《机器人学导论》大作业

——基于SCARA机器人

**摘 要**：SCARA机器人简化后是一种RRP机器人，具有两个旋转关节和一个平移关节，文章以此为研究对象对机器人的位姿表示、运动学模型、动力学模型、轨迹规划方面进行求解，首先对规划好的路径进行无外力干扰的仅控制位置的位置跟踪；此后，还设计了“位置+力”的混合控制，即阻抗控制；最后，在阻抗控制器的基础上加入外力的干扰，设计了串级控制系统以满足了机器人系统对位置的跟踪需求。

**关键词**：SCARA机器人 运动学 动力学 轨迹规划 运动控制

# SCARA基本信息

文章首先对机器人进行简化，保留第一、第二旋转，锁定第三关节，保留第四平移关节，变成了一个具有两个旋转自由度和一个平移自由度的三自由度机械臂，其简化模型及参数如下。

假设旋转关节得关节角空间的范围是.

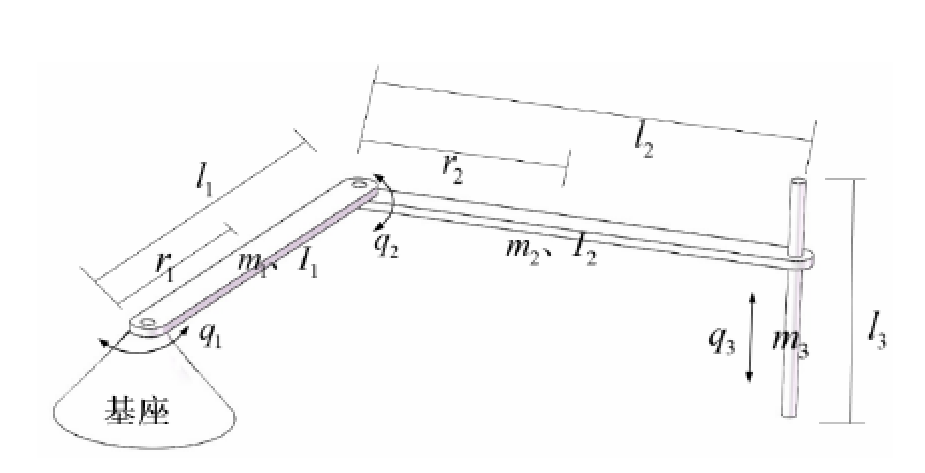


图 SCARA机器人简化模型

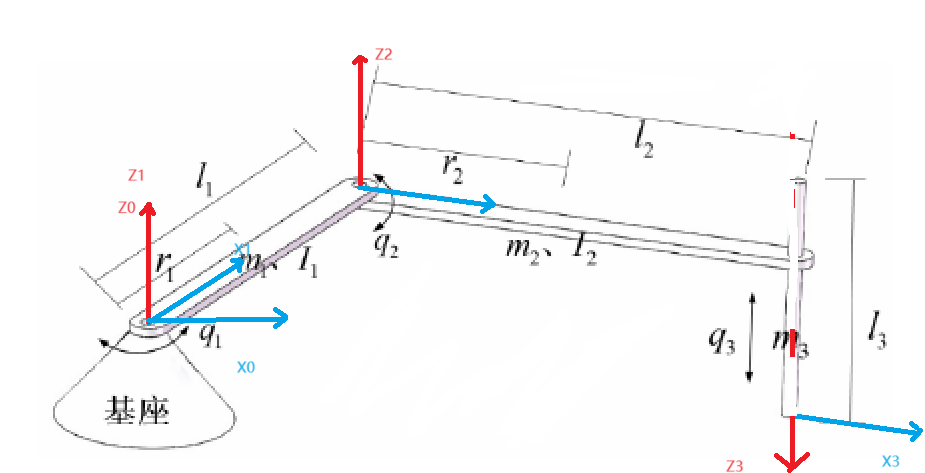
表 SCARA机器人各连杆参数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 连杆 |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| 3 |  |  | 0 |  |

# 机器人的基本模型

## DH参数表

坐标系建立说明，坐标系与基坐标系原点重合，则与重合，竖直向下，沿连杆1指向第二关节；同理，坐标系同样如此建立，建立坐标系如下图。



则有DH参数表如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| 1 |  |  |  | () |
| 2 |  |  |  | () |
| 3 |  |  | () |  |

DH参数表符号说明：

## 运动学模型

正运动学，即关节角到笛卡尔坐标系的映射关系。

根据***几何法***基本参数可以得到如下

说明：，

构建连杆变换矩阵通式如下：

由***递推法***根据（1.2.2）式可以得出、、如下

则有

则有从角关节空间到笛卡尔空间的映射关系，即运动学模型式（2.2.7）（2.2.8）

## 逆运动学模型

### 方法一

由上，逆运动学则是由笛卡尔空间向关节角空间的映射。

整理得

且反余弦函数的取值范围为，所以该关节存在多解，在文章假设前提下，有

对于第三关节，显然有

对于第一关节，

与第二关节角同理，考虑机器人角空间的范围和反余弦函数的解的范围，同样存在多解

综上，上式（1.3.4）、（1.3.5）、（1.3.6）、（1.3.8）则为此机器人的逆运动学解，在编程中需要在计算后判断每个角是否处于合法范围中在进行下一步的操作，否则会出错。

### 方法二

从正运动学中变换得到：

展开计算则有：

整理得到：

因此，从中抽离出等式：

## 雅可比矩阵

已知机器人的运动学模型（[详细见2.运动学模型](#_运动学模型)）：

则有雅可比矩阵如下：

## 动力学模型

考虑到需要用计算机仿真计算，此处选用牛顿-欧拉递推动力学算法求解，由[2.运动学模型](#_运动学模型)转移矩阵可得出转移矩阵得出如下：

说明：质量均匀分布，采用同一种材质，但是由于长度不一样，每一个连杆的质量都不相同，为了方便计算，此处我们进行简化，假设操作臂的质量分布非常简单，每个连杆的质量集中在连杆的末端，设其质量分别为，在此基础上计算操作臂的动力学模型。

首先，确定牛顿-欧拉递推公式中个参量的值，每个连杆质心的位置矢量为：

已知惯性张量如下：

并且其中的元素为:

由于假设为集中质量，因此每个连杆质心的惯性张量为**零矩阵：**

末端执行器上不挂在任何工具和作业阻力，因而有：

机器人的底座固定在大地上，不能旋转，因此有：

包括重力因素，有

考虑到需要用计算机进行仿真和计算，文章选用牛顿-欧拉法来推理动力学模型，其采用递推形式，更适合计算机的运算。

且

则有如下递推结果

### 从连杆1到连杆3向外迭代计算连杆的*速度*和*加速度*

对于连杆1则有：

对于连杆2则有：

对于连杆3则有：

### 从连杆3到连杆1向内迭代计算连杆间的相互作用力和力矩以及关节驱动力矩

对于关节3则有：

对关节2则有：

对于关节1则有：

+

对此，根据牛顿—欧拉法迭代推理出来了该机器人动力学方程。

# 轨迹规划（关节空间）

起始时刻、中间时刻、终止时刻分别为：，初始位置、中间位置、终止位置（关节空间）分别为

利用两端三次曲线进行轨迹规划，即用以下三次多项式：

为关节变量，旋转角度或者平移距离。

根据题意，经过每一个点都需要停止***（注意：由于未交代中间点的速度，所以此处默认中间点速度为0m/s，停止后再次运动起来需要重新计时，因此）*，**因此，结合式（3.4）可以求解得到两端三次多项式：

第一关节空间：

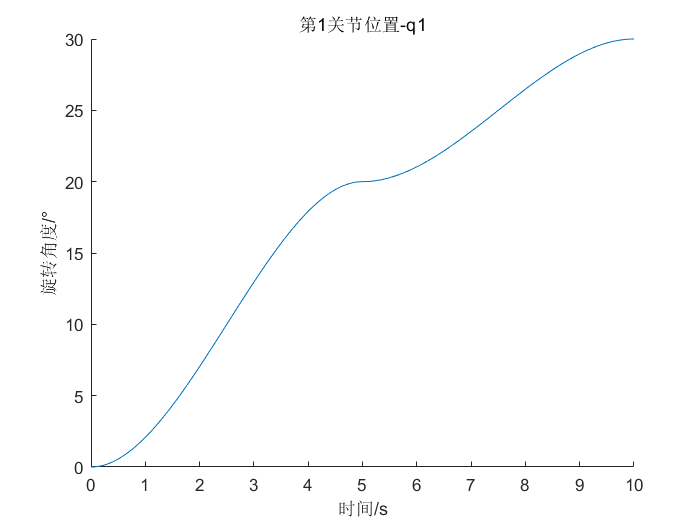
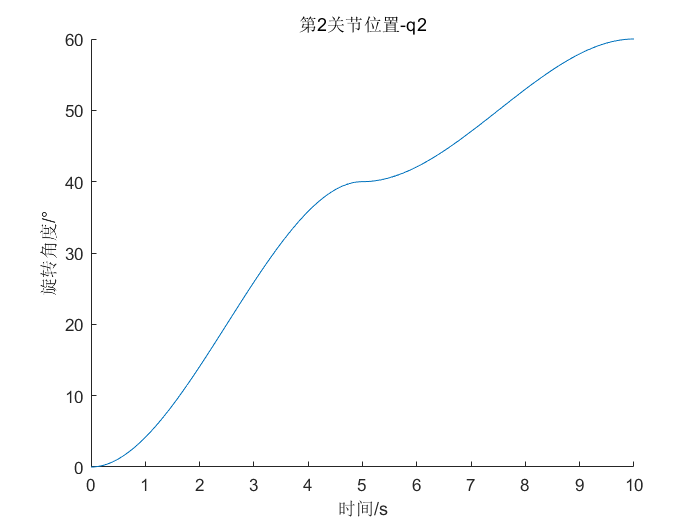
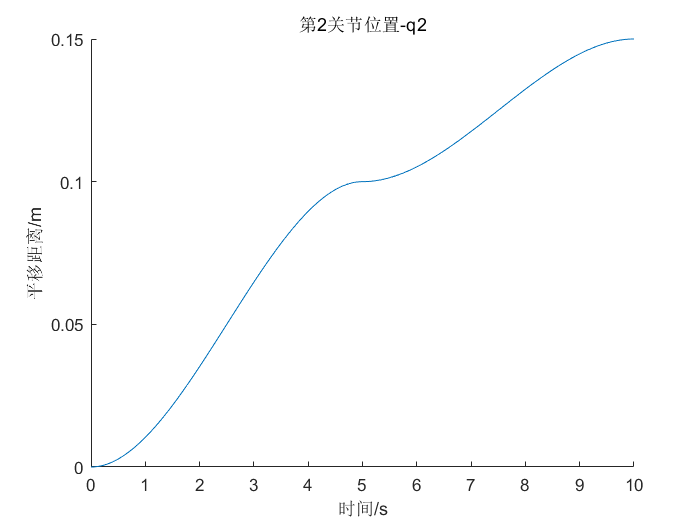
（第二段重新计时）

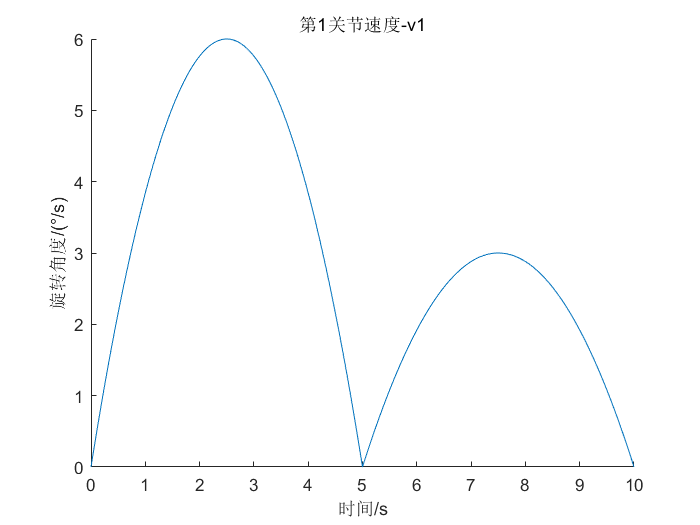
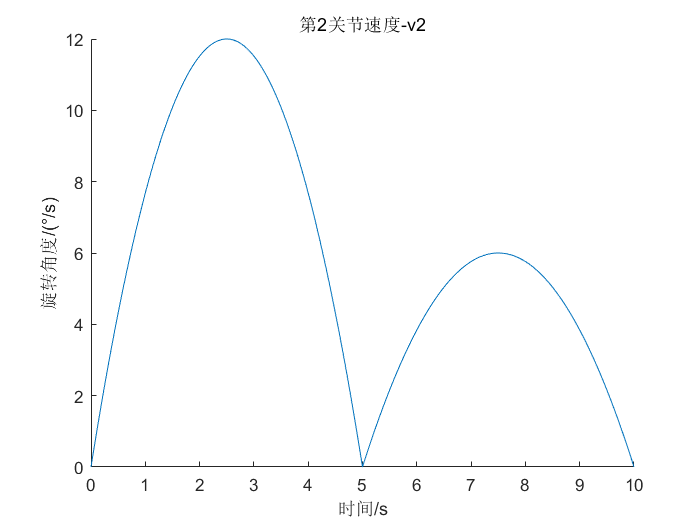
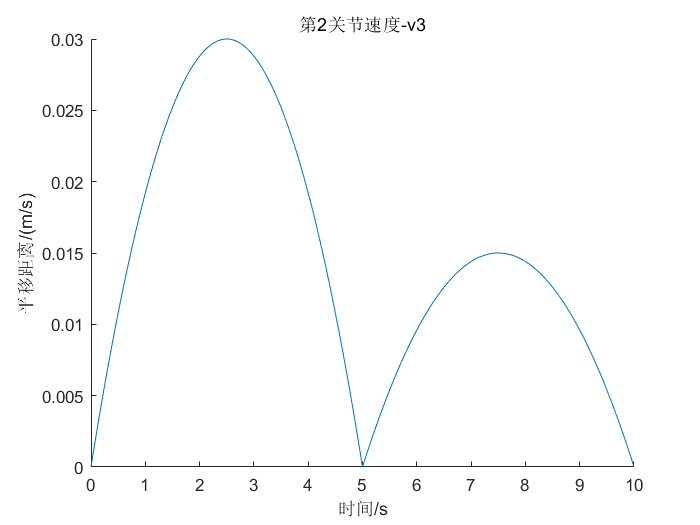
（第二段不重新计时，为了不产生误解，后续均采用此种形式表达这段10s的运动）

