探究$Stewart$平台运动学

林元莘 5201070910077

一、实验目的

1、通过二维版本的 Stewart 平台，掌握求解 Stewart 平台的前向运动学问题的方法，探究姿态数量和支杆长度的关系。

2、拓展探究三维、6 自由度的 Stewart 平台前向动力系统。

3、掌握 MATLAB 的基础功能和使用方法。

二、实验问题

考虑 Stewart 平台二维版本，该控制器由三个支杆控制的平面上的一个三角形平台构成，内部三角形表示平面 Stewart 平台，其对应的维数由三个长度L_1, L_2, L_3定义。令γ表示边L_1所对角度，平台位置由三个长度p_1, p_2, p_3控制，这对应三个支杆变化的长度。给定三个支杆长度，找到平台位置。推导相关方程并写出 MATLAB 程序，解决相关问题。并在最后，推导并确定一个方程，来表示三维、6 自由度的 Stewart 平台前向动力系统。写出 MATLAB 程序，并验证其用于求解前向动力系统。

三、建立数学模型

如图建立几何模型。

使用简单的三角方法包含下面三个方程：

$$\begin{aligned} p_1^2 &= x^2 + y^2 \\ p_2^2 &= (x + A_2)^2 + (y + B_2)^2 \\ p_3^2 &= (x + A_2)^2 + (y + B_3)^2 \end{aligned}$$

在方程中，

$$\begin{aligned} A_2 &= L_3 \cos\theta - x_1 \\ B_2 &= L_3 \sin\theta \\ A_3 &= L_2[\cos\theta\cos\gamma - \sin\theta\sin\gamma] - x_2 \\ B_3 &= L_2[\cos\theta\sin\gamma + \sin\theta\cos\gamma] - y_2 \end{aligned}$$

第一个方程代入后两个方程得

$$p_2^2 = p_1^2 + 2A_2x + 2B_2y + A_2^2 + B_2^2$$

$$p_3^2 = p_1^2 + 2A_3x + 2B_3y + A_3^2 + B_3^2$$

只要令$D = 2(A_2B_3 - A_3B_2) \neq 0$,x 和 y 可求解为:

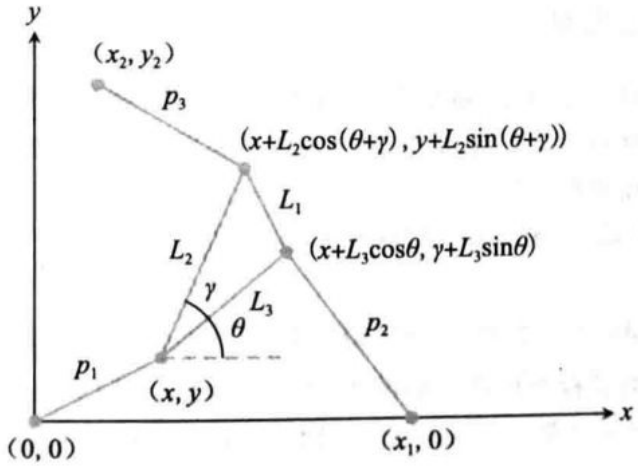
$$x = \frac{N_1}{D} = \frac{B_3(p_2^2 - p_1^2 - A_2^2 - B_2^2) - B_2(p_3^2 - p_1^2 - A_3^2 - B_3^2)}{2(A_2B_3 - B_2A_3)}$$

$$y=\frac{N_2}{D}=\frac{-A_3(p_2^2-p_1^2-A_2^2-B_2^2)+A_2(p_3^2-p_1^2-A_3^2-B_3^2)}{2(A_2B_3-B_2A_3)}$$

整理可得仅含一个未知数θ的方程

$$f=N_1^2+N_2^2-p_1^2D^2=0$$

通过$f(\theta)=0$解得θ，从而解得(x,y)，平台姿态可被确定。



四、程序实现与问题求解

1、写出$f(\theta)$的函数文件。并将参数设成

$$L_1=2, L_2=L_3=\sqrt{2}, \gamma=\frac{\pi}{2}, p_1=p_2=p_3=\sqrt{5}, (x_1,0)=(4,0), (x_2,y_2)=(0,4)$$

令$\theta=\frac{\pi}{4}$或$-\frac{\pi}{4}$，验证$f(\theta)=0$

a) 创建$f(\theta)$的函数文件

按照公式推导步骤可得

```
function out=f(theta)%为了求得 theta,构造 f(theta)=0 所需 f
format long;
L1=input('L1=');
L2=input('L2=');
L3=input('L3=');
g=input('gamma(rad)=') ;
p1=input('p1=');
p2=input('p2=');
p3=input('p3=');
x1=input('x1=');
x2=input('x2=');
y2=input('y2=');%输入需要的固定的参数
A2=L3.*cos(theta)-x1;
B2=L3.*sin(theta);
A3=L2.*(cos(theta).*cos(g)-sin(theta).*sin(g))-x2;
B3=L2.*(cos(theta).*sin(g)+sin(theta).*cos(g))-y2;
```

```
D=2.*(A2.*B3-B2.*A3);
M1=p2.^2-p1.^2-A2.^2-B2.^2;
M2=p3.^2-p1.^2-A3.^2-B3.^2;
N1=B3.*M1-B2.*M2;
N2=-A3.*M1+A2.*M2;%按照上述数学公式推导，计算所需关系式
out=N1.^2+N2.^2-p1.^2*D.^2;%输出函数
end
```

为了操作简单，将参数输入改为直接赋值

```
function out=f(theta)%为了求得 theta,构造 f(theta)=0 所需 f
format long;
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;%直接给参数赋值
A2=L3.*cos(theta)-x1;
B2=L3.*sin(theta);
A3=L2.*(cos(theta).*cos(gamma)-sin(theta).*sin(gamma))-x2;
B3=L2.*(cos(theta).*sin(gamma)+sin(theta).*cos(gamma))-y2;
D=2.*(A2.*B3-B2.*A3);
M1=p2.^2-p1.^2-A2.^2-B2.^2;
M2=p3.^2-p1.^2-A3.^2-B3.^2;
N1=B3.*M1-B2.*M2;
N2=-A3.*M1+A2.*M2;%按照上述数学公式推导，计算所需关系式
out=N1.^2+N2.^2-p1.^2*D.^2;%输出函数
end
```

b)代值验证

```
f(pi/4)

ans =

    -4.547473508864641e-13

f(-pi/4)

ans =

    -4.547473508864641e-13
```

