

苏州大学

工业机器人及应用 课程 期末考核作业

考试形式：开卷

日期：2022 年 6 月 20日

学院：\_\_\_机电工程学院\_\_\_\_\_\_

专业：\_\_\_智能制造工程\_\_\_\_\_\_

年级：\_\_\_\_\_\_\_2019\_\_\_\_\_\_\_\_\_\_

学号：\_\_\_\_\_1929409018\_\_\_\_\_\_

姓名：\_\_\_\_\_\_ 孙晨兴\_\_\_\_\_\_\_

成绩：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

基于MATLAB Robotics Toolbox的六轴机械臂

轨迹规划与运动仿真

**一、实验目的**

基于机器人学理论知识，掌握标准D-H参数法建立关节型机器人的数学模型，并学会使用Matlab的Robotics Toolbox工具包搭建模型,完成机械臂运动行为仿真。实验要求如下：

1.基于机器人学理论知识，利用标准D-H参数法建立关节型机器人的数学模型。

2.使用Matlab的Robotics Toolbox工具包搭建模型，并介绍相关函数用法。

3.自定义障碍位置，利用A星算法完成避障的轨迹规划，实现六轴机械臂在障碍环境中的小球搬运。

4.利用matlab中的相关函数，完成机械臂运动行为仿真，绘制过程中关节运动的速度，加速度曲线，并生成gif文件。

**二、实验内容**

**1.本实验中使用的Matlab函数及用法**

**1）Link函数**

用于定义六轴机器人的一个轴。包含了机器人的运动学参数、动力学参数、刚体惯性矩参数、电机和传动参数；可采用DH法建立模型,其中包含参数：关节转角，关节距离，连杆长度，连杆转角，关节类型（0转动，1移动）。其中输人的参数顺序分别是关节角、连杆偏距、连杆长度、连杆转角、关节类型。

% 定义六轴机器人的一个轴

L(1) = Link([theta1, D1, A1, alpha1, offset1], 'standard')

**2)** **SerialLink()函数**

用于构建[机械臂](https://so.csdn.net/so/search?q=%E6%9C%BA%E6%A2%B0%E8%87%82&spm=1001.2101.3001.7020)，该机制是用Denavit-Hartenberg参数描述的，每个关节一组。主要参数如下：

links ：连杆向量（1×N）

gravity ：重力加速度 [gx gy gz]

base ：基座标系 （4×4其次变换形式）

tool：与基座标系的变换矩阵 （4×4其次变换形式）

qlim ：关节极限位置 （N×2）

offset ：关节偏移量 （N×1）

name ：机器人的名字

manuf ：制造者的名字

comment：注释

plotopt：plot（）方法选择（单元格数组）

fast：使用RNE的MEX版本，如果mex文件存在只能被设置为true，默认值为true

此外，它的类函数比较多，包括显示机器人、动力学、逆动力学、雅可比等。

% 创建一个六连杆机械臂

robot = SerialLink([L1,L2,L3,L4,L5,L6],'name','ABB IRB1200');

**3)fkine()正解函数**

用于求解出末端位姿p

tr = fkine(robot, q)，参数robot为一个机器人对象，tr为由q定义的每个前向运动学的正解。

**4)ikine()逆解函数**

采用数值迭代法求解逆运动学，及求解出关节角q。其中ikine函数的调用格式有三种：

q = ikine(robot,t);q = ikine(robot,t,q);q = ikine(robot,t,m)

参数robot为一个机器人对象，q为初始猜测点（默认为0），t为要反解的变换矩阵。当反解的机器人对象的自由度少于6时，要用m进行忽略某个关节自由度。

**5）ikunc()函数**

采用最优化法求解逆运动学，及求解出关节角q。

**6）teach()函数**

用于示教模式，驱动机器人运动。

**7）变换矩阵函数**

利用[MATLAB](https://so.csdn.net/so/search?q=MATLAB&spm=1001.2101.3001.7020" \t "_blank)中Robotics Toolbox工具箱中的transl、rotx、roty和rotz可以实现用齐次变换矩阵表示平移变换和旋转变换；rpy2tr可以实现欧拉角转齐次变换矩阵T，参数为对应的欧拉角度。

函数transl(a,b,c)创建了一个有平移但无旋转的相对位姿，函数rotx (alpha)则返回一个绕 x 轴旋转alpha角度的3×3旋转矩阵，而函数trotx (alpha)则返回一个绕 x 轴旋转alpha角度的 4 × 4 齐次变换矩阵：其旋转部分与rotx(alpha)相同，平移部分为零。

rpy2tr()返回欧拉角的齐次变换矩阵T。

同理roty、rotz绕 y、z轴旋转alpha角度的3×3旋转矩阵， troty和trotz返回一个绕 y、z 轴旋转alpha角度的4×4齐次变换矩阵。

% transl(0.5,0,0) %机器人在x轴方向平移了0.5米

% rotx(pi/4) %机器人绕x轴旋转45度，

**8）jtraj()函数**

**jtraj函数**用于规划关节空间轨迹，是五次[多项式](https://so.csdn.net/so/search?q=%E5%A4%9A%E9%A1%B9%E5%BC%8F&spm=1001.2101.3001.7020)插值函数。它的输出值有三项，分别为