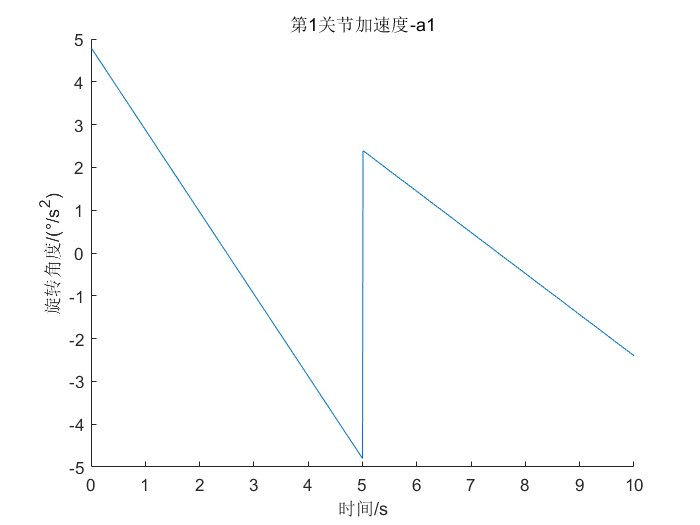
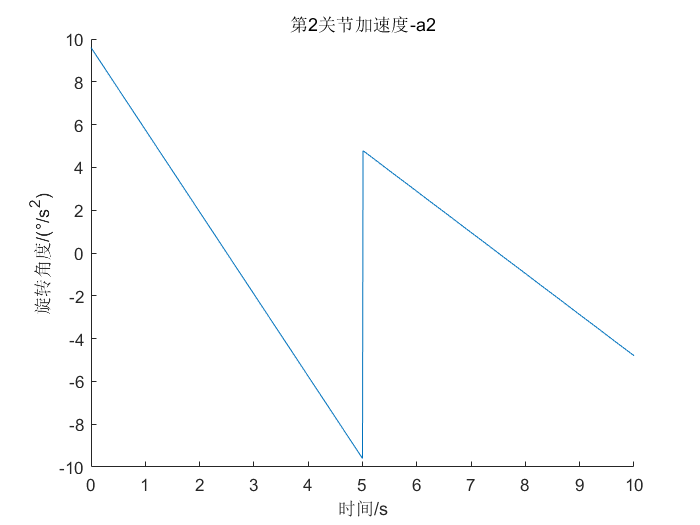
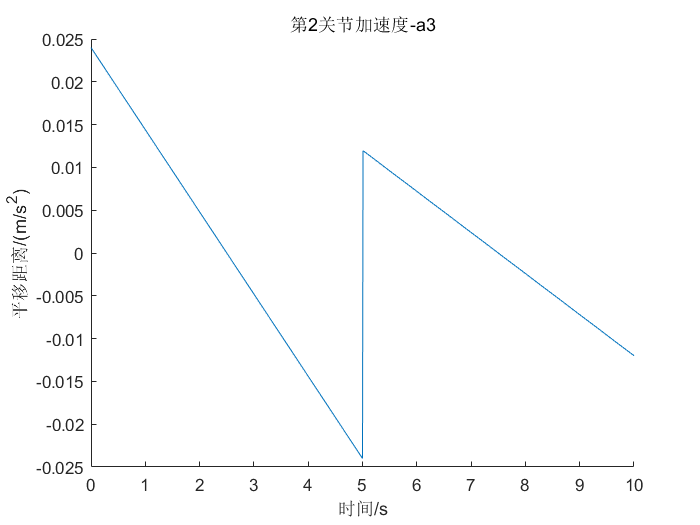
第二关节空间：

第三关节空间：

利用Matlab绘制关节位置（）、速度（）即加速度（）曲线如下（注：速度和加速度对位置函数进行一次求导和二次求导即可得到）：、

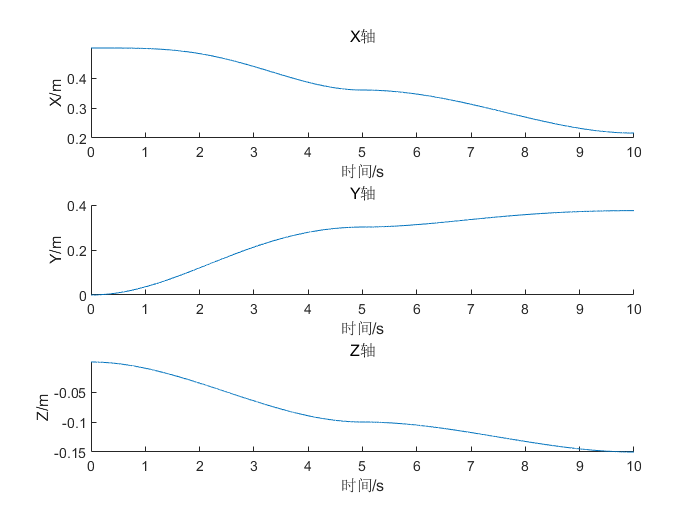
位置曲线

速度曲线

加速度曲线

# 机械臂末端轨迹（笛卡尔空间）

查看Matlab附件[Solution.m（见附件）](Solution.m)，运行函数Get\_Pose\_Trace\_of\_the\_end\_of\_Robot\_R()和Get\_Pose\_Trace\_of\_the\_end\_of\_Robot\_T() 即可以观察到机械臂末端的运动状态

机械臂末端的轨迹如下图所示：

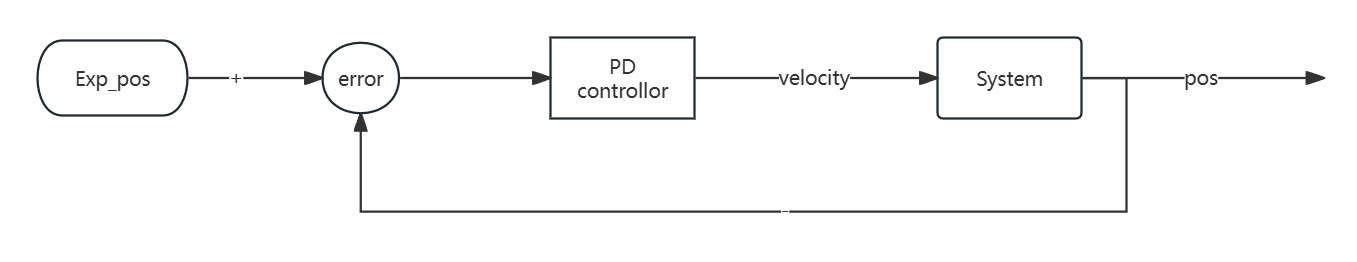
根据上文得到的轨迹，文章将会给出机械臂末端的速度曲线如下：

对上式进行求导即可得到笛卡尔空间下的速度关于时间的函数：

进行绘图则有（具体实现见 Matlab Solution.m（见附件） 文件中的Plot\_End\_velocity\_InXYZ() 函数）

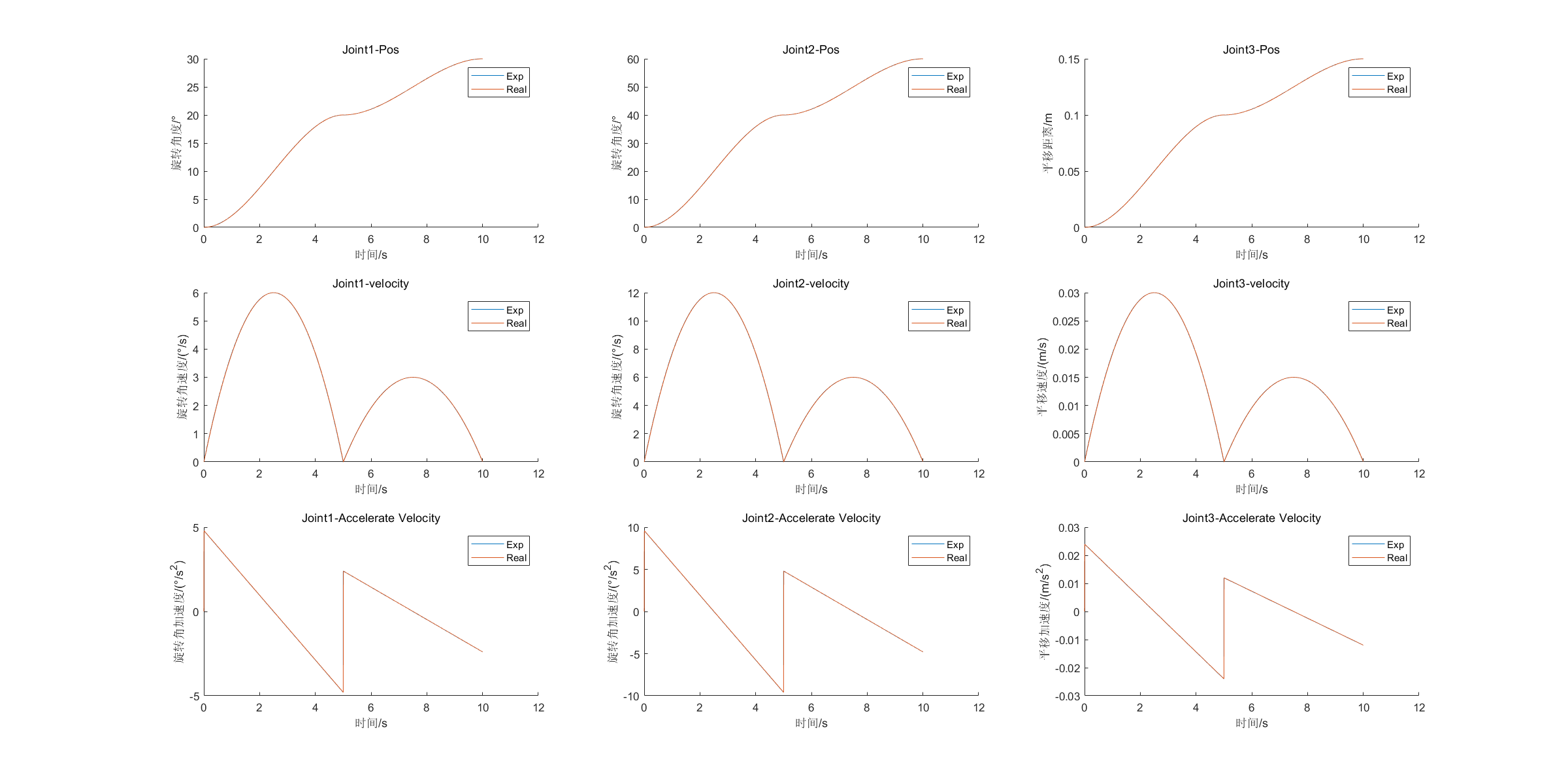
# PD控制器（关节空间轨迹跟踪）

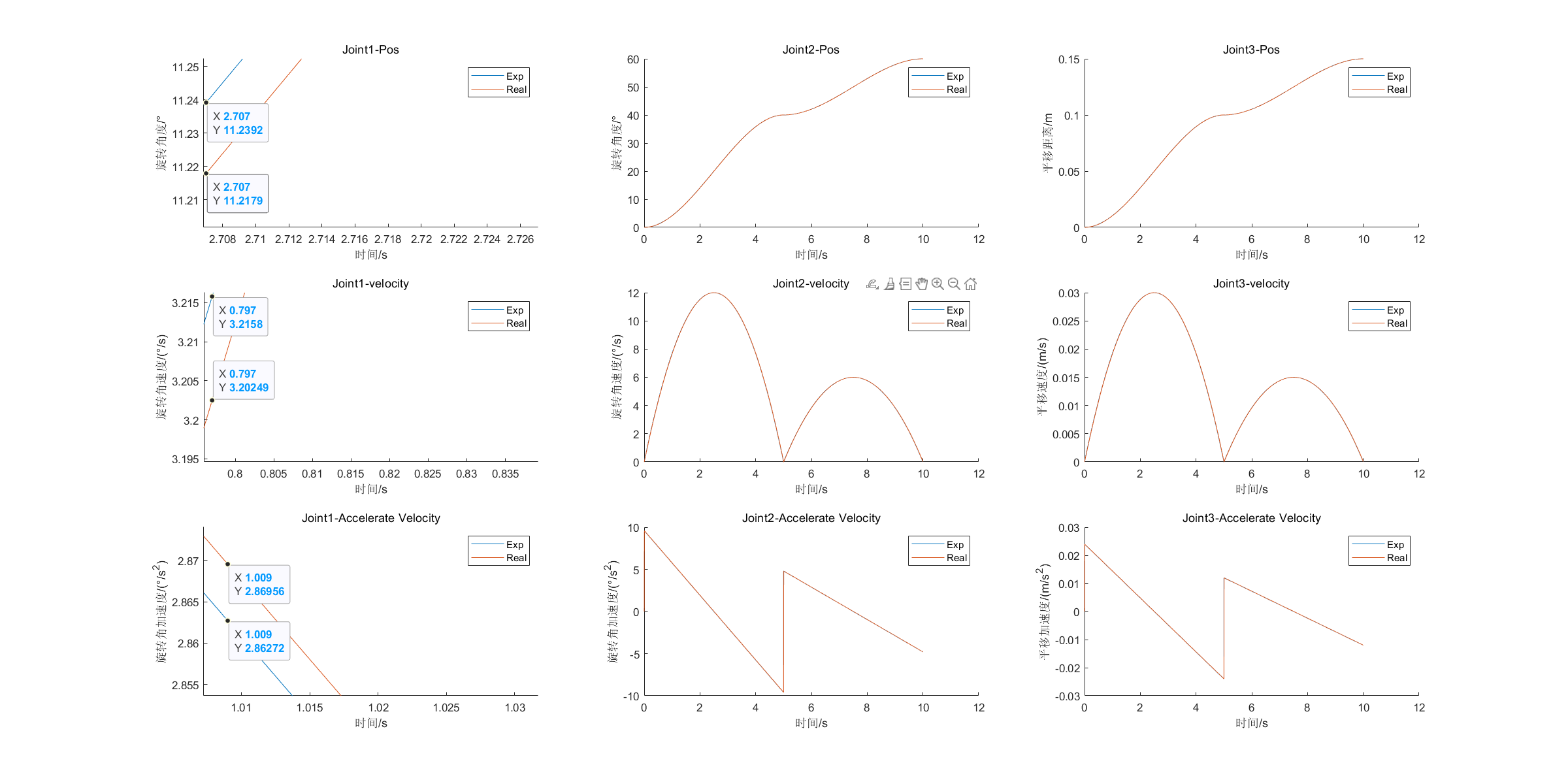
假设系统输出量为关节空间下的位置，输入量为关节空间下各个关节电机的旋转速度，因此可以得到以下系统：