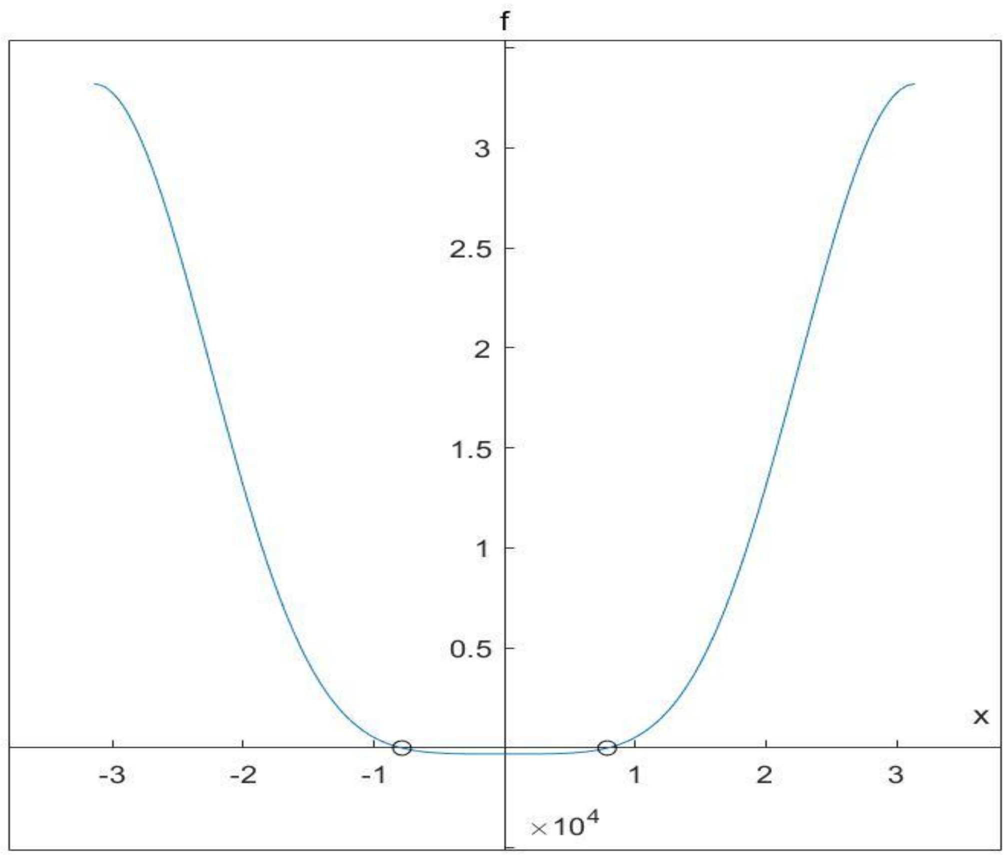
由于两个函数值均为 10^(-13)的数量级，$f(-\frac{\pi}{2}) = f(\frac{\pi}{2}) = 0$成立。

2、画出 $f(\theta)$ 在 $[-\pi, \pi]$ 上的图像并检查其在 $\pm\frac{\pi}{4}$ 处有根

沿用 1 中函数

```
ezplot(@f,[-pi,pi]);hold on%画出[-pi,pi]区间内的 f 图像
plot([pi/4 -pi/4],[0 0],'ko');%描出点(-pi/4,0),(pi/4,0)
ax=gca;
ax.XAxisLocation = 'origin';
ax.YAxisLocation = 'origin';%设置过原点的坐标轴
axis([-inf inf -inf inf]);%坐标轴范围随函数值域而调整
```

导出图像如下，黑色小圆处为点(-pi/4,0),(pi/4,0)：



由图可知，f 在$\pm\frac{\pi}{4}$处有根

或通过图像求解$\frac{\pi}{4}$附近以及$-\frac{\pi}{4}$附近该函数的根，探究和$\pm\frac{\pi}{4}$的误差。

```
x01=fzero(@f,pi/4)
x02=fzero(@f,-pi/4)
Ef1=abs(x01-pi/4)%前向误差
Ef2=abs(x02+pi/4)
Eb1=abs(f(x01))%后向误差
Eb2=abs(f(x02))

Ef1 =

     3.330669073875470e-16
```

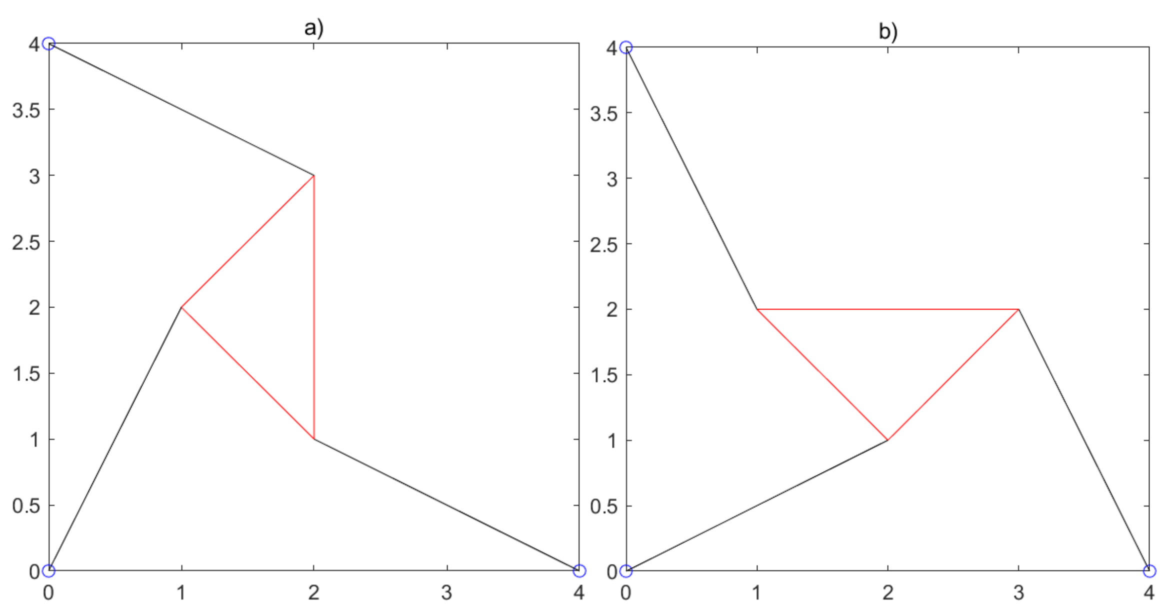
```
Ef2 =

     3.330669073875470e-16

Eb1 =

     2.273736754432321e-13

Eb2 =

     2.273736754432321e-13
```

口差为 10^(-13)的数量级，则 f 在$\pm\frac{\pi}{4}$处有根

3、生成 1 的情况下，Stewart 平台的两种姿态图

```
figure;%创建图窗窗口
subplot(1,2,1);%在第一个分块创建坐标区
plot([1 2 2 1],[2 3 1 2],'r');hold on%画三角形
plot([0 4 0],[0 0 4],'bo');hold on%描出锚点
plot([0 1 0],[0 2 0],'k');hold on
plot([4 2 4],[0 1 0],'k');hold on
plot([0 2 0],[4 3 4],'k')%画出支杆
title('a)');%给图像标号
subplot(1,2,2);%在第二个分块创建坐标区
plot([2 1 3 2],[1 2 2 1],'r');hold on%画三角形
plot([0 4 0],[0 0 4],'bo');hold on%描出锚点
plot([0 2 0],[0 1 0],'k');hold on
plot([4 3 4],[0 2 0],'k');hold on
plot([0 1 0],[4 2 4],'k')%画出支杆
title('b)');%给图像标号
```

