HW8

吴程锴 ckwu1201@163.com

一、模型预测控制算法简介

基于模型预测控制（MPC）的无人机轨迹跟踪是现代控制理论研究中的一个重点，其原理是：在当前时刻，通过被控对象的状态方程、输出方程和控制输入预测出系统未来的输出，优化控制输入使得输出与参考输出的误差最小，将得到的最优控制输入的第一个元素作用于被控对象；一直重复该过程，直到跟踪结束。

二、无约束模型预测控制

2.1 模型

根据四旋翼无人机系统的微分平坦特性，平坦输出

$$\sigma = [x, y, z, \psi]^T \tag{0.1}$$

及其各阶导数能够线性表示无人机位姿、速度、角速度等状态变量和输入变量，其中$[x, y, z]^T$为无人机的三维空间坐标，ψ为无人机的偏航角。并且，在这里我们不考虑无人机的姿态，即不考虑偏航角ψ，所以我们可以直接控制$[x, y, z]^T$使得无人机跟踪参考轨迹。且无人机的三个维度相互独立，可以分开控制。

使用三阶积分器来建立系统模型

$$\begin{cases} \dot{p}_m = v_m \\ \dot{v}_m = a_m \\ \dot{a}_m = j_m \end{cases} \tag{0.2}$$

其中$[p_m, v_m, a_m, j_m]$分别为m轴的位置、速度、加速度和加加速度，$m = x, y, z$。系统输入为加加速度

$$u_m = j_m \tag{0.3}$$

2.2 预测

采用直接离散化的方式来定义输入的参数空间，设预测时间间隔为dt，系统的离散模型为

$$\begin{cases} p_{k+1}=p_k+v_k dt+\dfrac{1}{2}a_k dt^2+\dfrac{1}{6}j_k dt^3 \\ v_{k+1}=v_k+a_k dt+\dfrac{1}{2}j_k dt^2 \\ a_{k+1}=a_k+j_k dt \end{cases} \tag{0.4}$$

设预测时域和控制时域均为N，k时刻预测$k+1$到$k+N$的系统状态和输入为

$$\begin{cases} P_k=[p_{k+1},p_{k+2},\cdots,p_{k+N}]^T \\ V_k=[v_{k+1},v_{k+2},\cdots,v_{k+N}]^T \\ A_k=[a_{k+1},a_{k+2},\cdots,a_{k+N}]^T \\ J_k=[j_k,j_{k+2},\cdots,j_{k+N-1}]^T \end{cases} \tag{0.5}$$

将(0.4)代入(0.5)，将 k 时刻预测的系统状态$[P_k,V_k,A_k]$写成输入J_k的形式

$$\begin{cases} P_k=T_p J_k+B_p \\ V_k=T_v J_k+B_v \\ A_k=T_a J_k+B_a \end{cases} \tag{0.6}$$

2.3 控制

优化问题为得到最优的输入J_k，使得无人机轨迹与参考轨迹P_k^r的距离最小，并且，我们希望速度、加速度、加加速度越小越好，让无人机更加省力，则目标函数为

$$\min_{J_k}\ \omega_1(P_k-P_k^r)^T(P_k-P_k^r)+\omega_2 {V_k}^T V_k+\omega_3 {A_k}^T A_k+\omega_4 {J_k}^T J_k \tag{0.7}$$

代入(0.6)，则(0.7)等价于

$$\begin{aligned}\min_{J_k}\ & J^T(\omega_1 {T_p}^T T_p+\omega_2 {T_v}^T T_v+\omega_3 {T_a}^T T_a+\omega_4 I)J \\ & +2\big(\omega_1(B_p-P_k^r)^T T_p+\omega_2 {B_v}^T T_v+\omega_3 {B_a}^T T_a\big)J\end{aligned} \tag{0.8}$$