并且离散化后对PD控制器进行差分化得到：

利用Matlab进行编写，运行结果如下：

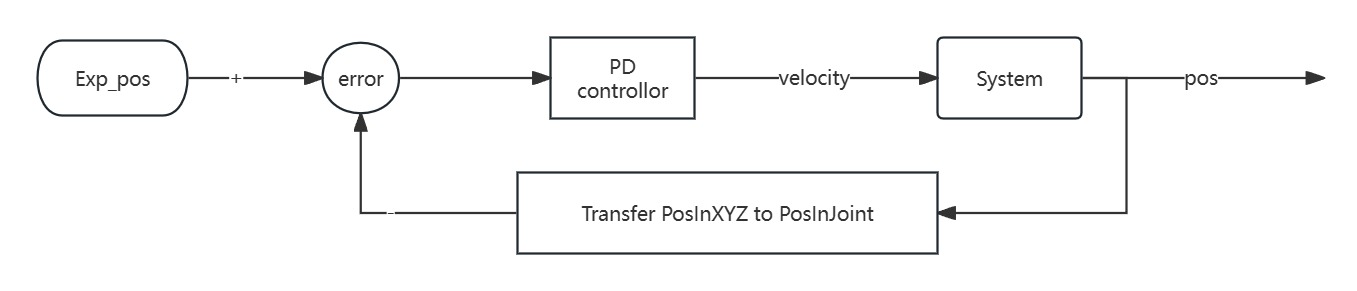
*注：见Solution.m(见附件)文件中的函PD\_Controllor\_For\_Trace\_Forward()*

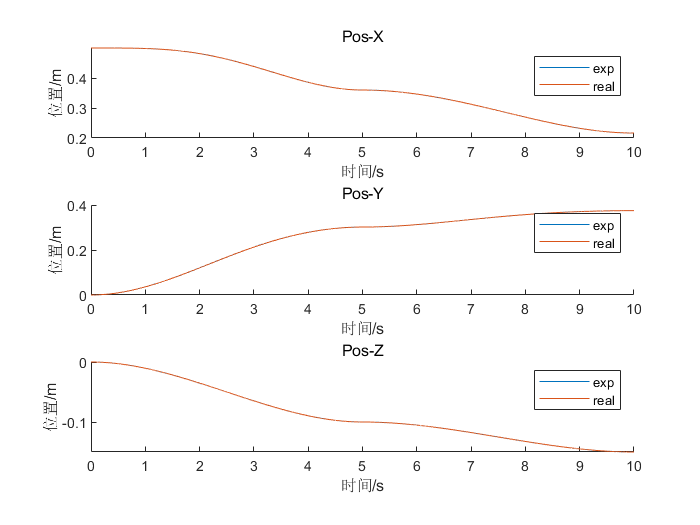
对局部放大后（见下图）可见其跟踪误差控制在0.02之内，跟踪效果很好。

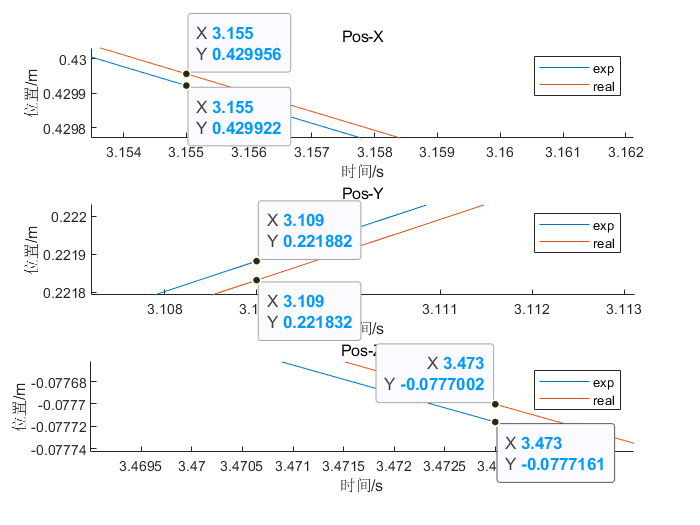
# 逆动学PD控制器（笛卡尔空间轨迹跟踪）

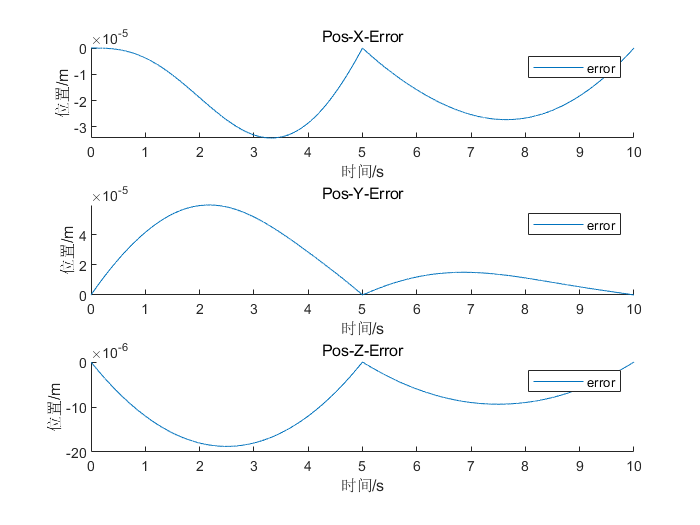
与PD控制器大同小异，输出量变成了笛卡尔空间下的位置坐标，在跟踪的时候需要先对其求逆解，求解出其在关节空间下的对应值，只不过此时存在多解问题，需要在多个解中筛选出有效值。

对应五.增加坐标系变换求逆解则有控制系统：

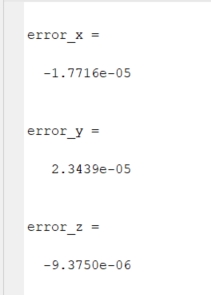


机械臂末端在笛卡尔坐标系下跟踪的效果如下图所示：

对局部放大后可以得到

可以观察到其误差很小，对跟踪误差进行绘制则有：

求解平均计算得到：



由此可以看出，此PD控制器跟踪的效果很好，得到为位置误差很小，X轴可以控制在mm级别，Y轴可以控制在cm级别，Z轴可有控制在nm级别。

# 阻抗控制器（位置&力混合控制）

## 路径规划

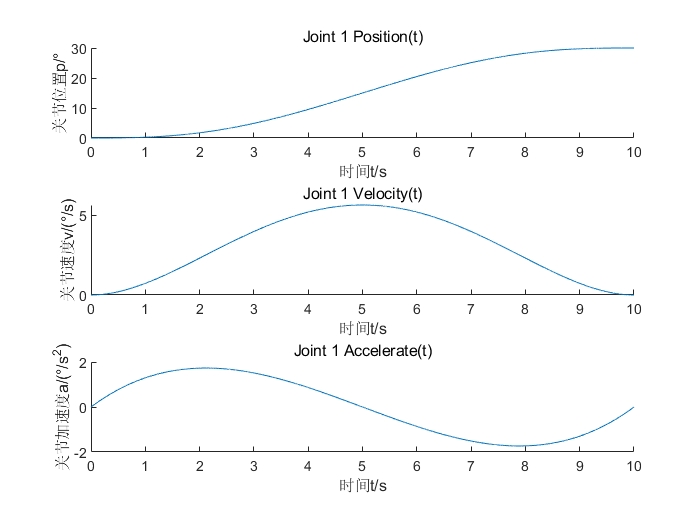
机器人的初始位置为;

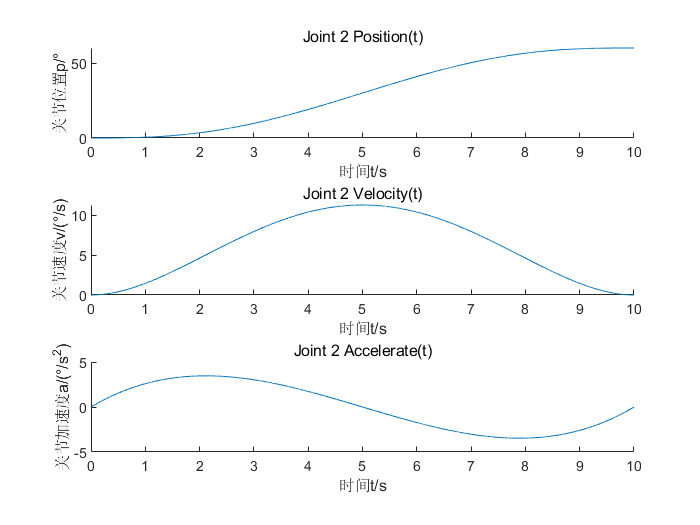
期望位置为;

