1. bisection:

$f'(x) = -e^{-x} + \sin x \qquad f'(0) = -1 < 0 \qquad f'(1) = 0.474 > 0$

Iteration	1	2	3	4	5	6	7	8
x_m	0.5	0.75	0.625	0.5625	0.59375	0.578125	0.5859375	0.58984375
$f'(x_m)$	-0.127	0.209	0.0498	-0.0365	0.0072	-0.0145	-0.0036	0.001817
interval	[0.5, 1]	[0.5, 0.75]	[0.5, 0.625]	[0.5625, 0.625]	[0.5625, 0.59375]	[0.578125, 0.59375]	[0.5859375, 0.59375]	[0.5859375, 0.58984375]
ε	0.5	0.25	0.125	0.0625	0.03125	0.015625	0.0078125	0.00390625

Iteration	9	10	11	12	13	14	15
x_m	0.587890625	0.5874140625	0.58815234375	0.588521484375	0.588706054688	0.588613769532	0.588567626953
$f'(x_m)$	0.000496	-0.001552	-0.000528	-0.00001562	0.0002403	0.0001124	0.0000484
interval	[0.5859375, 0.587890625]	[0.5874140625, 0.587890625]	[0.58815234375, 0.587890625]	[0.588521484375, 0.587890625]	[0.588521484375, 0.588706054688]	[0.588521484375, 0.588613769532]	[0.588521484375, 0.588567626953]
ε	0.001953125	0.0009765625	0.00048828125	0.000244140625	0.0001220703125	0.00006103515	0.000030517578

Iteration	16	17
x_m	0.588544555664	0.58853302002
$f'(x_m)$	0.0000164	3.828×10^{-7}
interval	[0.588521484375, 0.588544555664]	[0.588521484375, 0.58853302002]
ε	0.00001525878	0.00000762939

Compare the number of iterations:

bisection: $(1-0)(\frac{1}{2})^{n_1} \le 10^{-5} \qquad n_1 \in N$

$5\log_2 10 \le n_1 \Rightarrow n_1 \geq 16.61 \Rightarrow \min(n_1) = 17$

gold section: $(1-0)(0.618)^{n_2} \le 10^{-5} \qquad n_2 \in N$

$-5\log_{0.618} 10 \le n_2 \Rightarrow n_2 \geq 23.92 \Rightarrow \min(n_2) = 24$

$\min(n_1) < \min(n_2)$

The number of iterations of bisection is less than gold section.

```
% Bisection method
xl = 0;
xr = 1;
iter_bi = 0;
while xr - xl > 10^(-5)
    xm = 0.5 * (xl + xr);
    if derivative(xm) > 0
        xr = xm;
    else
        xl = xm;
    end
    iter_bi = iter_bi + 1;
end
disp('the solution by bisection is ')
disp(xm)
disp('the number of iterations by bisection is')
disp(iter_bi)
```

```
%% Golden section method
xl = 0;
xr = 1;
phi = (3 - sqrt(5)) / 2;
iter_gold = 0;
while xr - xl > 10^(-5)
    x1 = phi * xr + (1 - phi) * xl;
    x2 = phi * xl + (1 - phi) * xr;
    if  f(x1) <  f(x2)
        xr = x2;
    else
        xl = x1;
    end
    iter_gold = iter_gold + 1;
end
disp('the solution by golden section is ')
disp(0.5*(xl + xr))
disp('the number of iterations by golden section is')
disp(iter_gold)
```

```
>> search
the solution by bisection is
    0.5885

the number of iterations by bisection is
    17

the solution by golden section is
    0.5885

the number of iterations by golden section is
    24
```

2. a) $\nabla f(x)^T d = -(\nabla f(x)_j)\nabla f(x)^T e_j = -(\nabla f(x)_j)^2 = -\|\nabla f(x)\|_\infty^2 < 0$ since $\nabla f(x) \neq 0$

b) $\nabla f(x)^T d = \nabla f(x)^T\left(-\frac{1}{\|\nabla f(x)\|}\nabla f(x)\right) = -\frac{1}{\|\nabla f(x)\|}(\nabla f(x)^T \nabla f(x)) = -\frac{\|\nabla f(x)\|^2}{\|\nabla f(x)\|} = -\|\nabla f(x)\| < 0$ since $\nabla f(x) \neq 0$

c) $\nabla f(x)^T d = \sum_{i=1}^{n}\left[(\nabla f(x)_i)\cdot\frac{-(\nabla f(x)_i)}{\max\{\nabla^2 f(x)_{ii}, \varepsilon\}}\right] = \sum_{i=1}^{n} -(\nabla f(x)_i)^2 \cdot \frac{1}{\max\{\nabla^2 f(x)_{ii}, \varepsilon\}}$