导出图像为



4、将参数设成

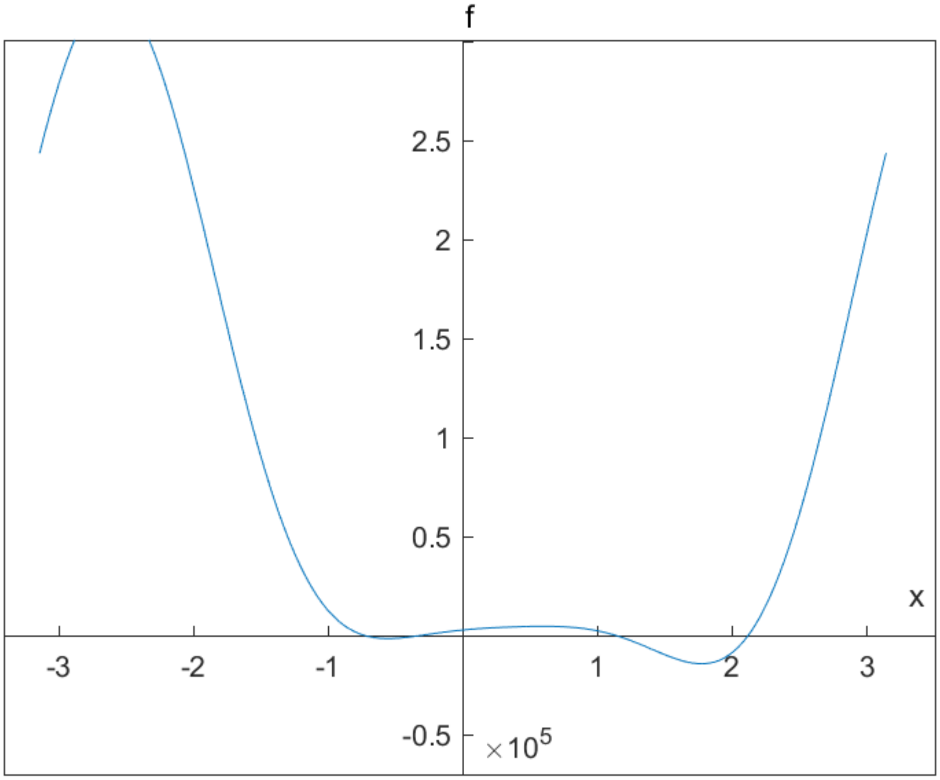
$$L_2 = 3\sqrt{2}, L_1 = L_3 = 3, \gamma = \frac{\pi}{4}, p_1 = p_2 = 5, p_3 = 3, (x_1, 0) = (5, 0), (x_2, y_2) = (0, 6)$$

画出 $f(\theta)$，利用方程求解技术找出 4 个位置，并画出这些位置，通过验证p_1, p_2, p_3检查结果

a)更新函数并画出函数图像

```
ezplot(@f,[-pi,pi]);hold on%画出函数图像
ax=gca;
ax.XAxisLocation = 'origin';
ax.YAxisLocation = 'origin';%画出原点处坐标轴
axis([-inf inf -inf inf]);%坐标轴范围随函数值域调整

function out=f(theta)%为了求得 theta,构造 f(theta)=0 所需 f
format long;
L1=3;
L2=3*sqrt(2);
L3=3;
gamma=pi/4;
p1=5;
p2=5;
p3=3;
x1=5;
x2=0;
y2=6;%更改所需参数
A2=L3.*cos(theta)-x1;
B2=L3.*sin(theta);
A3=L2.*(cos(theta).*cos(gamma)-sin(theta).*sin(gamma))-x2;
B3=L2.*(cos(theta).*sin(gamma)+sin(theta).*cos(gamma))-y2;
D=2.*(A2.*B3-B2.*A3);
M1=p2.^2-p1.^2-A2.^2-B2.^2;
M2=p3.^2-p1.^2-A3.^2-B3.^2;
N1=B3.*M1-B2.*M2;
N2=-A3.*M1+A2.*M2;%按照上述数学公式推导，计算所需关系式
out=N1.^2+N2.^2-p1.^2*D.^2;%输出函数
end
```

导出图像，可观察得 4 个解



b)求解 4 个位置时的θ

```
t1=fzero(@f,2)

t1 =

   2.115909014086458

t2=fzero(@f,1)

t2 =

   1.143685517821374

t3=fzero(@f,0)

t3 =

  -0.331005184283869

t4=fzero(@f,-1)%根据函数图像，求解出四个位置根的数值解

t4 =

  -0.720849204460390
```

c)计算画图相关点坐标

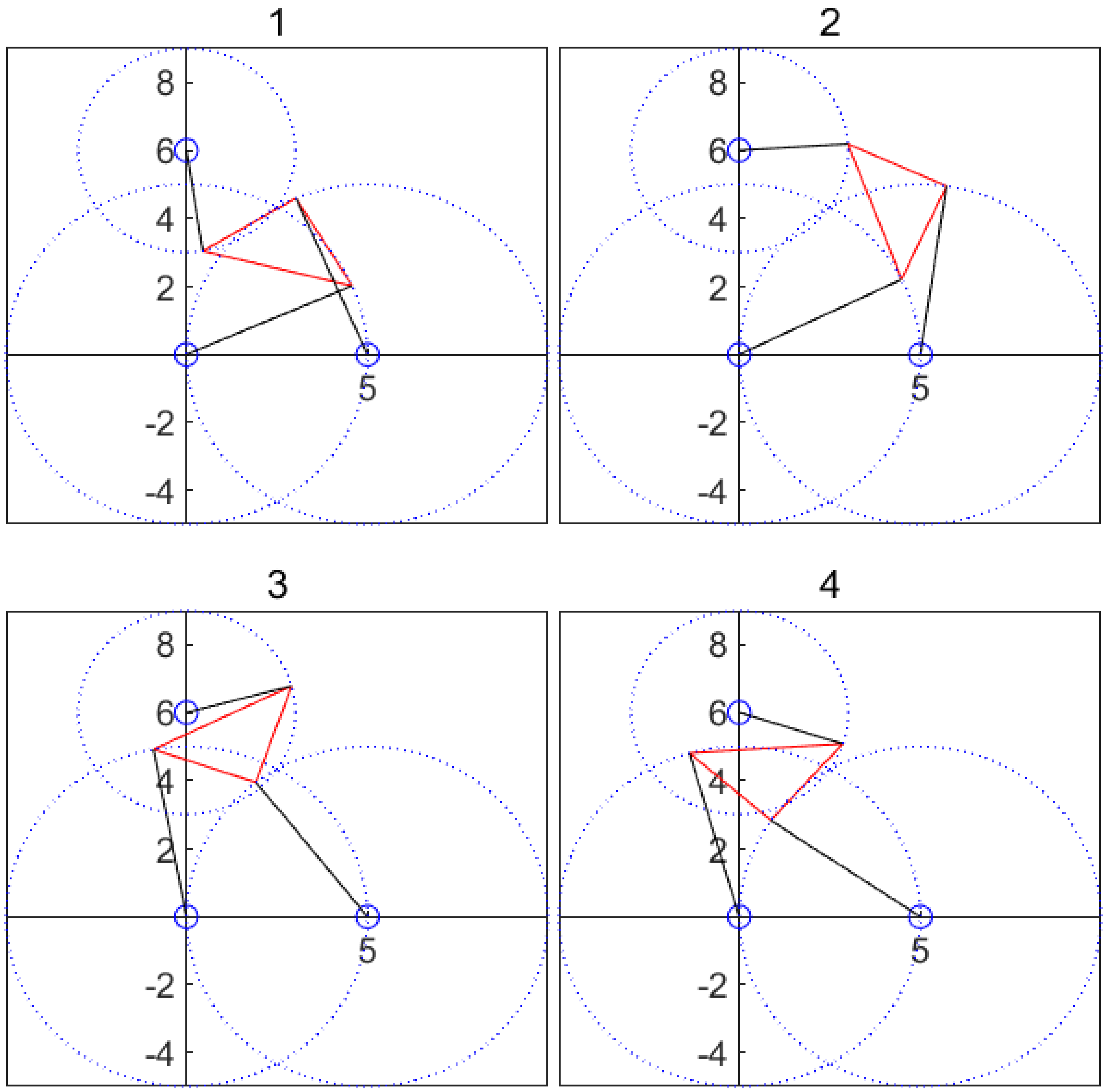
```
theta=[t1,t2,t3,t4];%解向量
L1=3;
L2=3*sqrt(2);
L3=3;
```

```
gamma=pi/4;
p1=5;
p2=5;
p3=3;
x1=5;
x2=0;
y2=6;%参数
A2=L3.*cos(theta)-x1;
B2=L3.*sin(theta);
A3=L2.*(cos(theta).*cos(gamma)-sin(theta).*sin(gamma))-x2;
B3=L2.*(cos(theta).*sin(gamma)+sin(theta).*cos(gamma))-y2;
D=2.*(A2.*B3-B2.*A3);
M1=p2.^2-p1.^2-A2.^2-B2.^2;
M2=p3.^2-p1.^2-A3.^2-B3.^2;
N1=B3.*M1-B2.*M2;
N2=-A3.*M1+A2.*M2;%计算相关式子
u1=N1./D;
v1=N2./D;%计算第一个点横纵坐标
u2=u1+L3.*cos(theta);
v2=v1+L3.*sin(theta);%计算第二个点横纵坐标
u3=L2.*cos(theta+gamma)+u1;
v3=L2.*sin(theta+gamma)+v1;%计算第三个点横纵坐标
```

