HW8

吴程锴 ckwu1201@163.com

# 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。

# 无约束模型预测控制

## 模型

根据四旋翼无人机系统的微分平坦特性，平坦输出



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，其中为无人机的三维空间坐标，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，所以我们可以直接控制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



其中分别为轴的位置、速度、加速度和加加速度，。系统输入为加加速度



## 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，系统的离散模型为



设预测时域和控制时域均为，时刻预测到的系统状态和输入为



将代入，将k时刻预测的系统状态写成输入的形式



## 控制

优化问题为得到最优的输入，使得无人机轨迹与参考轨迹的距离最小，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



代入，则等价于



求解无约束优化问题得到最优的输入



将作用到系统，在时刻重复以上步骤直到跟踪完成。

## 仿真

设初始状态为



想要无人机悬停在处。仿真结果如图 1所示

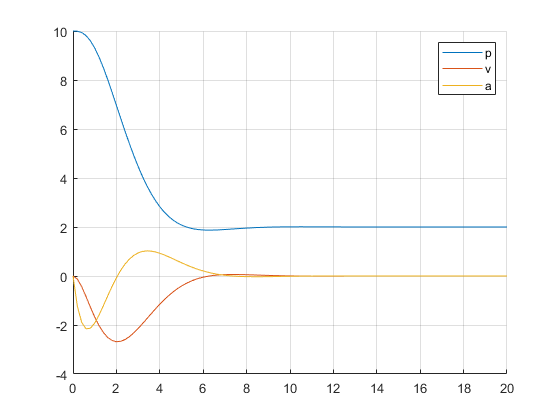


图 1无约束MPC

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。

## 代码

1. clc,clear
2. close all
3. %%
4. p0=10;
5. v0=-2;
6. a0=0;
7. N=20;
8. dt=0.2;
9. log=[0,p0,v0,a0];
10. w1=1;
11. w2=1;
12. w3=1;
13. w4=1;
14. w5=1e4;
15. Pr=ones(N,1)\*2;
16. **for** t=0.2:0.2:20
17. [Tp,Tv,Ta,Bp,Bv,Ba]=getPredictionMatrix(N,dt,p0,v0,a0);
18. H=w1\*(Tp'\*Tp)+w2\*(Tv'\*Tv)+w3\*(Ta'\*Ta)+w4\*eye(N);
19. H=blkdiag(H,w5\*eye(2\*N));
20. Bp=Bp-Pr;
21. F=w1\*Bp'\*Tp+w2\*Bv'\*Tv+w3\*Ba'\*Ta;
22. F=[F,zeros(1,2\*N)];
23. 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)];
24. b=[2\*ones(N,1)-Bv;ones(N,1)+Bv;ones(N,1)-Ba;ones(N,1)+Ba;zeros(N,1);zeros(N,1)];
25. J=quadprog(H,F);
26. j=J(1);
27. % 模拟系统
28. p0=p0+v0\*dt+0.5\*a0\*dt^2+1/6\*j\*dt^3;
29. v0=v0+a0\*dt+0.5\*j\*dt^2;
30. %     **if** t==4
31. %         v0=v0+4;
32. %     end
33. a0=a0+j\*dt;
34. log=[log;t,p0,v0,a0];
35. end
36. figure()
37. hold on
38. grid on
39. plot(log(:,1),log(:,2))
40. plot(log(:,1),log(:,3))
41. plot(log(:,1),log(:,4))
42. legend('p','v','a')
44. function [Tp,Tv,Ta,Bp,Bv,Ba]=getPredictionMatrix(N,dt,p0,v0,a0)
46. Tp=zeros(N);
47. Tv=zeros(N);
48. Ta=zeros(N);
50. **for** i=1:N
51. Ta(i,1:i)=ones(1,i)\*dt;
52. end
54. **for** i=1:N
55. **for** j=1:i
56. Tv(i,j)=(i-j+0.5)\*dt^2;
57. end
58. end
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
66. Bp=ones(N,1)\*p0;
67. Bv=ones(N,1)\*v0;
68. Ba=ones(N,1)\*a0;
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

