

SHANGHAI JIAO TONG UNIVERSITY

课程项目报告

PROJECT REPORT OF CURRICULUM



机器人学课程大作业

小组成员:

指导教师: 吴建华 熊振华

学院(系): 机械与动力工程学院

目录

一、项目背景

1.六自由度机器人应用背景………………………………………………………………… …3

2.国内外研究现状 ………………………………………………………………………………… 3

二、结构设计与建模………………………………………………………………………………………4

1.基本参数设计………………………………………………………………………………………4

2.精度分析………………………………………………………………………………………………4

i.原理与结构 ………………………………………………………………………………………4

ii. 走线方式 ………………………………………………………………………………………5

iii.元器件选择 ……………………………………………………………………………………6

3.实物建模 ……………………………………………………………………………………………8

三、数学建模与仿真

1.数学建模……………………………………………………………………………………………10

i. 坐标系与运动学……………………………………………………………………… 10

ii. 动力学与逆运动学反解 ……………………………………………………………11

iii. 雅可比矩阵 ……………………………………………………………………………… 13

2.仿真……………………………………………………………………………………………………15

i.运动学仿真 ……………………………………………………………………………………15

ii.动力学仿真 …………………………………………………………………………………16

3.机器人控制程序…………………………………………………………………………………16

四、总结与展望

1.项目体会与建议 ………………………………………………………………………………17

2.应用展望……………………………………………………………………………………………17

五、参考文献………………………………………………………………………………………………17

六、附录代码………………………………………………………………………………………………18

**一、项目背景**

**1.六自由度机器人应用背景**

在以智能制造为主题的“工业4.0”和“中国制造2025“的背景下，为进一步提高工业领域的自动化、智能化程度，协作机器人已经被广泛应用于各大制造行业。根据国际工业机器人联合会（IFR）的统计，在2020年间，全球使用中的工业机器人达到3,014,879台，相较去年增长10%；亚洲成为世界最大的工业机器人市场，20年亚洲新增工业机器人数量占全球新增的71%（19年占比67%）；中国（44%）、日本（10%）、美国（8%）、韩国（8%）和德国（6%）是全球五大工业机器人市场（76%）；全球每年新增工业机器人数量基本成上升趋势，但其增速由于新冠疫情有所放缓甚至下降。

图表, 条形图

描述已自动生成

图1 近十年工业机器人增长量/年

在设计工业机器人时，考虑到使机器人末端在三维空间内实现无约束的自由移动需要描述刚体位置的三个自由度以及描述其姿态的另外三个自由度，因此六自由度机器人理所当然地成为了制造行业的普遍选择，并被广泛应用于焊接、装配、搬运以及喷涂等领域。

**2.国内外研究现状**

协作机器人是一种被设计成能与人类在共同工作空间进行近距离互动的机器人。本小组设计了一种经典的六轴关节型协作机器人。

关节机器人(Robot joints)，也称关节手臂机器人或关节机械手臂，是当今工业领域中最常见的工业机器人的形态之一。适合用于诸多工业领域的机械自动化作业，比如，自动装配、喷漆、搬运、焊接等工作。

图示, 工程绘图

描述已自动生成

图2 关节机器人示意

然而，国际机器人联合会（IFR）的统计数据表明近几年工业机器人的出货量有所回落,工业机器人市场面临瓶颈。2018年全球工业机器人销售额为165亿美元，全球出货量为42.2万台，比2017年增长6％，但是作为全球最大市场的中国已经出现弱势，工业机器人年销售 15.6 万台，同比下降 1.7%。工业机器人的发展瓶颈不是技术造成的，而是工业市场决定的。工业机器人的发展因此需要全新的技术变革推动。

**二、结构设计与建模**

**1.基本参数设计**

考虑到电机性能的影响，根据尽量减小负载质量以及电机的尺寸以保证机器人结构紧凑的原则，选择小负载、中等臂展。从指导老师给出的参数中，我们选择如下基本参数组合：

表1 六自由度机器人基本参数

|  |  |
| --- | --- |
| 负载/kg | 臂展/m |
| 3.0 | 1.0 |

**2.精度分析**

**i.原理与结构**

关节机器人按摆动方向可分为铅锤和水平方向两种，因此这类机器人又可分为垂直关节机器人和水平关节机器人。

垂直关节机器人模拟了人类的手臂功能，由垂直于地面的腰部旋转轴（相当于大臂旋转的肩部旋转轴）带动小臂旋转的肘部旋转轴以及小臂前端的手腕等构成。手腕通常由2～3个自由度构成。其动作空间近似一个球体，所以也称多关节球面机器人。其优点是可以自由地实现三维空间的各种姿势，可以生成各种复杂形状的轨迹。相对机器人的安装面积。其动作范围很宽；缺点是结构刚度较低，动作的绝对位置精度磨较低。它广泛应用于代替人完成装配作业、货物搬运、电弧焊接、喷涂、点焊接等作业场合。

水平关节机器人在结构上具有串联配置的二个能够在水平面内旋转的手臂，其自由度可以根据用途选择2～4个，动作空间为一圆柱体。水平关节机器人的优点是在垂直方向上的刚性好，能方便地实现二维平面上的动作，在装配作业中得到普遍应用。

本组所设计的协作机器人为垂直6关节手腕机器人。

图片包含 游戏机, 规模

描述已自动生成图片包含 游戏机, 物体, 风向标

描述已自动生成

|  |  |
| --- | --- |
| 垂直关节机器人 | 水平关节机器人 |
| 图3 六轴机器人关节示意图 | |

**ii.走线方式**

机器人通过控制设置在各关节处的电机进行各种动作，需采用电连接线连接电机、电路板及电源等.机器人走线形式分为内部布线和外部布线两种形式，内部布线需要在建模时就考虑过机器人内部过线结构，关节处预留空间等等比较麻烦，所以我们计划参考节卡机器人的管线包，采用外部走线的方式，通过在各关节处将电线直接引出，包入外接管线包中的方式走线，这样即可以避免线路对机器人关节的干涉影响，又可以保护线缆。