d)画出 4 个位置姿态，并验证p_1, p_2, p_3是图中支杆长度

```
t=linspace(0,2*pi);%为画圆设置参数
figure%创建图窗窗口
%画出 4 种位置姿态图
for i=1:4
    subplot(2,2,i);%在第 i 个窗口
    plot([u1(i) u2(i) u3(i) u1(i)],[v1(i) v2(i) v3(i) v1(i)],'r');hold on%画出三角形平台
    plot([0 x1 x2],[0 0 y2],'bo');hold on%描出锚点
    plot([0 u1(i) 0],[0 v1(i) 0],'k');hold on
    plot([x1 u2(i) x1],[0 v2(i) 0],'k');hold on
    plot([x2 u3(i) x2],[y2 v3(i) y2],'k');hold on%画出支杆
    plot((p1.*cos(t)),(p1*sin(t)),'b:');hold on
    plot((p2.*cos(t)+x1),(p2*sin(t)),'b:');hold on
    plot((p3.*cos(t)+x2),(p3*sin(t)+y2),'b:')%画出关节运动圆，以检查结果
    ax=gca;
    ax.XAxisLocation = 'origin';
    ax.YAxisLocation = 'origin';
    axis([-inf inf -inf inf]);%画出原点处坐标轴
    title(i)%名称
end
```

导出姿态图如下




```
Ep1=max(abs([(norm([u1(1),v1(1)])-p1),(norm([u2(1)-x1,v2(1)])-p2),(norm([u3(1)-x2,v3(1)-y2])-p3)]))
Ep2=max(abs([(norm([u1(2),v1(2)])-p1),(norm([u2(2)-x1,v2(2)])-p2),(norm([u3(2)-x2,v3(2)-y2])-p3)]))
Ep3=max(abs([(norm([u1(3),v1(3)])-p1),(norm([u2(3)-x1,v2(3)])-p2),(norm([u3(3)-x2,v3(3)-y2])-p3)]))
Ep4=max(abs([(norm([u1(4),v1(4)])-p1),(norm([u2(4)-x1,v2(4)])-p2),(norm([u3(4)-x2,v3(4)-y2])-p3)]))
%关于支杆长度的前向误差
Ep1 =

     8.881784197001252e-16
Ep2 =

     8.881784197001252e-16
Ep3 =

     0
Ep4 =

     4.440892098500626e-16
```