Since $-(\nabla f(x)_i)^2 \leq 0$, $\frac{1}{\max\{\nabla^2 f(x)_{ii}, \varepsilon\}} > 0$

Then $-(\nabla f(x)_i)^2 \cdot \frac{1}{\max\{\nabla^2 f(x)_{ii}, \varepsilon\}} \leq 0$

So $\nabla f(x)^T d = \sum_{i=1}^{n} -(\nabla f(x)_i)^2 \cdot \frac{1}{\max\{\nabla^2 f(x)_{ii}, \varepsilon\}} < 0$ Since $\nabla f(x) \neq 0$

3. a

```
clear;
clc;
x = [3; -3];
tol = 10^(-5);
maxit = 100;
iter = 0;
a = 2;
while norm(gradient(x)) > tol
 % Doing exact line search
    xl = x;
    xr = x - a * gradient(x);
    phi = (3 - sqrt(5)) / 2;

    it = 0;
    while norm(xr - xl) > 10^(-6) && it < maxit
        x1 = phi * xr + (1 - phi) * xl;
        x2 = phi * xl + (1 - phi) * xr;
        if  f(x1) >  f(x2)
            xl = x1;
        else
            xr = x2;
        end
        it = it + 1;
    end
   xtemp = (xr + xl) / 2;

    iter = iter + 1;
    x = xtemp;

end
disp('exact line:')
disp('x = ')
disp(x)
disp('the number of iterations')
disp(iter)
```

```
exact line:
x =
   -2.0000
    1.0000

the number of iterations
    14
```

```
clear;clc;
x = [3; -3];
tol = 10^(-5);
gamma = 0.1;
sigma = 0.5;

iter = 0;
while norm(gradient(x)) > tol
     d = gradient(x);
     t = 1;

     xtemp = x - t * d;
     while f(xtemp) >= f(x) - gamma * t * gradient(x)' * d
         t = t * sigma;
         xtemp = x - t * d;
     end
    iter = iter + 1;
    x = xtemp;
end
disp('backtracking: ')
disp('x = ')
disp(x)
disp('the number of iterations')
disp(iter)
```

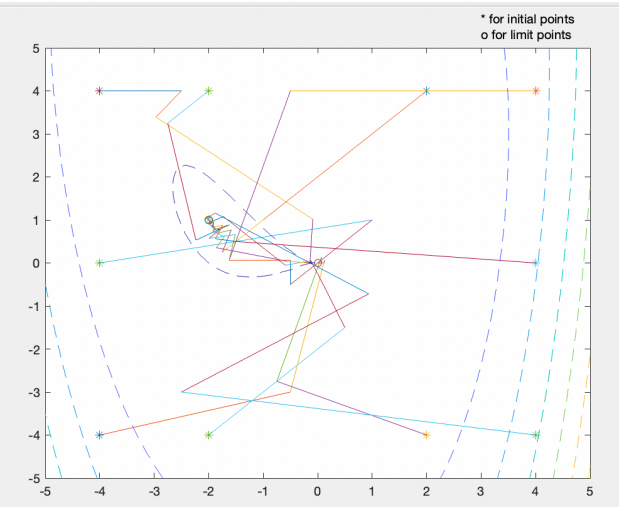
```
backtracking:
x =
   -2.0000
    1.0000

the number of iterations
    36
```

exact line search and backtracking converge to $(-2, 1)$

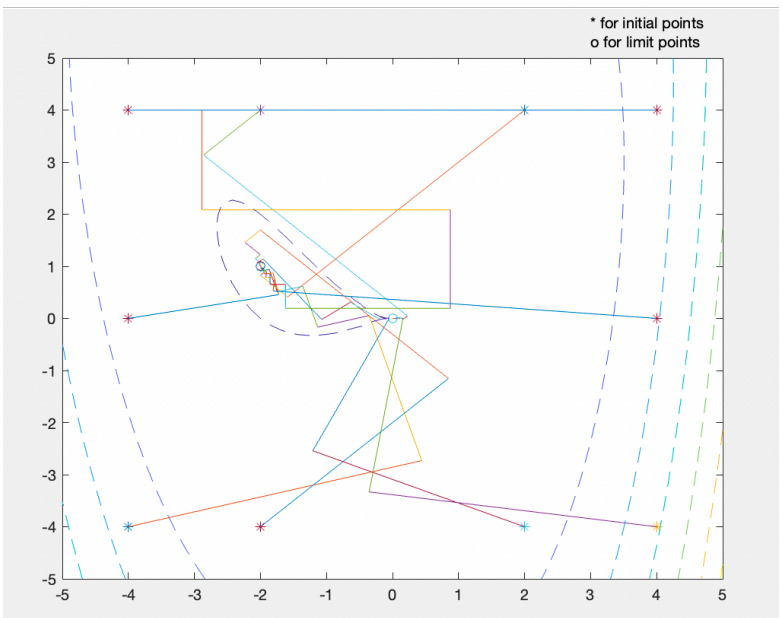
the number of iterations of exact line search is less than backtracking.