求解无约束优化问题(0.8)得到最优的输入

$$J_k^*=[j_k^*,j_{k+1}^*,\cdots,j_{k+N-1}^*]^T \tag{0.9}$$

将j_k^*作用到系统，在$k+1$时刻重复以上步骤直到跟踪完成。

2.4 仿真

设初始状态为

$$\begin{cases} p_0=10 \\ v_0=0 \\ a_0=0 \end{cases} \tag{0.10}$$

想要无人机悬停在$p=2$处。仿真结果如图 1 所示

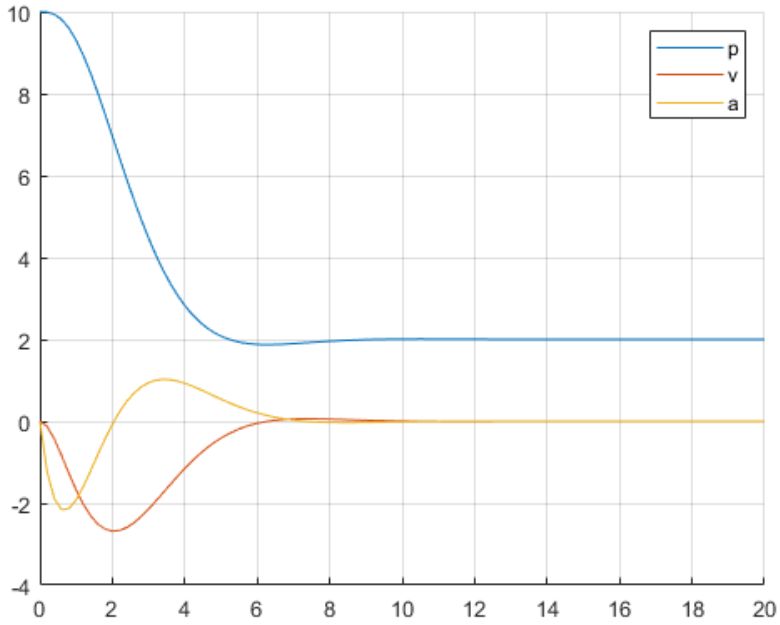

图 1 无约束 MPC

从图 1 可以看出无人机轨迹的超调量很小，且很快收敛到 2。

2.5 代码

```
clc,clear
close all
%%
p0=10;
v0=-2;
a0=0;
N=20;
dt=0.2;
log=[0,p0,v0,a0];
w1=1;
w2=1;
w3=1;
w4=1;
w5=1e4;
Pr=ones(N,1)*2;
for t=0.2:0.2:20
    [Tp,Tv,Ta,Bp,Bv,Ba]=getPredictionMatrix(N,dt,p0,v0,a0);
    H=w1*(Tp'*Tp)+w2*(Tv'*Tv)+w3*(Ta'*Ta)+w4*eye(N);
    H=blkdiag(H,w5*eye(2*N));
    Bp=Bp-Pr;
    F=w1*Bp'*Tp+w2*Bv'*Tv+w3*Ba'*Ta;
```

```
    F=[F,zeros(1,2*N)];
    A=[Tv,zeros(N),-eye(N);-Tv,-eye(N),zeros(N);Ta,zeros(N),zeros(N);-Ta,zeros(N),zeros(N);zeros(size(Ta)),-eye(N),zeros(N);zeros(size(Ta)),zeros(N),-eye(N)];
    b=[2*ones(N,1)-Bv;ones(N,1)+Bv;ones(N,1)-Ba;ones(N,1)+Ba;zeros(N,1);zeros(N,1)];
    J=quadprog(H,F);
    j=J(1);
    % 模拟系统
    p0=p0+v0*dt+0.5*a0*dt^2+1/6*j*dt^3;
    v0=v0+a0*dt+0.5*j*dt^2;
%     if t==4
%         v0=v0+4;
%     end
    a0=a0+j*dt;
    log=[log;t,p0,v0,a0];
end
figure()
hold on
grid on
plot(log(:,1),log(:,2))
plot(log(:,1),log(:,3))
plot(log(:,1),log(:,4))
legend('p','v','a')

function [Tp,Tv,Ta,Bp,Bv,Ba]=getPredictionMatrix(N,dt,p0,v0,a0)

Tp=zeros(N);
Tv=zeros(N);
Ta=zeros(N);

for i=1:N
    Ta(i,1:i)=ones(1,i)*dt;
end

for i=1:N
    for j=1:i
        Tv(i,j)=(i-j+0.5)*dt^2;
    end
end

for i=1:N
    for j=1:i
        Tp(i,j)=((i-j+1)*(i-j)/2+1/6)*dt^3;
    end
```

```
end

Bp=ones(N,1)*p0;
Bv=ones(N,1)*v0;
Ba=ones(N,1)*a0;

for i=1:N
    Bp(i)=Bp(i)+i*dt*v0+i^2/2*a0*dt^2;
    Bv(i)=Bv(i)+i*dt*a0;
end
```