为了满足初始时刻和终止时刻的速度、加速都为零，文章采用五次多项式来进行路径的规划，公式如下：

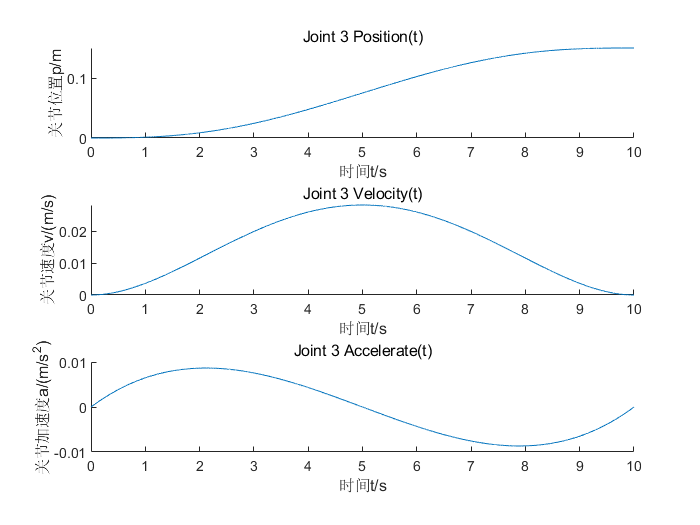
根据要求的信息，我们可以得出路径规划，其关节位置关于时间的函数如下：

每个关节的规划轨迹如下图所示：

第一关节：

第二关节：

第三关节：

综上，即为满足假设的路径规划，由此我们求导后还可以得到关节速度、加速度关于时间的函数，此处不再赘述。

## 阻抗控制器

间接力控制：通过运动控制实现对力的控制，不需要力反馈闭环。

不同于五、六点所提到的跟踪需求，五、六点的跟踪是在末端没有里的需求的情况下工作的，现在机器人系统中加入了力的控制，需要考虑机械手末端的受力情况。

并且，题设给出了机械臂末端的刚性和柔性需求如下：

期望刚性：

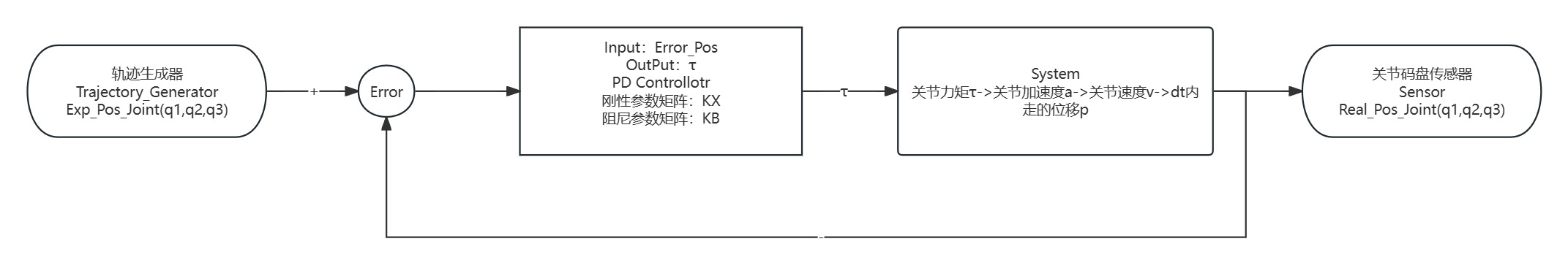
阻尼参数：

根据上述信息，文章设计出满足此需求的阻抗控制器。

文章假设力的传感信息来自机械臂末端六维传感器，且传感器的输出信息不存在噪声，不需要做滤波处理。

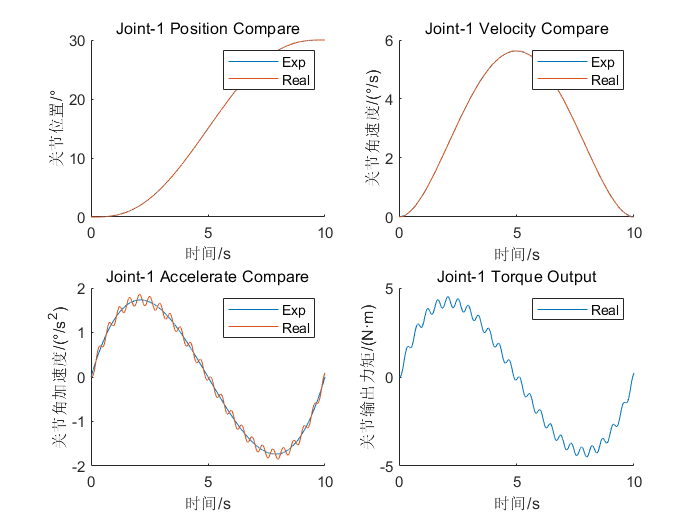
位置误差为：

根据阻抗控制器公式则有：

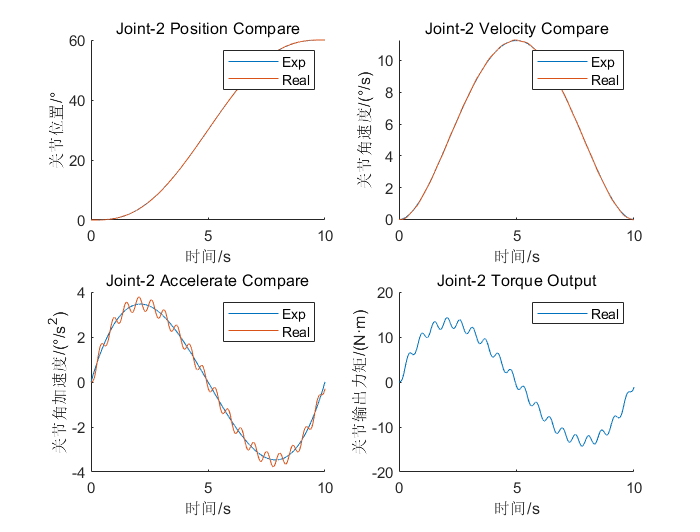
控制系统式设计见图如下图：

根据控制系统设计简图和控制器公式，运行Matlab Solution.m 文件中的函数 Impedance\_Control()，则有运行的结果如下：

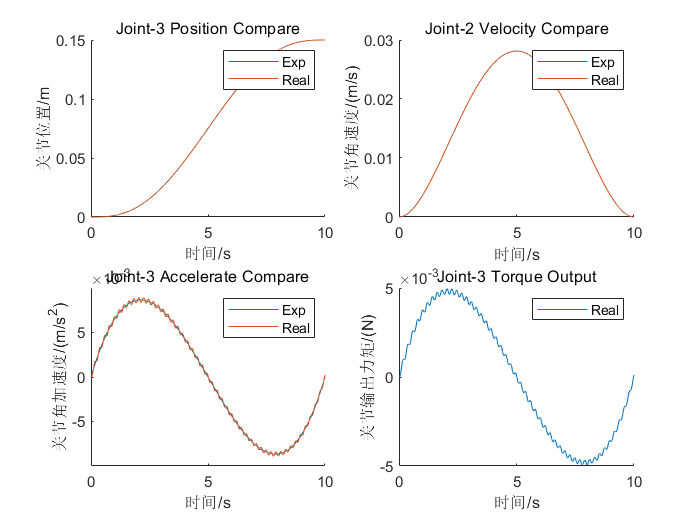
第一关节：

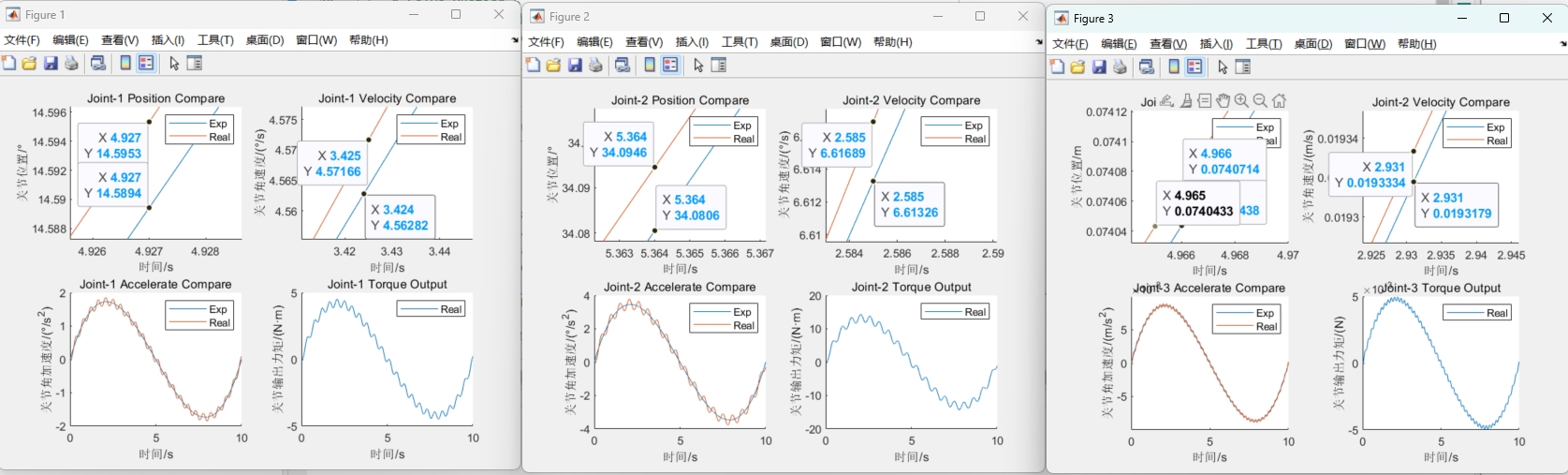


第二关节：



第三关节：

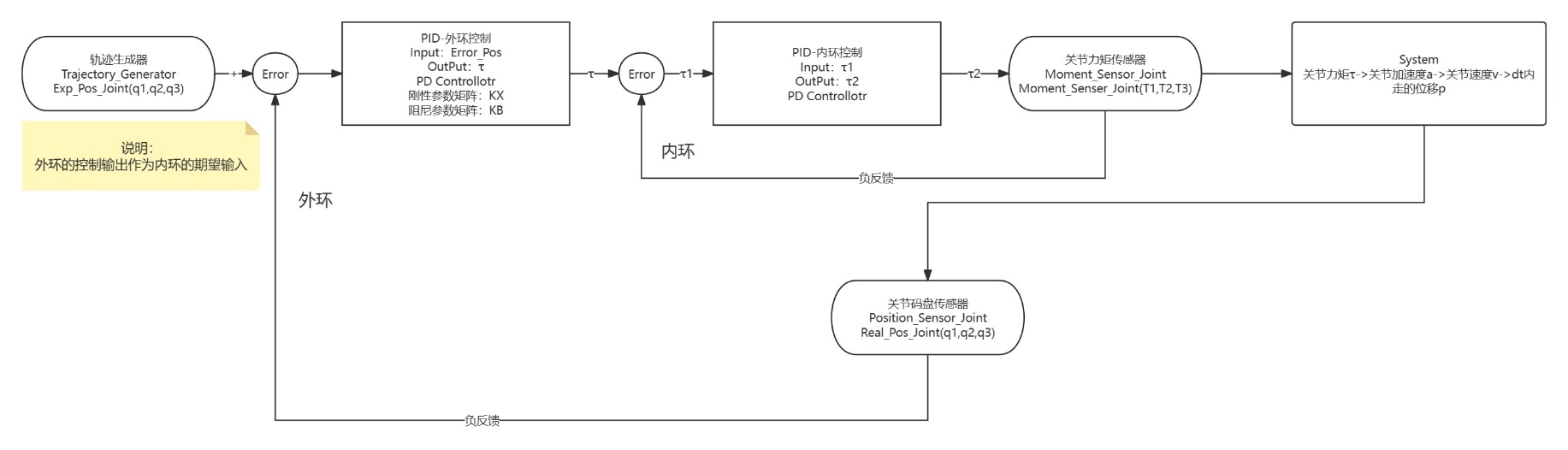


对其中的部分数据放大后可观察到：

可见在阻抗控制器下的跟踪误差很小，基本上实现了无差跟踪。

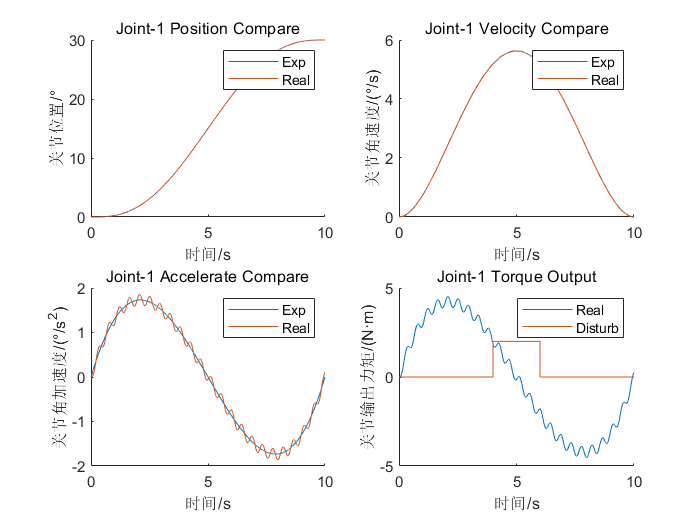
# 带有力矩传感器的阻抗控制器

前提假设：在[七.](#_阻抗控制器（位置&力混合控制）)的基础上，在4~6s的时候，给机机械臂施加外力如下，不考虑重力补偿，每一个关节的质量均为10kg。

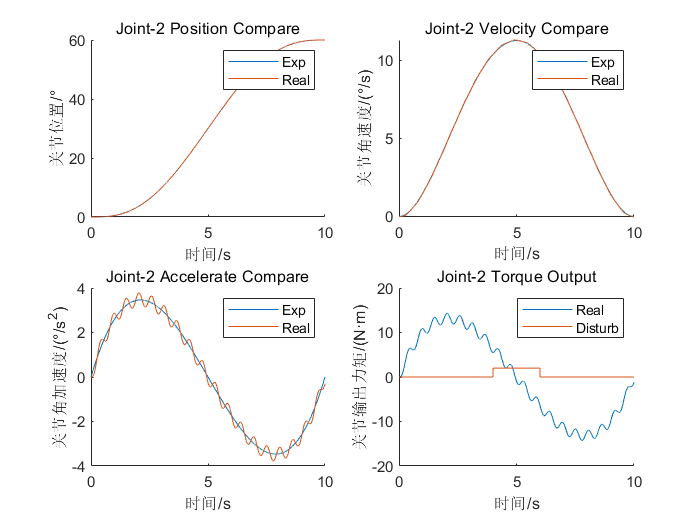
为了抵抗这一外力的干扰，并且维护系统的稳定性，文章在七.的控制系统的基础上进行改善，增加了内环控制，就是为了更好的控制点击力矩的输出，新的控制系统简图如下：

对上图所表示的串级控制系统进行编写，运行名为Solution.m的Matlab文件中的函数Impedance\_Control\_With\_Disturb()，可以得到每一个关节的运动情况，包括关节位置，关节速度，关节加速度以及关节的力矩输出情况（以及外力施加情况）。

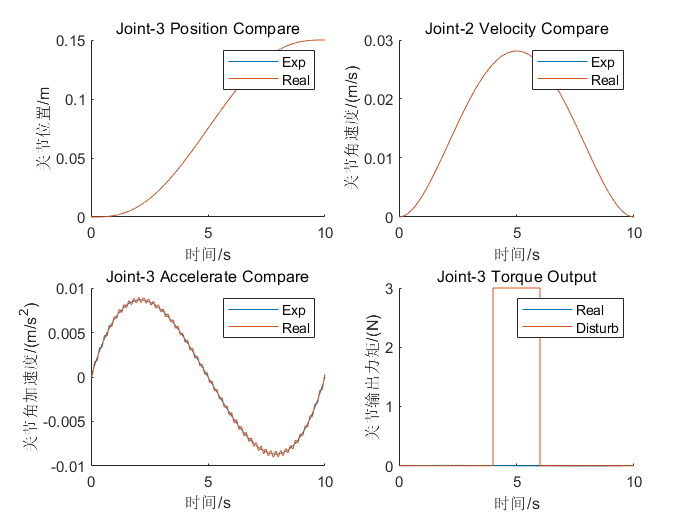
第一关节：



第二关节：

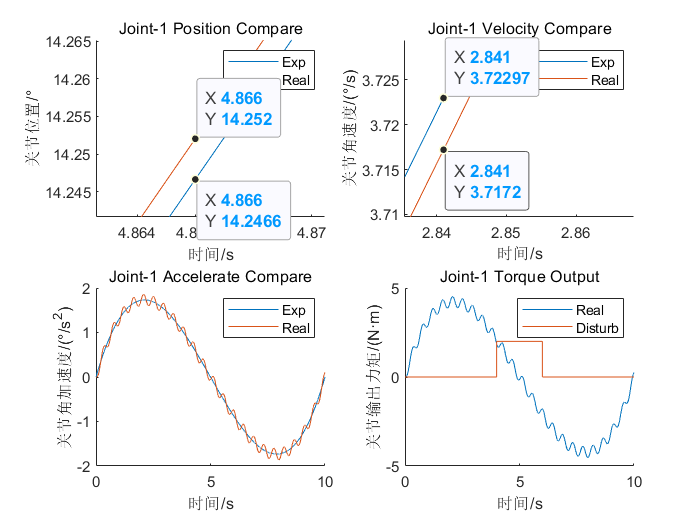


第三关节：

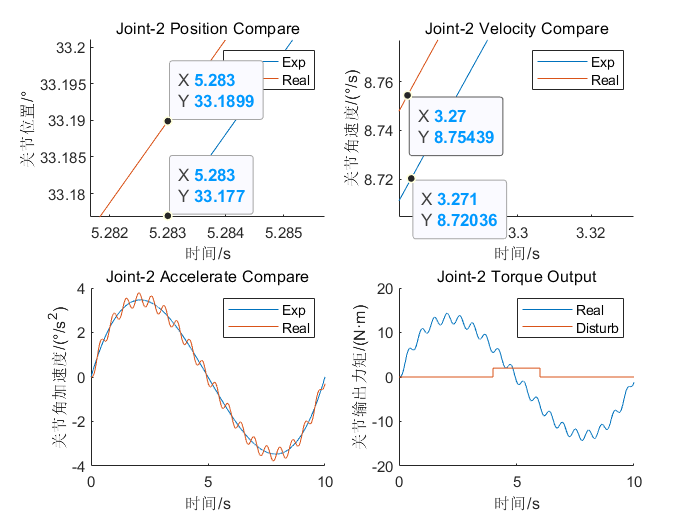


为了更好的的观察跟踪的效以及系统的稳定性，文章对上述三幅图进行局部的发放大，得到如下三幅图片：

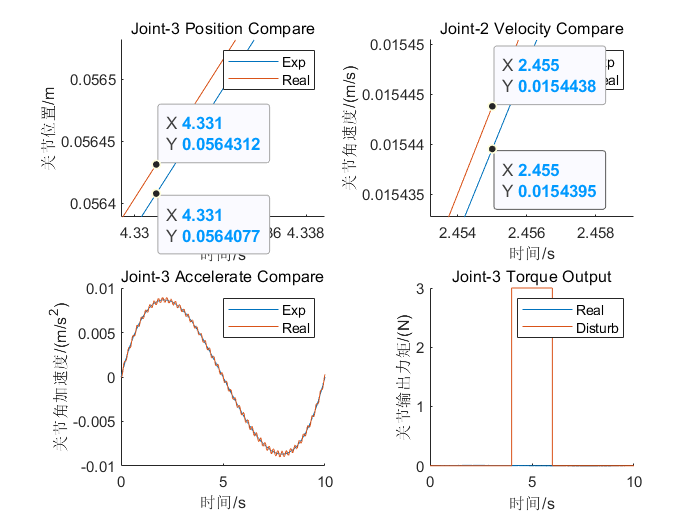
第一关节（局部放大）：



第二关节（局部放大）：



第三关节（局部放大）：



综上，我们可以观察到，双闭环即串级控制系统对于有外力施加的情况下，文档也能够很好的稳定住，并控制住误差，没有使系统出现发散的去情况。

# 附录

%% Solution.m

% 运行对应的代码即可得到对应的结果

% 文件说明(1-6行)：

% 文件一共分为两个部分，分别使第一部分、第二部分

% 第一部分(7-45行)：函数的调用，对对应的题目进行求解

% 第二部分(46-1323行)：函数的定义，对第一部分需要使用到的所有进行定义

%% -------------Part 1.函数的调用，对对应的题目进行求解----------------------

% 清除工作区和命令行窗口

clc;

clear;