qt：不同时间内各个关节的角度，

qdt，角速度，

qddt，角加速度。

输入参数：q0初始各个关节的角度。q1最终点的各个关节的角度，tv：可以理解为时间，也可以理解为步数。

**9)display()函数**

用于显示机器人信息，如D-H参数表、类型等。

**2.实验步骤**

**步骤1.标准DH参数法建立关节型机器人的数学模型**

为了描述机器人笛卡尔空间和关节空间的映射关系，广泛使用的是D-H（Denavit-Hartenberg）参数法，D-H参数法是一种对连杆的坐标描述，而关节机器人本质上就是一系列连杆通过关节连接起来而组成的空间开式运动链。对于开链式机构机器人，其可以看作由若干刚性杆件首尾相连而成。给机械臂的每一个杆件建立一个随体坐标系，并用齐次变换矩阵描述这些坐标系的相对位置和姿态，进而推导出机器人末端执行器相对于参考坐标系（一般取基座坐标系）齐次变换矩阵，得到机器人的运动学方程。

对于连杆本身，其功能在于保持其两端的关节轴线具有固定的几何关系，连杆的特性由轴线决定，通常用四个连杆参数来描述，连杆长度，连杆扭转角，连杆偏移量和关节角。

本次实验构建的机械臂以ABB的IRB1200为例；D-H参数表如下：



建立坐标系

采用D-H法建立坐标系，简述如下：

（1）确定Z轴：对于旋转关节，z轴按照右手螺旋定则旋转的方向，旋转角θ为关节变量；对于滑动关节，z轴为沿着直线运动的方向，连杆长度是关节变量；

（2）确定x轴：定义x\_n轴的方向为z(n-1)和zn轴的公垂线方向；有以下特别情况：若z(n-1)和zn平行，此时有无数条公垂线，选取与前一关节的公垂线共线的公垂线；

若z(n-1)和zn相交，此时没有公垂线，选取两条z轴叉积的方向作为x轴的方向。

按照上述方法，建立如下坐标系：

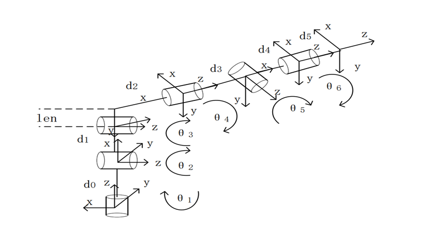


图1 机器人坐标系

再根据各个坐标系之间的传递矩阵可建立机器人手部坐标系到基座坐标系的变换矩阵T06即：T06=T01\*T12\*T23\*T34\*T45\*T56

根据上述变换矩阵关系可以求得机器人的正逆运动学模型，便于计算分析。

**步骤二、自定义障碍位置，利用A星算法完成避障的轨迹规划，实现六轴机械臂在障碍环境中的小球搬运。**

**1.A\*算法**：

启发式搜索:在状态空间中,对每-个要搜索的位置进行评估(估价),得到最好的位置。再以这个位置进行搜索,直到到达目标位置。这样可以减少搜索范围，降低问题复杂度,从而提高了效率。在启发式搜索中,对位置的估价,需要用到估价函数，这是十分重要的。采用了不同的估价函数可以得到不同的效果。

Aster算法是在Djkstra算法基础上发展出来的，是在静态路径中用于求解最优路径有效的直接搜索算法，比djkstra算法多了一个启发式的搜索函数，也就是通过一个代价函数来确定搜索方向(从起点开始向周围扩张，通过代价函数，计算得到周围每个节点的代价值，选出最小代价节点作为下一个扩展点，重复这个过程直到到达目标点。

A\*搜索算法是Greedy Best-First算法的改进，因为Greedy Best-First算法只考虑将预估距离（当前节点到目标点的预估距离）作为遴选开节点集的条件。即：

priority = heuristic(goal,next)

而A\*算法考虑将起始点到当前节点的实际路径耗散也作为遴选的条件，因此有：

priority = cost + heuristic(goal,next)

一般又记为：

f(x)= g(x) + h(x)

其中f(x)为计算出来的遴选值，g(x)为路径耗散，h(x)为预估距离，也称为启发函数，A\*搜索算法也是一种典型的启发式搜索算法。

**算法逻辑如下**：

开启列表(open list):存放待检查方格及其周围的状态为可移动的方格。

关闭列表(close list):存放开启列表中保存的且不需要再次检查的方格。

\* 初始化open\_list和close\_list；

\* 将起点加入open\_list中，并设置优先级为0（优先级最高）；

\* 如果open\_list不为空，则从open\_list中选取优先级最高的节点n：

\* 如果节点n为终点，则：

\* 从终点开始逐步追踪parent节点，一直达到起点；

\* 返回找到的结果路径，算法结束；

\* 如果节点n不是终点，则：

\* 将节点n从open\_list中删除，并加入close\_list中；

\* 遍历节点n所有的邻近节点：

\* 如果邻近节点m在close\_list中，则：

\* 跳过，选取下一个邻近节点

\* 如果m不在open\_list中:

\* 设置节点m的parent为节点n

\* 计算节点m的优先级f=g+h

\* 将节点m加入open\_list中

\* 如果m已在open\_list中:

\* 重新计算m.g, int new\_g=n.g+m.price;

\* 如果new\_g < old\_g :

\* 设置节点m的parent为节点n

\* 更新m.g和m.f

\* 否则:

\* 跳过，选取下一个邻近节点

**3.障碍物**

本次实验为了更好地模拟A\*算法，在不影响机器人工作空间的前提下设置了自定义障碍，自定义障碍如下：两个大球和两个圆柱体，。

参数如下：

两个大球的中心坐标：（1，2，-1）,(7.5，5，7)半径均为1；

两个圆柱体的底面圆心坐标：（1，6,0），（5，-1,0）底面半径和高分别为2,3；

障碍物生成图：

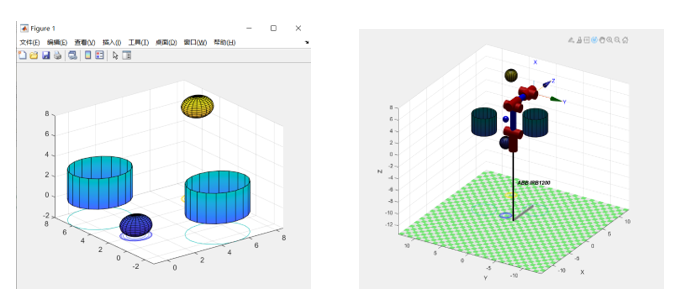


图2 障碍物布置图 图3 机械臂与障碍空间图

图中蓝色实心小球即为目标搬运小球，另外两个圆柱体和大球分别为自定义障碍。

4.目标小球搬运思路

目标搬运小球初始坐标为start=[1，2，3];目标搬运位置goal=[5，5，5];

为了方便起见，本次实验中，自定义障碍物的设置不会影响机械臂从初始位置到小球start位置，也不会影响机械臂搬运完小球以后，从goal位置回到机器人初始位位置的归位运动，因此，只需要在小球搬运过程中对机械臂使用A\*算法进行避障路径规划。

搬运思路：

第一步，机械臂移动到目标小球。根据目标小球的空间位置，使用变换矩阵函数transl、rpy2tr函数返回出对应的齐次变换矩阵T，再利用ikunc函数反解出该次变换矩阵对应的关节变量，在机器人初始关节变量与目标关节变量之间进行关节空间的轨迹插补。

第二步：小球的搬运。利用A\*算法完成对目标小球从start到goal之间的轨迹规划，并对路径上个点做记录，进行插补，完成机械臂关节空间的轨迹插补。如下图红色即为小球避障规划的搬运轨迹

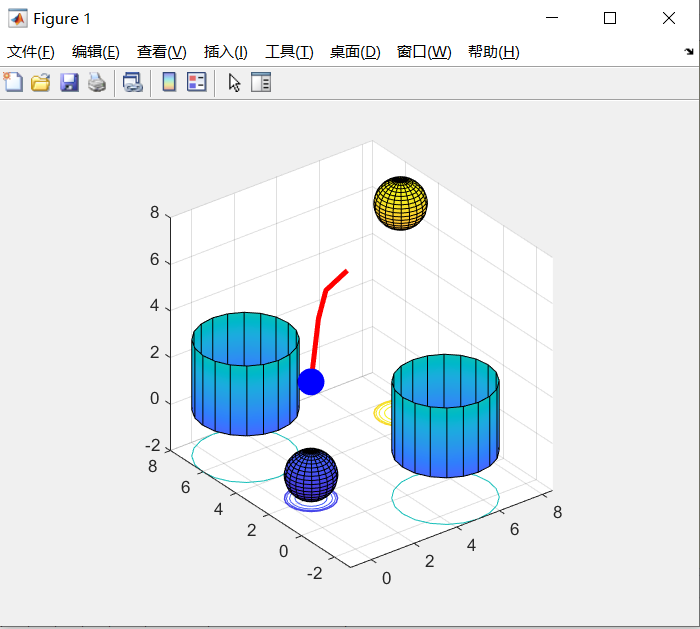


图4 轨迹图

第三步：机械臂归位。从goal位置出发，使机械臂回到初始状态，先算出goal位置对应的其次变换矩阵T，再利用ikunc函数反解出该次变换矩阵对应的关节变量，在机器人goal位置对应的关节变量与机器人初始位置对应的关节变量之间进行关节空间的轨迹插补。