# 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。

## 硬约束



## 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，输入变量变为



这里以速度约束为例进行讲解





当，等价于。

## 仿真

设初始状态为



约束为



想要无人机悬停在处。仿真结果如图 2所示。

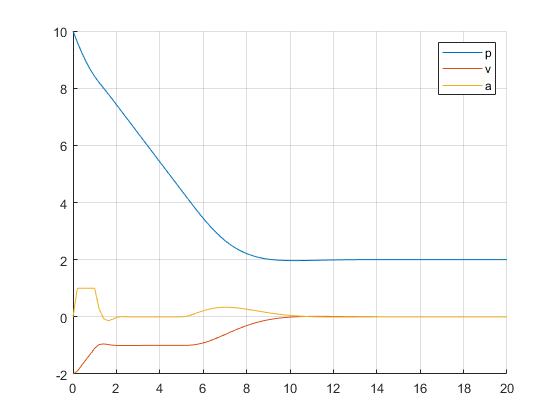


图 2有约束MPC

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。

## 代码

1. clc,clear
2. close all
3. %%
4. p0=10;
5. v0=-2;
6. a0=0;
7. N=20;
8. dt=0.2;
9. log=[0,p0,v0,a0];
10. w1=1;
11. w2=1;
12. w3=1;
13. w4=1;
14. w5=1e4;
15. Pr=ones(N,1)\*2;
16. **for** t=0.2:0.2:20
17. [Tp,Tv,Ta,Bp,Bv,Ba]=getPredictionMatrix(N,dt,p0,v0,a0);
18. H=w1\*(Tp'\*Tp)+w2\*(Tv'\*Tv)+w3\*(Ta'\*Ta)+w4\*eye(N);
19. H=blkdiag(H,w5\*eye(2\*N));
20. Bp=Bp-Pr;
21. F=w1\*Bp'\*Tp+w2\*Bv'\*Tv+w3\*Ba'\*Ta;
22. F=[F,zeros(1,2\*N)];
23. 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)];
24. b=[2\*ones(N,1)-Bv;ones(N,1)+Bv;ones(N,1)-Ba;ones(N,1)+Ba;zeros(N,1);zeros(N,1)];
25. J=quadprog(H,F,A,b);
26. j=J(1);
27. % 模拟系统
28. p0=p0+v0\*dt+0.5\*a0\*dt^2+1/6\*j\*dt^3;
29. v0=v0+a0\*dt+0.5\*j\*dt^2;
30. %     **if** t==4
31. %         v0=v0+4;
32. %     end
33. a0=a0+j\*dt;
34. log=[log;t,p0,v0,a0];
35. end
36. figure()
37. hold on
38. grid on
39. plot(log(:,1),log(:,2))
40. plot(log(:,1),log(:,3))
41. plot(log(:,1),log(:,4))
42. legend('p','v','a')
44. function [Tp,Tv,Ta,Bp,Bv,Ba]=getPredictionMatrix(N,dt,p0,v0,a0)
46. Tp=zeros(N);
47. Tv=zeros(N);
48. Ta=zeros(N);
50. **for** i=1:N
51. Ta(i,1:i)=ones(1,i)\*dt;
52. end
54. **for** i=1:N
55. **for** j=1:i
56. Tv(i,j)=(i-j+0.5)\*dt^2;
57. end
58. end
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
66. Bp=ones(N,1)\*p0;
67. Bv=ones(N,1)\*v0;
68. Ba=ones(N,1)\*a0;
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

