MATLAB 第 3 次作业参考答案

几点反馈：

- 作业的参考答案来自同学们的作业，只是稍作调整，为保护隐私，隐去姓名。
- 请注意作业格式。作业的 WORD 书写样例、文件命名格式之前都发给过大家，请大家仔细阅读，参照执行。
- 提交作业前请检查一下是否缺文件，建议大家将 WORD 文档与源代码文件夹打包到一个 zip 直接上传至 CANVAS。

第一题

hw3_1_1.m

```
%本文件用于对多项式求根
%用符号函数求根
syms x;
f = 0.5*x^3 -8*x -4;
x1 = solve(f==0,x);
x1 = double(x1)
%用 roots 函数求根
p = [0.5 0 -8 -4];
x2 = roots(p)
```

输出结果：

```
>> hw3_1_1

x1 =

   -3.7216
   -0.5082
    4.2298


x2 =

    4.2298
   -3.7216
   -0.5082
```

hw3_1_2.m

```
f=@(x) 0.5*x.^3-8*x-4;

% 二分法
```

```
p=6;
TOL=0.5*10^(-p);
xb=zeros(3,1);
a1=-4; b1=-3;
xb(1)=Bisection(f,a1,b1,TOL);
a2=-1; b2=0;
xb(2)=Bisection(f,a2,b2,TOL);
a3=4; b3=5;
xb(3)=Bisection(f,a3,b3,TOL);
% fzero
xz=zeros(3,1);
xz(1)=fzero(f,-4);
xz(2)=fzero(f,-1);
xz(3)=fzero(f,4);
fprintf("二分法\n")
disp(xb)
fprintf("fzero\n")
disp(xz)
```

Bisection.m

```
function x=Bisection(f,a,b,TOL)
    % 检验 a 和 b 是否满足条件
    if(sign(f(a))==sign(f(b)))
        error('wrong input');
    end
    while (b-a)/2 > TOL
        c=(a+b)/2;
        fc=f(c);
        if(fc==0)
            break;
        end
        if sign(fc)==sign(f(b))
            b=c;
        else
            a=c;
        end
    end
    x=(a+b)/2;
end
```

二分法的 3 组迭代初值（对应 3 个根）为：

$$\begin{cases} a_1 = -4 \\ b_1 = -3 \end{cases}$$

$$\begin{cases} a_2 = -1 \\ b_2 = 0 \end{cases}$$

$$\begin{cases} a_3 = 4 \\ b_3 = 5 \end{cases}$$

fzero 的 3 个迭代初值（对应 3 个根）为：

$x_1 = -4$

$x_2 = -1$

$x_3 = 4$

执行结果如下：

```
二分法
  -3.721611499786377
  -0.508203029632568
   4.229815006256104

fzero
  -3.721611706223407
  -0.508203376730105
   4.229815082953511
```

注意：很多同学没有正确设置二分法的终止条件，精确到小数点后 6 位对应的终止条件应该是(b-a)/2<0.5e-6;

