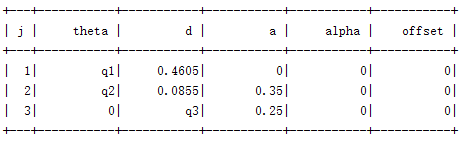
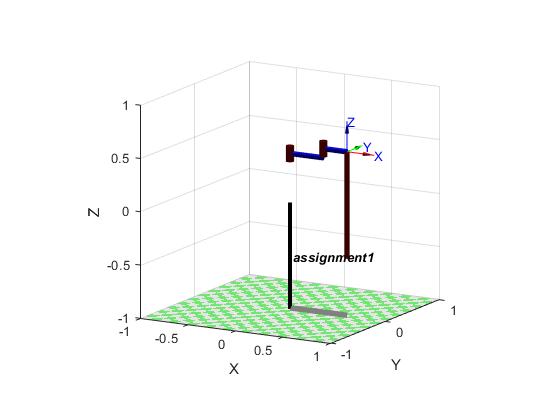
机械臂建模与控制

1机械臂建模





Code

clear all;

clc;

L1=Link([0,0.4605,0,0,0],'modified');

L2=Link([0,0.0855,0.350,0,0],'modified');

L3=Link([0,0,0.250,0,1],'modified');

robot=SerialLink([L1 L2 L3],'name','assignment1');

robot.plot([0 0 0],'workspace',[-1 1 -1 1 -1 1])

robot.display()

2单关节轨迹跟随控制

使用二阶方程对单关节建模，



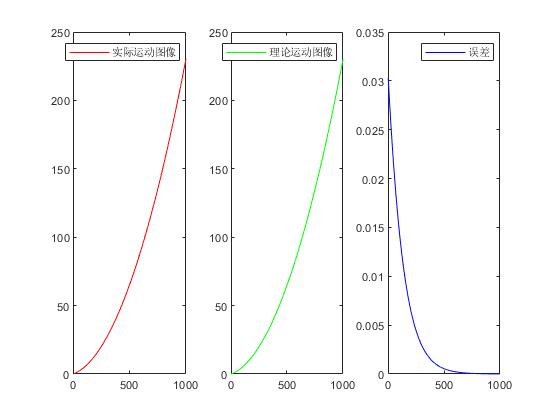




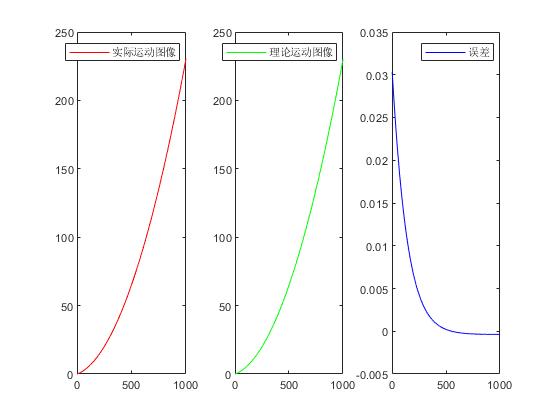
带入后，

上式可以编程画出机械臂的实际运动图像以及与理论图像的误差

Image



可以看出，随着时间增加，实际轨迹与目标轨迹之间的差别越来越小。但积分系数ki过大时，则会产生较为明显的超调。例如令ki=0.0001，则



Code

%%单关节控制

%%本次控制为轨迹跟踪控制，包含恒定干扰

%%控制对象为实际的运动轨迹x(t),使其图像尽可能靠近目标轨迹xd(t)