# 无人机轨迹跟踪

## 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

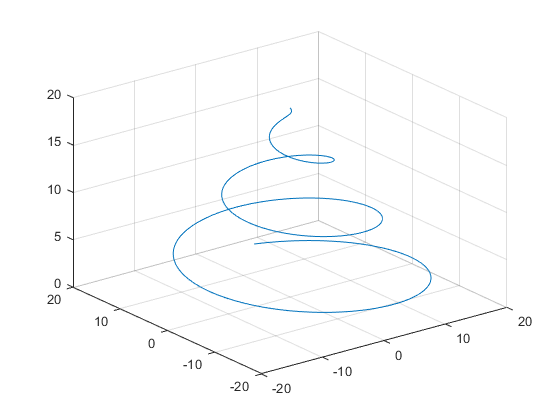


图 3参考轨迹

轨迹的角速度为，z轴速度为

约束为



设定无人机初始状态为



其中为参考轨迹的初始位置。仿真结果如图 4和图 5所示。

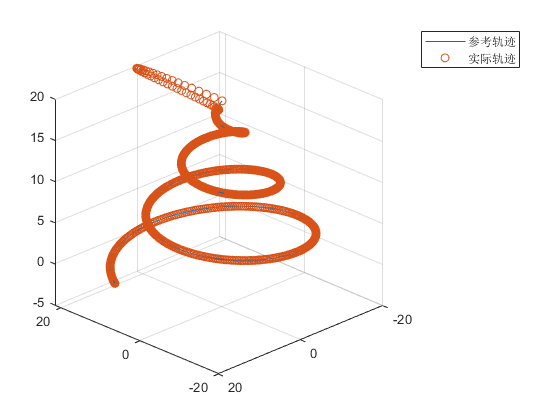


图 4跟踪结果

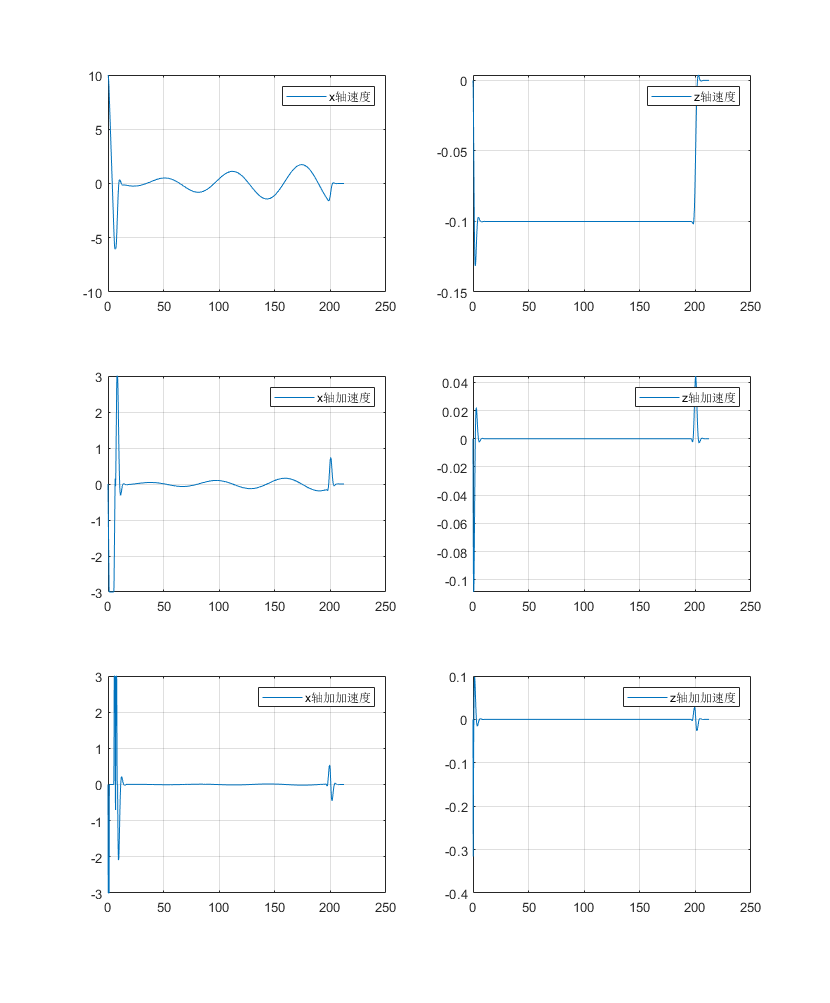


图 5无人机数据

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。

## 代码

1. clc,clear
2. close all
3. %%
4. dt=0.2;
5. N=20;
6. H=20;
7. R=20;
8. V=-0.1;
9. W=0.1;
10. path=getPath(dt,H,R,V,W);
11. figure()
12. plot3(path(:,1),path(:,2),path(:,3))
13. grid on
15. p0=[0,0,H];
16. v0=[10,0,0];
17. a0=[0,0,0];
18. j=[0,0,0];
20. v\_bound=[-6,6;-6,6;-1,6];
21. a\_bound=[-3,3;-3,3;-1,3];
22. j\_bound=[-3,3;-3,3;-2,2];
24. p\_noise=normrnd(0,0.01,size(path,1),3);
25. v\_noise=normrnd(0,0.01,size(path,1),3);
26. a\_noise=normrnd(0,0.01,size(path,1),3);
28. p\_noise=zeros(size(path,1),3);
29. v\_noise=zeros(size(path,1),3);
30. a\_noise=zeros(size(path,1),3);
32. log=zeros(size(path,1)+1,13);
33. log(1,:)=[0,p0,v0,a0,j];
34. w1=10;
35. w2=1;
36. w3=1;
37. w4=1;
38. w5=1e4;
40. % Pr=path();
42. **for** i=1:size(path,1)
43. t=i\*dt;
44. **for** k=1:3
45. %         tic
46. [Tp,Tv,Ta,Bp,Bv,Ba]=getPredictionMatrix(N,dt,p0(k),v0(k),a0(k));
47. H=w1\*(Tp'\*Tp)+w2\*(Tv'\*Tv)+w3\*(Ta'\*Ta)+w4\*eye(N);
48. H=blkdiag(H,w5\*eye(2\*N));
49. **if** i<size(path,1)-N+1
50. Bp=Bp-path(i:i+N-1,k);
51. **else**
52. Bp=Bp-[path(i:end,k);ones(N-size(path,1)+i-1,1)\*path(end,k)];
53. end
54. F=w1\*Bp'\*Tp+w2\*Bv'\*Tv+w3\*Ba'\*Ta;
55. F=[F,zeros(1,2\*N)];
56. 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)];
57. 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)];
58. J=quadprog(H,F,A,b);
