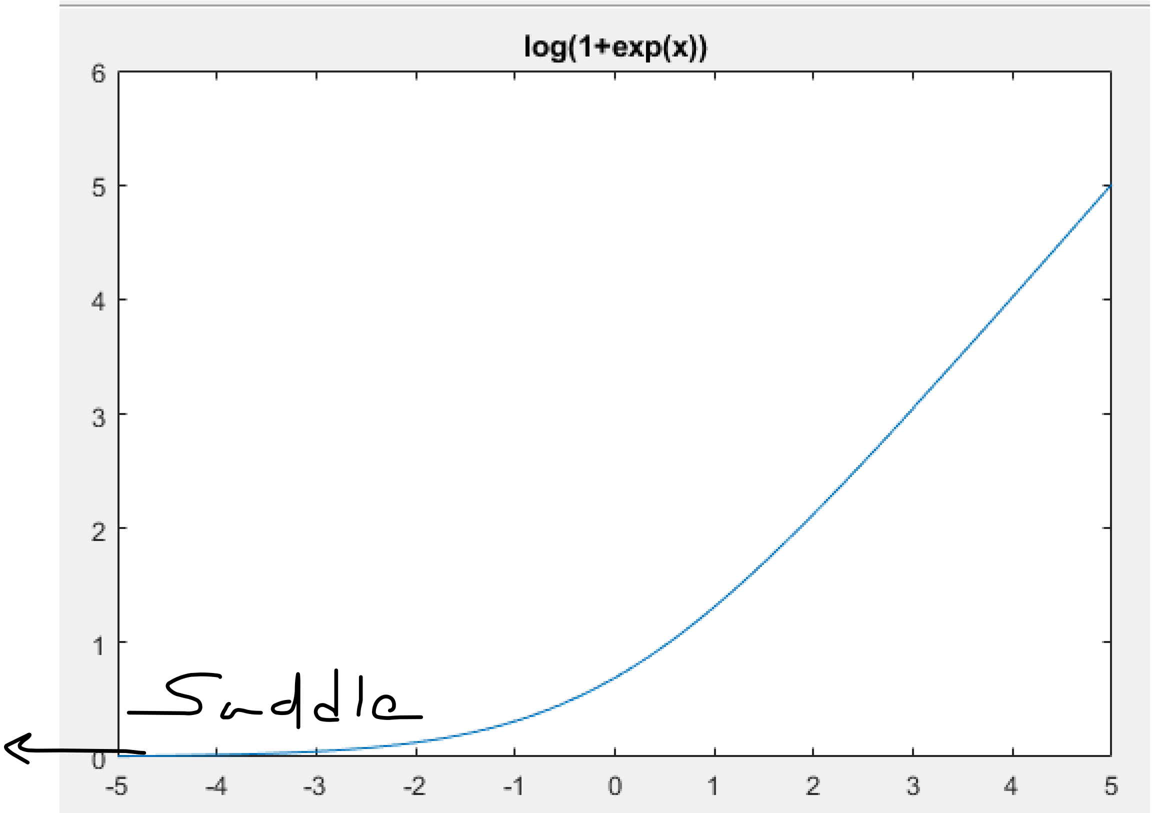
Problem 3.1
a.

```
%%
x=linspace(-5,5,100);
eqn = x.*log(x) - log(1 - x).*(x - 1);
plot(eqn,x);
title('x.*log(x) - log(1 - x).*(x - 1)')
```