% Solve Qustion 2

% 计算并且绘制出关节的位置、速度、加速度曲线（关节空间下）

% Plot\_Posotion\_Velocity\_AcceleratedVelocity\_Curve()

% Solve Question 3

% 绘制机械臂末端的位置和姿态

% Get\_Pose\_Trace\_of\_the\_end\_of\_Robot\_R()

% Solve Question 4

% 绘制机械臂末端位置轨迹（笛卡尔空间下）

% Plot\_End\_Position\_InXYZ()

% 绘制机械臂末端速度轨迹（笛卡尔空间下）

% Plot\_End\_velocity\_InXYZ()

% Solve Question 5

% PD控制器进行轨迹跟踪（正运动学）

% PD\_Controllor\_For\_Trace\_Forward()

% Solve Qustion 6

% PD控制器进行轨迹跟踪（逆运动学）

% PD\_Controllor\_For\_Trace\_Reverse()

% Solve Question 7

% 求解函数参数

% Solve\_For\_Function2()

% Trajectory\_Planning\_2()

% Plot\_For\_Trajectory\_Planning\_2

% 阻抗控制

% Impedance\_Control();

% Solve Question 8

% Get Disturb Data

% Get\_Disturb\_Data()

Impedance\_Control\_With\_Disturb()

%% --------------Part 2.函数的定义-各种功能函数的定义------------------------

%--------------------------Function Definition-----------------------------

% function 1：计算位置坐标

function Data\_Position=Calculate\_Robot\_Position()

time=0:0.001:10;

Data\_Position=zeros(10001,4);

index = 1;

for t=time

% Position(t)

[q1,q2,q3]=Position\_Calculate\_Function(t);

Data\_Position(index,:)=[t,q1,q2,q3];

index = index + 1;

end

end

% function 2: 绘制关节空间下的位置、速度、加速度关于时间t的函数图像

function []=Plot\_Posotion\_Velocity\_AcceleratedVelocity\_Curve()

Joint\_Data = Get\_Joint\_Data()

% ---------------subplot绘图-----------------

% ----------绘制位置--------

figure(1)

hold on

title("第1关节位置-q1")

ylabel("旋转角度/°")

xlabel("时间/s")

plot(Joint\_Data(:,1),Joint\_Data(:,2))

hold off

figure(2)

hold on

title("第2关节位置-q2")

ylabel("旋转角度/°")

xlabel("时间/s")

plot(Joint\_Data(:,1),Joint\_Data(:,3))

hold off

figure(3)

hold on

title("第2关节位置-q2")

ylabel("平移距离/m")

xlabel("时间/s")

plot(Joint\_Data(:,1),Joint\_Data(:,4))

hold off

% ----------------绘制速度-----------------------

figure(4)

hold on

title("第1关节速度-v1")

ylabel("旋转角度/(°/s)")

xlabel("时间/s")

plot(Joint\_Data(:,1),Joint\_Data(:,5))

hold off

figure(5)

hold on

title("第2关节速度-v2")

ylabel("旋转角度/(°/s)")

xlabel("时间/s")

plot(Joint\_Data(:,1),Joint\_Data(:,6))

hold off

figure(6)

hold on

title("第2关节速度-v3")

ylabel("平移距离/(m/s)")

xlabel("时间/s")

plot(Joint\_Data(:,1),Joint\_Data(:,7))

hold off

% ---------------绘制加速度----------------------

figure(7)

hold on

title("第1关节加速度-a1")

ylabel("旋转角度/(°/s^2)")

xlabel("时间/s")

plot(Joint\_Data(:,1),Joint\_Data(:,8))

hold off

figure(8)

hold on

title("第2关节加速度-a2")

ylabel("旋转角度/(°/s^2)")

xlabel("时间/s")

plot(Joint\_Data(:,1),Joint\_Data(:,9))

hold off

figure(9)

hold on

title("第2关节加速度-a3")

ylabel("平移距离/(m/s^2)")

xlabel("时间/s")

plot(Joint\_Data(:,1),Joint\_Data(:,10))

hold off

end

% function 3: 计算位置的函数

function [q1,q2,q3]=Position\_Calculate\_Function(t)

if t<=5

q1=(12/5)\*t^2-(8/25)\*t^3;

q2=(24/5)\*t^2-(16/25)\*t^3;

q3=(3/250)\*t^2-(1/625)\*t^3;

else

q1=(6/5)\*(t-5)^2-(4/25)\*(t-5)^3+20;

q2=(12/5)\*(t-5)^2-(8/25)\*(t-5)^3+40;

q3=(3/500)\*(t-5)^2-(1/1250)\*(t-5)^3+0.1;

end

end

% function 4: 计算速度的函数

function [v1,v2,v3]=Velocity\_Calculate\_Function(t)

if t<=5

v1=(24/5)\*t-(24/25)\*t^2;

v2=(48/5)\*t-(48/25)\*t^2;

v3=(3/125)\*t-(3/625)\*t^2;

else

v1=(12/5)\*(t-5)-(12/25)\*(t-5)^2;

v2=(24/5)\*(t-5)-(24/25)\*(t-5)^2;

v3=(3/250)\*(t-5)-(3/1250)\*(t-5)^2;

end

end

% function 5: 计算加速度

function [a1,a2,a3]=Accelerated\_Velocity\_Calculate\_Function(t)

if t<=5

a1=(24/5)-(48/25)\*t;

a2=(48/5)-(96/25)\*t;

a3=(3/125)-(6/625)\*t;

else

a1=(12/5)-(24/25)\*(t-5);

a2=(24/5)-(48/25)\*(t-5);

a3=(3/250)-(3/625)\*(t-5);

end

end

% function 6: 绘制机械臂末端的位姿T

function []=Get\_Pose\_Trace\_of\_the\_end\_of\_Robot\_T()

% robot base data

Joint\_Position = Calculate\_Robot\_Position();

% 旋转关节-角度制

EndPose = zeros(4,4,10001);

index=1;

for i=1:10:10001

theta1=Joint\_Position(i,2);

theta2=Joint\_Position(i,3);

d3=Joint\_Position(i,4);

T = Get\_T\_Matrix(theta1,theta2,d3);

EndPose(:,:,index)=T;

index=index+1;

end

view(3)

tranimate(EndPose,'axis',[0,250,0,200,-100,0])

end

% function 7:计算矩阵T

function T03=Get\_T\_Matrix(theta1,theta2,d3)

% 旋转关节-转为弧度制

theta1 = pi\*(theta1/180);

theta2 = pi\*(theta2/180);

% 平移关节

d3=1\*d3;

% 连杆长度

l1=0.250;

l2=0.250;

T03 = [cos(theta1+theta2) sin(theta1+theta2) 0 cos(theta1+theta2)\*l2+cos(theta1)\*l1;

sin(theta1+theta2) -cos(theta1-theta2) 0 sin(theta1+theta2)\*l2+sin(theta1)\*l1;

0 0 -1 -1\*d3;

0 0 0 1];

end

% function 8：计算矩阵R

function R03=Get\_R\_Matrix(theta1,theta2,d3)% 旋转关节-转为弧度制

theta1 = pi\*(theta1/180);

theta2 = pi\*(theta2/180);

% 平移关节

d3=1\*d3;

% 连杆长度

l1=0.250;

l2=0.250;

R03 = [cos(theta1+theta2) sin(theta1+theta2) 0 ;

sin(theta1+theta2) -cos(theta1-theta2) 0 ;

0 0 -1 ];

end

% function 9:绘制机械臂末端的位姿R

function []=Get\_Pose\_Trace\_of\_the\_end\_of\_Robot\_R()

% robot base data

Joint\_Position = Calculate\_Robot\_Position();

% 旋转关节-角度制

EndPose = zeros(3,3,100);

index=1;

for i=1:10:1000

theta1=Joint\_Position(i,2);

theta2=Joint\_Position(i,3);

d3=Joint\_Position(i,4);

R = Get\_R\_Matrix(theta1,theta2,d3);

EndPose(:,:,index)=R;

index=index+1;

end

view(3)

tranimate(EndPose)

end

% function 10:绘制机械臂末端的轨迹曲线（基于笛卡尔坐标系）

function []=Plot\_End\_Position\_InXYZ()

clc;

clear;

Joint\_Position = Calculate\_Robot\_Position();

% 连杆长度

l1=0.250;

l2=0.250;

PosXYZ=zeros(10001,4);

dt=0.001;

for i=1:1:10001

% 获取数据

theta1=Joint\_Position(i,2);

theta2=Joint\_Position(i,3);

d3=Joint\_Position(i,4);

% 转为弧度制

theta1 = pi\*(theta1/180);

theta2 = pi\*(theta2/180);

x=cos(theta1+theta2)\*l2+cos(theta1)\*l1;

y=sin(theta1+theta2)\*l2+sin(theta1)\*l1;

z=-d3;

PosXYZ(i,:)=[dt\*(i-1),x,y,z];

end

subplot(3,1,1)

hold on

xlabel('时间/s')

ylabel('X/m')

title('X轴')

plot(PosXYZ(:,1),PosXYZ(:,2));

hold off

subplot(3,1,2)

hold on

xlabel('时间/s')

ylabel('Y/m')

title('Y轴')

plot(PosXYZ(:,1),PosXYZ(:,3));

hold off

subplot(3,1,3)

hold on

xlabel('时间/s')

ylabel('Z/m')

title('Z轴')

plot(PosXYZ(:,1),PosXYZ(:,4));

hold off

end

% function 11:绘制机械臂末端的速度曲线（基于笛卡尔坐标系）

function []=Plot\_End\_velocity\_InXYZ()

clc;

clear;

Joint\_Position = Get\_Joint\_Data();

% 连杆长度

l1=0.250;

l2=0.250;

VeloXYZ=zeros(10001,4);

for i=2:1:10001

% 获取数据

theta1=Joint\_Position(i,2);theta1\_fore=Joint\_Position(i-1,2);

theta2=Joint\_Position(i,3);theta2\_fore=Joint\_Position(i-1,3);

d3=Joint\_Position(i,4);d3\_fore=Joint\_Position(i-1,4);

% 转为弧度制

theta1 = pi\*(theta1/180);theta1\_fore=pi\*(theta1\_fore/180);

theta2 = pi\*(theta2/180);theta2\_fore=pi\*(theta2\_fore/180);

% delta

delta\_theta1 = theta1-theta1\_fore;

delta\_theta2 = theta2-theta2\_fore;

delta\_d3 = d3-d3\_fore;

x=(-sin(theta1+theta2)\*l2-sin(theta1)\*l1)\*delta\_theta1-(sin(theta1+theta2)\*l2)\*delta\_theta2;

y=(cos(theta1+theta2)\*l2+cos(theta1)\*l1)\*delta\_theta1+(cos(theta1+theta2)\*l2)\*delta\_theta2;

z=-1\*delta\_d3;

VeloXYZ(i,:)=[i/1000,x,y,z];

end

subplot(1,3,1)

hold on

xlabel('时间/s')

ylabel('X/(m/s)')

title('X轴')

plot(VeloXYZ(:,1),VeloXYZ(:,2));

hold off

subplot(1,3,2)

hold on

xlabel('时间/s')

ylabel('Y/(m/s)')

title('Y轴')

plot(VeloXYZ(:,1),VeloXYZ(:,3));

hold off

subplot(1,3,3)

hold on

xlabel('时间/s')

ylabel('Z/(m/s)')

title('Z轴')

plot(VeloXYZ(:,1),VeloXYZ(:,4));

hold off

end

% function 12:得到关节空间的的信息

function Joint\_Data = Get\_Joint\_Data()

Joint\_Data=zeros(10001,10);

% 计算位置

dt=0.001;

for i=1:10001

t=dt\*(i-1);

% Position(t)

[q1,q2,q3]=Position\_Calculate\_Function(t);

% Velocity(t)

[v1,v2,v3]=Velocity\_Calculate\_Function(t);

% Accelerated\_Velocity(t)

[a1,a2,a3]=Accelerated\_Velocity\_Calculate\_Function(t);

Joint\_Data(i,:)=[t,q1,q2,q3,v1,v2,v3,a1,a2,a3];

end

end

% function 13：获得笛卡尔坐标系下末端位置坐标的数据

function PosXYZ=Get\_End\_Pos\_InXYZ()

%Joint\_Position = Calculate\_Robot\_Position();

Joint\_Position = Get\_Joint\_Data();

% 连杆长度（cm）

l1=0.250;

l2=0.250;

PosXYZ=zeros(10001,4);

dt = 0.001;

for i=1:1:10001

% 获取数据

theta1=Joint\_Position(i,2);

theta2=Joint\_Position(i,3);

d3=Joint\_Position(i,4);

% 转为弧度制

theta1 = pi\*(theta1/180);

theta2 = pi\*(theta2/180);

% 计算末端位置

x=cos(theta1+theta2)\*l2+cos(theta1)\*l1;

y=sin(theta1+theta2)\*l2+sin(theta1)\*l1;

z=-d3;