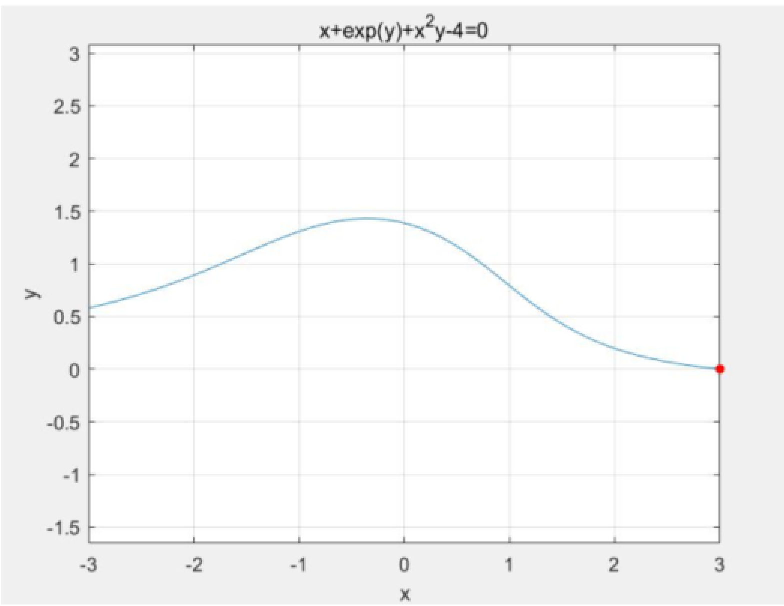
第二题

hw3_2.m

```
%用匿名函数表示出 y 关于 x 的表达式
y = @(x) fzero(@(y) x+exp(y)+x^2*y-4,1);
%绘制 y 关于 x 的图像
x = linspace(-3,3,200);
yo = zeros(200,1);
for i = 1:200
    yo(i) = y(x(i));
end
figure;
plot(x,yo);
```

```
title('x+exp(y)+x^2y-4=0');
xlabel('x'); ylabel('y');
axis equal; grid on;
hold on;
%二分法迭代求解， 初始区间为[2.5,3.5],精确到小数点后六位
a = 2.5; b = 3.5;
format long;
while (b-a)/2>0.5e-6
    c = (a+b)/2;
    fc = y(c);
    if fc==0
        break;
    elseif sign(fc)==sign(f(b))
        b = c;
    else
        a = c;
    end
    x = c;
end
display('二分法所求零点为');
display(x);
plot(x,y(x),'r.','MarkerSize',16);
```

图像为：



二分法求得零点为：

```
二分法所求零点为

x =

    2.999999046325684
```

第三题

（1）$g(x)=\frac{1}{2}(x+\frac{x_0}{x})$，$x_0$为被求平方根的数。

迭代的收敛速度取决于函数在不动点处的导数，因此应当使$g(x)$的导数尽可能小。如上选取的$g(x)$在$x=\sqrt{x_0}$的导数为 0，因此收敛速度很快。

mysqrt.m

```
function X=mysqrt(x)
    X=x;
    f=@(t) 0.5*(t+x./t);
    k=1;
    while(1)
        X=f(X);
        k=k+1;
        if(norm(X-sqrt(x))<1e-6)
            break
        end
    end
end
```

（2）已知$x^2-x_0=0$，若化为$x=\frac{x_0}{x}\Leftrightarrow g(x)$，则$g(x)$在$x=\sqrt{x_0}$的导数为 1，此时无法收敛，所以这个$g(x)$无法用于不动点法。

第四题

迭代初值：

$\begin{cases}x_{1a}=0.2\\x_{2a}=0.2\end{cases}$，$\begin{cases}x_{1b}=-0.2\\x_{2b}=0.2\end{cases}$

$g(x)$ 构造：

$$g(x)=\begin{cases} \dfrac{\dfrac{2x_2}{5x_1}+x_1}{2} \\ 0.25(\sin x_1+\cos x_2) \end{cases}$$

选择原因：转化简便，收敛快。

计算结果：

```
Solution1:
x1=0.360620
x2=0.325117
Solution2:
x1=-0.268134
x2=0.179740
```

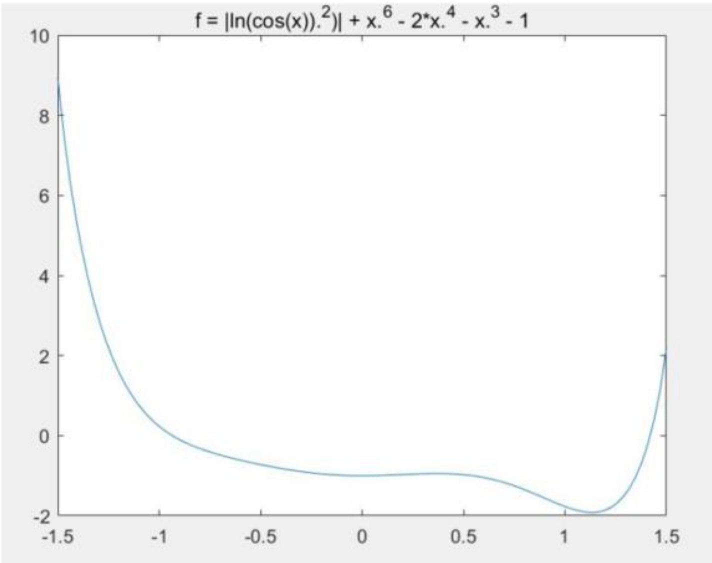
hw3_4.m

```
clear,clc;
g=@(x)[(0.4*x(2)/x(1)+x(1))/2;
      0.25*(sin(x(1))+cos(x(2)))];
x0=[0.2 -0.2;0.2 0.2];
x1=g(x0);
while norm(x1-x0)>1e-6
      x0=x1;
      x1=g(x0);
end
fprintf('Solution: \nx1=%f\nx2=%f\n',x1(1),x1(2));
```