验证了图中支杆长度为实际支杆长度

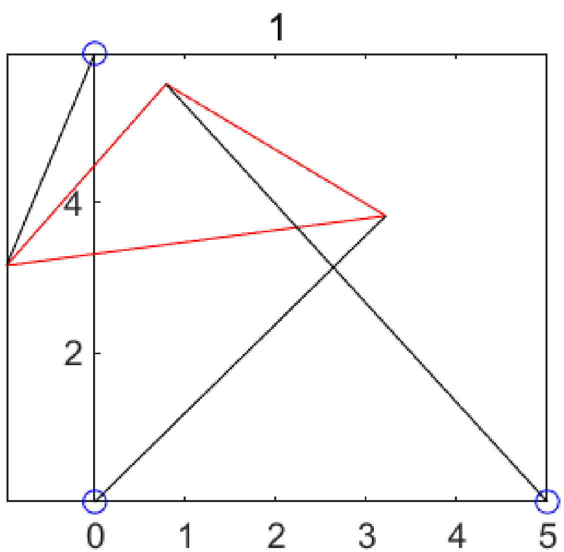
5、$p_2 = 7$其他条件不变重新求解问题

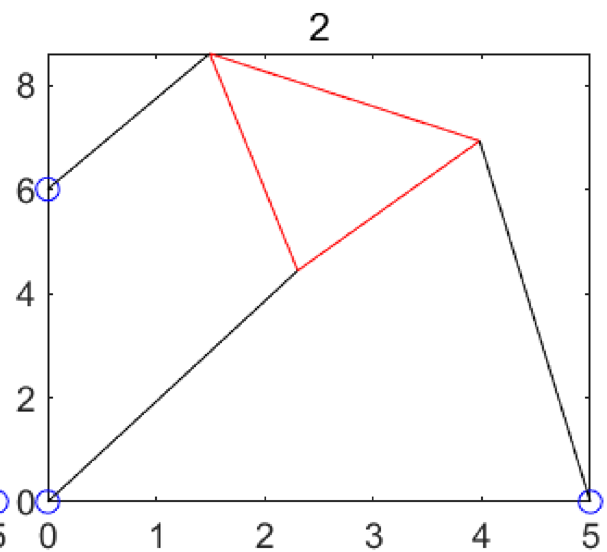
与 4 中代码同理

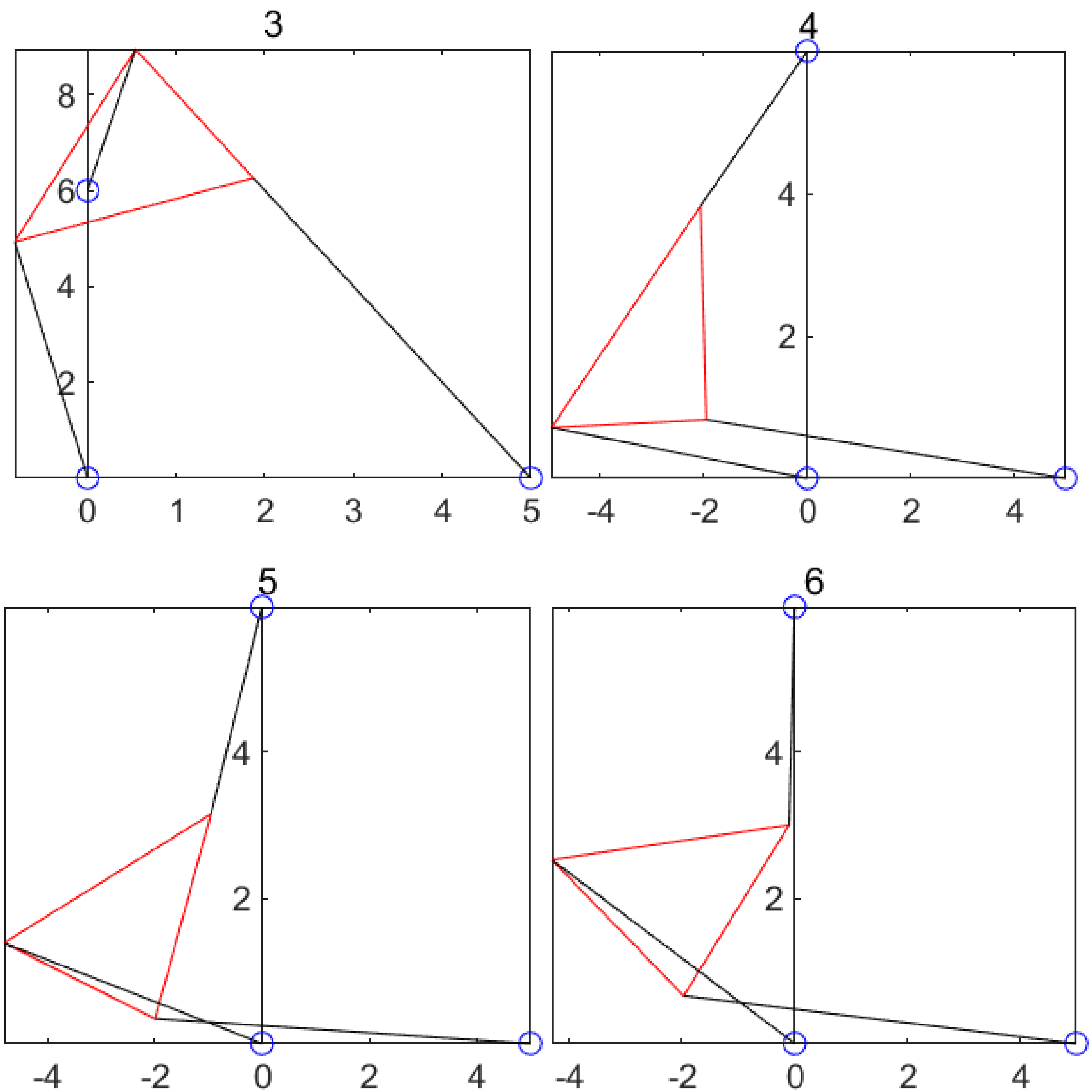
```
ezplot(@f,[-pi,pi])
ax=gca;
ax.XAxisLocation = 'origin';
ax.YAxisLocation = 'origin';
axis([-inf inf -inf inf]);
L1=3;
L2=3*sqrt(2);
L3=3;
gamma=pi/4;
p1=5;
p2=7;
p3=3;
x1=5;
x2=0;
y2=6;
t1=fzero(@f,2)
t2=fzero(@f,1)
t3=fzero(@f,0.5)
t4=fzero(@f,0)
t5=fzero(@f,-0.3)
t6=fzero(@f,-1)
theta=[t1,t2,t3,t4,t5,t6];
A2=L3.*cos(theta)-x1;
B2=L3.*sin(theta);
A3=L2.*(cos(theta).*cos(gamma)-sin(theta).*sin(gamma))-x2;
B3=L2.*(cos(theta).*sin(gamma)+sin(theta).*cos(gamma))-y2;
D=2.*(A2.*B3-B2.*A3);
M1=p2.^2-p1.^2-A2.^2-B2.^2;
M2=p3.^2-p1.^2-A3.^2-B3.^2;
N1=B3.*M1-B2.*M2;
N2=-A3.*M1+A2.*M2;
u1=N1./D;
v1=N2./D;
u2=u1+L3.*cos(theta);
v2=v1+L3.*sin(theta);
u3=L2.*cos(theta+gamma)+u1;
v3=L2.*sin(theta+gamma)+v1;
figure
for i=1:6
    subplot(3,2,i);
    plot([u1(i) u2(i) u3(i) u1(i)],[v1(i) v2(i) v3(i) v1(i)],'r');hold on
    plot([0 x1 x2],[0 0 y2],'bo');hold on
    plot([0 u1(i) 0],[0 v1(i) 0],'k');hold on
    plot([x1 u2(i) x1],[0 v2(i) 0],'k');hold on
    plot([x2 u3(i) x2],[y2 v3(i) y2],'k')
    ax=gca;
    ax.XAxisLocation = 'origin';
    ax.YAxisLocation = 'origin';
    axis([-inf inf -inf inf]);
    title(i)
end

function out=f(theta)
```

```
format long;
L1=3;
L2=3*sqrt(2);
L3=3;
gamma=pi/4;
p1=5;
p2=7;
p3=3;
x1=5;
x2=0;
y2=6;
A2=L3.*cos(theta)-x1;
B2=L3.*sin(theta);
A3=L2.*(cos(theta).*cos(gamma)-sin(theta).*sin(gamma))-x2;
B3=L2.*(cos(theta).*sin(gamma)+sin(theta).*cos(gamma))-y2;
D=2.*(A2.*B3-B2.*A3);
M1=p2.^2-p1.^2-A2.^2-B2.^2;
M2=p3.^2-p1.^2-A3.^2-B3.^2;
N1=B3.*M1-B2.*M2;
N2=-A3.*M1+A2.*M2;
out=N1.^2+N2.^2-p1.^2*D.^2;
end
```