% 存储末端位置

PosXYZ(i,:)=[dt\*(i-1),x,y,z];

end

end

% function 14：PD控制器进行轨迹跟踪

function []=PD\_Controllor\_For\_Trace\_Forward()

% 获得期望下关节空间的信息

Exp\_Joint\_Data = Get\_Joint\_Data();

% 获取期望的轨迹（关节空间下）

Exp\_Pos\_Trace\_Joint = [Exp\_Joint\_Data(:,1) ,Exp\_Joint\_Data(:,2:4)];

% 获得期望的速度（关节空间下）

Exp\_Velocity\_Joint = [Exp\_Joint\_Data(:,1) ,Exp\_Joint\_Data(:,5:7)];

% 获得期望的加速度

Exp\_Accelerate\_Velocity\_Joint = [Exp\_Joint\_Data(:,1) ,Exp\_Joint\_Data(:,8:10)];

% PD-Controllor-parameter(kp,kd,dt)

kp = 200;

kd = 10;

delta\_t = 0.001;

% input\_v

current\_input\_v = zeros(1,3);

next\_input\_v = zeros(1,3);

% Pos\_error

delta\_pos\_i = zeros(1,3);

delta\_pos\_i\_1 = zeros(1,3);

delta\_pos\_i\_2 = zeros(1,3);

% real Position

Real\_Pos\_Joint = zeros(10001,4);

Real\_Velocity\_Joint = zeros(10001,4);

Real\_Accelerate\_Velocity\_Joint = zeros(10001,4);

% 列向量

% trace by PD

for i=1:10001

% Input-Joint-v

% Output-Joint-Pos

% 记录速度

Real\_Velocity\_Joint(i,:)=[0.001\*i,current\_input\_v];

% calculate the real position

if i==1

Real\_Pos\_Joint(i,1)=0;

Real\_Pos\_Joint(i,2:4)=current\_input\_v\*delta\_t;

else

Real\_Pos\_Joint(i,1)=0.001\*i;

Real\_Pos\_Joint(i,2:4)=Real\_Pos\_Joint(i-1,2:4)+current\_input\_v\*delta\_t;

end

% Pos\_delta(1,3)

delta\_pos= Exp\_Pos\_Trace\_Joint(i,2:4)-Real\_Pos\_Joint(i,2:4);

if i==1

delta\_pos\_i =delta\_pos;

elseif i==2

delta\_pos\_i\_1 = delta\_pos\_i;

delta\_pos\_i =delta\_pos;

else

delta\_pos\_i\_2 = delta\_pos\_i\_1;

delta\_pos\_i\_1 = delta\_pos\_i;

delta\_pos\_i =delta\_pos;

end

delta\_v = kp\*(delta\_pos\_i-delta\_pos\_i\_2)+kd\*(delta\_pos\_i\_1-2\*delta\_pos\_i\_1+delta\_pos\_i\_2);

% 计算出下一时刻的速度

next\_input\_v = current\_input\_v + delta\_v;

% 加速度

Real\_Accelerate\_Velocity\_Joint(i,:)=[0.001\*i,delta\_v/delta\_t];

% 更新速度

current\_input\_v = next\_input\_v;

end

Plot\_for\_PD(Exp\_Pos\_Trace\_Joint,Real\_Pos\_Joint,Exp\_Velocity\_Joint,Real\_Velocity\_Joint,Exp\_Accelerate\_Velocity\_Joint,Real\_Accelerate\_Velocity\_Joint)

end

% function 15：绘制期望与实际的图像

function []=Plot\_for\_PD(Exp\_Pos\_Joint,Real\_Pos\_Joint,Exp\_Velocity\_Joint,Real\_Velocity\_Joint,Exp\_Accelerate\_Velocity\_Joint,Real\_Accelerate\_Velocity\_Joint)

% Position Trace

Plot\_for\_PD\_inJoint\_Pos(Exp\_Pos\_Joint,Real\_Pos\_Joint);

% Velocity Trace

Plot\_for\_PD\_inJoint\_Velocity(Exp\_Velocity\_Joint,Real\_Velocity\_Joint);

% Accelerate Velocity Trace

Plot\_for\_PD\_inJoint\_Accelerate\_Velocity(Exp\_Accelerate\_Velocity\_Joint,Real\_Accelerate\_Velocity\_Joint);

end

% function 16: 绘制期望与实际（关节空间下的位置跟踪）

function []=Plot\_for\_PD\_inJoint\_Pos(Exp\_Pos\_Joint,Real\_Pos\_Joint)

% 绘制位置跟踪

subplot(3,3,1)

hold on

plot(Exp\_Pos\_Joint(:,1),Exp\_Pos\_Joint(:,2))

plot(Real\_Pos\_Joint(:,1),Real\_Pos\_Joint(:,2))

title("Joint{1}-Pos")

ylabel("旋转角度/°")

xlabel("时间/s")

legend('Exp','Real')

subplot(3,3,2)

hold on

plot(Exp\_Pos\_Joint(:,1),Exp\_Pos\_Joint(:,3))

plot(Real\_Pos\_Joint(:,1),Real\_Pos\_Joint(:,3))

title("Joint{2}-Pos")

legend('Exp','Real')

ylabel("旋转角度/°")

xlabel("时间/s")

subplot(3,3,3)

hold on

plot(Exp\_Pos\_Joint(:,1),Exp\_Pos\_Joint(:,4))

plot(Real\_Pos\_Joint(:,1),Real\_Pos\_Joint(:,4))

title("Joint{3}-Pos")

legend('Exp','Real')

ylabel("平移距离/m")

xlabel("时间/s")

end

% function 17: 绘制期望与实际（关节空间下的速度跟踪）

function []=Plot\_for\_PD\_inJoint\_Velocity(Exp\_Velocity\_Joint,Real\_Velocity\_Joint)

% 绘制位置跟踪

subplot(3,3,4)

hold on

plot(Exp\_Velocity\_Joint(:,1),Exp\_Velocity\_Joint(:,2))

plot(Real\_Velocity\_Joint(:,1),Real\_Velocity\_Joint(:,2))

title("Joint{1}-velocity")

ylabel("旋转角速度/(°/s)")

xlabel("时间/s")

legend('Exp','Real')

subplot(3,3,5)

hold on

plot(Exp\_Velocity\_Joint(:,1),Exp\_Velocity\_Joint(:,3))

plot(Real\_Velocity\_Joint(:,1),Real\_Velocity\_Joint(:,3))

title("Joint{2}-velocity")

legend('Exp','Real')

ylabel("旋转角速度/(°/s)")

xlabel("时间/s")

subplot(3,3,6)

hold on

plot(Exp\_Velocity\_Joint(:,1),Exp\_Velocity\_Joint(:,4))

plot(Real\_Velocity\_Joint(:,1),Real\_Velocity\_Joint(:,4))

title("Joint{3}-velocity")

legend('Exp','Real')

ylabel("平移速度/(m/s)")

xlabel("时间/s")

end

% function 18: 绘制期望与实际（关节空间下的速度跟踪）

function []=Plot\_for\_PD\_inJoint\_Accelerate\_Velocity(Exp\_Accelerate\_Velocity\_Joint,Real\_Accelerate\_Velocity\_Joint)

% 绘制位置跟踪

subplot(3,3,7)

hold on

plot(Exp\_Accelerate\_Velocity\_Joint(:,1),Exp\_Accelerate\_Velocity\_Joint(:,2))

plot(Real\_Accelerate\_Velocity\_Joint(:,1),Real\_Accelerate\_Velocity\_Joint(:,2))

title("Joint{1}-Accelerate Velocity")

ylabel("旋转角加速度/(°/s^2)")

xlabel("时间/s")

legend('Exp','Real')

subplot(3,3,8)

hold on

plot(Exp\_Accelerate\_Velocity\_Joint(:,1),Exp\_Accelerate\_Velocity\_Joint(:,3))

plot(Real\_Accelerate\_Velocity\_Joint(:,1),Real\_Accelerate\_Velocity\_Joint(:,3))

title("Joint{2}-Accelerate Velocity")

legend('Exp','Real')

ylabel("旋转角加速度/(°/s^2)")

xlabel("时间/s")

subplot(3,3,9)

hold on

plot(Exp\_Accelerate\_Velocity\_Joint(:,1),Exp\_Accelerate\_Velocity\_Joint(:,4))

plot(Real\_Accelerate\_Velocity\_Joint(:,1),Real\_Accelerate\_Velocity\_Joint(:,4))

title("Joint{3}-Accelerate Velocity")

legend('Exp','Real')

ylabel("平移加速度/(m/s^2)")

xlabel("时间/s")

end

% function 19：PD控制器（逆运动学）

function []=PD\_Controllor\_For\_Trace\_Reverse()

% 机器人的基本信息

l1=0.250;

l2=0.250;

% 获得期望下关节空间的信息

Exp\_Joint\_Data = Get\_Joint\_Data();

% 获取期望的轨迹（关节空间下）

Exp\_Pos\_Joint = [Exp\_Joint\_Data(:,1) ,Exp\_Joint\_Data(:,2:4)];

% 获得期望的速度（关节空间下）

Exp\_Velocity\_Joint = [Exp\_Joint\_Data(:,1) ,Exp\_Joint\_Data(:,5:7)];

% 获得期望的加速度

Exp\_Accelerate\_Velocity\_Joint = [Exp\_Joint\_Data(:,1) ,Exp\_Joint\_Data(:,8:10)];

% 期望机械臂末端的位置

Exp\_Pos\_XYZ = Get\_End\_Pos\_InXYZ();

% PD-Controllor-parameter(kp,kd,dt)

kp = 1000;

kd = 400;

dt = 0.001;

% input\_v

current\_joint\_v = zeros(1,3);

next\_joint\_v = zeros(1,3);

% Pos\_error

delta\_pos\_i = zeros(1,3);

delta\_pos\_i\_1 = zeros(1,3);

delta\_pos\_i\_2 = zeros(1,3);

% real Position

Real\_Pos\_Joint = zeros(10001,4);

Real\_Velocity\_Joint = zeros(10001,4);

Real\_Accelerate\_Velocity\_Joint = zeros(10001,4);

Real\_Pos\_XYZ = zeros(10001,4);

% 列向量

% trace by PD

for i=1:10001

% Input-Joint-v

% Output-Joint-Pos

% 记录速度

t = dt\*(i-1);

Real\_Velocity\_Joint(i,:)=[t,current\_joint\_v];

% 关节空间下的关节位置

if i==1

Real\_Pos\_Joint(i,1)=0;

Real\_Pos\_Joint(i,2:4)=current\_joint\_v\*dt;

else

Real\_Pos\_Joint(i,1)=t;

Real\_Pos\_Joint(i,2:4)=Real\_Pos\_Joint(i-1,2:4)+current\_joint\_v\*dt;

end

% 逆运动学：从笛卡尔空间到关节空间

% 计算笛卡尔空间的位置

q1=Real\_Pos\_Joint(i,2);q2=Real\_Pos\_Joint(i,3);q3=Real\_Pos\_Joint(i,4);

% 角度转弧度

q1=pi\*q1/180;q2=pi\*q2/180;

x=cos(q1+q2)\*l2+cos(q1)\*l1;

y=sin(q1+q2)\*l2+sin(q1)\*l1;

z=-q3;

Real\_Pos\_XYZ(i,:)=[t,x,y,z];

% Pos\_delta(1,3)

delta\_pos= Exp\_Pos\_Joint(i,2:4)-Real\_Pos\_Joint(i,2:4);

if i==1

delta\_pos\_i =delta\_pos;

elseif i==2

delta\_pos\_i\_1 = delta\_pos\_i;

delta\_pos\_i =delta\_pos;

else

delta\_pos\_i\_2 = delta\_pos\_i\_1;

delta\_pos\_i\_1 = delta\_pos\_i;

delta\_pos\_i =delta\_pos;

end

delta\_joint\_v = kp\*(delta\_pos\_i-delta\_pos\_i\_2)+kd\*(delta\_pos\_i\_1-2\*delta\_pos\_i\_1+delta\_pos\_i\_2);

% 计算出下一时刻的速度

next\_joint\_v = current\_joint\_v + delta\_joint\_v;

% 加速度（关节）

Real\_Accelerate\_Velocity\_Joint(i,:)=[t,delta\_joint\_v/dt];

% 更新速度（关节）

current\_joint\_v = next\_joint\_v;

end

% 机械臂末端位置变化(机械臂基座的笛卡尔坐标系下)

figure(1)

subplot(3,1,1)

hold on

plot(Exp\_Pos\_XYZ(:,1),Exp\_Pos\_XYZ(:,2));

plot(Real\_Pos\_XYZ(:,1),Real\_Pos\_XYZ(:,2));

legend('exp','real')

xlabel('时间/s')

ylabel('位置/m')

title('Pos-X')

subplot(3,1,2)

hold on

plot(Exp\_Pos\_XYZ(:,1),Exp\_Pos\_XYZ(:,3));

plot(Real\_Pos\_XYZ(:,1),Real\_Pos\_XYZ(:,3));

legend('exp','real')

xlabel('时间/s')

ylabel('位置/m')

title('Pos-Y')

subplot(3,1,3)

hold on

plot(Exp\_Pos\_XYZ(:,1),Exp\_Pos\_XYZ(:,4));

plot(Real\_Pos\_XYZ(:,1),Real\_Pos\_XYZ(:,4));

legend('exp','real')

xlabel('时间/s')

ylabel('位置/m')

title('Pos-Z')

% 机械臂末端位置误差

figure(2)

subplot(3,1,1)

hold on

plot(Exp\_Pos\_XYZ(:,1),Exp\_Pos\_XYZ(:,2)-Real\_Pos\_XYZ(:,2));

legend('error')

xlabel('时间/s')

ylabel('位置/m')

title('Pos-X-Error')

subplot(3,1,2)

hold on

plot(Exp\_Pos\_XYZ(:,1),Exp\_Pos\_XYZ(:,3)-Real\_Pos\_XYZ(:,3));

legend('error')

xlabel('时间/s')

ylabel('位置/m')

title('Pos-Y-Error')

subplot(3,1,3)

hold on

plot(Exp\_Pos\_XYZ(:,1),Exp\_Pos\_XYZ(:,4)-Real\_Pos\_XYZ(:,4));

legend('error')

xlabel('时间/s')

ylabel('位置/m')

title('Pos-Z-Error')