根据尽可能使走线结构清晰、易于检查和维护等原则，进行走线规划设计如下：

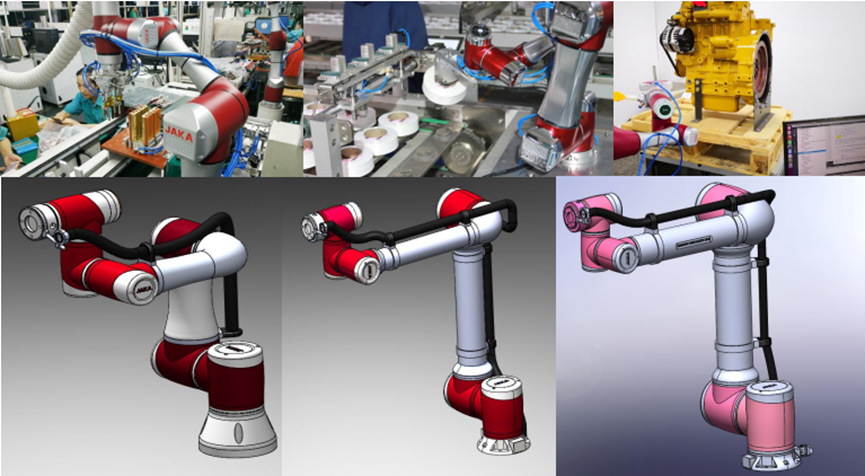


图4 六轴机器人走线方式示意图

**iii.元器件选择**

根据确定的结构（负载、臂展、两岸质量）等参数，选择符合设计要求同时兼顾经济性的元器件型号以及参数汇总如下：

表2 相关元器件型号和参数

|  |  |
| --- | --- |
| 序号 | 元件名称 |
| 1 | KAS-1A谐波减速器 |
| 2 | M221X力矩传感器 |

各元件选择理由如下：

1. 谐波减速器

通过动力学仿真可以看出，机器人关节最大扭矩为30Nm，仅有伺服电机无法满足转矩要求，在工业机器人系统中，大多数电机都需要安装谐波减速器。

在查找选择相关谐波减速器时，我们查看了国内谐波减速器的龙头企业绿地谐波，在查询中我们发现了一款比较适合的机电一体化执行器KAS系列实心轴旋转执行器。

其主要特点有：

1. 高转矩输出及转矩密度，如KAH-32旋转执行器输出转矩最高达459N.m
2. 旋转执行器定位精度高达40弧秒以内
3. 提供丰富产品选型，满足多样化需求
4. 防护等级高达IP67，适宜于恶劣工作环境
5. 可选择与KED系列EtherCAT总线型伺服驱动器配套使用，实现超低振动控制及可靠平稳运行，提供驱动控制整体解决方案。
6. 安装简单操作维护方便。

其中KAS-1A型平均最大转矩32NM，启停允许最大转矩42NM，满足仿真需求。半径79mm，适合小型机械臂的使用。

图片包含 游戏机, 物体, 发动机

描述已自动生成

图5 KAS系列实心轴旋转执行器

2）力矩传感器

为了提高机器人力控精度，引进传感器是必不可少的。机器人力控技术按传感方式可分为三类：基于电机电流的力控制，基于关节转矩传感器的力控制和基于力/力矩传感器的力控制。

较为直观的获取外部力信息的方法是在每个关节都安装力矩传感器进行测量，进而估算出机器人末端的外部力。关节力矩传感器一般安装在机械臂关节减速器的输出端。

常见的六维力和力矩传感器制造商有美国ATI，德国ME，瑞士BOTA，德国HBM，宇立仪器等公司，在实际制作机械臂时，我们可以通过所需尺寸，测量范围选择力矩传感器。

考虑到本小组项目制造的为小型机械臂，我们选择宇立仪器的M221X单轴扭矩传感器，M221X系列是单轴扭矩传感器，特别适合用于机器人关节的扭矩测量，在协作机器人上有大量应用。SRI的扭矩传感器厚度薄至7mm，直径41至146mm，扭矩量程从10Nm至400Nm。传感器内置放大器，直流 5V输入，0 - 5V输出。其官方网站上有着各种具体型的尺寸表，方便我们进行电机以及减速器配合。

|  |
| --- |
|  |
| 图6 M221X单轴扭矩传感器 |

**3.实物建模**

|  |  |
| --- | --- |
|  |  |
| 7-1 关节1 | 7-2 关节2 |
|  |  |
| 7-3 关节3 | 7-4 关节4 |
|  |  |
| 7-5 关节5 | 7-6 关节6 |
|  | |
| 7-7 基座 | |

图7 六轴机器人三维建模图纸

|  |
| --- |
|  |
|  |
| 图8 机器人模型图 |

**三、数学建模与仿真**

**1.数学建模**

**i．正运动学**

表3.标准D-H

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| i |  |  |  |  |
| 1 | -0.22/2.8 | pi/2 | 0.2/2.8 | [-180,180] |
| 2 | -1.069/2.8 | 0 | 0 | [-135,135] |
| 3 | 0 | -pi/2 | 0.1/2.8 | [-90,90] |
| 4 | 0 | pi/2 | 1.04/2.8 | [-135,135] |
| 5 | 0 | pi/2 | 0 | [-90,90] |
| 6 | 0 | pi | 0.495/2.8 | [-180,180] |

由相邻连杆系的连杆变化矩阵

确定各连杆的变化矩阵分别为：

末端坐标系相对基座的位姿变换矩阵：

其中：

c23=c2\*c3-s2\*s3=cos(th2+th3)

s23=c2\*s3+c3\*s2=sin(th2+th3)

nx=s6\*(c4\*s1+s4\*c1\*c23)-c6\*(c5\*(s1\*s4-c4\*c1\*c23)+s5\*c1\*s23)

ny=c6\*(c5\*(c1\*c4+c4\*s1\*c23)-s5\*s1\*s23)-s6\*(c1\*c4-s4\*s1\*c23)

nz=c6\*(s5\*c23 +c4\*c5\*s23)+s4\*s6\*s23

ox=s6\*(c5\*(s1\*s4-c4\*c1\*c23)+s5\*c1\*s23)-c6\*(c4\*s1+s4\*c1\*c23)

oy=c6\*(c1\*c4-s4\*s1\*c23)-s6\*(c5\*(c1\*s4+c4\*s1\*c23)-c5\*s1\*s23)

oz=-s6\*(s5\*c23+c4\*c5\*s23)-c6\*s4\*s23

ax=s5\*(s1\*s4-c4\*c1\*c23)-c5\*c1\*s23

ay=-s5\*(c1\*s4+c4\*s1\*c23)-c5\*s1\*s23

az=c5\*(c23-c4\*s5\*s23)

px=a1\*c1-d4\*c1\*s23-d6\*(s5\*(s1\*s4-c4\*c1\*c23)-c5\*c1\*s23)+d3\*s1+a2\*c1^2;

py=d6\*(s5\*(c1\*s4+c4\*s1\*c23)+c5\*s1\*s23)-d4\*s1\*s23-d3\*c1+a1\*s1+a2\*c1\*s1;

pz=d1+d4\*c23+a2\*s1-d6\*(c5\*c23)-c4\*s5\*s23);

关节空间

操作空间:末端位姿

正向运动学方程

矩阵前3列分别为末端的x ,y, z轴的方向向量；若已知各个关节，可求末端位姿。

**ii．逆运动学**

已知机器人末端坐标系相对于基座标系的位姿矩阵，求对应的关节角q.