**步骤三、绘制过程中关节运动的速度，加速度曲线**

利用[MATLAB](https://so.csdn.net/so/search?q=MATLAB&spm=1001.2101.3001.7020" \t "_blank)中Robotics Toolbox工具箱中的**jtraj函数可以求解机械臂在运动时的速度与加速度。**

**jtraj函数**用于规划关节空间轨迹，是五次[多项式](https://so.csdn.net/so/search?q=%E5%A4%9A%E9%A1%B9%E5%BC%8F&spm=1001.2101.3001.7020)插值函数。它的输出值有三项，分别为

qt：不同时间内各个关节的角度，

qdt，角速度，

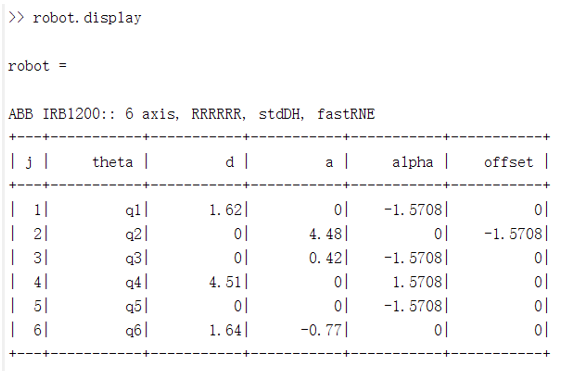
qddt，角加速度。

输入参数：q0初始各个关节的角度。q1最终点的各个关节的角度，tv：可以理解为时间，也可以理解为步数。