% 关节角度变化(期望与实际)

% figure(3)

% subplot(3,1,1)

% hold on

% plot(Exp\_Pos\_Joint(:,1),Exp\_Pos\_Joint(:,2));

% plot(Real\_Pos\_Joint(:,1),Real\_Pos\_Joint(:,2));

% plot(Real\_Pos\_Joint(:,1),Exp\_Pos\_Joint(:,2)-Real\_Pos\_Joint(:,2));

% legend('exp','real','error')

% subplot(3,1,2)

% hold on

% plot(Exp\_Pos\_Joint(:,1),Exp\_Pos\_Joint(:,3));

% plot(Real\_Pos\_Joint(:,1),Real\_Pos\_Joint(:,3));

% legend('exp','real')

% subplot(3,1,3)

% hold on

% plot(Exp\_Pos\_Joint(:,1),Exp\_Pos\_Joint(:,4));

% plot(Real\_Pos\_Joint(:,1),Real\_Pos\_Joint(:,4));

% legend('exp','real')

% 绘制实际角度变化

% figure(4)

% subplot(3,1,1)

% plot(Real\_Pos\_Joint(:,1),Real\_Pos\_Joint(:,2));

% subplot(3,1,2)

% plot(Real\_Pos\_Joint(:,1),Real\_Pos\_Joint(:,3));

% subplot(3,1,3)

% plot(Real\_Pos\_Joint(:,1),Real\_Pos\_Joint(:,4));

error\_x=0;error\_y=0;error\_z=0;

for i=1:10001

error\_x = error\_x + Exp\_Pos\_XYZ(i,2)-Real\_Pos\_XYZ(i,2);

error\_y = error\_y + Exp\_Pos\_XYZ(i,3)-Real\_Pos\_XYZ(i,3);

error\_z = error\_z + Exp\_Pos\_XYZ(i,4)-Real\_Pos\_XYZ(i,4);

end

error\_x=error\_x/10000

error\_y=error\_y/10000

error\_z=error\_z/10000

end

% function 20：轨迹生成器2（两点间的轨迹规划）

function Planning\_Data\_PVA=Trajectory\_Planning\_Generator\_2()

% 规划时间：10s

% 起始位置：q0-[0 0 0]

% 期望位置：qd-[30 60 0.15]

% 规划要求：初始和期望关节的速度、加速度均为0

% 前提假设：没有最大速度限制

% 所用模型：五次多项式 q(t) =>a0=0;a1=0;a2=0;

% qi a0 a1 a2 a3 a4 a5

% q1 0 0 0 3/10 -9/200 9/5000

% …………运行function21即可得到所有参数

Planning\_Data\_PVA = zeros(10001,10);

dt = 0.001;

for i=1:10001

t = dt\*(i-1);

Pos\_Joint = Planning\_Pos\_Formula(t);

Vel\_Joint = Planning\_Velocity\_Formula(t);

Acc\_Joint = Planning\_Accelerate\_Formula(t);

% Planning\_Data\_PVA(i,:)=[t,Pos\_Joint(1),Pos\_Joint(2),Pos\_Joint(3),Vel\_Joint(1),Vel\_Joint(2),Vel\_Joint(3),Acc\_Joint(1),Acc\_Joint(2),Acc\_Joint(3)];

Planning\_Data\_PVA(i,:)=[t,Pos\_Joint,Vel\_Joint,Acc\_Joint];

end

end

% function 21:解方程，解析参数

function []=Solve\_For\_Function2()

% 定义参数

syms a3 a4 a5

% 定义方程组

eqns1 = [1000\*a3+10000\*a4+100000\*a5==30,300\*a3+4000\*a4+50000\*a5==0,60\*a3+1200\*a4+20000\*a5==0];

solve(eqns1)

eqns2 = [1000\*a3+10000\*a4+100000\*a5==60,300\*a3+4000\*a4+50000\*a5==0,60\*a3+1200\*a4+20000\*a5==0];

solve(eqns2)

eqns3 = [1000\*a3+10000\*a4+100000\*a5==0.15,300\*a3+4000\*a4+50000\*a5==0,60\*a3+1200\*a4+20000\*a5==0];

solve(eqns3)

end

% function 22:关节位置函数

function Real\_Time\_Postion=Planning\_Pos\_Formula(t)

q1=(3/10)\*t^3 +(-9/200)\*t^4 +(9/5000)\*t^5;

q2=(3/5)\*t^3 +(-9/100)\*t^4 +(9/2500)\*t^5;

q3=(3/2000)\*t^3 +(-9/40000)\*t^4 +(9/1000000)\*t^5;

Real\_Time\_Postion = [q1,q2,q3];

end

% function 23：关节速度函数

function Real\_Time\_Velocity=Planning\_Velocity\_Formula(t)

v1=(9/10)\*t^2 +(-36/200)\*t^3 +(45/5000)\*t^4;

v2=(9/5)\*t^2 +(-36/100)\*t^3 +(45/2500)\*t^4;

v3=(9/2000)\*t^2 +(-36/40000)\*t^3 +(45/1000000)\*t^4;

Real\_Time\_Velocity = [v1,v2,v3];

end

% function 24：关节加速度函数

function Real\_Time\_Accelerate=Planning\_Accelerate\_Formula(t)

a1=(18/10)\*t +(-108/200)\*t^2 +(180/5000)\*t^3;

a2=(18/5)\*t +(-108/100)\*t^2 +(180/2500)\*t^3;

a3=(18/2000)\*t +(-108/40000)\*t^2 +(180/1000000)\*t^3;

Real\_Time\_Accelerate = [a1,a2,a3];

end

% function 25：对得到的位置、速度、加速度绘图

function []=Plot\_For\_Trajectory\_Planning\_2()

Planning\_Data\_PVA = Trajectory\_Planning\_Generator\_2();

% Joint 1

% Position Velocity Accelerate

figure(1)

subplot(3,1,1)

hold on

title("Joint 1 Position(t)")

xlabel("时间t/s")

ylabel("关节位置p/°")

plot(Planning\_Data\_PVA(:,1),Planning\_Data\_PVA(:,2))

subplot(3,1,2)

hold on

title("Joint 1 Velocity(t)")

xlabel("时间t/s")

ylabel("关节速度v/(°/s)")

plot(Planning\_Data\_PVA(:,1),Planning\_Data\_PVA(:,5))

subplot(3,1,3)

hold on

title("Joint 1 Accelerate(t)")

xlabel("时间t/s")

ylabel("关节加速度a/(°/s^2)")

plot(Planning\_Data\_PVA(:,1),Planning\_Data\_PVA(:,8))

% Joint 2

% Position Velocity Accelerate

figure(2)

subplot(3,1,1)

hold on

title("Joint 2 Position(t)")

xlabel("时间t/s")

ylabel("关节位置p/°")

plot(Planning\_Data\_PVA(:,1),Planning\_Data\_PVA(:,3))

subplot(3,1,2)

hold on

title("Joint 2 Velocity(t)")

xlabel("时间t/s")

ylabel("关节速度v/(°/s)")

plot(Planning\_Data\_PVA(:,1),Planning\_Data\_PVA(:,6))

subplot(3,1,3)

hold on

title("Joint 2 Accelerate(t)")

xlabel("时间t/s")

ylabel("关节加速度a/(°/s^2)")

plot(Planning\_Data\_PVA(:,1),Planning\_Data\_PVA(:,9))

% Joint 3

% Position Velocity Accelerate

figure(3)

subplot(3,1,1)

hold on

title("Joint 3 Position(t)")

xlabel("时间t/s")

ylabel("关节位置p/m")

plot(Planning\_Data\_PVA(:,1),Planning\_Data\_PVA(:,4))

subplot(3,1,2)

hold on

title("Joint 3 Velocity(t)")

xlabel("时间t/s")

ylabel("关节速度v/(m/s)")

plot(Planning\_Data\_PVA(:,1),Planning\_Data\_PVA(:,7))

subplot(3,1,3)

hold on

title("Joint 3 Accelerate(t)")

xlabel("时间t/s")

ylabel("关节加速度a/(m/s^2)")

plot(Planning\_Data\_PVA(:,1),Planning\_Data\_PVA(:,10))

end

% function 26: 阻抗控制(位置&力混合控制)

% 要求：绘制出关节位置，速度，加速度，控制力矩曲线

% 目的：阻抗控制是通过控制方法使机械手末端呈现需要的刚性和阻尼

% 通常：

% 对于需要进行位置控制的自由度，要求大纲性，及表现出很硬的特性以确保位置控制精度

% 对于需要进行力控制的自由度，则要求在该方向有合适的阻抗特性，以满足接触力的要求

% 机械臂连杆的参数：

% l1=0.250;

% l2=0.250;

% 计算位置的时候为了计算精度，单位应该均统一为mm，但是在计算力(N)和力矩(N·m)的时候，需要转换为m

% 暂时无重力补偿

function []=Impedance\_Control()

% 机器人的物理模型参数

m1=2.4312;m2=3.7860;m3=0.5552;

% 令惯性矩阵为m

m = [m1 0 0;0 m2 0;0 0 m3];

% 进行轨迹规划，获取规划信息，即期望信息（关节空间下）

% 数据（10001，10）

% 数据说明：[t,Px,Py,Pz,Vx,Vy,Vz,Ax,Ay,Az]

Exp\_Data\_Joint = Trajectory\_Planning\_Generator\_2();

% 期望刚性KX

K\_X = [500 0 0;0 500 0;0 0 500];

% 阻尼参数KB

K\_B = [50 0 0;0 50 0;0 0 30];

% 时间步长

dt = 0.001;

% 记录控制过程中的实时数据（关节空间）

% 数据存储格式

% 1-t

% (2,3,4)-(P1,P2,P3)

% (5,6,7)-(V1,V2,V3)

% (8,9,10)-(a1,a2,a3)

% (11,12,13)-(T1,T2,T3)

Real\_Data\_Joint=zeros(10001,13);

% i时刻的位置误差

Pos\_Error\_i = zeros(1,3);

Pos\_Error\_i\_1 = zeros(1,3);

Pos\_Error\_i\_2 = zeros(1,3);

% Torque(i时刻，q1 q2 q3关节的力矩)

Current\_Torque\_Joint = zeros(1,3);

% 加速度

Current\_Accelerate\_Joint=zeros(1,3);

% 速度

Current\_Velocity\_Joint= zeros(1,3);

% 位置

Current\_Position\_Joint=zeros(1,3);

t=0;

% 控制器

for i=1:10001

% 获取当前时间并保存

t= dt\*(i-1);

Real\_Data\_Joint(i,1)=t;

% 获取当前时刻，即i时刻的位置误差

if i==1

Pos\_Error\_i = Exp\_Data\_Joint(i,2:4);

else

Pos\_Error\_i = Exp\_Data\_Joint(i,2:4)-Real\_Data\_Joint(i-1,2:4);

end

% delta\_torque (3,1)

delta\_torque\_31 = K\_X\*(Pos\_Error\_i-Pos\_Error\_i\_1)'+K\_B\*(Pos\_Error\_i-2\*Pos\_Error\_i\_1+Pos\_Error\_i\_2)';

% delta\_torque (1,3)

delta\_torque\_13 = delta\_torque\_31';

% Current\_Torque

Current\_Torque\_Joint = Current\_Torque\_Joint + delta\_torque\_13;

% Save the Current\_Torque to Real\_Data\_Joint(i,11:13)

Real\_Data\_Joint(i,11:13)=Current\_Torque\_Joint;

% 计算加速度,保存加速度

Current\_Torque\_31 = Current\_Torque\_Joint';

Current\_Accelerate\_31 = inv(m) \* Current\_Torque\_31;

Current\_Accelerate\_Joint = Current\_Accelerate\_31';

Real\_Data\_Joint(i,8:10) = Current\_Accelerate\_Joint;

% 通过得到加速度计算新的速度,并保存下来

Current\_Velocity\_Joint = Current\_Velocity\_Joint + Current\_Accelerate\_Joint\*dt;

Real\_Data\_Joint(i,5:7) = Current\_Velocity\_Joint;

% 计算位移,并保存下来

Current\_Position\_Joint = Current\_Position\_Joint + Current\_Velocity\_Joint \* dt;

Real\_Data\_Joint(i,2:4) = Current\_Position\_Joint;

% 更新误差

Pos\_Error\_i\_2 = Pos\_Error\_i\_1;

Pos\_Error\_i\_1 = Pos\_Error\_i;

end

% plot(Real\_Data\_joint(:,1),Real\_Data\_joint(:,2))

Plot\_For\_Impedance\_Controller\_Resault(Exp\_Data\_Joint,Real\_Data\_Joint)

end

% function 27: 为阻抗控制器的结果进行图像绘制

% 内容说明：对实际运动过程的位置、速度、加速度、关节扭矩输出进行绘制，并且与期望值进行比较

% 函数调用位置：在阻抗控制器计算完成后进行绘图

% 传入参数说明：期望值数据(10001,10)、真实值数据(10001,13)

function Plot\_For\_Impedance\_Controller\_Resault(Exp\_Data\_Joint,Real\_Data\_Joint)

% 关节一（位置、速度、加速度、力矩）

figure(1)

subplot(2,2,1) % 期望位置&实际位置

hold on

title("Joint-1 Position Compare")

xlabel("时间/s")

ylabel("关节位置/°")

plot(Exp\_Data\_Joint(:,1),Exp\_Data\_Joint(:,2))

plot(Real\_Data\_Joint(:,1),Real\_Data\_Joint(:,2))

legend("Exp","Real")

subplot(2,2,2)

hold on

title("Joint-1 Velocity Compare")

xlabel("时间/s")

ylabel("关节角速度/(°/s)")

plot(Exp\_Data\_Joint(:,1),Exp\_Data\_Joint(:,5))

plot(Real\_Data\_Joint(:,1),Real\_Data\_Joint(:,5))

legend("Exp","Real")

subplot(2,2,3)

hold on

title("Joint-1 Accelerate Compare")

xlabel("时间/s")

ylabel("关节角加速度/(°/s^2)")

plot(Exp\_Data\_Joint(:,1),Exp\_Data\_Joint(:,8))

plot(Real\_Data\_Joint(:,1),Real\_Data\_Joint(:,8))

legend("Exp","Real")

subplot(2,2,4)

hold on

title("Joint-1 Torque Output")

xlabel("时间/s")

ylabel("关节输出力矩/(N·m)")

plot(Real\_Data\_Joint(:,1),Real\_Data\_Joint(:,11))

legend("Real")