(1)的求解

由

令两边矩阵元素相等可以获得若干代数方程。

由（3，3）相等、（3，4）相等联立解得

其中

(2)的求解

由基座到末端的齐次变换矩阵知，总是成对出现，无法独立求解，故一起求解。

同理解得

式中，;求解后，对应的在关节空间内有无穷多解。

（3）的求解

由pz\*c23-(px+py)c1\*c23=d4-d6\*c5;可解得:

式中，

雅可比矩阵：在matlab程序中使用向量积求解实现。

**iii．动力学**

使用拉格朗日方程进行动力学建模：

，L=T-V；

各个机械臂总动能 为连杆 的动能, 为关节电机的动能。

在进行动力学建模时，简化机械臂内部结构，将电机质量和机械臂壳体质量共同计算，即只对各连杆建模，整体质量不变。

（1）动能的计算

首先计算连杆i上一定点A的动能，A的空间坐标为，式中，为A点在基座标系中的坐标，为第i关节到第0关节的位姿变换矩阵，为A点在连杆i中的局部坐标表示。则A的速度，加速度和动能满足：

速度：

加速度：

动能，式中，dm为点A的微元质量，trace表示矩阵的迹。

对连杆i进行积分即可得到连杆i的动能为：

其中，表示连杆i上的A点惯量属性。

取A点为质心，则可得到连杆i的惯量矩阵：



式中，为连杆i的等效质量，为连杆i质心在连杆i上的坐标表示。

则机械臂六个连杆总动能：

（2）势能的计算

连杆i的势能为： ,点A 为质心。

则六个连杆总势能为：

（3）建立动力学方程

则



同理可得

则机械臂系统的动力学方程为：

又

 式中，

后两行为全零行。

令 ,则

令

令：

则：

对于六轴串联机械臂，可以采用关节转角表述广义坐标，则动力学方程可展开为：





**2.运动学和动力学仿真**

**i.运动学仿真**

本项目选择在Matlab中进行空中抓球运动学仿真，可以实现抓取1个和2个球；具体代码附于文末，仿真截图如下：

电脑屏幕的照片

低可信度描述已自动生成

图9 Matlab运动学仿真

**ii．动力学仿真**

利用matlab robotics库的内置函数获得在规划路径下的关节空间，关节速度和关节加速度，求解获得关节力矩，输出力矩曲线：

图示, 工程绘图

描述已自动生成图示, 工程绘图

描述已自动生成

图示, 工程绘图

描述已自动生成 图示, 工程绘图

描述已自动生成

图10 Matlab动力学仿真

根据输出的力矩曲线特征，确定关节电机型号。

**3.机器人控制程序**

机器人控制程序以及求解雅各比矩阵的子程序附于文末，在此不做展开。

**四、总结与展望**

**1.项目体会与建议**

通过对于机器人学项目的学习，本组通过不同的分工实现了合作完成任务，收获颇丰；我们完成了从机器人的应用背景、基本原理与公式推导等理论知识，到设计与建模、程序设计、运动学和动力学仿真等实际操作等一系列工作，从理论上和实践上都加深了对于机器人学的理解与认识，出色的完成了设计与实现。

**2.应用展望**

近些年来工业机器人领域具有以下发展特点：

1.结合了机器视觉和其他传感系统的人工智能软件被融入到机器人中，使得机器人具有完成诸如精确的物品拾取等更复杂任务的能力。以工业机器人的轨迹控制为例，相对于传统的点位控制（PTP）、连续轨迹控制（CP）、力、力矩控制，依赖于人工神经网络、遗传算法等的智能控制方式使机器人具有更强的环境适应性和自学习能力，是机器人控制的未来发展趋势。

2.机器人更多地被应用于智能工厂中。与传统的固定机器人流水线工作方式不同，智能工厂中的机器人都具有一定的移动能力，相较于前者更加灵活；所以当产品模型完全更改时，工厂不需要像以前一样拆除生产线，只需要对这些机器人重新编程。此外，机器人与人类在没有区域分隔的条件下相互协作也是工业机器人未来的发展方向。

本次课程项目设计了一个基础的六轴协作机器人，通过项目设计的过程掌握到了基础的知识和技术，为日后的学习与研究提供了坚实的基础。所设计的机器人未来可以与人工智能、机器视觉等技术相结合。从而为工业机器人注入新的发展生机与活力。

**五、参考文献**

[1]International Fedration of Robotics. Executive Summary World Robotics 2021 Industrial Robots[R].IFR.2021

[2]IFR Press Room. *Top 5 Robot Trends 2021*[Z].IFR.2021

http://www.dongmao-drive.com/product/83.html

https://www.jaka.com/jszl.html

https://www.srisensor.com.cn/36.html

http://www.maxsine.com/news/knowledge/93.html

**六、附录**

1．运动学仿真代码

1. clear all;
2. clc;
3. % %机器人建模
4. th(1) = 0; d(1) = 0.2;   a(1) = -0.220;  alp(1) = pi/2;
5. th(2) = 0; d(2) = 0;     a(2) = -1.069; alp(2) = 0;
6. th(3) = 0; d(3) = 0.1;   a(3) = 0;      alp(3) = -pi/2;
7. th(4) = 0; d(4) = 1.04;  a(4) = 0;      alp(4) = pi/2;
8. th(5) = 0; d(5) = 0;     a(5) = 0;      alp(5) = pi/2;
9. th(6) = 0; d(6) = 0.495; a(6) = 0;      alp(6) = pi;
10. % DH parameters  th     d    a    alpha  sigma
11. L(1) = Link([th(1), d(1), a(1), alp(1), 0], 'revolute');
12. L(2) = Link([th(2), d(2), a(2), alp(2), 0], 'revolute');L(2).offset=-pi/2;
13. L(3) = Link([th(3), d(3), a(3), alp(3), 0], 'revolute');
14. L(4) = Link([th(4), d(4), a(4), alp(4), 0], 'revolute');L(4).offset=pi;
15. L(5) = Link([th(5), d(5), a(5), alp(5), 0], 'revolute');L(5).offset=-pi;
16. L(6) = Link([th(6), d(6), a(6), alp(6), 0], 'revolute');
17. **global** robot
18. robot = SerialLink(L); %SerialLink 类函数
19. robot.name='B09';

22. %% 设定球的参数
23. g=1/3;
24. **global** time\_step;
25. time\_step=0.05;
26. %球的初速度、位置值
27. ball\_initial\_pos=[6,0,1];
28. velx=-1;
29. vely=-1/6;
30. velz=1;
32. endtime=(velz+sqrt(velz\*velz+2\*g\*(1+ball\_initial\_pos(3))))/g;%到地板（z=-1）停止
33. time\_vec=0:time\_step:endtime;
34. time\_vec=transpose(time\_vec);
35. ball\_x=velx\*time\_vec+ball\_initial\_pos(1);
36. ball\_y=vely\*time\_vec+ball\_initial\_pos(2);
37. ball\_z=velz\*time\_vec-g/2\*time\_vec.^2+ball\_initial\_pos(3);

40. %% 求解机械臂接球轨迹
41. data=catchball(ball\_x,ball\_y,ball\_z,time\_vec);
42. **if** isempty(data)
43. q\_targ\_t=zeros(length(time\_vec),6,1);
44. "Catch Ball Fail"
45. **else**
46. "Catch Ball Success"
47. q\_targ\_t=data(:,2:7);
48. time\_vec=data(:,1);
50. end