三、有约束模型预测控制

物理上，无人机有最大速度、加速度等，所以，我们需要给系统加上约束。约束又分为软约束和硬约束，其中软约束往往约束系统的状态，如速度、加速度等，硬约束往往约束系统的输入。这是由于系统的状态可能受到外界干扰，导致系统状态初值超出约束范围，如果将系统状态作为硬约束，则优化问题将无解；而系统的输入往往是人工给定的，如果给定的输入过大会导致系统损坏，所以要将输入约束设定为硬约束。

3.1 硬约束

$$J_{\min} \leqslant J_k \leqslant J_{\max} \tag{1.1}$$

3.2 软约束

一种将硬约束转化为软约束的方法是加入辅助变量L，输入变量变为

$$J^{'} = \begin{bmatrix} J \\ L \end{bmatrix} \tag{1.2}$$

这里以速度约束为例进行讲解

$$V_{\min} \leqslant V_k = T_v J_k + B_v \leqslant V_{\max} \tag{1.3}$$

$$\begin{aligned} &\min\ \omega_5 L^T L \\ &\begin{cases} T_v J_k \leqslant V_{\max} - B_v + L \\ -T_v J_k \leqslant B_v - V_{\min} + L \\ -L \leqslant 0 \end{cases} \end{aligned} \tag{1.4}$$

当$\omega_5 = inf$，(1.3)等价于(1.4)。

3.3 仿真

设初始状态为

$$\begin{cases} p_0 = 10 \\ v_0 = -2 \\ a_0 = 0 \end{cases} \tag{1.5}$$

约束为

$$\begin{cases} -1 \leqslant v_k \ll 1 \\ -1 \leqslant a_k \ll 1 \\ -1 \leqslant j_k \ll 1 \end{cases}, \forall k \tag{1.6}$$