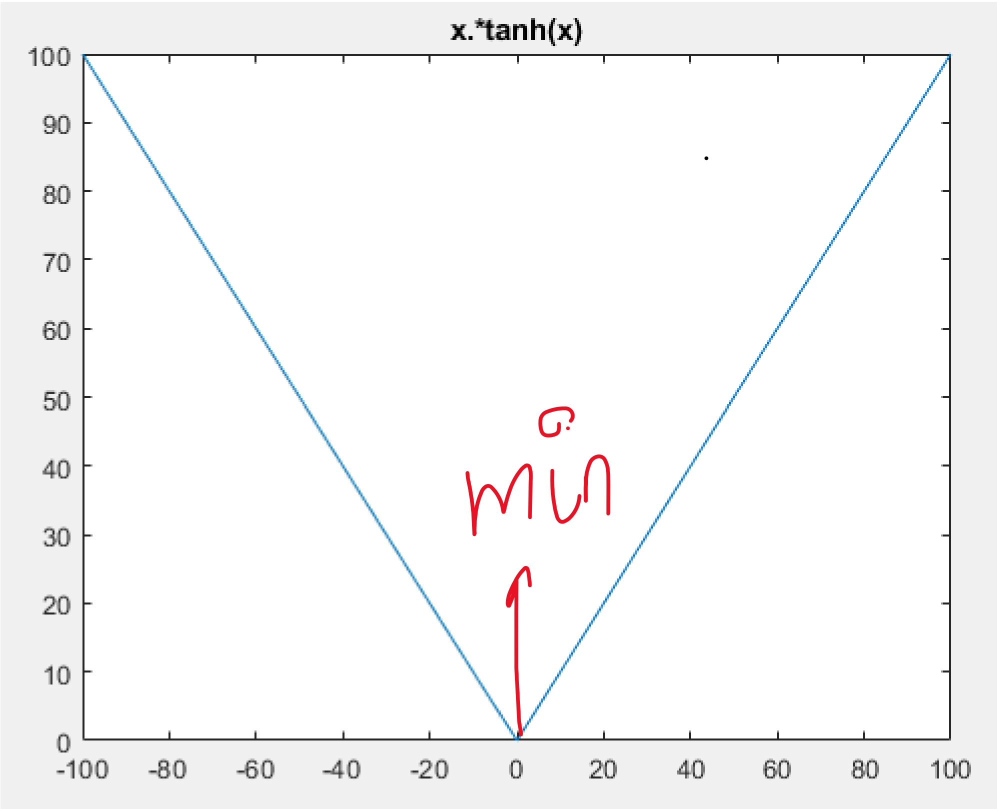


b.

```
x=linspace(-5,5,100);
eqn = log(1+exp(x));
plot(x,eqn);
title('log(1+exp(x))')
```




```
x = [-100:10:100];
y = x.*tanh(x);
plot(x, y)
title('x.*tanh(x)')
```



d.

```
C= [2 1;1 3]

b=[1; 1]

X= linsolve(C,-1*b)
```

Exercise

3.1 Condition for optimality.

1) $g(w) = w\log(w) + (1-w)\log(1-w)$. $\quad w \Rightarrow (0,1)$

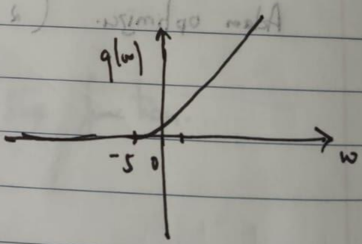
1st order condition, $\frac{d}{dw}(g(w))$,

$$= \cancel{w \times \frac{1}{w}} + \log w + \frac{1}{(1-w)}\cancel{(1-w)} + (-1)\log(1-w).$$

$$= 2 + \log w - \log(1-w) = 0$$

$$w = \frac{1}{(e^2+1)} \approx 0.1192$$

2) $g(w) = \log(1+e^w)$



1st order condition $\frac{d}{dw}(g(w))$

$$= \left(\frac{1}{1+e^w}\right) \cdot e^w = 0$$

$\therefore w = \infty$ (infinity)

3)

(3) Cannot solve

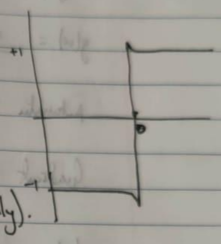
3) $g(w) = w \tanh(w)$

$\Rightarrow$ 1st order condition, $\frac{d}{dw}(g(w)) =$

$\Rightarrow \tanh(w) - w \times (\tanh(w)^2 - 1) = 0$

Cannot be solved.

unable to solve symbollically, $\delta = 0$ (numerically).