b. i)

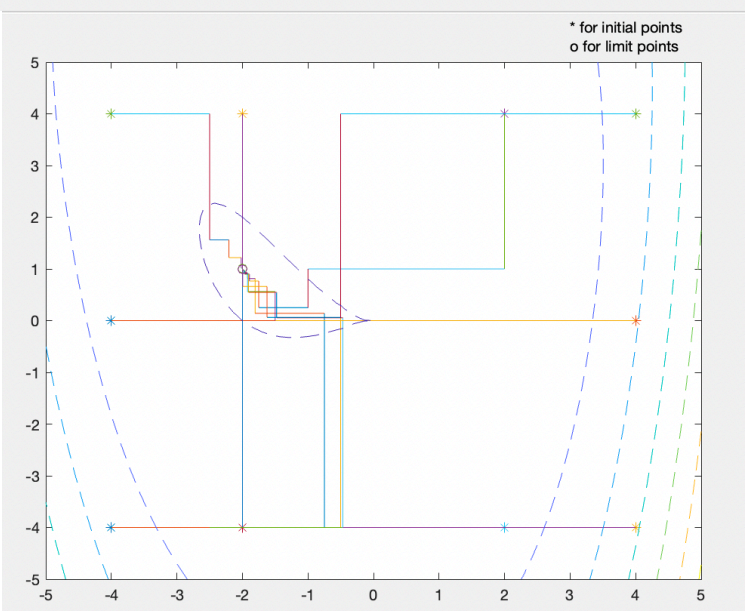
```
clear;clc;
X = [-4 -4 -4 -2 -2 2 2 4 4 4;
     -4 0 4 -4 4 -4 4 -4 0 4];
x = [0;0];

tol = 10^(-5);
gamma = 0.1;
sigma = 0.5;

for i = 1:10
    x(1) = X(1, i);
    x(2) = X(2, i);
    plot(x(1),x(2),'*');%'*' stands for initial points
    hold on;
    while norm(gradient(x)) > tol
         d = gradient(x);
         t = 1;

         xtemp = x - t * d;
         while f(xtemp) >= f(x) - gamma * t * gradient(x)' * d
             t = t * sigma;
             xtemp = x - t * d;
         end
        plot([x(1), xtemp(1)], [x(2), xtemp(2)], '-');
        hold on;
        x = xtemp;
    end
    plot(x(1),x(2),'o');%o stands for limit points
    hold on;
end
func = @(n,n) n^4 + 2*(n-n)*n^2+4*n^2;
fcontour(func,'--');
hold on;
txt = {'* for initial points','o for limit points'};
text(3,5.5,txt);
hold on;
```




```
X = [-4 -4 -4 -2 -2 2 2 4 4 4;
     -4 0 4 -4 4 -4 4 -4 0 4];
x = [0;0];

tol = 10^(-5);%for gradient descent
tol_search = 10^(-6);
maxit = 100;
a = 2;

for i = 1:10
    x(1) = X(1,i);
    x(2) = X(2,i);
    plot(x(1),x(2),'*');%'*' stands for initial points
    hold on;
    while norm(gradient(x)) > tol
        xl = x;
        xr = x - a * gradient(x);
        phi = (3 - sqrt(5)) / 2;

        it = 0;
        while norm(xr - xl) > tol_search && it < maxit
            x1 = phi * xr + (1 - phi) * xl;
            x2 = phi * xl + (1 - phi) * xr;
            if  f(x1) >  f(x2)
                xl = x1;
            else
                xr = x2;
            end
            it = it + 1;
        end
        xtemp = (xr + xl) / 2;

        plot([x(1), xtemp(1)], [x(2), xtemp(2)], '-');
        hold on;
        x = xtemp;
    end
    plot(x(1),x(2),'o');%o stands for limit points
    hold on;
end
```




```
X = [-4 -4 -4 -2 -2 2 2 4 4 4;
     -4 0 4 -4 4 -4 4 -4 0 4];
x = [0;0];

tol = 10^(-5);
gamma = 0.1;
sigma = 0.5;

for i = 1:10
    x(1) = X(1, i);
    x(2) = X(2, i);
    plot(x(1),x(2),'*');%'*' stands for initial points
    hold on;
    while norm(gradient(x)) > tol
        d = gradient(x);
        if abs(d(1)) > abs(d(2))
            d(2) = 0;
        else
            d(1) = 0;
        end

        t = 1;
        xtemp = x - t * d;
        while f(xtemp) >= f(x) - gamma * t * gradient(x)' * d
            t = t * sigma;
            xtemp = x - t * d;
        end
        plot([x(1), xtemp(1)], [x(2), xtemp(2)], '-');
        hold on;
        x = xtemp;
    end
    plot(x(1),x(2),'o');%o stands for limit points
    hold on;
end
func = @(m,n) m^4 + 2*(m-n)*m^2+4*n^2;
fcontour(func,'--');
hold on;
txt = {'* for initial points','o for limit points'};
text(3,5.5,txt);
hold on;
```