想要无人机悬停在$p = 2$处。仿真结果如图 2 所示。

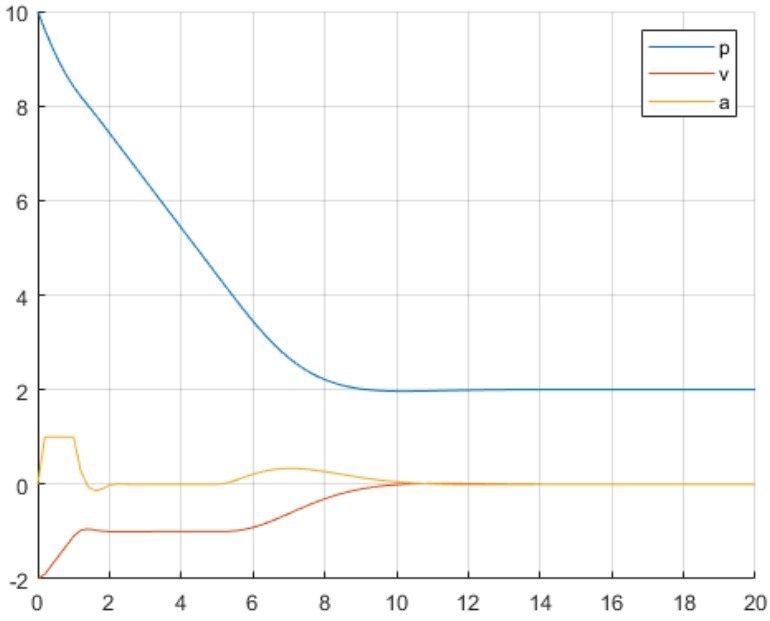

图 2 有约束 MPC

从图 2 可以看出无人机的速度一开始超出了约束，控制器优先将速度控制到约束范围内。

3.4 代码

```
clc,clear
close all
%%
p0=10;
v0=-2;
a0=0;
```

```
N=20;
dt=0.2;
log=[0,p0,v0,a0];
w1=1;
w2=1;
w3=1;
w4=1;
w5=1e4;
Pr=ones(N,1)*2;
for t=0.2:0.2:20
    [Tp,Tv,Ta,Bp,Bv,Ba]=getPredictionMatrix(N,dt,p0,v0,a0);
    H=w1*(Tp'*Tp)+w2*(Tv'*Tv)+w3*(Ta'*Ta)+w4*eye(N);
    H=blkdiag(H,w5*eye(2*N));
    Bp=Bp-Pr;
    F=w1*Bp'*Tp+w2*Bv'*Tv+w3*Ba'*Ta;
    F=[F,zeros(1,2*N)];
    A=[Tv,zeros(N),-eye(N);-Tv,-eye(N),zeros(N);Ta,zeros(N),zeros(N);-Ta,zeros(N),zeros(N);zeros(size(Ta)),-eye(N),zeros(N);zeros(size(Ta)),zeros(N),-eye(N)];
    b=[2*ones(N,1)-Bv;ones(N,1)+Bv;ones(N,1)-Ba;ones(N,1)+Ba;zeros(N,1);zeros(N,1)];
    J=quadprog(H,F,A,b);
    j=J(1);
    % 模拟系统
    p0=p0+v0*dt+0.5*a0*dt^2+1/6*j*dt^3;
    v0=v0+a0*dt+0.5*j*dt^2;
%     if t==4
%         v0=v0+4;
%     end
    a0=a0+j*dt;
    log=[log;t,p0,v0,a0];
end
figure()
hold on
grid on
plot(log(:,1),log(:,2))
plot(log(:,1),log(:,3))
plot(log(:,1),log(:,4))
legend('p','v','a')

function [Tp,Tv,Ta,Bp,Bv,Ba]=getPredictionMatrix(N,dt,p0,v0,a0)

Tp=zeros(N);
Tv=zeros(N);
Ta=zeros(N);
```

```
49.
50. for i=1:N
51.     Ta(i,1:i)=ones(1,i)*dt;
52. end
53.
54. for i=1:N
55.     for j=1:i
56.         Tv(i,j)=(i-j+0.5)*dt^2;
57.     end
58. end
59.
60. for i=1:N
61.     for j=1:i
62.         Tp(i,j)=((i-j+1)*(i-j)/2+1/6)*dt^3;
63.     end
64. end
65.
66. Bp=ones(N,1)*p0;
67. Bv=ones(N,1)*v0;
68. Ba=ones(N,1)*a0;
69.
70. for i=1:N
71.     Bp(i)=Bp(i)+i*dt*v0+i^2/2*a0*dt^2;
72.     Bv(i)=Bv(i)+i*dt*a0;
73. end
```