53. %% 画图
54. fig=figure(1);
55. fig.Position=[100 100 1400 1200];
56. i=1;
57. radius=0.1;
58. [ball\_x\_mesh,ball\_y\_mesh,ball\_z\_mesh]=sphere;
59. ball\_x\_mesh=ball\_x\_mesh\*radius;
60. ball\_y\_mesh=ball\_y\_mesh\*radius;
61. ball\_z\_mesh=ball\_z\_mesh\*radius;
63. v=[70 30];
64. w=[-3 7 -4 4 -1 4];
65. range\_w=[w(2)-w(1),w(4)-w(3),w(6)-w(5)];
66. **for** i=1:1:length(time\_vec)
67. ball=surf(ball\_x\_mesh+ball\_x(i),ball\_y\_mesh+ball\_y(i),ball\_z\_mesh+ball\_z(i),'EdgeColor','none');
68. hold on;
69. shooter=plot3(ball\_initial\_pos(1),ball\_initial\_pos(2),ball\_initial\_pos(3),'Marker','o','MarkerSize',15,'LineWidth',3);
70. trace\_ball=plot3(ball\_x(1:i),ball\_y(1:i),ball\_z(1:i),'LineStyle','--','LineWidth',2);
71. linez\_ball=plot3([ball\_x(i),ball\_x(i)],[ball\_y(i),ball\_y(i)],[-1,ball\_z(i)],'LineStyle','--');
72. liney\_ball=plot3([ball\_initial\_pos(1),ball\_initial\_pos(1)],[ball\_initial\_pos(2),ball\_y(i)],[-1,-1],'LineStyle','--');
73. linex\_ball=plot3([ball\_initial\_pos(1),ball\_x(i)],[ball\_y(i),ball\_y(i)],[-1,-1],'LineStyle','--');
74. axis(w);
75. pbaspect(range\_w);
76. robot.plot3d(q\_targ\_t(i,:),'tilesize',0.5,'workspace',w,'path','.\stl','nowrist','view',v,'perspective');
78. **if** i<length(time\_vec)
79. clf(fig)
80. end
81. end
82. %%
83. function data=catchball(ball\_x,ball\_y,ball\_z,time\_vec)
84. **global** robot;
85. **global** time\_step;
86. N=length(time\_vec);
87. q\_init=[0 0 0 0 0 0];
88. iscatch=-1;
89. **for** i=1:1:N
90. targ\_p=[ball\_x(i);ball\_y(i);ball\_z(i)];
91. %首先根据工作空间排除达不到的点↓
92. **if** **not**((targ\_p(1)>-2) & (targ\_p(1)<2) & (targ\_p(2)>-2) & (targ\_p(2)<2) & (targ\_p(3)>-2) & (targ\_p(3)<3))
93. **continue**;
94. end
95. %排除达不到的点↓
96. p=[0 0 -1 0;0 -1 0 0;-1 0 0 0;0 0 0 1];
97. p(1:3,4)=targ\_p;
98. q\_targ=ikine(robot,p);
99. **if**(isempty(q\_targ))
100. **continue**;
101. end
102. %排除时间层面上来不及达到的点↓
103. time=0.8;
104. qd\_limit=2.2;
105. robot\_time\_vec=0:time\_step:time;
106. [q\_targ\_t ,qd\_t, qdd\_t]=jtraj(q\_init,q\_targ,robot\_time\_vec);
108. **while**(max(abs(qd\_t),[],'all')>qd\_limit)
109. time=time\*max(abs(qd\_t),[],'all')/qd\_limit+time\_step;
110. robot\_time\_vec=0:time\_step:time;
111. [q\_targ\_t ,qd\_t, qdd\_t]=jtraj(q\_init,q\_targ,robot\_time\_vec);
112. end
113. %可以达到的点↓
114. **if** time<=time\_vec(i)
115. iscatch=1;
116. **break**;
117. end
118. end
119. %输出结果↓
120. **if** iscatch==-1
121. data=[];
122. **else**
123. n=length(robot\_time\_vec);
124. data=zeros(i,7,1);
125. data(:,1)=time\_vec(1:i);
126. data(i-n+1:i,2:7)=q\_targ\_t;
128. end
129. end

运动学仿真（抓取2球）

1. clear all;
2. clc;
3. % %机器人建模
4. th(1) = 0; d(1) = 0.2;   a(1) = -0.220;  alp(1) = pi/2;
5. th(2) = 0; d(2) = 0;     a(2) = -1.069; alp(2) = 0;
6. th(3) = 0; d(3) = 0.1;   a(3) = 0;      alp(3) = -pi/2;
7. th(4) = 0; d(4) = 1.04;  a(4) = 0;      alp(4) = pi/2;
8. th(5) = 0; d(5) = 0;     a(5) = 0;      alp(5) = pi/2;
9. th(6) = 0; d(6) = 0.495; a(6) = 0;      alp(6) = pi;
10. % DH parameters  th     d    a    alpha  sigma
11. L(1) = Link([th(1), d(1), a(1), alp(1), 0], 'revolute');
12. L(2) = Link([th(2), d(2), a(2), alp(2), 0], 'revolute');L(2).offset=-pi/2;
13. L(3) = Link([th(3), d(3), a(3), alp(3), 0], 'revolute');
14. L(4) = Link([th(4), d(4), a(4), alp(4), 0], 'revolute');L(4).offset=pi;
15. L(5) = Link([th(5), d(5), a(5), alp(5), 0], 'revolute');L(5).offset=-pi;
16. L(6) = Link([th(6), d(6), a(6), alp(6), 0], 'revolute');
17. global robot
18. robot = SerialLink(L); %SerialLink 类函数
19. robot.name='B09';