%%xd(t)存在二阶导数

%%恒定干扰数值：t\_dist=10

%% m=5 b=10 k=8

%%kv=2,kp=1

clear all

kv=2;

kp=1;

ki=0.0000001;

v=3;%%初始速度

m=5;%%初始化m b k

b=10;

k=8;

e=10\*rand();%%实际x在输出时由于各种因素产生的随机干扰误差，本次没添加

err\_sum=0;

err\_dot=0;

err\_last=0;

err=0;

t=0:0.01:10;

record=zeros(1,size(t,2)-1);%%记录实际x与目标轨迹xd的误差变化

x=zeros(1,size(t,2)-1);%%实际的x轨迹，初始化

t\_dist=10;

num=2;

xd=2\*t.\*t+3\*t;%%预设x的轨迹

f=m\*4+b\*(4\*t+3)+k\*xd+t\_dist;%%模型法提供的力

for i=0.01:0.01:10

err=xd(num)-x(num-1);

err\_dot=(err-err\_last)/0.01;

err\_sum=err\_sum+err;

f\_servo=4+kv\*err\_dot+kp\*err+ki\*err\_sum;%%伺服法提供的力

record(num-1)=err

x(num)=x(num-1)+0.5\*f\_servo\*0.01^2+v\*0.01%%print 实际轨迹x在各个时间点的值，初中物理

v\*t+0.5\*a\*t^2

err\_last=err;

v=v+f\_servo\*0.01;

num=num+1;

end

subplot(1,3,1)

plot(x,'r')

legend('实际运动图像')

subplot(1,3,2)

plot(xd,'g')

legend("理论运动图像")

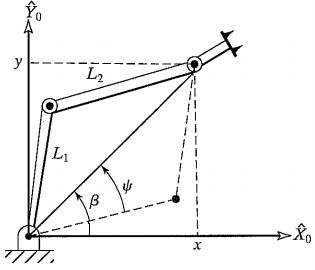
subplot(1,3,3)

plot(record,'b')

legend("误差")

3关节角反解

设杆1、杆2长度分别为，由建模知，变量为



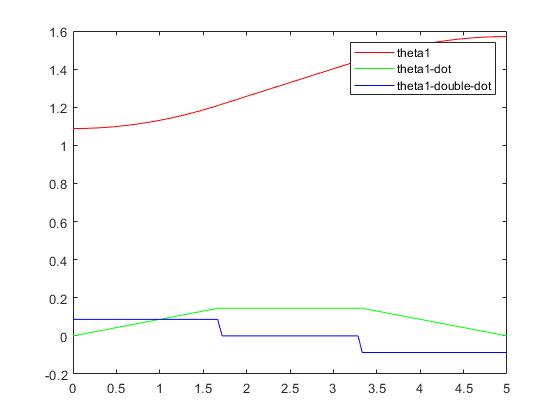


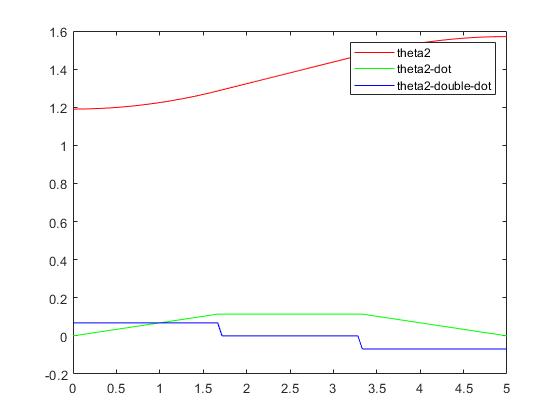


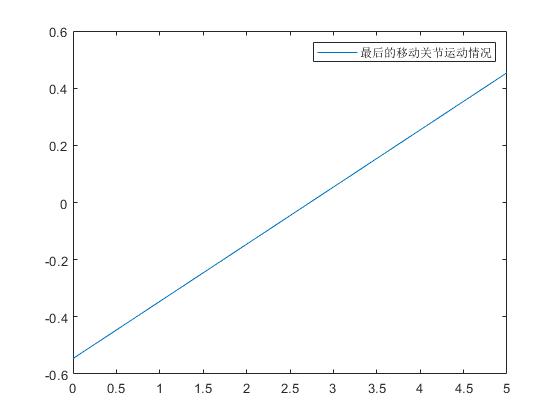
4轨迹规划

本仿真的轨迹规划不涉及中间点，采用二次函数过渡的方案，使用matlab中的lspb函数辅助进行轨迹设计，为方便轨迹设计，采用默认速度，不对速度进行额外约束。针对多解情况，采用使所有关节角转动变化量最小的原则来选择目标解。