(4) $g(w) = \frac{1}{2} w^T C w + b^T w$ $\quad C = \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}$ $\quad b = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

2x2 matrix $\quad$ 2x1 vector

1st order condition, $\frac{d}{dw}(g(w))$

$\Rightarrow \nabla g(w) = \frac{1}{\cancel{2}} \times \cancel{2} C\bar{w} + b = C\bar{w} + b = 0.$

$C\bar{w} = -b$

$\Rightarrow \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix} \cdot \bar{w} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$ $\quad$ Soln is Stationary point of the original function.

$\Rightarrow \bar{w} = \begin{bmatrix} -0.4 \\ -0.2 \end{bmatrix}$ Ans.

Question 3.3

3 may

3.3 Rayleigh Quotient. 'N×N' matrix 'C'

$g(w) = \frac{w^T C w}{w^T w}$ where $w \neq 0_{N\times 1}$

Automatic differentiation in deeplearning toolbox

Gradient descent algorithm.

$g(w) = \frac{w^T C w}{w^T w}$

we know that Rayleigh's quotient is for vectors (w) & C (N×N) matrix.

~~Let $w = \begin{bmatrix} 1 \\ \vdots \\ \vdots \\ N \end{bmatrix}$ N×1 vector.~~

~~$w^T = [1 \; . \; . \; . \; . \; . \; N]$ 1×N vector~~

~~denominator $w^T \cdot w$ = (scalar)~~

Let ~~$N = v^T A v$~~, ~~$D = v^T v$~~

Let $N = w^T C w$
$D = w^T w$

then $g'(v) = \frac{d}{dv}\left(\frac{N}{D}\right)$

$= \frac{N'}{D} - \frac{ND'}{D^2} = \frac{N'D - ND'}{D^2}$

when finding stationary point, $g'(w) = 0$

Can we solve in matlab. this matrix vector differentiation?

3.5) taking a lot of time to compute 1000 iterations?

(3.8) Matrix size.