% 关节二（位置、速度、加速度、力矩）

figure(2)

subplot(2,2,1)

hold on

title("Joint-2 Position Compare")

xlabel("时间/s")

ylabel("关节位置/°")

plot(Exp\_Data\_Joint(:,1),Exp\_Data\_Joint(:,3))

plot(Real\_Data\_Joint(:,1),Real\_Data\_Joint(:,3))

legend("Exp","Real")

subplot(2,2,2)

hold on

title("Joint-2 Velocity Compare")

xlabel("时间/s")

ylabel("关节角速度/(°/s)")

plot(Exp\_Data\_Joint(:,1),Exp\_Data\_Joint(:,6))

plot(Real\_Data\_Joint(:,1),Real\_Data\_Joint(:,6))

legend("Exp","Real")

subplot(2,2,3)

hold on

title("Joint-2 Accelerate Compare")

xlabel("时间/s")

ylabel("关节角加速度/(°/s^2)")

plot(Exp\_Data\_Joint(:,1),Exp\_Data\_Joint(:,9))

plot(Real\_Data\_Joint(:,1),Real\_Data\_Joint(:,9))

legend("Exp","Real")

subplot(2,2,4)

hold on

title("Joint-2 Torque Output")

xlabel("时间/s")

ylabel("关节输出力矩/(N·m)")

plot(Real\_Data\_Joint(:,1),Real\_Data\_Joint(:,12))

legend("Real")

% 关节三（位置、速度、加速度、力矩）

figure(3)

subplot(2,2,1)

hold on

title("Joint-3 Position Compare")

xlabel("时间/s")

ylabel("关节位置/m")

plot(Exp\_Data\_Joint(:,1),Exp\_Data\_Joint(:,4))

plot(Real\_Data\_Joint(:,1),Real\_Data\_Joint(:,4))

legend("Exp","Real")

subplot(2,2,2)

hold on

title("Joint-2 Velocity Compare")

xlabel("时间/s")

ylabel("关节角速度/(m/s)")

plot(Exp\_Data\_Joint(:,1),Exp\_Data\_Joint(:,7))

plot(Real\_Data\_Joint(:,1),Real\_Data\_Joint(:,7))

legend("Exp","Real")

subplot(2,2,3)

hold on

title("Joint-3 Accelerate Compare")

xlabel("时间/s")

ylabel("关节角加速度/(m/s^2)")

plot(Exp\_Data\_Joint(:,1),Exp\_Data\_Joint(:,10))

plot(Real\_Data\_Joint(:,1),Real\_Data\_Joint(:,10))

legend("Exp","Real")

subplot(2,2,4)

hold on

title("Joint-3 Torque Output")

xlabel("时间/s")

ylabel("关节输出力矩/(N)")

plot(Real\_Data\_Joint(:,1),Real\_Data\_Joint(:,13))

legend("Real")

end

% function 28:系统施加外力下的阻抗控制

% 说明：系统在4~6s内添加了外力[2N·m 2N·m 3N]

% 假设：机械臂感受到的外力由每个关节上的力矩传感器感知

function []=Impedance\_Control\_With\_Disturb()

% 机器人的物理模型参数

m1=2.4312;m2=3.7860;m3=0.5552;

% 令惯性矩阵为m

m = [m1 0 0;0 m2 0;0 0 m3];

% 进行轨迹规划，获取规划信息，即期望信息（关节空间下）

% 数据（10001，10）

% 数据说明：[t,Px,Py,Pz,Vx,Vy,Vz,Ax,Ay,Az]

Exp\_Data\_Joint = Trajectory\_Planning\_Generator\_2();

% 期望刚性KX

K\_X = [500 0 0;0 500 0;0 0 500];

% 阻尼参数KB

K\_B = [50 0 0;0 50 0;0 0 30];

% 时间步长

dt = 0.001;

% 记录控制过程中的实时数据（关节空间）

% 数据存储格式

% 1-t

% (2,3,4)-(P1,P2,P3)

% (5,6,7)-(V1,V2,V3)

% (8,9,10)-(a1,a2,a3)

% (11,12,13)-(T1,T2,T3)

Real\_Data\_Joint=zeros(10001,13);

% i时刻的位置误差

Pos\_Error\_i = zeros(1,3);

Pos\_Error\_i\_1 = zeros(1,3);

Pos\_Error\_i\_2 = zeros(1,3);

% Torque(i时刻，q1 q2 q3关节的力矩)

Current\_Torque\_Joint = zeros(1,3);

% 加速度

Current\_Accelerate\_Joint=zeros(1,3);

% 速度

Current\_Velocity\_Joint= zeros(1,3);

% 位置

Current\_Position\_Joint=zeros(1,3);

% 内环PID需要的参数

% 传感器感知到的力矩大小(10001,4)

Moment\_Sensor\_Joint = Get\_Disturb\_Data();

% 期望输出力矩

Exp\_Moment\_Joint = zeros(1,3);

% 当前输出力矩

Current\_Moment\_Joint = zeros(1,3);

% 当前传感器的数值

Current\_Sensor\_Joint = zeros(1,3);

% 当前时刻力矩的输出误差、前一时刻的误差、前两个时刻的误差

Moment\_Error\_i = zeros(1,3);

Moment\_Error\_i\_1 = zeros(1,3);

Moment\_Error\_i\_2 = zeros(1,3);

% 内环PID的参数

Moment\_Kp = [0.01 0 0;0 0.01 0;0 0 0.00001];

Moment\_Kd = [0.01 0 0;0 0.01 0;0 0 0.00001];

% 控制器

for i=1:10001

% 获取当前时间并保存

t= dt\*(i-1);

Real\_Data\_Joint(i,1)=t;

% 获取当前时刻，即i时刻的位置误差

if i==1

Pos\_Error\_i = Exp\_Data\_Joint(i,2:4);

else

Pos\_Error\_i = Exp\_Data\_Joint(i,2:4)-Real\_Data\_Joint(i-1,2:4);

end

% 外环PID

% delta\_torque (3,1)

delta\_torque\_31 = K\_X\*(Pos\_Error\_i-Pos\_Error\_i\_1)'+K\_B\*(Pos\_Error\_i-2\*Pos\_Error\_i\_1+Pos\_Error\_i\_2)';

% delta\_torque (1,3)

delta\_torque\_13 = delta\_torque\_31';

% Current\_Torque

Current\_Torque\_Joint = Current\_Torque\_Joint + delta\_torque\_13;

% 内环PID

Current\_Moment\_Joint = Current\_Torque\_Joint;

% 传感器

dissturb = Moment\_Sensor\_Joint(i,2:4);

% 计算当前误差

Moment\_Error\_i = -1\* dissturb;

% 通过PID重新调整输出力矩（采用增量式PID）

delta\_moment\_31 = Moment\_Kp\*(Moment\_Error\_i - Moment\_Error\_i\_1)'+Moment\_Kd\*(Moment\_Error\_i - 2\*Moment\_Error\_i\_1 + Moment\_Error\_i\_2)';

% 转置

delta\_moment\_13 = delta\_moment\_31';

% 更新输出力矩

Current\_Moment\_Joint = Current\_Moment\_Joint + delta\_moment\_13;

% 更新误差

Moment\_Error\_i\_2 = Moment\_Error\_i\_1;

Moment\_Error\_i\_1 = Moment\_Error\_i;

% 将内环输出传递给下一个环节

Current\_Torque\_Joint = Current\_Moment\_Joint;

% Save the Current\_Torque to Real\_Data\_Joint(i,11:13)

Real\_Data\_Joint(i,11:13)=Current\_Torque\_Joint;

% 计算加速度,保存加速度

Current\_Torque\_31 = Current\_Torque\_Joint';

Current\_Accelerate\_31 = inv(m) \* Current\_Torque\_31;

Current\_Accelerate\_Joint = Current\_Accelerate\_31';

Real\_Data\_Joint(i,8:10) = Current\_Accelerate\_Joint;

% 通过得到加速度计算新的速度,并保存下来

Current\_Velocity\_Joint = Current\_Velocity\_Joint + Current\_Accelerate\_Joint\*dt;

Real\_Data\_Joint(i,5:7) = Current\_Velocity\_Joint;

% 计算位移,并保存下来

Current\_Position\_Joint = Current\_Position\_Joint + Current\_Velocity\_Joint \* dt;

Real\_Data\_Joint(i,2:4) = Current\_Position\_Joint;

% 更新误差

Pos\_Error\_i\_2 = Pos\_Error\_i\_1;

Pos\_Error\_i\_1 = Pos\_Error\_i;

end

% plot(Real\_Data\_joint(:,1),Real\_Data\_joint(:,2))

Plot\_For\_Impedance\_Controller\_Resault\_With\_Disturb(Exp\_Data\_Joint,Real\_Data\_Joint,Moment\_Sensor\_Joint)

end

%function 29：设置系统施加的外力

function Disturb\_Data = Get\_Disturb\_Data()

Disturb\_Data=zeros(10001,4);

dt=0.001;

for i=1:10001

t = dt\*(i-1);

Disturb\_Data(i,1)=t;

if t>=4 && t<=6

Disturb\_Data(i,2:4)=[2 2 3];

end

end

% test for Disturb Data

% figure(1)

% hold on

% plot(Disturb\_Data(:,1),Disturb\_Data(:,2))

% plot(Disturb\_Data(:,1),Disturb\_Data(:,3))

% plot(Disturb\_Data(:,1),Disturb\_Data(:,4))

% legend('Joint-1-Disturb','Joint-2-Disturb','Joint-3-Disturb')

end

%function 30: 为带有外力干扰的阻抗控制器的内环进行绘图

function Plot\_For\_Impedance\_Controller\_Resault\_With\_Disturb(Exp\_Data\_Joint,Real\_Data\_Joint,Disturb\_Data)

% 关节一（位置、速度、加速度、力矩）

figure(1)

subplot(2,2,1) % 期望位置&实际位置

hold on

title("Joint-1 Position Compare")

xlabel("时间/s")

ylabel("关节位置/°")

plot(Exp\_Data\_Joint(:,1),Exp\_Data\_Joint(:,2))

plot(Real\_Data\_Joint(:,1),Real\_Data\_Joint(:,2))

legend("Exp","Real")

subplot(2,2,2)

hold on

title("Joint-1 Velocity Compare")

xlabel("时间/s")

ylabel("关节角速度/(°/s)")

plot(Exp\_Data\_Joint(:,1),Exp\_Data\_Joint(:,5))

plot(Real\_Data\_Joint(:,1),Real\_Data\_Joint(:,5))

legend("Exp","Real")

subplot(2,2,3)

hold on

title("Joint-1 Accelerate Compare")

xlabel("时间/s")

ylabel("关节角加速度/(°/s^2)")

plot(Exp\_Data\_Joint(:,1),Exp\_Data\_Joint(:,8))

plot(Real\_Data\_Joint(:,1),Real\_Data\_Joint(:,8))

legend("Exp","Real")

subplot(2,2,4)

hold on

title("Joint-1 Torque Output")

xlabel("时间/s")

ylabel("关节输出力矩/(N·m)")

plot(Real\_Data\_Joint(:,1),Real\_Data\_Joint(:,11))

plot(Disturb\_Data(:,1),Disturb\_Data(:,2))

legend("Real","Disturb")

% 关节二（位置、速度、加速度、力矩）

figure(2)

subplot(2,2,1)

hold on

title("Joint-2 Position Compare")

xlabel("时间/s")

ylabel("关节位置/°")

plot(Exp\_Data\_Joint(:,1),Exp\_Data\_Joint(:,3))

plot(Real\_Data\_Joint(:,1),Real\_Data\_Joint(:,3))

legend("Exp","Real")

subplot(2,2,2)

hold on

title("Joint-2 Velocity Compare")

xlabel("时间/s")

ylabel("关节角速度/(°/s)")

plot(Exp\_Data\_Joint(:,1),Exp\_Data\_Joint(:,6))

plot(Real\_Data\_Joint(:,1),Real\_Data\_Joint(:,6))

legend("Exp","Real")

subplot(2,2,3)

hold on

title("Joint-2 Accelerate Compare")

xlabel("时间/s")

ylabel("关节角加速度/(°/s^2)")

plot(Exp\_Data\_Joint(:,1),Exp\_Data\_Joint(:,9))

plot(Real\_Data\_Joint(:,1),Real\_Data\_Joint(:,9))

legend("Exp","Real")

subplot(2,2,4)

hold on

title("Joint-2 Torque Output")

xlabel("时间/s")

ylabel("关节输出力矩/(N·m)")

plot(Real\_Data\_Joint(:,1),Real\_Data\_Joint(:,12))

plot(Disturb\_Data(:,1),Disturb\_Data(:,3))

legend("Real","Disturb")

% 关节三（位置、速度、加速度、力矩）

figure(3)

subplot(2,2,1)

hold on

title("Joint-3 Position Compare")

xlabel("时间/s")

ylabel("关节位置/m")

plot(Exp\_Data\_Joint(:,1),Exp\_Data\_Joint(:,4))

plot(Real\_Data\_Joint(:,1),Real\_Data\_Joint(:,4))

legend("Exp","Real")

subplot(2,2,2)

hold on

title("Joint-2 Velocity Compare")

xlabel("时间/s")

ylabel("关节角速度/(m/s)")

plot(Exp\_Data\_Joint(:,1),Exp\_Data\_Joint(:,7))

plot(Real\_Data\_Joint(:,1),Real\_Data\_Joint(:,7))

legend("Exp","Real")

subplot(2,2,3)

hold on

title("Joint-3 Accelerate Compare")

xlabel("时间/s")

ylabel("关节角加速度/(m/s^2)")

plot(Exp\_Data\_Joint(:,1),Exp\_Data\_Joint(:,10))

plot(Real\_Data\_Joint(:,1),Real\_Data\_Joint(:,10))

legend("Exp","Real")

subplot(2,2,4)

hold on

title("Joint-3 Torque Output")

xlabel("时间/s")

ylabel("关节输出力矩/(N)")

plot(Real\_Data\_Joint(:,1),Real\_Data\_Joint(:,13))

plot(Disturb\_Data(:,1),Disturb\_Data(:,4))

legend("Real","Disturb")

end