四、无人机轨迹跟踪

4.1 仿真

最后，将控制器扩展到三维空间，要求无人机跟随的轨迹如图 3 所示

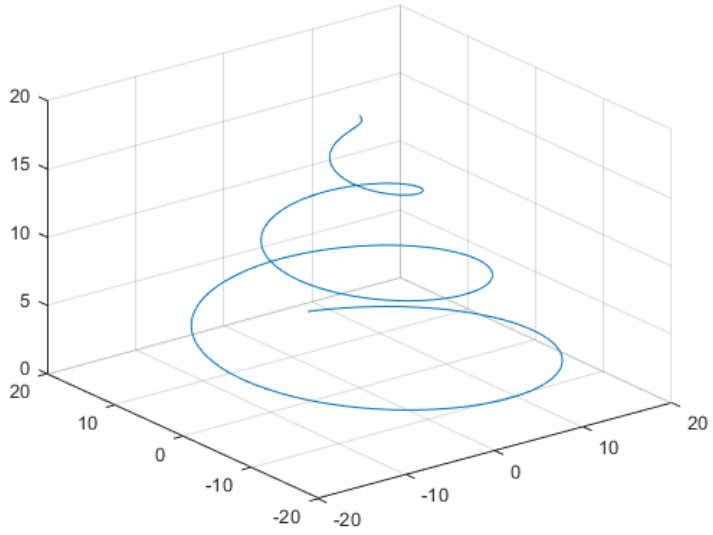

图 3 参考轨迹

轨迹的角速度为$w=0.1rad/s$，z 轴速度为$v_z=-0.1m/s$

约束为

$$\begin{aligned}
&-6\leqslant v_{x,y}\leqslant 6\\
&-1\leqslant v_z\leqslant 6\\
&-3\leqslant a_{x,y}\leqslant 3\\
&-1\leqslant a_z\leqslant 3\\
&-3\leqslant j_{x,y}\leqslant 3\\
&-2\leqslant j_z\leqslant 2
\end{aligned} \tag{2.1}$$