$$\Rightarrow \frac{N'D - ND'}{D^2} = 0$$

$$\Rightarrow N'D - ND' = 0$$

$$\Rightarrow N'D = ND' \quad \Rightarrow \ldots \text{(1)}$$

where $N' = \frac{d}{dw}(w^T C w) = w^T (C + C^T)$

& $D = w^T D$, $D' = 2w^T$

Substituting in (1),

$$\cancel{\boxed{w^T (C + C^T) . w^T . w = (w^T C w) . 2w^T}} \quad \text{(2)}$$

~~Solve eq. (2) to determine w value.~~

Assuming C is symmetric, $C + C^T = 2C$

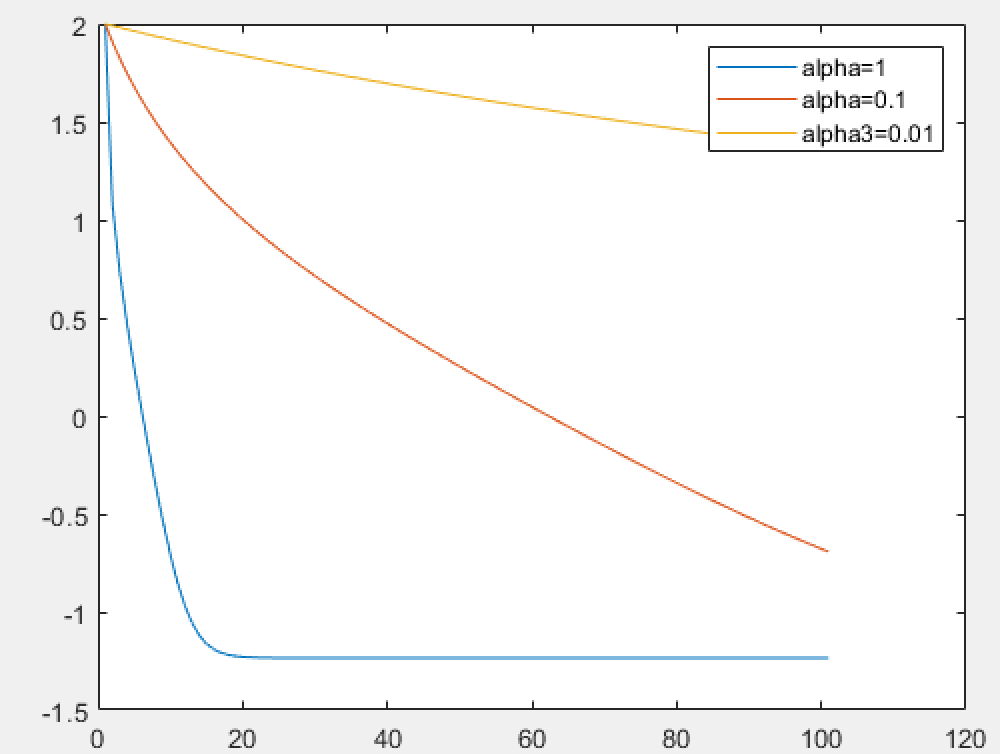
which gives us,

$$g'(w) = \frac{2w^T C}{w^T w} - \frac{w^T C w . 2w^T}{(w^T w)^2} = \frac{2}{w^T w}\left[w^T C - g(w) \right.$$

The stationary points, the eigenvectors of 'C' are the stationary points for $g(w)$.

Problem 3.5

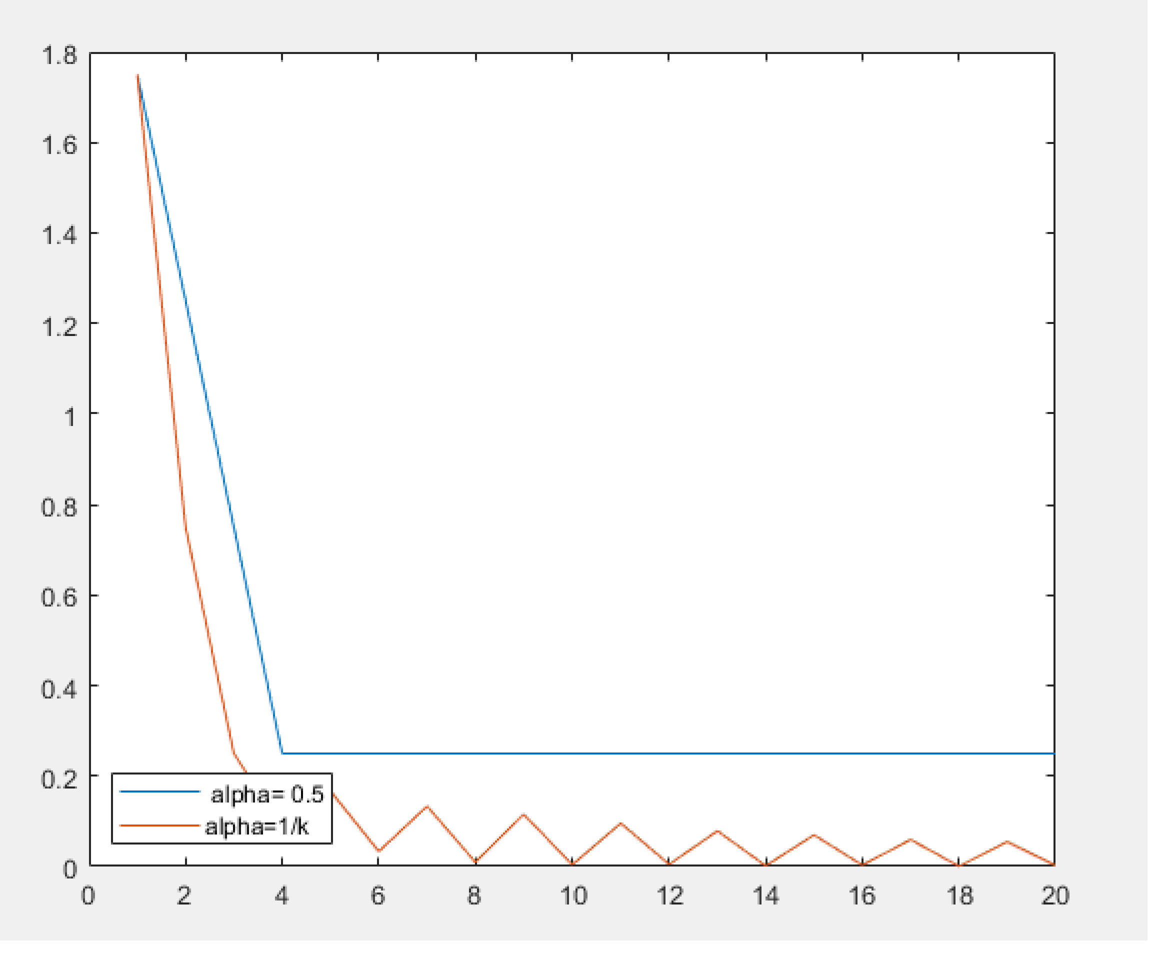
```
x=2;
alpha=1;
max_its = 100;
out = (x^4+x^2+10*x)/50;
iter1=1;
iter2=1;
iter3=1;
doutsym=(2*x^3)/25 + x/25 + 1/5;

A1=[];
A1(1,1)=x;
A2=[];
A2(1,1)=x;
A3=[];
A3(1,1)=x;

%%
 while iter1 <= max_its
% doutsym= gradient(out);
doutsym=(2*x^3)/25 + x/25 + 1/5;
x = x - alpha*doutsym;
A1(iter1+1,:)=x;
iter1=iter1+1;
 end
 alpha2=0.1;
 x=2
 while iter2 <= max_its
% doutsym= gradient(out);
doutsym=(2*x^3)/25 + x/25 + 1/5;
x = x - alpha2*doutsym;
A2(iter2+1,:)=x;
iter2=iter2+1;
 end
 alpha3=0.01;
 x=2;
  while iter3 <= max_its
% doutsym= gradient(out);
doutsym=(2*x^3)/25 + x/25 + 1/5;
x = x - alpha3*doutsym;
A3(iter3+1,:)=x;
iter3=iter3+1;
 end
plot (A1)
hold on
plot (A2)
hold on
plot (A3)
legend('alpha=1','alpha=0.1','alpha3=0.01');
hold off
```