导出 6 个位置姿态的图片







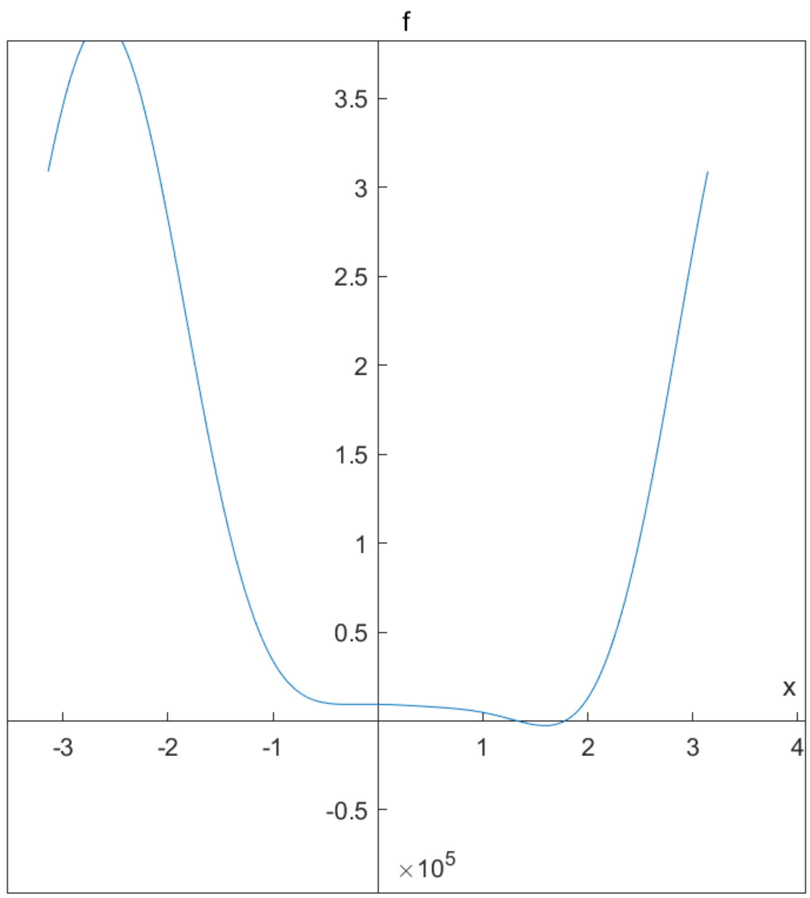
6、找出p_2,其他参数相同，使得其中只有两个姿态

取$p_2=4$

画出对应函数图像

```
ezplot(@f,[-pi,pi])
ax=gca;
ax.XAxisLocation = 'origin';
ax.YAxisLocation = 'origin';
axis([-inf inf -inf inf]);

function out=f(theta)
format long;
L1=3;
L2=3*sqrt(2);
L3=3;
gamma=pi/4;
p1=5;
```

```
p2=4;
p3=3;
x1=5;
x2=0;
y2=6;
A2=L3.*cos(theta)-x1;
B2=L3.*sin(theta);
A3=L2.*(cos(theta).*cos(gamma)-sin(theta).*sin(gamma))-x2;
B3=L2.*(cos(theta).*sin(gamma)+sin(theta).*cos(gamma))-y2;
D=2.*(A2.*B3-B2.*A3);
M1=p2.^2-p1.^2-A2.^2-B2.^2;
M2=p3.^2-p1.^2-A3.^2-B3.^2;
N1=B3.*M1-B2.*M2;
N2=-A3.*M1+A2.*M2;
out=N1.^2+N2.^2-p1.^2*D.^2;
end
```



7、计算p_2区间，使得其中分别有 0、2、4、6 个姿态。

a)创建相关函数

求解θ的函数f

```
function out=f(theta,p2)%将 p2 也设为变量，以便对 p2 进行更换
format long;
L1=3;
```

```
L2=3*sqrt(2);
L3=3;
gamma=pi/4;
p1=5;
p3=3;
x1=5;
x2=0;
y2=6;
A2=L3.*cos(theta)-x1;
B2=L3.*sin(theta);
A3=L2.*(cos(theta).*cos(gamma)-sin(theta).*sin(gamma))-x2;
B3=L2.*(cos(theta).*sin(gamma)+sin(theta).*cos(gamma))-y2;
D=2.*(A2.*B3-B2.*A3);
M1=p2.^2-p1.^2-A2.^2-B2.^2;
M2=p3.^2-p1.^2-A3.^2-B3.^2;
N1=B3.*M1-B2.*M2;
N2=-A3.*M1+A2.*M2;
out=N1.^2+N2.^2-p1.^2*D.^2;
end
```

求 f 零点个数的函数 numofroot

```
function m=numofroot(f,a,b,tol)%计算连续函数在某区间内零点的数量
format long;
x=a:tol:b;%对区间进行细分成点, 间距由 tol 控制
m=0;%初始化零点个数
%由于函数连续, 对于本题的特殊情况, 不存在固定存在的非变号零点, 我们仅寻找变号零点即可
for i=1:(length(x)-1)
    p=f(x(i));
    q=f(x(i+1));
    if p>0&&q<0%若相邻两个分点函数值异号, 则说明变号零点存在
        m=m+1;
    elseif p<0&&q>0
        m=m+1;
    elseif p==0%若分点处函数值为 0, 则计入零点
            m=m+1;
    end
end
if f(b)==0%最后检查未在循环内的边缘点
    m=m+1;
end
end
```

b)粗略估计各区间边界

由于求解零点数量所需计算量较大, 无法对 p2 进行细分后一个个代入计算。此时我们采取粗糙估计后二分法求边界, 以争取减小计算量。

```
format long
%由前面的题目以及几何模型可以估计[3,10]遍历所有可能姿态
p2=3:0.5:10;%0.5 为间隔粗糙取点, 大概得出各姿态粗糙区间
j=0;
n(1)=0;
%遍历取点, 得出粗糙区间
for i=2:length(p2)
    g=@(x)f(x,p2(i));
    n(i)=numofroot(g,-pi,pi,10^(-4));%计算零点数量
    if n(i-1)~=n(i)%若相邻两个 p_2 分点处函数零点个数不一致, 则记录
```

```
            j=j+1;
            p2_0(1:2,2*j-1)=[p2(i-1);n(i-1)];
            p2_0(1:2,2*j)=[p2(i);n(i)];
        end
    end
    p2_0%检查所得区间边界点（第一行）以及对应函数零点数（第二行）
```

```
p2_0 = 2×12

3.500000000000000   4.000000000000000   4.500000000000000   5.000000000000000
6.500000000000000   7.000000000000000   7.000000000000000   7.500000000000000
7.500000000000000   8.000000000000000   9.000000000000000   9.500000000000000

0   2.000000000000000   2.000000000000000   4.000000000000000   4.000000000000000
6.000000000000000   6.000000000000000   4.000000000000000   4.000000000000000
2.000000000000000   2.000000000000000                   0
```

由程序以及函数连续性可知，$[0,3.5]\bigcup[9.5,+\infty)$有 0 个姿态，$[4,4.5]\bigcup[8,9]$有 2 个姿态，$[5,6.5]\bigcup\{7.5\}$有 2 个姿态，7 处有 2 个姿态。

c)二分法较准确估计各区间边界

```
    %利用函数的连续性
    %细分利用到二分法，对于每次得到的相邻区间边界进行二分法划分，以缩小相邻区间边界距离，得到较准确区间
    for i=1:j
        dp2(i)=p2_0(1,2*i)-p2_0(1,2*i-1);
    end%初始化各相邻区间边界距离
    while max(dp2)>=5*10^(-3)%各相邻区间边界距离精确到两位小数
        for i=1:j
            pp2=(p2_0(1,2*i)+p2_0(1,2*i-1))/2;%二分法
            g=@(x)f(x,pp2);
            m=numofroot(g,-pi,pi,10^(-4));
            if m==p2_0(2,2*i-1)%若中点所算得零点个数与左侧相等，则左侧相应区间的右边界拓宽至该点
                p2_0(1:2,2*i-1)=[pp2;m];
            else%反之，则右侧相应区间的左边界拓宽至该点
                p2_0(1:2,2*i)=[pp2;m];
            end
            dp2(i)=p2_0(1,2*i)-p2_0(1,2*i-1);%计算各相邻区间边界距离
        end
    end
    p2_0%输出所得区间边界点（第一行）以及对应函数零点数（第二行）
```

```
p2_0 = 2×12

3.707031250000000   3.710937500000000   4.863281250000000   4.867187500000000
6.964843750000000   6.968750000000000   7.019531250000000   7.023437500000000
7.847656250000000   7.851562500000000   9.261718750000000   9.265625000000000

0   2.000000000000000   2.000000000000000   4.000000000000000   4.000000000000000
6.000000000000000   6.000000000000000   4.000000000000000   4.000000000000000
2.000000000000000   2.000000000000000                   0
```