因此，利用**jtraj函数的返回值可以直接求出关节型机械臂在关节空间运动时的角速度与角加速度。**

**三、实验结果**

输出的机械臂D-H参数表



利用标准D-H参数法建立关节型机器人的数学模型如下图所示：

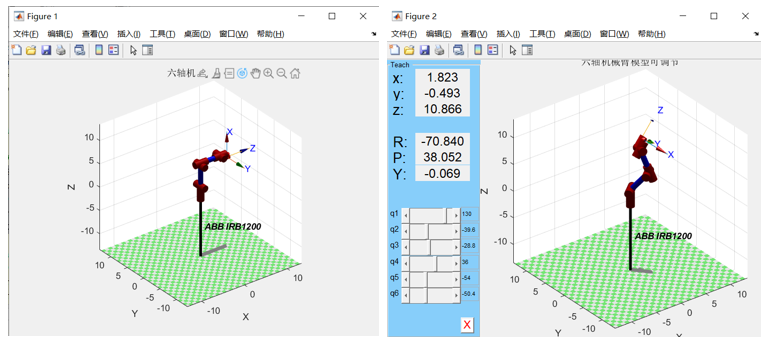


图5 机械臂空间图 图6 示教模式

上图中，左图为关节型机器人的三维立体图形，右图是利用teach函数的示教模式，可以在机械臂关节空间范围内驱动机器人运动，完成示教。

搬运小球动画：

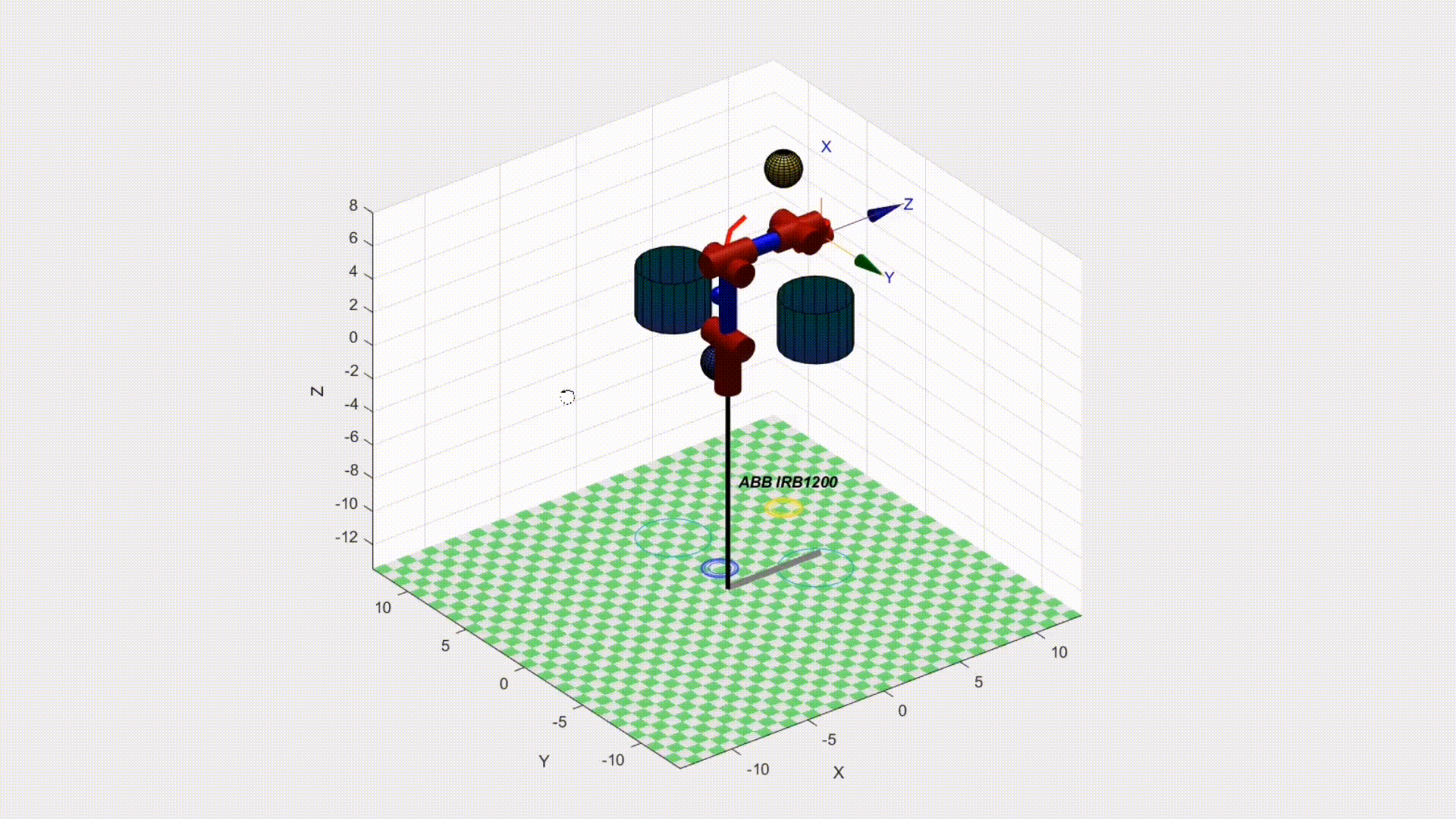


图7 搬运动画

机械臂运动过程中的关节变量位置、速度、加速度曲线变化图

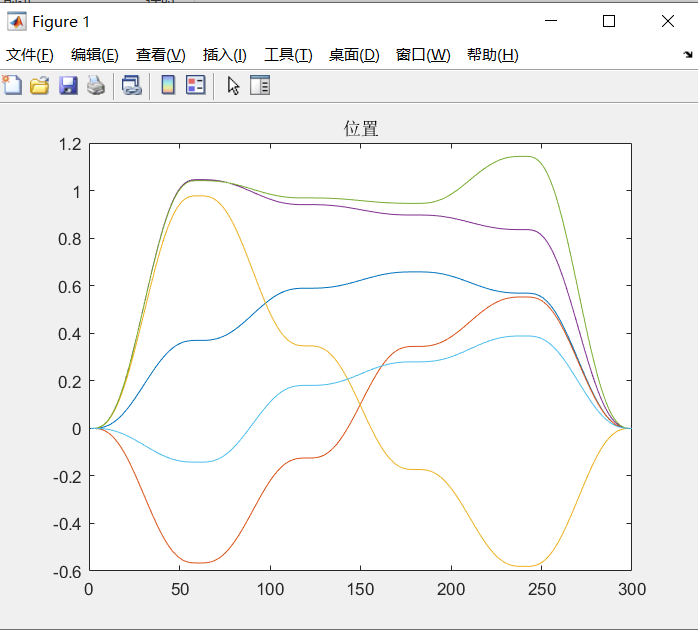
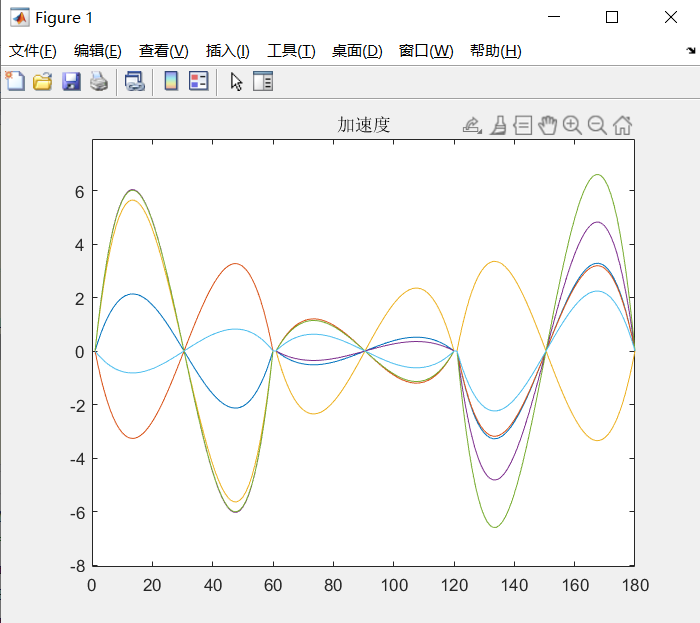
 