4 a

```
clear;clc;
x = [-1;-0.5];
tol = 10^(-7);
sigma = 0.5;
gamma = 10^(-4);
gamma1 = 10^(-6);
gamma2 = 0.1;

alwaysUtilizeNewtonDirection = 1;
alwayUseFullStepSize = 1;
iter = 0;
while norm(gradient(x)) > tol
    s = -(hessian(x))\gradient(x);
    if -gradient(x)'*s >= gamma1*min([1;(norm(s))^(gamma2)])*(norm(s))^2
        d = s;
    else
        d = -gradient(x);
        alwaysUtilizeNewtonDirection = 0;
    end
    t = 1;
    xtemp = x + t * d;
    while f(xtemp) >= f(x) + gamma * t * gradient(x)' * d
        t = t * sigma;
        xtemp = x + t * d;
        alwayUseFullStepSize = 0;
    end
    x = xtemp;
    iter = iter + 1;
end
```

```
globalized Newton method:
x =
   0.999999999999661
   0.999999999999000

the number of iterations
    20

the Newton method always utilizes the Newton direction
the method does not always use full step size
```

```
x = [-1; -0.5];
tol = 10^(-7);
gamma = 10^(-4);
sigma = 0.5;
iter = 0;
while norm(gradient(x)) > tol
     d = gradient(x);
     t = 1;
     xtemp = x - t * d;
     while f(xtemp) >= f(x) - gamma * t * gradient(x)' * d
         t = t * sigma;
         xtemp = x - t * d;
     end
    x = xtemp;
    iter = iter + 1;
end
disp('backtracking: ')
disp('x = ')
disp(x)
disp('the number of iterations')
disp(iter)
```

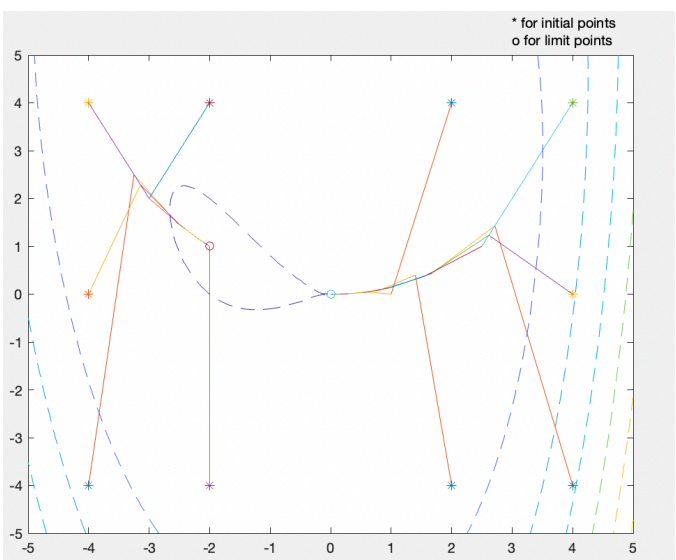
```
backtracking:
x =
    1.0000
    1.0000

the number of iterations
       16896
```

The number of iterations of the Globalized Newton Method is less than the gradient method with backtracking.
the Globalized Newton Method may have better performance.

Newton method always utilize the Newton direction in this case.
the method doesn't always use full step sizes in this case.