这一题部分同学因为没有预判解的情况而漏解

第五题

（1） 首先做出图像为：



观察两个零点近似值分别为-1，1.3，用 Newton’s Method 和 The Secant Method 分别编程求解。

hw3_5.m

```
f = @(x) abs(log((cos(x)).^2)) + x.^6 - 2*x.^4 - x.^3 - 1;
x = linspace(-1.5,1.5);
y = f(x);
%做出图像，找零点近似值
figure;
plot(x,y);
title('f = |ln(cos(x)).^2)| + x.^6 - 2*x.^4 - x.^3 - 1');
%牛顿法求解
display('使用牛顿法求解');
tic;
x1 = myNewtn(-1)
x2 = myNewtn (1.3)
toc;
%割线法求解
display('使用割线法求解');
tic;
x1 = mySecnt(-1.2,-0.8)
x2 = mySecnt (1,1.5)
toc;
function x = myNewtn (xo)
f = @(x) abs(log((cos(x)).^2)) + x.^6 - 2*x.^4 - x.^3 - 1;
```

```
fprime = @(x) 2*tan(x) + 6*x^5 - 8*x^3 - 3*x^2;
TOL = 1e-4;
x = xo - f(xo)/fprime(xo);
while abs(x - xo)> TOL
      xo = x;
      x = xo - f(xo)/fprime(xo);
end
end
function x2 = mySecnt (xo,x1)
f = @(x) abs(log((cos(x)).^2)) + x.^6 - 2*x.^4 - x.^3 - 1;
x2 = x1 - f(x1)*(x1 - xo)/(f(x1) - f(xo));
TOL = 1e-4;
while abs(x2 - x1)> TOL
      xo = x1;
      x1 = x2;x2 = x1 - f(x1)*(x1 - xo)/(f(x1) - f(xo));
end
end
```

计算结果为：

```
>> hw3_5
使用牛顿法求解

x1 =

   -0.9352


x2 =

    1.4203

时间已过 0.001308 秒。
```

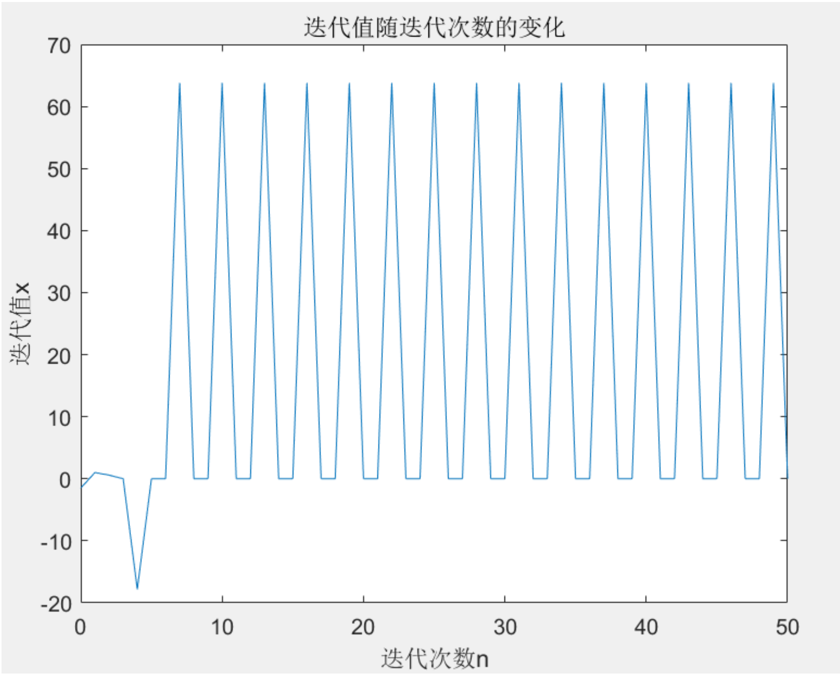
```
使用割线法求解

x1 =

   -0.9352


x2 =

    1.4203

时间已过 0.001575 秒。
```