22. %% 设定球的参数
23. g=1/3;
24. global time\_step;
25. time\_step=0.05;
26. %球的初速度、位置值
27. ball\_initial\_pos=[6,0,1];
28. ball\_initial\_pos2=[6,0,2];
29. radius=0.1;
30. velx=-1;
31. vely=1/6;
32. velz=1;
34. velx2=-1;
35. vely2=-1/6;
36. velz2=1;
38. endtime=(velz+sqrt(velz\*velz+2\*g\*(1+ball\_initial\_pos(3))))/g;%到地板（z=-1）停止
39. time\_vec=0:time\_step:endtime;
40. time\_vec=transpose(time\_vec);
41. ball\_x=velx\*time\_vec+ball\_initial\_pos(1);
42. ball\_y=vely\*time\_vec+ball\_initial\_pos(2);
43. ball\_z=velz\*time\_vec-g/2\*time\_vec.^2+ball\_initial\_pos(3);
45. endtime2=(velz2+sqrt(velz2\*velz2+2\*g\*(1+ball\_initial\_pos2(3))))/g;%到地板（z=-1）停止
47. time\_vec2=0:time\_step:endtime2;
48. time\_vec2=transpose(time\_vec2);
49. ball\_x2=velx2\*time\_vec2+ball\_initial\_pos2(1);
50. ball\_y2=vely2\*time\_vec2+ball\_initial\_pos2(2);
51. ball\_z2=velz2\*time\_vec2-g/2\*time\_vec2.^2+ball\_initial\_pos2(3);
52. %% 求解机械臂接球轨迹
53. global i\_stop;
54. i\_stop=0;
55. global ball\_pos\_fir ball\_pos\_sec;
56. global iscatch2;
57. data=catchball2(ball\_x,ball\_y,ball\_z,time\_vec,ball\_x2,ball\_y2,ball\_z2,time\_vec2);
58. iscatch2
59. **if** isempty(data)
60. q\_targ\_t=zeros(length(time\_vec),6,1);
61. "Catch Ball Fail"
62. **else**
63. "Catch Ball Success"
64. q\_targ\_t=data(:,2:7);
65. time\_vec=data(:,1);
67. end
69. ball\_x=ball\_pos\_fir(:,1);
70. ball\_y=ball\_pos\_fir(:,2);
71. ball\_z=ball\_pos\_fir(:,3);
72. ball\_x2=ball\_pos\_sec(:,1);
73. ball\_y2=ball\_pos\_sec(:,2);
74. ball\_z2=ball\_pos\_sec(:,3);
76. %% 画图
78. fig=figure(1);
79. fig.Position=[100 100 1400 1200];
80. i=1;
81. radius=0.1;
82. [ball\_x\_mesh,ball\_y\_mesh,ball\_z\_mesh]=sphere;
83. ball\_x\_mesh=ball\_x\_mesh\*radius;
84. ball\_y\_mesh=ball\_y\_mesh\*radius;
85. ball\_z\_mesh=ball\_z\_mesh\*radius;
87. v=[70 30];
88. w=[-3 7 -4 4 -1 4];
89. range\_w=[w(2)-w(1),w(4)-w(3),w(6)-w(5)];
90. **for** i=1:1:i\_stop
91. ball=surf(ball\_x\_mesh+ball\_x(i),ball\_y\_mesh+ball\_y(i),ball\_z\_mesh+ball\_z(i),'EdgeColor','none');
92. hold on;
93. ball2=surf(ball\_x\_mesh+ball\_x2(i),ball\_y\_mesh+ball\_y2(i),ball\_z\_mesh+ball\_z2(i),'EdgeColor','none');
95. shooter=plot3(ball\_initial\_pos(1),ball\_initial\_pos(2),ball\_initial\_pos(3),'Marker','o','MarkerSize',15,'LineWidth',3);
96. shooter2=plot3(ball\_initial\_pos2(1),ball\_initial\_pos2(2),ball\_initial\_pos2(3),'Marker','o','MarkerSize',15,'LineWidth',3);
97. trace\_ball=plot3(ball\_x(1:i),ball\_y(1:i),ball\_z(1:i),'LineStyle','--','LineWidth',2,'Color','r');
98. trace\_ball2=plot3(ball\_x2(1:i),ball\_y2(1:i),ball\_z2(1:i),'LineStyle','--','LineWidth',2,'Color','b');
100. axis(w);
101. pbaspect(range\_w);
102. robot.plot3d(q\_targ\_t(i,:),'tilesize',0.5,'workspace',w,'path','.\stl','nowrist','view',v,'perspective');
104. clf(fig)
105. end
107. **for** i=i\_stop+1:1:size(data,1)
108. ball2=surf(ball\_x\_mesh+ball\_x2(i),ball\_y\_mesh+ball\_y2(i),ball\_z\_mesh+ball\_z2(i),'EdgeColor','none');
109. hold on;
110. shooter=plot3(ball\_initial\_pos(1),ball\_initial\_pos(2),ball\_initial\_pos(3),'Marker','o','MarkerSize',15,'LineWidth',3);
111. shooter2=plot3(ball\_initial\_pos2(1),ball\_initial\_pos2(2),ball\_initial\_pos2(3),'Marker','o','MarkerSize',15,'LineWidth',3);
112. trace\_ball2=plot3(ball\_x2(1:i),ball\_y2(1:i),ball\_z2(1:i),'LineStyle','--','LineWidth',2,'Color','b');
113. axis(w);
114. pbaspect(range\_w);
115. robot.plot3d(q\_targ\_t(i,:),'tilesize',0.5,'workspace',w,'path','.\stl','nowrist','view',v,'perspective');
117. **if** i<length(time\_vec)
118. clf(fig)
119. end
120. end

123. %%
124. function data=catchball2(ball\_x,ball\_y,ball\_z,time\_vec,ball\_x2,ball\_y2,ball\_z2,time\_vec2)
125. global robot;
126. global time\_step;
127. N=length(time\_vec);
128. q\_init=[0 0 0 0 0 0];
129. iscatch=-1;
130. global iscatch2;
131. iscatch2=-1;
132. **for** i=1:1:N
133. targ\_p=[ball\_x(i);ball\_y(i);ball\_z(i)];
134. targ\_p2=[ball\_x2(i);ball\_y2(i);ball\_z2(i)];
135. %首先根据工作空间排除达不到的点↓
136. **if** (not((targ\_p(1)>-2) & (targ\_p(1)<2) & (targ\_p(2)>-2) & (targ\_p(2)<2) & (targ\_p(3)>-2) & (targ\_p(3)<3)))...
137. &(not((targ\_p2(1)>-2) & (targ\_p2(1)<2) & (targ\_p2(2)>-2) & (targ\_p2(2)<2) & (targ\_p2(3)>-2) & (targ\_p2(3)<3)))
138. **continue**;
139. end
140. %排除达不到的点↓
141. p=[0 0 -1 0;0 -1 0 0;-1 0 0 0;0 0 0 1];
142. p2=[0 0 -1 0;0 -1 0 0;-1 0 0 0;0 0 0 1];
143. p(1:3,4)=targ\_p;
144. p2(1:3,4)=targ\_p2;
145. q\_targ=ikine(robot,p);
146. q\_targ2=ikine(robot,p2);
147. **if**(isempty(q\_targ))&(isempty(q\_targ2))
148. **continue**;
149. end
150. global ball\_pos\_fir ball\_pos\_sec;
151. **if** not (isempty(q\_targ))
152. ball\_pos\_fir=[ball\_x,ball\_y,ball\_z,time\_vec];
153. ball\_pos\_sec=[ball\_x2,ball\_y2,ball\_z2,time\_vec2];
154. **else**
155. ball\_pos\_sec=[ball\_x,ball\_y,ball\_z,time\_vec];
156. ball\_pos\_fir=[ball\_x2,ball\_y2,ball\_z2,time\_vec2];
157. q\_targ=q\_targ2;
158. end
160. time=0.5;
161. qd\_limit=2.2;
162. robot\_time\_vec=0:time\_step:time;
163. [q\_targ\_t ,qd\_t, qdd\_t]=jtraj(q\_init,q\_targ,robot\_time\_vec);
164. **while**(max(abs(qd\_t),[],'all')>qd\_limit)
165. time=time\*max(abs(qd\_t),[],'all')/qd\_limit+time\_step;
166. robot\_time\_vec=0:time\_step:time;
167. [q\_targ\_t ,qd\_t, qdd\_t]=jtraj(q\_init,q\_targ,robot\_time\_vec);
168. end
170. **if** time<=time\_vec(i)
171. iscatch=1;
172. q\_init=q\_targ;
173. **break**;
174. end
175. end
176. global i\_stop;
177. i\_stop=i;
178. i\_start=i+4;
179. time\_start\_sec=time\_vec(i\_start);
180. N=size(ball\_pos\_sec,1);
182. **for** i=i\_start:1:N;
184. targ\_p=[ball\_pos\_sec(i,1),ball\_pos\_sec(i,2),ball\_pos\_sec(i,3)];
185. **if** (not((targ\_p(1)>-2) & (targ\_p(1)<2) & (targ\_p(2)>-2) & (targ\_p(2)<2) & (targ\_p(3)>-2) & (targ\_p(3)<3)))
186. **continue**;
187. end
188. p=[0 0 -1 targ\_p(1);0 -1 0 targ\_p(2);-1 0 0 targ\_p(3);0 0 0 1];
189. p(1:3,1:3)=[targ\_p(1)/sqrt(targ\_p(1)^2+targ\_p(2)^2) -targ\_p(2)/sqrt(targ\_p(1)^2+targ\_p(2)^2) 0;targ\_p(2)/sqrt(targ\_p(1)^2+targ\_p(2)^2) targ\_p(1)/sqrt(targ\_p(1)^2+targ\_p(2)^2) 0;0 0 1]\*p(1:3,1:3);
190. q\_targ=ikine(robot,p);
191. **if** isempty(q\_targ)
192. **continue**;
193. end
194. time2=0.4;
195. qd\_limit=4;
196. robot\_time\_vec2=0:time\_step:time2;
197. [q\_targ\_t2 ,qd\_t2, qdd\_t2]=jtraj(q\_init,q\_targ,robot\_time\_vec2);
198. **while**(max(abs(qd\_t2),[],'all')>qd\_limit)
199. time2=time2\*max(abs(qd\_t2),[],'all')/qd\_limit+time\_step;
200. robot\_time\_vec2=0:time\_step:time2;
201. [q\_targ\_t2,qd\_t2, qdd\_t2]=jtraj(q\_init,q\_targ,robot\_time\_vec2);
202. end
203. **if** time2+time\_start\_sec<=ball\_pos\_sec(i,4)
204. iscatch2=1;
205. **break**;
206. end
207. end