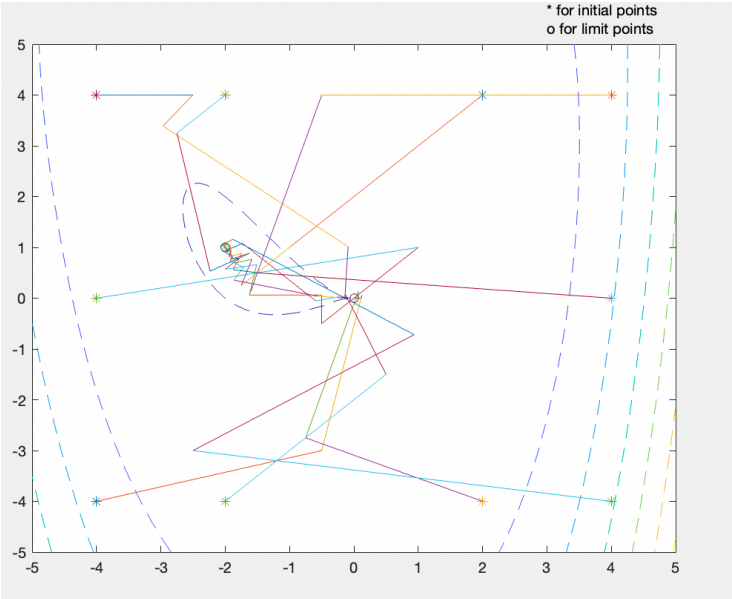
b.

```
X = [-4 -4 -4 -2 -2 2 2 4 4 4;
     -4 0 4 -4 4 -4 4 -4 0 4];
x = [0;0];

gamma1 = 10^(-6);
gamma2 = 0.1;
tol = 10^(-5);
gamma = 0.1;
sigma = 0.5;

for i = 1:10
    x(1) = X(1, i);
    x(2) = X(2, i);
    plot(x(1),x(2),'*');%'*' stands for initial points
    hold on;

    iter = 0;
    while norm(gradient(x)) > tol
        s = -(hessian(x))\gradient(x);
        if -gradient(x)'*s >= gamma1*min([1;(norm(s))^(gamma2)])*(norm(s))^2
            d = s;
        else
            d = -gradient(x);
        end
        t = 1;
        xtemp = x + t * d;
        while f(xtemp) >= f(x) + gamma * t * gradient(x)' * d
            t = t * sigma;
            xtemp = x + t * d;
        end
        plot([x(1), xtemp(1)], [x(2), xtemp(2)], '-');
        hold on;
        x = xtemp;
        iter = iter + 1;
    end
    plot(x(1),x(2),'o');%o stands for limit points
    hold on;
    disp('the number of iterations');
    disp(iter);
end
```




```
the number of iterations
     6

the number of iterations
     6

the number of iterations
     6

the number of iterations
     1

Warning: Matrix is close to s
> In newton (line 20)

the number of iterations
     6

the number of iterations
    12

the number of iterations
    12

the number of iterations
    14

the number of iterations
    14

the number of iterations
    14
```

```
X = [-4 -4 -4 -2 -2 2 2 4 4 4;
     -4 0 4 -4 4 -4 4 -4 0 4];
x = [0;0];

tol = 10^(-5);
gamma = 0.1;
sigma = 0.5;

for i = 1:10
    x(1) = X(1, i);
    x(2) = X(2, i);
    plot(x(1),x(2),'*');%'*' stands for initial points
    hold on;
    iter = 0;
    while norm(gradient(x)) > tol
        d = gradient(x);
        t = 1;

        xtemp = x - t * d;
        while f(xtemp) >= f(x) - gamma * t * gradient(x)' * d
            t = t * sigma;
            xtemp = x - t * d;
        end
        plot([x(1), xtemp(1)], [x(2), xtemp(2)], '-');
        hold on;
        x = xtemp;
        iter = iter + 1;
    end
    plot(x(1),x(2),'o');%o stands for limit points
    hold on;
    disp('the number of iterations');
    disp(iter);
end
func = @(m,n) m^4 + 2*(m-n)*m^2+4*n^2;
fcontour(func,'--');
hold on;
txt = {'* for initial points','o for limit points'};
text(3,5.5,txt);
hold on;
```




```
the number of iterations
   499

the number of iterations
    35

the number of iterations
    38

the number of iterations
    40

the number of iterations
    31

the number of iterations
   395

the number of iterations
    30

the number of iterations
    34

the number of iterations
    36

the number of iterations
    33
```