（2）

1)使用牛顿法编程比较简单，因为迭代时只与前一个 x 有关，只涉及两个变量，而割线法迭代时与前两个变量有关，共涉及三个变量，编程更复杂一些。或：使用割线法不需要求导数，提升计算效率。

1)小问答案不唯一，言之成理即可。

2）使用 The Secant Method 时，不能取 $x_0=-1.5$, $x_1=1$。因为其迭代数次之后会如下图进入循环，无法收敛。



（3）fzero 是 Matlab 的内置函数，结合使用了二等分法、正割法和逆二次插值方法。它与两种方法的效率比较见上面的执行结果图。在本题的条件下，即容许绝对误差为 0.0001，fzero 的效率比 Newton Method 和 Secant Method 低一个数量级。

```
>> tic,fzero(f,-1),fzero(f,1.3),toc

ans =

   -0.9352


ans =

    1.4203

时间已过 0.019548 秒。
```

第六题

hw3_6_1.m

clc;clear;close all

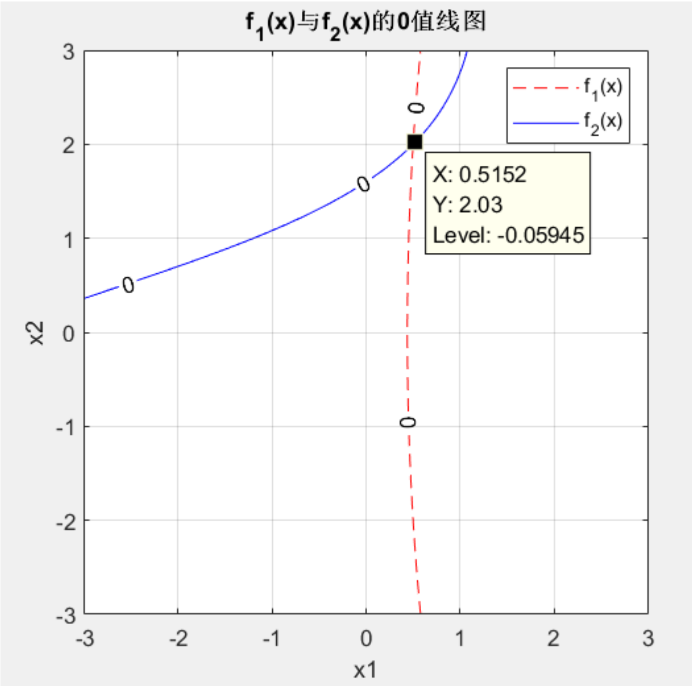
% 绘图确定解的范围

a = 4; b = 1/4;

x1 = linspace(-3,3,100);

x2 = linspace(-3,3,100);

```
[X1,X2] = meshgrid(x1,x2);
Z1 = a.*X1.^2-20.*X1+b.*X2.^2+8;
Z2 = 1/2.*X1.*X2.^2+2.*X1-5.*X2+8;
figure(1)
contour(X1,X2,Z1,[0 0],'ShowText','on','LineColor','r','LineStyle','--');
hold on
contour(X1,X2,Z2,[0 0],'ShowText','on','LineColor','b');
title('f_1(x)与 f_2(x)的 0 值线图');
xlabel('x1'),ylabel('x2');
legend({'f_1(x)' 'f_2(x)'})
axis equal,grid on
```