图8 关节位置变化曲线 图9 关节速度变化曲线

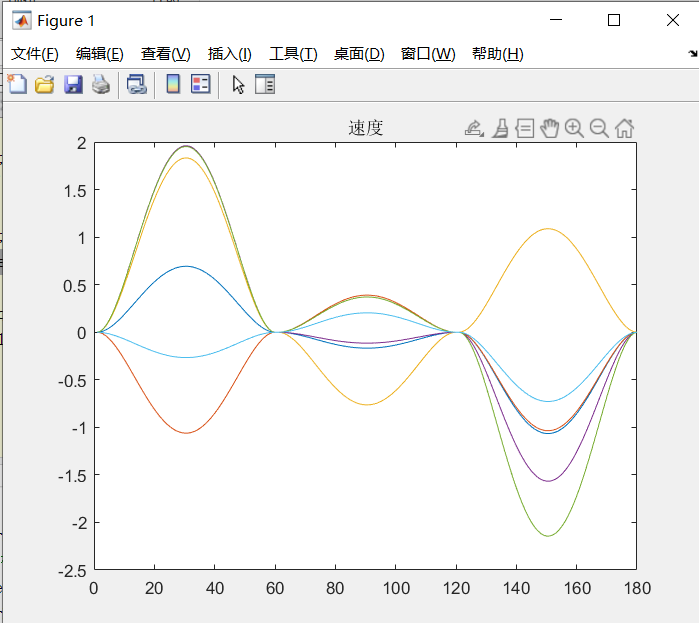


图10 关节加速度变化曲线

本次路径规划由机械臂初始状态、小球起点、小球目标点、再回到机械臂初始状态组成，其中小球起点与小球目标点之间有由A\*算法算出的路径点组成，在每两个点之间进行五次样条插补，将插补得到的关节角度组合起来，便得到机械臂运动的全过程因此速度与加速度会有相应的变化，但是角度位置变化比较平缓。

**总结**：本次实验通过Matlab的Robotics Toolbox工具包完成了机械臂行为运动的仿真，对机械臂的正逆运动学，轨迹规划插补，行为仿真都有了清晰的认识，也感受到了Matlab的Robotics Toolbox工具包的便捷之处。

代码如下：

clear;

close all;

clc

%% 实验一 利用标准D-H参数法建立关节型机器人的数学模型

%角度转换

angle=pi/180; %度

% DH法建立模型,关节转角，关节距离，连杆长度，连杆转角

L1=Link('d',1.62,'a',0, 'alpha',-pi/2);

L2=Link('d',0, 'a',4.48,'alpha',0, 'offset',-pi/2);

L3=Link('d',0, 'a',0.42, 'alpha',-pi/2);

L4=Link('d',4.51,'a',0, 'alpha',pi/2);

L5=Link('d',0, 'a',0, 'alpha',-pi/2);

L6=Link('d',1.64,'a',-0.77,'alpha',0);%机器人连杆参数

% 定义关节范围

L1.qlim =[-180\*angle, 180\*angle];

L2.qlim =[-180\*angle, 180\*angle];

L3.qlim =[-180\*angle, 180\*angle];

L4.qlim =[-180\*angle, 180\*angle];

L5.qlim =[-180\*angle, 180\*angle];

L6.qlim =[-180\*angle, 180\*angle];

%显示DH参数

%robot.display

% 显示机械臂

robot = SerialLink([L1,L2,L3,L4,L5,L6],'name','ABB IRB1200');

initial\_p = [0 0 0 0 0 0];%初始关节角度

figure(1)

view(3);robot.plot(initial\_p);

title('六轴机械臂模型');

% 加入teach指令，则可调整各个关节角度

figure(2)

view(3);robot.plot(initial\_p);

view(3);robot.teach

title('六轴机械臂模型可调节');

% 正解，给定关节角，求末端位姿

Theta=15\*ones(1,6);

% 利用逆解函数验证正逆解的结果

Theta=Theta/180\*pi; %换算成弧度

T=robot.fkine(Theta) %求正解的齐次变换矩阵

q1=robot.ikine(T)\*180/pi %求逆解的关节角 验证

%% 实验二 自定义障碍位置，利用A星算法完成避障的轨迹规划，实现六轴机械臂在障碍环境中的小球搬运。

% A\* algorithm实现

%自定义障碍物参数

obstacleMatrix=[[1 2 -1];[7.5 5 7]];

RobstacleMatrix =[1;1];

cylinderMatrix = [[1 6];[5,-1]];

cylinderRMatrix=[2;2];

cylinderHMatrix = [3;3];

start = [1 2 3];

goal = [5 5 5];

[numberOfSphere, ~] = size(obstacleMatrix);

[numberOfCylinder, ~] = size(cylinderMatrix);

Alldirec = [[1,0,0];[0,1,0];[0,0,1];[-1,0,0];[0,-1,0];[0,0,-1];...