211. %输出结果↓
212. **if** iscatch==-1
213. data=[];
214. **else**
216. **if**  iscatch2==-1
217. N=size(ball\_pos\_sec,1);
218. n1=length(robot\_time\_vec);
219. data=zeros(N,7,1);
220. data(:,1)=ball\_pos\_sec(1:N,4);
221. data(i\_stop-n1+1:i\_stop,2:7)=q\_targ\_t;
222. **for** j=i\_stop+1:1:N
223. data(j,2:7)=q\_targ\_t(n1,:);
224. end
225. **else**
226. n1=length(robot\_time\_vec);
227. n2=length(robot\_time\_vec2);
228. data=zeros(i,7,1);
229. data(:,1)=ball\_pos\_sec(1:i,4);
230. data(i\_stop-n1+1:i\_stop,2:7)=q\_targ\_t;
231. **for** j=i\_stop:1:i-n2
232. data(j,2:7)=q\_targ\_t(n1,:);
233. end
234. data(i-n2+1:i,2:7)=q\_targ\_t2;
236. end
237. end

240. end

2.动力学仿真代码

1. clear;
2. clc;
3. % %机器人建模
4. th(1) = 0; d(1) = 0.2/2.8;   a(1) = -0.220/2.8;  alp(1) = pi/2;
5. th(2) = 0; d(2) = 0;     a(2) = -1.069/2.8; alp(2) = 0;
6. th(3) = 0; d(3) = 0.1/2.8;   a(3) = 0;      alp(3) = -pi/2;
7. th(4) = 0; d(4) = 1.04/2.8;  a(4) = 0;      alp(4) = pi/2;
8. th(5) = 0; d(5) = 0;     a(5) = 0;      alp(5) = pi/2;
9. th(6) = 0; d(6) = 0.495/2.8; a(6) = 0;      alp(6) = pi;
10. % DH parameters  th     d    a    alpha  sigma
11. L(1) = Link([th(1), d(1), a(1), alp(1), 0], 'revolute');
12. L(2) = Link([th(2), d(2), a(2), alp(2), 0], 'revolute');L(2).offset=-pi/2;
13. L(3) = Link([th(3), d(3), a(3), alp(3), 0], 'revolute');
14. L(4) = Link([th(4), d(4), a(4), alp(4), 0], 'revolute');L(4).offset=pi;
15. L(5) = Link([th(5), d(5), a(5), alp(5), 0], 'revolute');L(5).offset=-pi;
16. L(6) = Link([th(6), d(6), a(6), alp(6), 0], 'revolute');
18. %动力学参量，质量，质心位置和惯量矩阵，其中末端质量加上了负载的3kg
19. L(1).r=[0.09851,0.27522,0.18156];
20. L(1).m=0.41418;
21. L(1).Jm=2.2e-4;
22. L(1).I=10^(-9)\*[1734891.83 -184792.26 -86325.71;
23. -184792.26 2376522.66 172929.64;
24. -86325.71 172929.64 1252382.38];
26. L(2).r=[-0.00925,0.53218,0.17848];
27. L(2).m=0.52455;
28. L(2).Jm=2.2e-4;
29. L(2).I=10^(-9)\*[11839437.44 -930504.09 -23887.6;
30. -930504.09 1072895.45 -76643.7;
31. -23887.6 -76643.7 11930866.95];
33. L(3).r=[0.10246,0.68597,0.13014];
34. L(3).m=0.45030;
35. L(3).Jm=2.2e-4;
36. L(3).G=-53.70364;
37. L(3).I=10^(-9)\*[2052918.21 -186784.56 -928086.57;
38. -186784.56 2731992.7 164232.49;
39. -928086.57 164232.49 2168655.87];
41. L(4).r=[0.20975,0.66891,-0.005];
42. L(4).m=0.34716;
43. L(4).Jm=2.2e-4;
44. L(4).I=10^(-9)\*[1379297.73 -51681.67 -291688.61;
45. -51681.67 2000353.98 110826.92;
46. -291688.61 110826.92 1230334.41];
48. L(5).r=[0.26130,0.65481,-0.06671];
49. L(5).m=0.14325;
50. L(5).Jm=2.2e-4;
51. L(5).I=10^(-9)\*[328240.83 10409.66 -19760.38;
52. 10409.66 217596.66 -27141.63;
53. -19760.38 -27141.63 313278.42];
55. L(6).r=[0.24833,0.54880,-0.03822];
56. L(6).m=3.03159;
57. L(6).Jm=2.2e-4;
58. L(6).I=10^(-9)\*[19773.04 1895.11 5905.68;
59. 1895.11 14851.04 -2539.11;
60. 5905.68 -2539.11 22694.09];
62. robot = SerialLink(L); %SerialLink 类函数
63. robot.name='B09';