使机械臂末端由(0, 0.5, 0)变换到（0.25，0.35，1）各个关节变化如图，同时程序运行结果将展示运动动画。







附录

反解+轨迹规划代码

cclear all

global l1 l2 d1 d2

l1=0.35;

l2=0.25;

d1=0.4605;

d2=0.0855;

coordinate1=input('输入初始坐标');

x1=coordinate1(1);

y1=coordinate1(2);

z1=coordinate1(3);

if (sqrt(x1^2+y1^2)>(l1+l2)||sqrt(x1^2+y1^2)<(l1-l2))

fprintf("坐标超范围")

return

end

coordinate2=input('输入变化坐标');

x2=coordinate2(1);

y2=coordinate2(2);

z2=coordinate2(3);

if (sqrt(x2^2+y2^2)>l1+l2||sqrt(x2^2+y2^2)<l1-l2)

fprintf("坐标超范围")

return

end

time=input('输入运动时间');

[theta1\_1,theta2\_1,z1]=converse(x1,y1,z1);

%%随便假定一种初始状态，本例假设theta2为正

%%求变化后的角度，设两个机械臂转动角度的损耗权值均为1.即转动同等重要，我们在规划时考虑使两个关节的转动角度之和最小

[theta1\_2,theta2\_2,z2]=converse(x2,y2,z2);

array1=theta1\_2-theta1\_1(1);

array2=theta2\_2-theta2\_1(1);

[num1,index1]=min(abs(array1));

[num2,index2]=min(abs(array2));

[s1,v1,a1]=lspb(theta1\_1(1),theta1\_2(index1),linspace(0,time,100));

[s2,v2,a2]=lspb(theta2\_1(1),theta2\_2(index2),linspace(0,time,100));

t=linspace(0,time,100);

%%动画部分

L1=Link([0,0.4605,0,0,0],'modified');

L2=Link([0,0.0855,0.350,0,0],'modified');

L3=Link([0,0,0.250,0,1],'modified');

robot=SerialLink([L1 L2 L3],'name','assignment1');

init\_ang = [theta1\_1(1),theta2\_1(1) z1];

targ\_ang = [theta1\_2(index1) theta2\_2(index2) z2];

[q,qd,qdd]=jtraj(init\_ang,targ\_ang,100);

robot.plot(q,'workspace',[-1 1 -1 1 -1 1])

%%数据打印

figure(1)

plot(t,s1,'R');

hold on

plot(t,v1,'G')

hold on

plot(t,a1,'B')

legend('theta1','theta1-dot','theta1-double-dot')

figure(2)

plot(t,s2,'R');

hold on

plot(t,v2,'G')

hold on

plot(t,a2,'B')

legend('theta2','theta2-dot','theta2-double-dot')

figure(3)

plot(t,linspace(z1,z2,100))

legend("最后的移动关节运动情况")

%%反解关节角

function [theta1,theta2,d3]=converse(x,y,z)

l1=0.35;

l2=0.25;

d1=0.4605;

d2=0.0855;

theta2=[acos((x^2+y^2-l1^2-l2^2)/2/l1/l2) -acos((x^2+y^2-l1^2-l2^2)/2/l1/l2)];

beta=atan2(y,x);

fai=acos((l1^2+x^2+y^2-l2^2)/(2\*l1\*sqrt(x^2+y^2)));

theta1=[beta-fai beta+fai];

d3=z-d1-d2;

end