Problem 3.6

```
syms x
out = abs(x);
y=linspace(-2,2,4);
z=subs(out,{y});
plot(y,z)
%%
iter1=1;
iter2=1;
iter3=1;
max_its = 20;
% doutsym=gradient(out)
x=1.75;
A1=[];
A1=[x iter1];
A2=[];
A2=[x iter2];
%%
alpha=0.5;
 while iter1 < max_its
     if x>0
         df=1;
     else
         df=-1;
     end
% doutsym= gradient(out);
x = x - alpha*df;
A1(iter1+1,:)=[x iter1+1];
iter1=iter1+1;
 end
 alpha2=1/iter2;
 x=1.75
 while iter2 < max_its
          if x>0
```

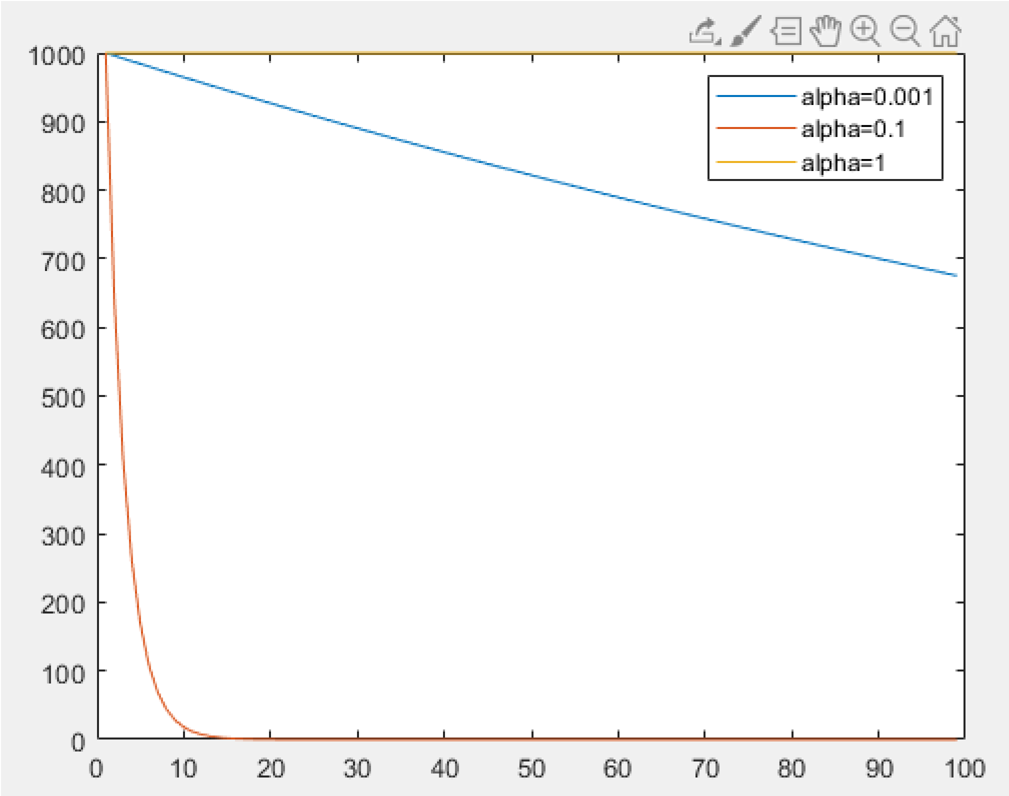
```
        df=1;
    else
        df=-1;
    end
 alpha2=1/iter2;
% doutsym= gradient(out);
x = x - alpha2*df;
A2(iter2+1,:)=[x iter2+1];
iter2=iter2+1;
 end

% plot (A1)
% hold on
% plot (A2)
% hold off

plot ((A1(:,2)),(abs(A1(:,1))))
hold on
plot ((A2(:,2)),(abs(A2(:,1))))
hold off
legend({' alpha= 0.5','alpha=1/k'},'Location','southwest')
```



3.8

```
syms x
out = transpose(x)*x;
y=10*ones(1,10);
z=subs(out,{y});
%%
%diff(transpose(x)*x)=2*transpose(x)
%
%%
iter1=1;
iter2=1;
iter3=1;
max_its = 100;
% doutsym=gradient(out)
A1=[];
A1=[y(iter1,1) iter1];
A2=[];
%A2=[y iter2];
%%
alpha=0.001;
y=10*ones(10,1);
df=2*(y);
 while iter1 < max_its
     z=transpose(y)*y;
A1(iter1,:)=[z iter1];
% doutsym= gradient(out);
y = y - alpha*df;
df=2*(y);
iter1=iter1+1;

r=y;
 end

%%
iter1=1;
iter2=1;
iter3=1;

% doutsym=gradient(out)
A2=[];
A2=[y(iter1,1) iter1];

%%
y=10*ones(10,1);
alpha=0.1;
df=2*(y);

z=subs(out,{y});
 while iter2 < max_its
     z=transpose(y)*y;
A2(iter2,:)=[z iter2];
% doutsym= gradient(out);
y = y - alpha*df;
df=2*(y);
iter2=iter2+1;

r=y;
 end
%plot(A2(:,2),A2(:,1))
```

```
%hold on
%%
%%
iter1=1;
iter2=1;
iter3=1;

% doutsym=gradient(out)
A3=[];
A3=[y(iter3,1) iter3];

%%
y=10*ones(10,1);
alpha=1;
df=2*(y);
y=10*ones(10,1);
z=subs(out,{y});
 while iter3 < max_its
% doutsym= gradient(out);
z=dot(y,transpose(y));
A3(iter3,:)=[z iter3];
y = y - alpha*df;
df=2*(y);
iter3=iter3+1;

r=y;
 end
%plot(A3(:,2),A3(:,1))
%hold off
plot(A1(:,2),A1(:,1))
hold on
plot(A2(:,2),A2(:,1))
hold on
plot(A3(:,2),A3(:,1))
legend('alpha=0.001','alpha=0.1','alpha=1');
hold off
```



1000
900
800
700
600
500
400
300
200
100
0
0
10
20
30
40
50
60
70
80
90
100
alpha=0.001
alpha=0.1
alpha=1