66. %机械臂的工作空间,关节3，5设置工作角度为[-90,90]
67. A=unifrnd(-pi,pi,[1,3000]);
68. B=unifrnd(-pi\*3/4,pi\*3/4,[1,3000]);
69. C=unifrnd(-pi/2,pi\*3/4,[1,3000]);
70. D=unifrnd(-pi\*3/4,3\*pi/4,[1,3000]);
71. E=unifrnd(-pi/2,pi/2,[1,3000]);
72. F=unifrnd(-pi\*3/4,pi\*3/4,[1,3000]);
74. G=cell(3000,3);%建立元胞数组
75. **for** n=1:3000
76. G{n}=[A(n) B(n) C(n) D(n) E(n) F(n)];
77. end                          %生成3000组随机点
78. H1=cell2mat(G);              %将元胞数组转化为矩阵
79. T0=double(robot.fkine(H1));      %机械臂正解
81. %% 运动学正解
82. theta2 = [0.1,0,0,0,0,0];               %关节角
83. p1=robot.fkine(theta2)  ;               %fkine正解函数，根据关节角theta，求解出末端位姿p
84. q1=ikine(robot,p1) ;                    %ikine逆解函数，根据末端位姿p，求解出关节角q
86. %已知初始点和终止点的位姿，五次多项式规划路径
87. T1=[ 0.5048   -0.7521    0.4236   -0.1557
88. -0.7693   -0.1694    0.6160    0.0370
89. -0.3915   -0.6369   -0.6641    0.3072
90. 0         0         0    1.0000
91. ];
92. T2=[   -0.5736    0.5505    0.6066    0.1039
93. -0.8132   -0.4712   -0.3415    0.4722
94. 0.0979   -0.6892    0.7180   -0.3047
95. 0         0         0    1.0000
96. ];
97. init\_ang=robot.ikine(T1);               %根据起始点位姿，得到起始点关节角
98. targ\_ang=robot.ikine(T2);               %根据终止点位姿，得到终止点关节角
100. %轨迹规划方法并生成动画
101. time=0:0.01:2;
102. subplot(3,2,[1,3]);                     %subplot 对画面分区 三行两列 占用1到3的位置
103. f=3;
104. figure(f);
105. [q ,qd, qdd]=jtraj(init\_ang,targ\_ang,time);%五次多项式轨迹，得到关节角度，角速度，角加速度
106. grid on
107. T=robot.fkine(q(:,:,:));                        %根据插值，得到末端执行器位姿
108. nT=T.T;
109. subplot(3,2,1);
110. plot3(squeeze(nT(1,4,:)),squeeze(nT(2,4,:)),squeeze(nT(3,4,:)));%输出末端轨迹
111. title('输出末端轨迹');
113. figure(f);
114. tau=robot.rne\_dh(q,qd,qdd);
115. subplot(3,2,2);
116. plot(time,tau);grid on;title('关节力矩');
117. xlabel('t/s');
118. ylabel('tau/Nm');
119. legend('关节1','关节2','关节3','关节4','关节5','关节6','Location','bestoutside');
120. grid on;
122. figure(f)
123. subplot(3, 2, 3);
124. **for** i=1:6
125. plot(time,q(:,i));
126. hold on;
127. end
128. legend('关节1','关节2','关节3','关节4','关节5','关节6','Location','bestoutside');
129. title('q');
130. xlabel('t/s');
131. ylabel('q/rad');
132. grid on;
134. figure(f)
135. subplot(3, 2, 4);
136. **for** i=1:6
137. plot(time,qd(:,i));
138. hold on;
139. end
140. legend('关节1','关节2','关节3','关节4','关节5','关节6','Location','bestoutside');
141. title('速度');
142. xlabel('t/s');
143. ylabel('w/rad/s');
144. grid on;
146. figure(f)
147. subplot(3, 2, 5);
148. **for** i=1:6
149. plot(time,qdd(:,i));
150. hold on;
151. end
152. legend('关节1','关节2','关节3','关节4','关节5','关节6','Location','bestoutside');
153. title('加速度');
154. xlabel('t/s');
155. ylabel('a/(rad/s/s)');
156. grid on;

3.机器人控制主程序

1. clear;
2. clc;
3. % %机器人建模
4. th(1) = 0; d(1) = 0.2;   a(1) = -0.220;  alp(1) = pi/2;
5. th(2) = 0; d(2) = 0;     a(2) = -1.069; alp(2) = 0;
6. th(3) = 0; d(3) = 0.1;   a(3) = 0;      alp(3) = -pi/2;
7. th(4) = 0; d(4) = 1.04;  a(4) = 0;      alp(4) = pi/2;
8. th(5) = 0; d(5) = 0;     a(5) = 0;      alp(5) = pi/2;
9. th(6) = 0; d(6) = 0.495; a(6) = 0;      alp(6) = pi;
10. % DH parameters  th     d    a    alpha  sigma
11. L(1) = Link([th(1), d(1), a(1), alp(1), 0], 'revolute');
12. L(2) = Link([th(2), d(2), a(2), alp(2), 0], 'revolute');L(2).offset=-pi/2;
13. L(3) = Link([th(3), d(3), a(3), alp(3), 0], 'revolute');
14. L(4) = Link([th(4), d(4), a(4), alp(4), 0], 'revolute');L(4).offset=pi;
15. L(5) = Link([th(5), d(5), a(5), alp(5), 0], 'revolute');L(5).offset=-pi;
16. L(6) = Link([th(6), d(6), a(6), alp(6), 0], 'revolute');
17. robot = SerialLink(L); %SerialLink 类函数
18. robot.name='B09';
20. L(1).qlim=[-180,180]/180\*pi;  %工作空间，3关节和5关节的空间空间到不了[-180,180]
21. L(2).qlim=[-135,135]/180\*pi;
22. L(3).qlim=[-90,90]/180\*pi;
23. L(4).qlim=[-135,135]/180\*pi;
24. L(5).qlim=[-90,90]/180\*pi;
25. L(6).qlim=[-180,180]/180\*pi;
27. %机械臂的工作空间,关节3，5设置工作角度为[-90,90]
28. A=unifrnd(-pi,pi,[1,3000]);
29. B=unifrnd(-pi\*3/4,pi\*3/4,[1,3000]);
30. C=unifrnd(-pi/2,pi\*3/4,[1,3000]);
31. D=unifrnd(-pi\*3/4,3\*pi/4,[1,3000]);
32. E=unifrnd(-pi/2,pi/2,[1,3000]);
33. F=unifrnd(-pi,pi,[1,3000]);
35. G=cell(3000,3);%建立元胞数组
36. **for** n=1:3000
37. G{n}=[A(n) B(n) C(n) D(n) E(n) F(n)];
38. end                          %生成3000组随机点
39. H1=cell2mat(G);              %将元胞数组转化为矩阵
40. T=double(robot.fkine(H1));      %机械臂正解
41. %figure(1)
42. %scatter3(squeeze(T(1,4,:)),squeeze(T(2,4,:)),squeeze(T(3,4,:))) %工作空间绘图
44. %% 运动学正解
45. theta2 = [0.1,0,0,0,0,0];               %关节角
46. p1=robot.fkine(theta2)  ;               %fkine正解函数，根据关节角theta，求解出末端位姿p
47. q1=ikine(robot,p1) ;                    %ikine逆解函数，根据末端位姿p，求解出关节角q
49. %已知初始点和终止点的位姿，五次多项式规划路径
50. init\_ang=robot.ikine(T(:,:,10));                %根据起始点位姿，得到起始点关节角
51. targ\_ang=robot.ikine(T(:,:,20));                %根据终止点位姿，得到终止点关节角
53. %轨迹规划方法并生成动画
54. step = 20;
55. f=3;
56. figure(f);
57. [q ,qd, qdd]=jtraj(init\_ang,targ\_ang,step); %五次多项式轨迹，得到关节角度，角速度，角加速度，20为采样点个数
58. grid on
59. T=robot.fkine(q(:,:,:));                        %根据插值，得到末端执行器位姿
60. nT=T.T;
61. plot3(squeeze(nT(1,4,:)),squeeze(nT(2,4,:)),squeeze(nT(3,4,:)));%输出末端轨迹
62. title('输出末端轨迹');
64. w=3\*[-1 1 -1 1 -1 2];
65. v=[35 20];
66. robot.plot3d(q,'tilesize',0.1,'workspace',w,'path','.\stl','nowrist','view',v);             %动画演示
68. %% 求解位置、速度、加速度变化曲线
69. f = 4;
70. figure(f)
71. subplot(3,2,[1,3]);                     %subplot 对画面分区 三行两列 占用1到3的位置
72. plot3(squeeze(nT(1,4,:)),squeeze(nT(2,4,:)),squeeze(nT(3,4,:)));%输出末端轨迹
73. robot.plot3d(q,'tilesize',0.1,'workspace',w,'path','.\stl','nowrist','view',v);                         %动画演示
75. figure(f)
76. subplot(3, 2, 2);
77. **for** i = 1:6
78. plot(q(:,i));
79. hold on;
80. end
81. title('位置');
82. grid on;
84. figure(f)
85. subplot(3, 2, 3);
86. **for** i = 1:6
87. plot(qd(:,i));
88. hold on;
89. end
90. title('速度');
91. grid on;
93. figure(f)
94. subplot(3, 2, 4);
95. **for** i = 1:6
96. plot(qdd(:,i));
97. hold on;
98. end
99. title('加速度');
100. grid on;
102. t = robot.fkine(q);                 %运动学正解
103. rpy=tr2rpy(t);                          %t中提取位置（xyz）
104. figure(f)
105. subplot(3,2,5);
106. plot2(rpy);
108. %% ctraj规划轨迹 考虑末端执行器在两个笛卡尔位姿之间移动
109. f = 5
110. T0 = robot.fkine(init\_ang);         %运动学正解
111. T1 = robot.fkine(targ\_ang);         %运动学正解
113. Tc = ctraj(T0,T1,step);                 %得到每一步的T阵
115. tt = transl(Tc);
116. figure(f)
117. plot2(tt,'r');
118. title('直线轨迹');