由于设置程序精确至两位小数，再取中点作为$[0,3.71]\bigcup[9.27,+\infty)$有 0 个姿态，$[3.71,4.86]\bigcup[7.85,9.26]$有 2 个姿态，$[4.87,6.96]\bigcup[7.02,7.85]$有 4 个姿态，$[6.97,7.02]$有 6 个姿态。

五、问题拓展：三维、6 自由度 Stewart 平台前向动力系统

将平台底座置于 xOy 平面上，并选取一点置于原点处，每个锚点可被如此表示$b_i = \begin{bmatrix} \mathrm{xx}_i \\ \mathrm{yy}_i \\ 0 \end{bmatrix}$。将平台也置于 xOy 平面时，选取原点处支杆另一端点也记为原点，每个平台定点可被如此表示$q_i = \begin{bmatrix} x_i \\ y_i \\ 0 \end{bmatrix}$，对应支杆长度为$p_i$

平台运动至相应位置时，进行了 6 个自由度变化，记其平移向量为$T = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$，

相应的转动矩阵$R_Z = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$，$R_Y = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$，$R_X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \sin\varphi & \cos\varphi \end{bmatrix}$

整个平台的转动矩阵即为

$$\mathrm{R} = \mathrm{R}_Z \mathrm{R}_Y \mathrm{R}_X$$

则平台各顶点相应位置坐标为

$$T + Rq_i$$

对应支杆向量为

$$l_i = T + Rq_i - b_i$$

两边取模必然有

$$p_i^2 = (T + Rq_i - b_i)^T (T + Rq_i - b_i)$$

构造目标函数

$$F(X, Y, Z, \psi, \theta, \varphi) = \sum_{i=1}^{6} [pi^2 - (T + Rq_i - b_i)^T (T + Rq_i - b_i)]^2$$

问题转为求该函数的极小值，即最小二乘问题。

记$v = \begin{bmatrix} X \\ Y \\ Z \\ \psi \\ \theta \\ \varphi \end{bmatrix}$，$g_i(v) = pi^2 - (T + Rq_i - b_i)^T (T + Rq_i - b_i)$，$g(v) = \begin{bmatrix} g_1(v) \\ g_2(v) \\ g_3(v) \\ g_4(v) \\ g_5(v) \\ g_6(v) \end{bmatrix}$，

$$F(v) = g(v)^T g(v)$$

利用高斯-牛顿迭代法，$J_k = \nabla g(u_k)$，迭代方程为：

$$v_{k+1} = v_k - J_k^{-1} g(v_k)$$

从而迭代逼近我们所需的解。

程序实现如下

```
format long;
syms X Y Z psi theta phi;
v=[X;Y;Z;psi;theta;phi];
Jg=jacobian(g(v),v);%计算雅克比矩阵
v0=[0;0;20;0;0;0];%设置初始值
J=vpa(subs(Jg,v,v0));%用 v0 赋值
v1=v0-inv(J)*g(v0);
while norm(v1-v0)>=0.5*10^(-4)%高斯-牛顿迭代法
    v0=v1;
    J=vpa(subs(Jg,v,v0));
    v1=v0-inv(J)*g(v0);
end
v1%输出结果

function out=g(v)%创建目标函数
format long;
p1=input('p1='); p2=input('p2='); p3=input('p3='); p4=input('p4='); p5=input('p5='); p6=input('p6=');%输入支杆
x1=input('x1='); x2=input('x2='); x3=input('x3='); x4=input('x4='); x5=input('x5=');%输入平台顶点横坐标
y1=input('y1='); y2=input('y2='); y3=input('y3='); y4=input('y4='); y5=input('y5=');%输入平台顶点纵坐标
xx1=input('xx1=');xx2=input('xx2='); xx3=input('xx3='); xx4=input('xx4='); xx5=input('xx5=');%输入平台底座定点横
yy1=input('yy1='); yy2=input('yy2='); yy3=input('yy3='); yy4=input('yy4'); yy5=input('yy5=');%输入平台底座定点纵
X=v(1);
Y=v(2);
Z=v(3);
psi=v(4);
theta=v(5);
phi=v(6);
p=[p1;p2;p3;p4;p5;p6];
T=[X;Y;Z];%平移矩阵
R=[cos(psi) -sin(psi) 0;sin(psi) cos(psi) 0;0 0 1]*[cos(theta) 0 sin(theta);0 1 0;-sin(theta) 0 cos(theta)]*[1
%旋转矩阵
xp=[0,x1,x2,x3,x4,x5];
yp=[0,y1,y2,y3,y4,y5];
z=zeros(1,6);
A=[xp;yp;z];
xb=[0,xx1,xx2,xx3,xx4,xx5];
yb=[0,yy1,yy2,yy3,yy4,yy5];
B=[xb;yb;z];
for i=1:6
    l(:,i)=T+R*A(:,i)-B(:,i);%计算支杆向量
end
for i=1:6
    out(i,1)=(p(i,1))^2-(l(:,i))'*l(:,i);%导出 gi,合并为 g
end
end
```