[1,1,0];[1,0,1];[0,1,1];[-1,-1,0];[-1,0,-1];[0,-1,-1];...

[1,-1,0];[-1,1,0];[1,0,-1];[-1,0,1];[0,1,-1];[0,-1,1];...

[1,1,1];[-1,-1,-1];[1,-1,-1];[-1,1,-1];[-1,-1,1];[1,1,-1];...

[1,-1,1];[-1,1,1]];

threshold = 1.5;

stop = threshold\*1.5;

g = [start, 0; goal,inf]; % 每一行前三个数为点坐标，第四个数为路径耗散

Path = [];

Parent = [];

Open = [start, g(findIndex(g,start),4) + getDist(start,goal)];

% 绘制障碍环境

figure(1)

[n,~] = size(obstacleMatrix);

for i = 1:n %绘制静态球障碍物

[x,y,z] = sphere();

surfc(RobstacleMatrix(i)\*x+obstacleMatrix(i,1),...

RobstacleMatrix(i)\*y+obstacleMatrix(i,2),...

RobstacleMatrix(i)\*z+obstacleMatrix(i,3));

hold on;

end

[n,~] = size(cylinderMatrix);

for i = 1:n %绘制圆柱体障碍物

[x,y,z] = cylinder(cylinderRMatrix(i));

z(2,:) = cylinderHMatrix(i);

surfc(x + cylinderMatrix(i,1),y + cylinderMatrix(i,2),...

z,'FaceColor','interp');

hold on;

end

r1 = 0.5;

plot\_sphere(start,r1,'b');

%bar1 = scatter3(start(1),start(2),start(3),80,'cyan','filled','o');hold on

%bar2 = scatter3(goal(1),goal(2),goal(3),80,'magenta','filled','o');

%axis equal

%set(gcf,'unit','centimeters','position',[30 10 20 15]);

% 主循环

while ~isempty(Open)

[xi, index] = findMin(Open);

Open(index,:) = [];

if getDist(xi, goal) < stop

break;

end

children = getChildren(xi, Alldirec, threshold, obstacleMatrix, RobstacleMatrix,...

cylinderMatrix, cylinderRMatrix, cylinderHMatrix);

%scatter3(children(:,1),children(:,2),children(:,3),10,'filled','o');

drawnow;

[n,~] = size(children);

for i = 1:n

child = children(i,:);

if findIndex(g, child) == 0 % child不在g

g = [g; child, inf];

end

a = g(findIndex(g, xi),4) + getDist(xi,child);

if a < g(findIndex(g, child),4)

g(findIndex(g, child),4) = a;

Parent = setParent(Parent, child,xi);

Open = setOpen(Open, child, a, goal);

end

end

end

lastPoint = xi;

% 回溯轨迹

x = lastPoint;

Path = x;

[n,~] = size(Parent);

while any(x ~= start)

for i = 1:n

if Parent(i,1:3) == x

Path = [Parent(i,4:6); Path];

break;

end

end

x = Parent(i,4:6);

end

plot3([Path(:,1);goal(1)],[Path(:,2);goal(2)],[Path(:,3);goal(3)],'LineWidth',3,'color','r');

% 计算轨迹距离

pathLength = 0;

[n,~] = size(Path);

for i = 1:n-1

pathLength = pathLength + getDist(Path(i,:),Path(i+1,:));

end

pathLength = pathLength + getDist(Path(end,:),goal);

fprintf('路径的长度为:%f',pathLength);

%%机器人搬运

robot = SerialLink([L1,L2,L3,L4,L5,L6],'name','ABB IRB1200');

%目标小球

r1 = 0.5;

plot\_sphere(start,r1,'b');

view(3);robot.plot([0 0 0 0 0 0]);

%轨迹插补

[a0, ~]=size(Path);

q=[]

q(a0+1,:)=robot.ikunc(transl(goal)\*rpy2tr(180,0,0));

for i=1:a0

q(i,:)=robot.ikunc(transl(Path(i,:))\*rpy2tr(180,0,0));

end

[a1,~]=size(q);

q\_all=[]

for i=1:a1-1

[q\_all(60\*(i-1)+1:60\*i,:) qd qdd]=jtraj(q(i,:),q(i+1,:),60);

end

%robot.plot(q\_all)

%动画演示

%机械臂由初始位置移动到小球

q0=[0 0 0 0 0 0]

T1 =transl(start)\*rpy2tr(180,0,0);

q1 = robot.ikunc(T1);

[q\_1,qd\_1,qdd\_1] = jtraj(q0,q1,60);

view(3);robot.plot(q\_1)

[q\_3,qd\_3,qdd\_3] = jtraj(q\_all(180,:),q0,60);

q = [q\_all;q\_3];

%搬运小球至目标位置并归位

for i = 1:240

qi = q(i,:);

if i < 181

Ti = robot.fkine(qi);

Pi = transl(Ti);

plot\_sphere(Pi,r1,'b');

[n,~] = size(obstacleMatrix);

for i = 1:n %绘制静态球障碍物

[x,y,z] = sphere();

surfc(RobstacleMatrix(i)\*x+obstacleMatrix(i,1),...