（2）选择迭代初值为 $x_1 = 0, x_2 = 2$

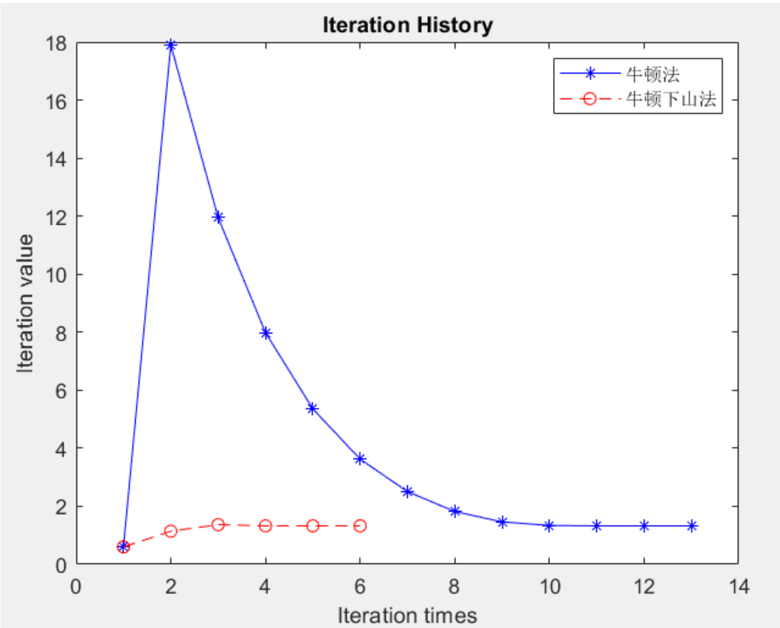
```
hw3_6_2.m
clear;
a=4;b=1/4;
f=@(x1,x2)[a*x1^2-20*x1+b*x2^2+8;
        0.5*x1*x2^2+2*x1-5*x2+8];
TOL=1e-5;x0=[0;2];
x=Mv_Newton(f,x0,TOL);
fprintf('Solution:\nx1=%f\nx2=%f\n',x);
```

```
Solution:
x1=0.500000
x2=2.000000
```

Mv_Newton.m

```
function x_solution=Mv_Newton(fun,x0,TOL)
f=fun;
x1=sym('x1'); x2=sym('x2');
y=f(x1,x2);
Jy=jacobian(y,[x1,x2]);
J=matlabFunction(Jy);

x01=x0(1);x02=x0(2);
x=[x01;x02]-J(x01,x02)\f(x01,x02);
while norm(x-[x01;x02])>TOL||norm(f(x(1),x(2)))>TOL
    x01=x(1);x02=x(2);
    x=[x01;x02]-J(x01,x02)\f(x01,x02);
end
x_solution=x;
end
```

第七题



牛顿法：x=1.324718

牛顿下山法:x=1.324718

（1）待改进之处：迭代值会出现大幅跳动，先远离真解之后再慢慢迭代回真解，计算效率较低。

（2）对比：牛顿下山法保证了迭代过程的单调性，避免了迭代值跳动，收敛更快。

hw3_7.m

```
clear,clc;
f=@(x)x.^3-x-1;
df=@(x)3*x.^2-1;
TOL=1e-5;
x(1)=0.6;
x(2)=x(1)-f(x(1))/df(x(1));
i=2;
while abs(x(i)-x(i-1))>TOL||abs(f(x(i)))>TOL
    x(i+1)=x(i)-f(x(i))/df(x(i));
    i=i+1;
end
plot(0:i-1,x, '-*b');
hold on;
xlabel 'Iteration times' ;ylabel 'Iteration value';
title 'Iteration History'
fprintf('牛顿法：x=%f\n',x(end));
%下山法
r(1)=0.6;
r(2)=r(1)-f(r(1))/df(r(1));
i=1;
while abs(r(i+1)-r(i))>TOL||abs(f(r(i+1)))>TOL
    lambda=1;
    while abs(f(r(i+1)))>abs(f(r(i)))
        lambda=lambda/2;
        r(i+1)=r(i)-lambda*f(r(i))/df(r(i));
    end
    i=i+1;
    r(i+1)=r(i)-f(r(i))/df(r(i));
end
fprintf('牛顿下山法:x=%f\n',r(end));
plot(0:i, '--or');
legend('牛顿法','牛顿下山法');
```

附加题

1.