代入相应数值，计算并画图验证

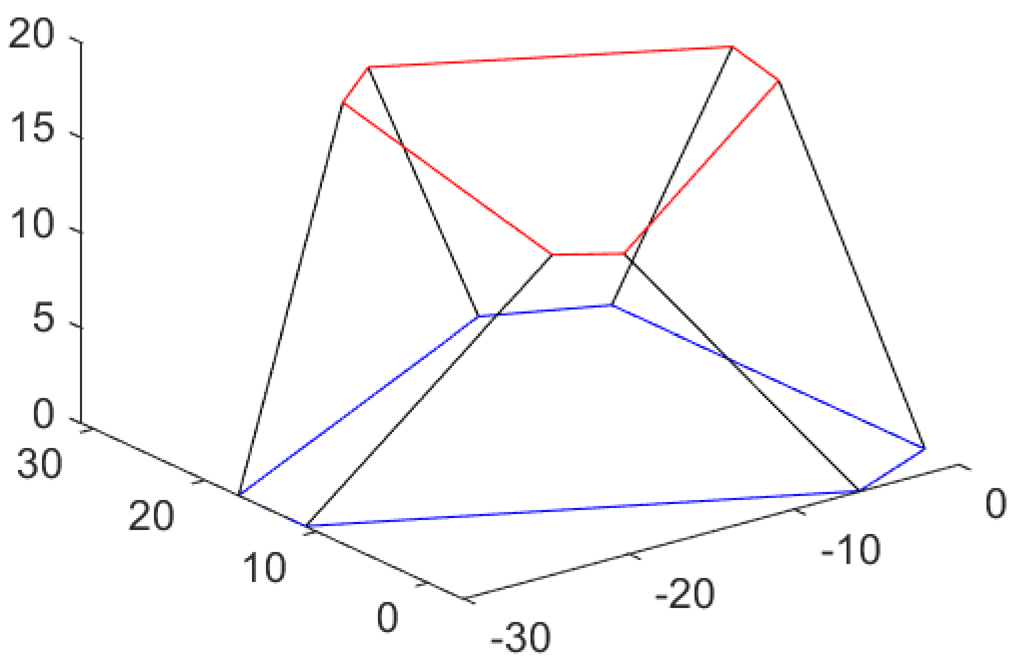
```
format long
```

```
syms X Y Z psi theta phi;%设置函数自变量
v=[X;Y;Z;psi;theta;phi];
Jg=jacobian(g(v),v);%计算雅克比矩阵
v0=[0;0;20;0;0;0];%设置初始值
J=vpa(subs(Jg,v,v0));%用 v0 赋值
v1=v0-inv(J)*g(v0);
while norm(v1-v0)>=0.5*10^(-4)%高斯-牛顿迭代法
    v0=v1;
    J=vpa(subs(Jg,v,v0));
    v1=v0-inv(J)*g(v0);
end
v1%输出结果
p1=20; p2=25; p3=26; p4=20; p5=20; p6=20;%输入支杆长度
x1=0; x2=-16; x3=-19; x4=-19; x5=-16;%输入平台顶点横坐标
y1=4; y2=13; y3=11; y4=-7; y5=-9;%输入平台顶点纵坐标
xx1=0;xx2=-6; xx3=-30; xx4=-30; xx5=-6;%输入平台底座定点横坐标
yy1=28; yy2=31; yy3=17; yy4=11; yy5=-3;%输入平台底座定点纵坐标
X1=v1(1);
Y1=v1(2);
Z1=v1(3);
psi1=v1(4);
theta1=v1(5);
phi1=v1(6);
p=[p1;p2;p3;p4;p5;p6];
T=[X1;Y1;Z1];%平移矩阵
R=[cos(psi1) -sin(psi1) 0;sin(psi1) cos(psi1) 0;0 0 1]*[cos(theta1) 0 sin(theta1);0 1 0;-sin(theta1) 0 cos(the
%旋转矩阵
xp=[0,x1,x2,x3,x4,x5];
yp=[0,y1,y2,y3,y4,y5];
z=zeros(1,6);
A=[xp;yp;z];
xb=[0,xx1,xx2,xx3,xx4,xx5];
yb=[0,yy1,yy2,yy3,yy4,yy5];
B=[xb;yb;z];
for i=1:6
    A(:,i)=T+R*A(:,i);%计算平台各顶点新的坐标
end
plot3([A(1,:),A(1,1)],[A(2,:),A(2,1)],[A(3,:),A(3,1)],'r');hold on
plot3([B(1,:),B(1,1)],[B(2,:),B(2,1)],[B(3,:),B(3,1)],'b');hold on
for i=1:6
    plot3([A(1,i),B(1,i),A(1,i)],[A(2,i),B(2,i),A(2,i)],[A(3,i),B(3,i),A(3,i)],'k');hold on
end%画出三维图

function out=g(v)
p1=20; p2=25; p3=26; p4=20; p5=20; p6=20;%输入支杆长度
x1=0; x2=-16; x3=-19; x4=-19; x5=-16;%输入平台顶点 x 坐标
y1=4; y2=13; y3=11; y4=-7; y5=-9;%输入平台顶点 y 坐标
xx1=0;xx2=-6; xx3=-30; xx4=-30; xx5=-6;%输入平台底座定点 x 坐标
yy1=28; yy2=31; yy3=17; yy4=11; yy5=-3;%输入平台底座定点 y 坐标
X=v(1);
Y=v(2);
Z=v(3);
psi=v(4);
theta=v(5);
phi=v(6);
p=[p1;p2;p3;p4;p5;p6];
T=[X;Y;Z];%平移矩阵
```

```
R=[cos(psi) -sin(psi) 0;sin(psi) cos(psi) 0;0 0 1]*[cos(theta) 0 sin(theta);0 1 0;-sin(theta) 0 cos(theta)]*[1
%旋转矩阵
xp=[0,x1,x2,x3,x4,x5];
yp=[0,y1,y2,y3,y4,y5];
z=zeros(1,6);
A=[xp;yp;z];
xb=[0,xx1,xx2,xx3,xx4,xx5];
yb=[0,yy1,yy2,yy3,yy4,yy5];
B=[xb;yb;z];
for i=1:6
    l(:,i)=T+R*A(:,i)-B(:,i);
end
for i=1:6
    out(i,1)=(p(i,1))^2-(l(:,i))'*l(:,i);%导出 gi,合并为 g
end
end
```

计算 6 个自由度相应结果为 $\begin{pmatrix} -3.4469557159719698767603820145102 \\ 7.9676162352545262027212995307 23 \\ 18.017646567180649831017508642222 \\ 0.023390306070863206015935701738644 \\ -0.078688271816232459173105224884905 \\ 0.19162395249659578920609236083318 \end{pmatrix}$,

画图得



六、结果分析

1-3 题、误差分析

```
x01=fzero(@f,pi/4)
x02=fzero(@f,-pi/4)
Ef1=abs(x01-pi/4)%前向误差
Ef2=abs(x02+pi/4)
Eb1=abs(f(x01))%后向误差
```

```
Eb2=abs(f(x02))

Ef1 =

     3.330669073875470e-16

Ef2 =

     3.330669073875470e-16

Eb1 =

     2.273736754432321e-13

Eb2 =

     2.273736754432321e-13
```

数量级均较小，逼近效果较好

4 题、误差分析

```
Eb1=abs(f(t1))%零点的后向误差
Eb2=abs(f(t2))
Eb3=abs(f(t3))
Eb4=abs(f(t4))

Eb1 =

     0
Eb2 =

     5.456968210637569e-12
Eb3 =

     0
Eb4 =

     0

Ep1=max(abs([(norm([u1(1),v1(1)])-p1),(norm([u2(1)-x1,v2(1)])-p2),(norm([u3(1)-x2,v3(1)-y2])-p3)]))
Ep2=max(abs([(norm([u1(2),v1(2)])-p1),(norm([u2(2)-x1,v2(2)])-p2),(norm([u3(2)-x2,v3(2)-y2])-p3)]))
Ep3=max(abs([(norm([u1(3),v1(3)])-p1),(norm([u2(3)-x1,v2(3)])-p2),(norm([u3(3)-x2,v3(3)-y2])-p3)]))
Ep4=max(abs([(norm([u1(4),v1(4)])-p1),(norm([u2(4)-x1,v2(4)])-p2),(norm([u3(4)-x2,v3(4)-y2])-p3)]))
%关于支杆长度的前向误差
Ep1 =

     8.881784197001252e-16
Ep2 =

     8.881784197001252e-16
Ep3 =

     0
Ep4 =

     4.440892098500626e-16
```

7 题、用时分析

□□ 26.105525 秒。

考虑本题的时候，对于对零点计数的函数思考了很多种。由于收敛零点的不确定性，构思了许多种方式都失效了，最后还是使用了细分点，考虑相邻变号情况以确定零点存在情况。对于每个 p2，都需要代入 f 大约 20000 次，而函数 f 本身计算也较复杂，在后续二分法中不断地计算，造成了用时较久。

8、迭代次数、误差分析

```
i=0;%检查迭代次数
while norm(v1-v0)>=1*10^(-50)%高斯-牛顿迭代法
    v0=v1;
    J=vpa(subs(Jg,v,v0));
    v1=v0-J\g(v0);
    i=i+1;
end
v1%输出结果
i%输出迭代次数
```

得到 i=23,即 23 次迭代就能得到范数精度在 10^(-50)的数量级，说明了高斯_牛顿迭代法的速度之快。令 g(v1),得到 $\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$，说明本数据下，高斯_牛顿迭代法收敛精准，适合用于三维 6 自由度 Stewart 前向运动学问题数值解。