设定无人机初始状态为

$$\begin{cases}
p_0=[0,0,H]\\
v_0=[10,0,0]\\
a_0=[0,0,0]
\end{cases} \tag{2.2}$$

其中p_0为参考轨迹的初始位置。仿真结果如图 4 和图 5 所示。

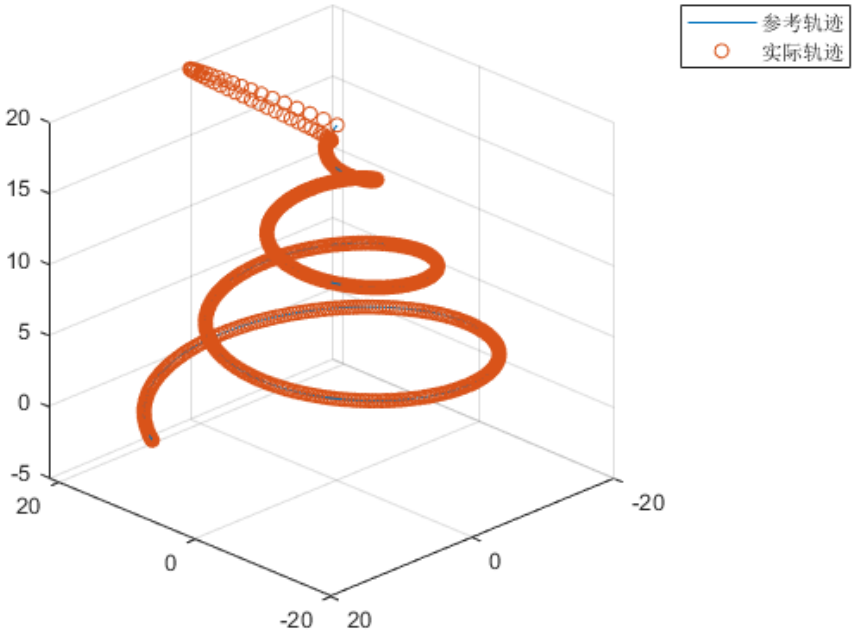

图 4 跟踪结果

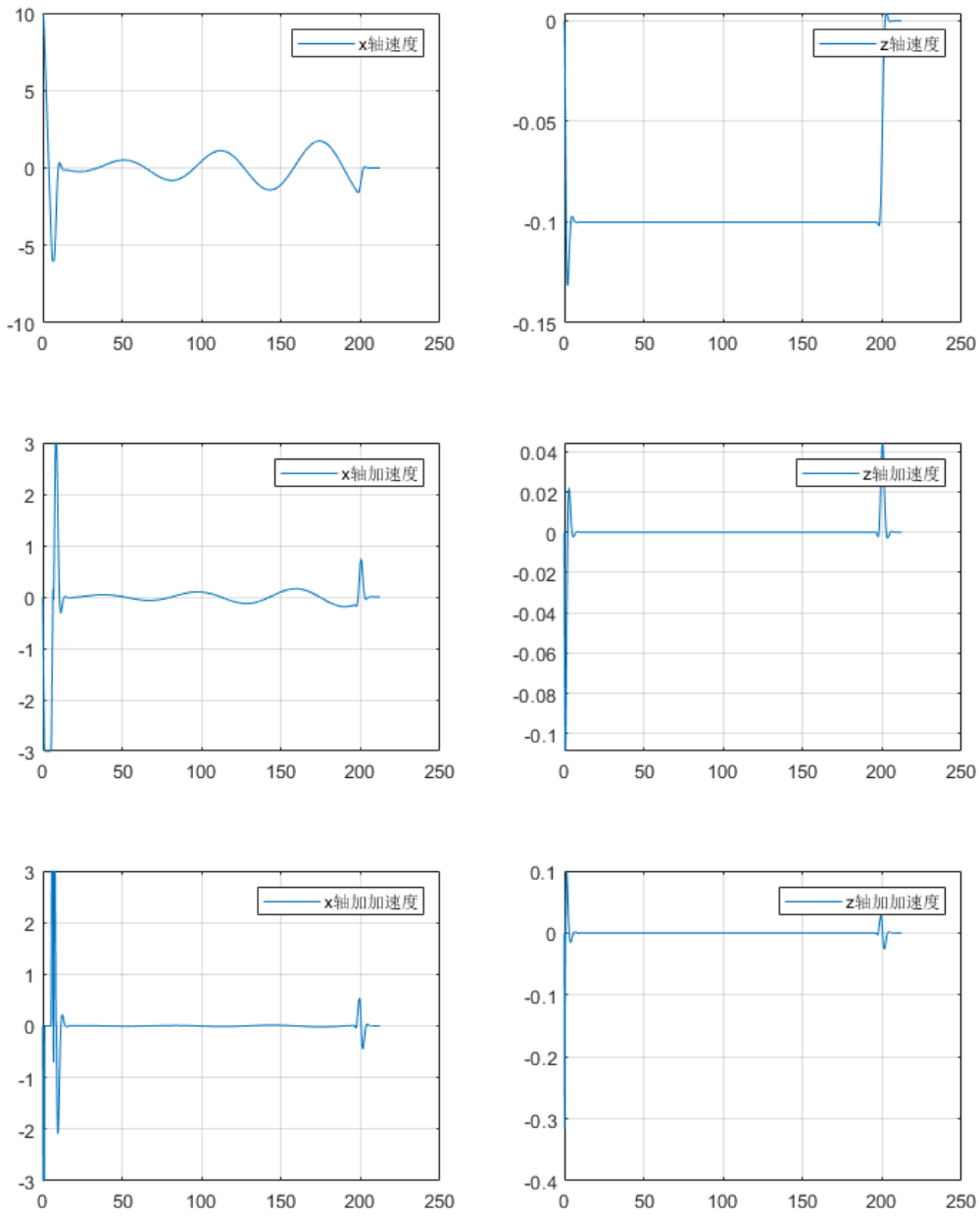

图 5 无人机数据

从结果中可以看出，即使初始速度非常糟糕，控制器也能够将无人机很快地拉到参考轨迹上，且没有出现无解的情况。

4.2 代码

```
clc,clear
close all
%%
dt=0.2;
N=20;
H=20;
R=20;
V=-0.1;
```

```
W=0.1;
path=getPath(dt,H,R,V,W);
figure()
plot3(path(:,1),path(:,2),path(:,3))
grid on

p0=[0,0,H];
v0=[10,0,0];
a0=[0,0,0];
j=[0,0,0];

v_bound=[-6,6;-6,6;-1,6];
a_bound=[-3,3;-3,3;-1,3];
j_bound=[-3,3;-3,3;-2,2];

p_noise=normrnd(0,0.01,size(path,1),3);
v_noise=normrnd(0,0.01,size(path,1),3);
a_noise=normrnd(0,0.01,size(path,1),3);

p_noise=zeros(size(path,1),3);
v_noise=zeros(size(path,1),3);
a_noise=zeros(size(path,1),3);

log=zeros(size(path,1)+1,13);
log(1,:)=[0,p0,v0,a0,j];
w1=10;
w2=1;
w3=1;
w4=1;
w5=1e4;

% Pr=path();

for i=1:size(path,1)
    t=i*dt;
    for k=1:3
%         tic
        [Tp,Tv,Ta,Bp,Bv,Ba]=getPredictionMatrix(N,dt,p0(k),v0(k),a0(k));
        H=w1*(Tp'*Tp)+w2*(Tv'*Tv)+w3*(Ta'*Ta)+w4*eye(N);
        H=blkdiag(H,w5*eye(2*N));
        if i<size(path,1)-N+1
            Bp=Bp-path(i:i+N-1,k);
        else
            Bp=Bp-[path(i:end,k);ones(N-size(path,1)+i-1,1)*path(end,k)];
        end
```

```
        F=w1*Bp'*Tp+w2*Bv'*Tv+w3*Ba'*Ta;
        F=[F,zeros(1,2*N)];
        A=[Tv,zeros(N),-eye(N);-Tv,-eye(N),zeros(N);Ta,zeros(N),zeros(N);-Ta,zeros(N),zeros(N);zeros(size(Ta)),-eye(N),zeros(N);zeros(size(Ta)),zeros(N),-eye(N);eye(N),zeros(N),zeros(N);-eye(N),zeros(N),zeros(N)];