59. j(k)=J(1);
60. % 模拟系统
61. p0(k)=p0(k)+v0(k)\*dt+0.5\*a0(k)\*dt^2+1/6\*j(k)\*dt^3+p\_noise(i,k);
62. v0(k)=v0(k)+a0(k)\*dt+0.5\*j(k)\*dt^2+v\_noise(i,k);
63. a0(k)=a0(k)+j(k)\*dt+a\_noise(i,k);
65. %         toc
66. end
67. log(i+1,:)=[t,p0,v0,a0,j];
68. end
69. figure()
70. plot3(path(:,1),path(:,2),path(:,3))
71. hold on
72. grid on
73. plot3(log(:,2),log(:,3),log(:,4),'o')
74. legend('参考轨迹','实际轨迹')
76. figure()
77. grid on
78. subplot(3,2,1)
79. plot(log(:,1),log(:,5))
80. legend('x轴速度')
81. grid on
83. subplot(3,2,2)
84. plot(log(:,1),log(:,7))
85. legend('z轴速度')
86. grid on
88. subplot(3,2,3)
89. plot(log(:,1),log(:,8))
90. legend('x轴加速度')
91. grid on
93. subplot(3,2,4)
94. plot(log(:,1),log(:,10))
95. legend('z轴加速度')
96. grid on
98. subplot(3,2,5)
99. plot(log(:,1),log(:,11))
100. legend('x轴加加速度')
101. grid on
103. subplot(3,2,6)
104. plot(log(:,1),log(:,13))
105. legend('z轴加加速度')
106. grid on
108. function [Tp,Tv,Ta,Bp,Bv,Ba]=getPredictionMatrix(N,dt,p0,v0,a0)
109. Tp=zeros(N);
110. Tv=zeros(N);
111. Ta=zeros(N);
112. **for** i=1:N
113. Ta(i,1:i)=ones(1,i)\*dt;
114. end
115. **for** i=1:N
116. **for** j=1:i
117. Tv(i,j)=(i-j+0.5)\*dt^2;
118. end
119. end
120. **for** i=1:N
121. **for** j=1:i
122. Tp(i,j)=((i-j+1)\*(i-j)/2+1/6)\*dt^3;
123. end
124. end
125. Bp=ones(N,1)\*p0;
126. Bv=ones(N,1)\*v0;
127. Ba=ones(N,1)\*a0;
128. **for** i=1:N
129. Bp(i)=Bp(i)+i\*dt\*v0+i^2/2\*a0\*dt^2;
130. Bv(i)=Bv(i)+i\*dt\*a0;
131. end
133. function path=getPath(dt,H,R,V,W)
134. T=H/abs(V);
135. t=0:dt:T;
136. r=R/T\*t;
137. x=r.\*cos(W\*t);
138. y=r.\*sin(W\*t);
139. z=H+t\*V;
140. path=[x',y',z'];
141. path=[path;ones(60,3).\*[x(end),y(end),z(end)]];