f.m

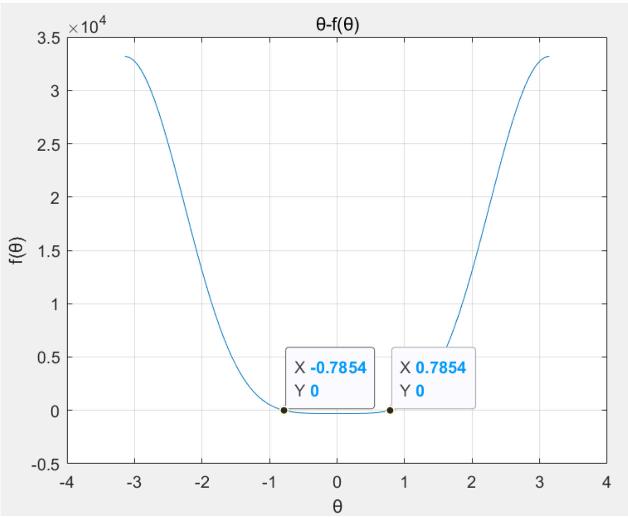
```
function out=f(theta)
L1=2;L2=sqrt(2);L3=sqrt(2);gamma=pi/2;
p1=sqrt(5);p2=sqrt(5);p3=sqrt(5);
x1=4;x2=0;y2=4;
A2=L3*cos(theta)-x1;
B2=L3*sin(theta);
A3=L2*cos(theta+gamma)-x2;
B3=L2*sin(theta+gamma)-y2;
N1=B3*(p2^2-p1^2-A2^2-B2^2)-B2*(p3^2-p1^2-A3^2-B3^2);
N2=-A3*(p2^2-p1^2-A2^2-B2^2)+A2*(p3^2-p1^2-A3^2-B3^2);
D=2*(A2*B3-B2*A3);
out=N1^2+N2^2-p1^2*D^2;
end
```

hw3_8_1.m

```
fprintf('test:\n');
fprintf('θ=-π/4， f(θ)=%f\n',f(-pi/4));
fprintf('θ= π/4， f(θ)=%f\n',f(pi/4));
```

```
test:
θ=-π/4, f(θ)=-0.000000
θ= π/4, f(θ)=-0.000000
```

2.



hw3_8_2.m

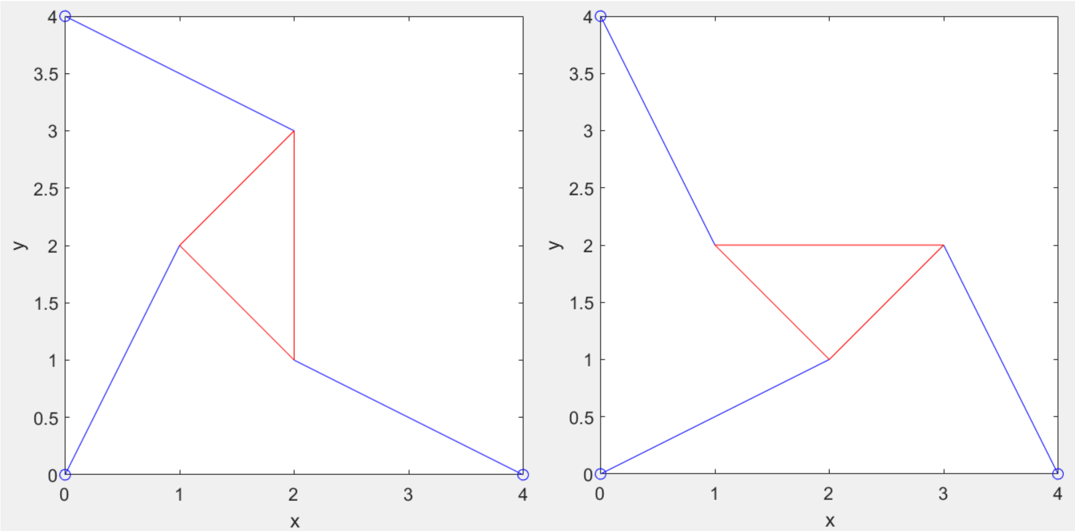
```
%题目里都是建议，那就不一定得用匿名函数和.
theta_list=linspace(-pi,pi);
y_list=zeros(1,length(theta_list));
for i=1:length(theta_list)
        y_list(i)=f(theta_list(i));
end
plot(theta_list,y_list);
```

```
xlabel 'θ';ylabel 'f(θ)';
grid on;hold on;
title 'θ-f(θ)'
plot([-pi/4,pi/4],[0,0],'*');
```