        b=[v_bound(k,2)*ones(N,1)-Bv;-v_bound(k,1)*ones(N,1)+Bv;a_bound(k,2)*ones(N,1)-Ba;-a_bound(k,1)*ones(N,1)+Ba;zeros(N,1);zeros(N,1);j_bound(k,2)*ones(N,1);-j_bound(k,1)*ones(N,1)];
        J=quadprog(H,F,A,b);
        j(k)=J(1);
        % 模拟系统
        p0(k)=p0(k)+v0(k)*dt+0.5*a0(k)*dt^2+1/6*j(k)*dt^3+p_noise(i,k);
        v0(k)=v0(k)+a0(k)*dt+0.5*j(k)*dt^2+v_noise(i,k);
        a0(k)=a0(k)+j(k)*dt+a_noise(i,k);

%         toc
    end
    log(i+1,:)=[t,p0,v0,a0,j];
end
figure()
plot3(path(:,1),path(:,2),path(:,3))
hold on
grid on
plot3(log(:,2),log(:,3),log(:,4),'o')
legend('参考轨迹','实际轨迹')

figure()
grid on
subplot(3,2,1)
plot(log(:,1),log(:,5))
legend('x 轴速度')
grid on

subplot(3,2,2)
plot(log(:,1),log(:,7))
legend('z 轴速度')
grid on

subplot(3,2,3)
plot(log(:,1),log(:,8))
legend('x 轴加速度')
grid on

```

```
subplot(3,2,4)
plot(log(:,1),log(:,10))
legend('z 轴加速度')
grid on

subplot(3,2,5)
plot(log(:,1),log(:,11))
    legend('x 轴加加速度')
    grid on

    subplot(3,2,6)
    plot(log(:,1),log(:,13))
    legend('z 轴加加速度')
    grid on

    function [Tp,Tv,Ta,Bp,Bv,Ba]=getPredictionMatrix(N,dt,p0,v0,a0)
    Tp=zeros(N);
    Tv=zeros(N);
    Ta=zeros(N);
    for i=1:N
        Ta(i,1:i)=ones(1,i)*dt;
    end
    for i=1:N
        for j=1:i
            Tv(i,j)=(i-j+0.5)*dt^2;
        end
    end
    for i=1:N
        for j=1:i
            Tp(i,j)=((i-j+1)*(i-j)/2+1/6)*dt^3;
        end
    end
    Bp=ones(N,1)*p0;
    Bv=ones(N,1)*v0;
    Ba=ones(N,1)*a0;
    for i=1:N
        Bp(i)=Bp(i)+i*dt*v0+i^2/2*a0*dt^2;
        Bv(i)=Bv(i)+i*dt*a0;
    end

    function path=getPath(dt,H,R,V,W)
    T=H/abs(V);
    t=0:dt:T;
    r=R/T*t;
    x=r.*cos(W*t);
```

```
y=r.*sin(W*t);
z=H+t*V;
path=[x',y',z'];
path=[path;ones(60,3).*[x(end),y(end),z(end)]];
```