RobstacleMatrix(i)\*y+obstacleMatrix(i,2),...

RobstacleMatrix(i)\*z+obstacleMatrix(i,3));

hold on;

end

[n,~] = size(cylinderMatrix);

for i = 1:n %绘制圆柱体障碍物

[x,y,z] = cylinder(cylinderRMatrix(i));

z(2,:) = cylinderHMatrix(i);

surfc(x + cylinderMatrix(i,1),y + cylinderMatrix(i,2),...

z,'FaceColor','interp');

hold on;

end

robot.plot(qi)

cla

else

[n,~] = size(obstacleMatrix);

for i = 1:n %绘制静态球障碍物

[x,y,z] = sphere();

surfc(RobstacleMatrix(i)\*x+obstacleMatrix(i,1),...

RobstacleMatrix(i)\*y+obstacleMatrix(i,2),...

RobstacleMatrix(i)\*z+obstacleMatrix(i,3));

hold on;

end

[n,~] = size(cylinderMatrix);

for i = 1:n %绘制圆柱体障碍物

[x,y,z] = cylinder(cylinderRMatrix(i));

z(2,:) = cylinderHMatrix(i);

surfc(x + cylinderMatrix(i,1),y + cylinderMatrix(i,2),...

z,'FaceColor','interp');

hold on;

end

plot\_sphere(goal,r1,'b');

robot.plot(qi)

cla

end

end

%T2 =transl(goal)\*rpy2tr(90,90,0);

%q2 = robot.ikunc(T2);

%t = 30;

%q\_2 = jtraj(q1,q2,t);

%[q\_3,qd\_3,qdd\_3] = jtraj(q\_all(180,:),q0,60);

%plot\_sphere(goal,r1,'b');

%view(3);robot.plot(q\_3)

%% 实验三 完成机械臂运动行为仿真，绘制过程中关节运动的速度，加速度曲线，并生成gif文件。

%绘制过程中关节运动的速度，加速度曲线

%关节变量位置

q = [q\_1;q\_all;q\_3];

plot(q);title('位置');

%关节变量速度

q\_d = [qd\_1;qd;qd\_3];

plot(q\_d);title('速度');

%关节变量加速度

q\_dd = [qdd\_1;qdd;qdd\_3];

plot(q\_dd);title('加速度');

注：以上代码所调用的自定义函数参见文件夹。

%% 函数

function children = getChildren(pos, Alldirec, step,circleCenter,circleR, cylinderCenter,cylinderR, cylinderH)

allchild = [];

[n,~] = size(Alldirec);

for i = 1:n

direc = Alldirec(i,:);

child = pos + direc \* step;

if ~checkCol(child, circleCenter,circleR, cylinderCenter,cylinderR, cylinderH)

continue;

end

allchild = [allchild; child];

end

children = allchild;

end

function flag = checkCol(pos, circleCenter,circleR, cylinderCenter,cylinderR, cylinderH)

[numberOfSphere, ~] = size(circleCenter);

[numberOfCylinder, ~] = size(cylinderCenter);

flag = true;

for i = 1:numberOfSphere

if getDist(pos, circleCenter(i,:)) <= circleR(i)

flag = false;

break;

end

end

for i = 1:numberOfCylinder

if getDist(pos(1:2), cylinderCenter(i,:)) <= cylinderR(i) && pos(3) <= cylinderH(i)

flag = false;

break;

end

end

if pos(3) <= 0 flag = false; end

end

function Par = setParent(Parent, xj, xi)

[n,~] = size(Parent);

if n == 0

Par = [xj, xi];

else

for i = 1:n

if Parent(i,1:3) == xj

Parent(i,4:6) = xi;

Par = Parent;

break;

end

if i == n

Par = [Parent; xj, xi];

end

end

end

end

function Ope = setOpen(Open, child, a, goal)

[n,~] = size(Open);

if n == 0

Ope = [child, a + getDist(child, goal)];

else

for i = 1:n

if Open(i,1:3) == child

Open(i,4) = a + getDist(child, goal);

Ope = Open;

end

if i == n

Ope = [Open; child, a + getDist(child, goal)];

end

end

end

end

function h = heuristic(pos, goal)

h = max([abs(goal(1) - pos(1)),abs(goal(2) - pos(2)),abs(goal(3) - pos(3))]);

end

function index = findIndex(g, pos)

[n,~] = size(g);

index = 0; % 表示没有找到索引

for i = 1:n

if g(i,1:3) == pos

index = i; % 索引为i

break;

end

end

end

function d = getDist(x,y)

d = sqrt(sum((x - y).^2));

end

function [pos, index] = findMin(Open)

[~,index] = min(Open(:,4));

pos = Open(index,1:3);

end