4．雅各比矩阵子程序

1. clear;
2. clc;
3. % %机器人建模
4. th(1) = 0; d(1) = 0.2/2.8;   a(1) = -0.22/2.8;  alp(1) = pi/2;
5. th(2) = 0; d(2) = 0;     a(2) = -1.069/2.8; alp(2) = 0;
6. th(3) = 0; d(3) = 0.1/2.8;   a(3) = 0;      alp(3) = -pi/2;
7. th(4) = 0; d(4) = 1.04/2.8;  a(4) = 0;      alp(4) = pi/2;
8. th(5) = 0; d(5) = 0;     a(5) = 0;      alp(5) = pi/2;
9. th(6) = 0; d(6) = 0.495/2.8; a(6) = 0;      alp(6) = pi;
10. % DH parameters  th     d    a    alpha  sigma
11. L(1) = Link([th(1), d(1), a(1), alp(1), 0], 'revolute');
12. L(2) = Link([th(2), d(2), a(2), alp(2), 0], 'revolute');L(2).offset=-pi/2;
13. L(3) = Link([th(3), d(3), a(3), alp(3), 0], 'revolute');
14. L(4) = Link([th(4), d(4), a(4), alp(4), 0], 'revolute');L(4).offset=pi;
15. L(5) = Link([th(5), d(5), a(5), alp(5), 0], 'revolute');L(5).offset=-pi;
16. L(6) = Link([th(6), d(6), a(6), alp(6), 0], 'revolute');
18. robot = SerialLink(L); %SerialLink 类函数
19. robot.name='B09';
21. %速度雅可比检验
22. q0=[0 -pi/2 0 pi -pi 0];
23. J0=Jacob\_SDH(q0);
25. %速度雅可比
26. function [ J ] = Jacob\_SDH( q )
27. %JACOB\_SDH 函数摘要
28. %   输入q0为逼近角，单位为弧度，矩阵大小1\*6;
29. %   输出J为速度雅各比矩阵，矩阵大小6\*6；
30. %   利用向量积的方法求解系统的雅各比矩阵
31. %   此求解方法基于SDH参数建模
33. d=[0.2,0,0.1,1.04,0,0.495]./2.8;
34. a=[-0.22,-1.069,0,0,0,0]./2.8;
35. alp=[pi/2,0,-pi/2,pi/2,pi/2,pi];
36. offset=[0,-pi/2,0,pi,-pi,0];
37. thd=q+offset;
39. % 求各个关节间的变换矩阵
40. T0=trotz(0)\*transl(0,0,0)\*trotx(0)\*transl(0,0,0);
41. T1=trotz(thd(1))\*transl(0,0,d(1))\*trotx(alp(1))\*transl(a(1),0,0);
42. T2=trotz(thd(2))\*transl(0,0,d(2))\*trotx(alp(2))\*transl(a(2),0,0);
43. T3=trotz(thd(3))\*transl(0,0,d(3))\*trotx(alp(3))\*transl(a(3),0,0);
44. T4=trotz(thd(4))\*transl(0,0,d(4))\*trotx(alp(4))\*transl(a(4),0,0);
45. T5=trotz(thd(5))\*transl(0,0,d(5))\*trotx(alp(5))\*transl(a(5),0,0);
46. T6=trotz(thd(6))\*transl(0,0,d(6))\*trotx(alp(6))\*transl(a(6),0,0);
48. % 求各个关节相对于惯性坐标系的变换矩阵
49. T00 = T0;
50. T01 = T1;
51. T02 = T1\*T2;
52. T03 = T1\*T2\*T3;
53. T04 = T1\*T2\*T3\*T4;
54. T05 = T1\*T2\*T3\*T4\*T5;
55. T06 = T1\*T2\*T3\*T4\*T5\*T6;
57. % 求各个关节相对于末端坐标系的变换矩阵
58. T06 = T1\*T2\*T3\*T4\*T5\*T6;
59. T16 = T2\*T3\*T4\*T5\*T6;
60. T26 = T3\*T4\*T5\*T6;
61. T36 = T4\*T5\*T6;
62. T46 = T5\*T6;
63. T56 = T6;
65. % 提取各变换矩阵的旋转矩阵
66. R00 = t2r(T00);
67. R01 = t2r(T01);
68. R02 = t2r(T02);
69. R03 = t2r(T03);
70. R04 = t2r(T04);
71. R05 = t2r(T05);
72. R06 = t2r(T06);
74. % 取旋转矩阵第3列，即Z轴方向分量
75. Z0 = R00(: , 3);
76. Z1 = R01(: , 3);
77. Z2 = R02(: , 3);
78. Z3 = R03(: , 3);
79. Z4 = R04(: , 3);
80. Z5 = R05(: , 3);
81. Z6 = R06(: , 3);
83. % 求末端关节坐标系相对于前面各个坐标系的位置，即齐次变换矩阵的第四列
84. % pi6为坐标系i和末端坐标系的相对位置在坐标系i下的表示
85. P06 = T06(1:3, 4);
86. P16 = T16(1:3, 4);
87. P26 = T26(1:3, 4);
88. P36 = T36(1:3, 4);
89. P46 = T46(1:3, 4);
90. P56 = T56(1:3, 4);
91. P66 = [0; 0; 0];
93. % 使用向量积求出雅可比矩阵
94. % R0i为坐标系0到坐标系i的旋转矩阵
95. % R0i\*Pi6指坐标系i和末端坐标系的相对位置在0坐标系下的表示
96. J1 = [cross(Z0, R00\*P06); Z0];
97. J2 = [cross(Z1, R01\*P16); Z1];
98. J3 = [cross(Z2, R02\*P26); Z2];
99. J4 = [cross(Z3, R03\*P36); Z3];
100. J5 = [cross(Z4, R04\*P46); Z4];
101. J6 = [cross(Z5, R05\*P56); Z5];
103. J = [J1, J2, J3, J4, J5, J6];
105. end