3.

hw3_8_3.m

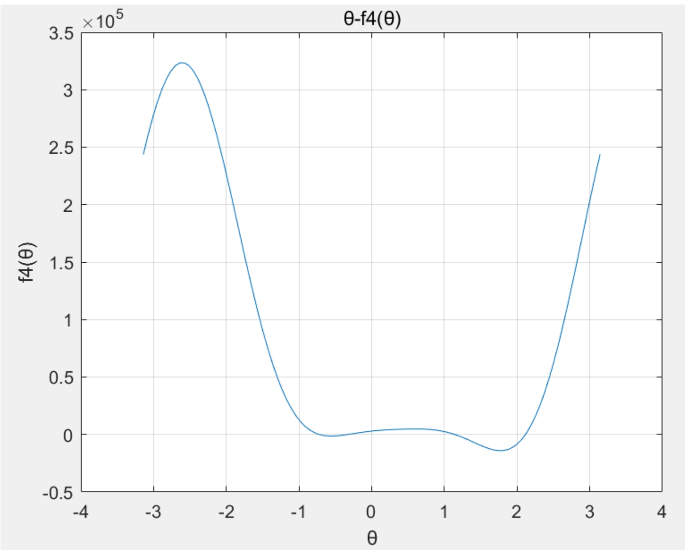
```
clear,clc;
L1=2;L2=sqrt(2);L3=sqrt(2);
gamma=pi/2;
p1=sqrt(5);p2=sqrt(5);p3=sqrt(5);
x1=4;x2=0;y2=4;
theta_list=[-pi/4,pi/4];
for i=1:2
    theta=theta_list(i);
    figure(i);
    A2=L3*cos(theta)-x1;
    B2=L3*sin(theta);
    A3=L2*cos(theta+gamma)-x2;
    B3=L2*sin(theta+gamma)-y2;
    N1=B3*(p2^2-p1^2-A2^2-B2^2)-B2*(p3^2-p1^2-A3^2-B3^2);
    N2=-A3*(p2^2-p1^2-A2^2-B2^2)+A2*(p3^2-p1^2-A3^2-B3^2);
    D=2*(A2*B3-B2*A3);
    u1=N1/D; v1=N2/D;
    u2=u1+L3*cos(theta);v2=v1+L3*sin(theta);
    u3=u1+L2*cos(theta+gamma);v3=v1+L2*sin(theta+gamma);
    plot([0,u1],[0,v1],'b',[x1,u2],[0,v2],'b',[x2,u3],[y2,v3],'b')
    hold on;axis equal;axis([0,4,0,4]);
    xlabel 'x';ylabel 'y';
    plot([u1 u2 u3 u1],[v1 v2 v3 v1],'r');
    plot([0 x1 x2],[0 0 y2],'bo');
end
```



4.

hw3_8_4_1.m

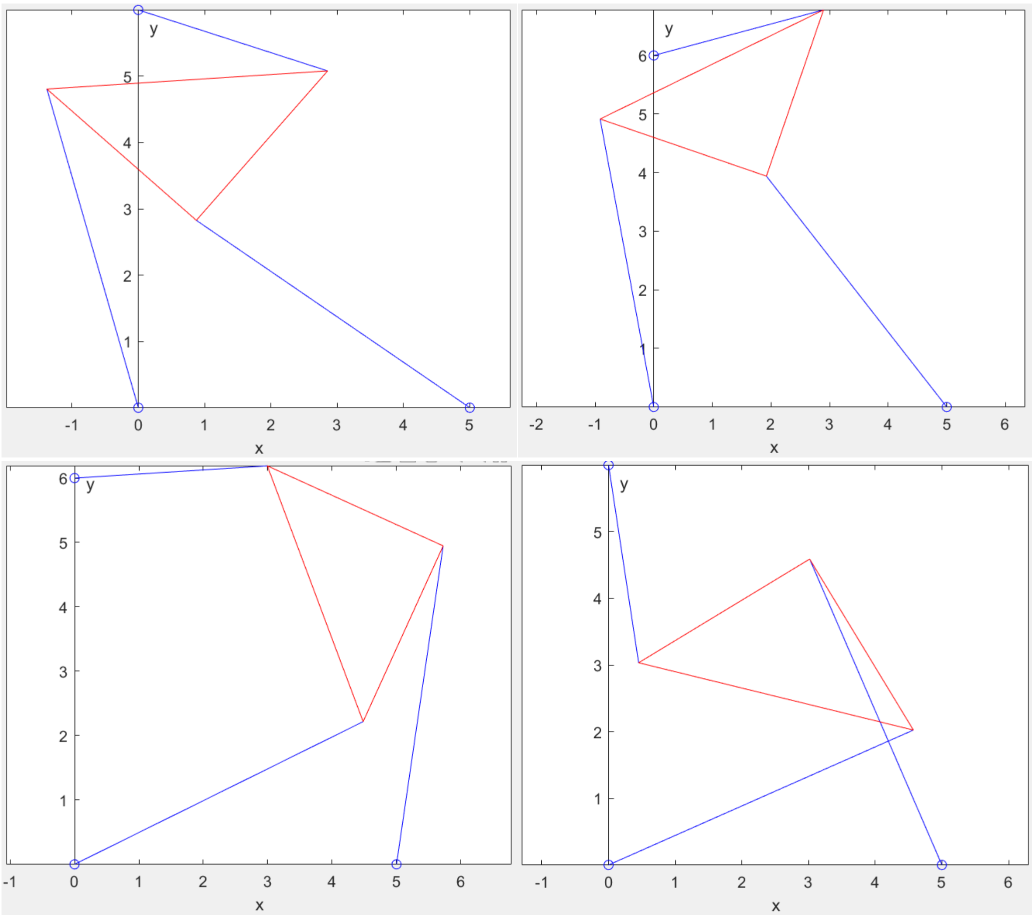
```
theta_list=linspace(-pi,pi);
y_list=zeros(1,length(theta_list));
for i=1:length(theta_list)
	y_list(i)=f4(theta_list(i));
end
plot(theta_list,y_list);
xlabel 'θ';ylabel 'f4(θ)';
grid on;hold on;
title 'θ-f4(θ)'
```



hw3_8_4_2.m

```
clear,clc;
f=@(theta)f4(theta);
init=[-1,0,1,2];
for i=1:4
    theta_list(i)=fzero(f,init(i));
    fprintf('solution%d;x=%f\n',i,theta_list(i));
end

L1=3;L2=3*sqrt(2);L3=3;gamma=pi/4;
p1=5;p2=5;p3=3;
x1=5;x2=0;y2=6;
for j=1:4
    figure(j);
    theta=theta_list(j);
    A2=L3*cos(theta)-x1;
    B2=L3*sin(theta);
    A3=L2*cos(theta+gamma)-x2;
    B3=L2*sin(theta+gamma)-y2;
    N1=B3*(p2^2-p1^2-A2^2-B2^2)-B2*(p3^2-p1^2-A3^2-B3^2);
    N2=-A3*(p2^2-p1^2-A2^2-B2^2)+A2*(p3^2-p1^2-A3^2-B3^2);
    D=2*(A2*B3-B2*A3);
    u1=N1/D; v1=N2/D;
    u2=u1+L3*cos(theta);v2=v1+L3*sin(theta);
    u3=u1+L2*cos(theta+gamma);v3=v1+L2*sin(theta+gamma);
    plot([0,u1],[0,v1],'b',[x1,u2],[0,v2],'b',[x2,u3],[y2,v3],'b')
    hold on;axis equal;%axis([0,4,0,4]);
    xlabel 'x';ylabel 'y';
    plot([u1 u2 u3 u1],[v1 v2 v3 v1],'r');
    plot([0 x1 x2],[0 0 y2],'bo');
    ax=gca;
    ax.YAxisLocation='origin';
end
```

```
solution1;x=-0.720849
solution2;x=-0.331005
solution3;x=1.143686
solution4;x=2.115909
```



y
x