本科生课程设计

题	目：	机械臂路径规划与仿真
专	业：	智能科学与技术
日	期：	2023年7月12日

机械臂路径规划与控制

欧阳家睿* 邓伟亮* 林佳盈* 吕韦辰* 邓伟亮* 周德峰* 李孟棠† 彭键清†

*对本文章有同等贡献

摘 要： 本篇报告介绍了我们小组完成机械臂路径规划与控制这个大作业的整个过程。我们在基于实验手册的基础上，对于每个函数充分理解，设计了控制器，制定了实验思路，完成了主函数。我们依次进行了轨迹生成、状态更新，最终在CoppeliaSim软件中进行模拟，完成了任务。

关键词： 机械臂、多项式规划、控制器、CoppeliaSim仿真

1 函数介绍

1.1 trajectoryGenerator.m轨迹生成

输入：

- 初始时刻末端执行器位形 Tseinit
- 期望末端执行器位形 Tsegoal

输出： Tseinit 到 Tsegoal 之间的轨迹result_traj(1×200cell)，其中每个cell都是4×4 double的齐次变换矩阵。

result_traj{1, 1}

1	2	3	4
-1	0	0	0.8170
0	0	1	0.1910
0	1	0	-0.0060
0	0	0	1

1.1.1 step1:Traj Generate

1. ScrewTrajectory轨迹生成函数

   代码及其注释如下：

```
function traj = ScrewTrajectory(Xstart, Xend, Tf, N, method)
    % 计算每个轨迹段之间的时间间隔
    timegap = Tf / (N - 1);
    % 初始化轨迹的单元数组
    traj = cell(1, N);
    % 为每个时间步生成轨迹
    for i = 1: N
        % 根据输入的方法选择适当的时间缩放方法
        if method == 3
            % 使用三次多项式时间缩放生成路径参数
            s = CubicTimeScaling(Tf, timegap * (i - 1));
        else
            % 使用五次多项式时间缩放生成路径参数
            s = QuinticTimeScaling(Tf, timegap * (i - 1));
        end
        % 计算当前时间步的轨迹
```

```
        traj{i} = Xstart * MatrixExp6(MatrixLog6(TransInv(Xstart) * Xend) * s);
    end
end
```

- **功能：**实现一个螺旋轨迹生成函数，用于生成从起始位姿Xstart到目标位姿Xend的平滑轨迹。
- **输入：**初始末端位形Xstart, 期望末端执行器位形Xend, 轨迹持续时间Tf, 轨迹步数N, 轨迹生成方法method（本函数中可用三次多项式、五次多项式规划）
- **输出：**返回一个轨迹traj，其中包含了每个时间步的位姿信息。

代码具体而言，

(a) 该函数通过将整个运动时间Tf等分为N个时间段，每个时间段的时间间隔为$timegap = Tf/(N-1)$。

(b) 通过循环从起始位姿到目标位姿逐步生成轨迹的各个时间步。

在每个时间步中，根据选择的轨迹生成方法method，分别调用CubicTimeScaling或QuinticTimeScaling函数来生成路径参数s。路径参数s控制了当前时间步的位姿在整个运动过程中的位置。

(c) 通过使用MatrixLog6、MatrixExp6和TransInv等函数对位姿变换矩阵进行计算，得到当前时间步的位姿值，并将其存储在轨迹traj的对应位置。

2. CubicTimeScaling三次多项式规划

代码如下：

```
function s = CubicTimeScaling(Tf, t)
  s = 3 * (t / Tf) ^ 2 - 2 * (t / Tf) ^ 3;
end
```

- 功能：实现了一个轨迹生成函数，用于生成一个满足特定条件的三次多项式轨迹。
- 输入：总运动时间Tf和当前时间t
- 输出：路径参数s。

具体而言，该函数生成的轨迹是一个**从静止开始到静止结束**的运动，其运动曲线由一个三次多项式定义。该三次多项式由以下公式(1)给出：

$$s = 3(\frac{t}{Tf})^2 - 2(\frac{t}{Tf})^3 \tag{1}$$

其中，t表示当前时间，Tf表示总运动时间。该公式中的t / Tf表示时间的归一化比例，将时间映射到区间[0, 1]上。通过对归一化时间的平方和立方进行线性组合，生成了满足起始和结束速度为零的平滑轨迹。

例如，当Tf = 2、t = 0.6时，调用该函数将返回s = 0.2160，表示在运动过程中当前时间为0.6时，对应的路径参数值为0.2160。

3. QuinticTimeScaling五次多项式规划

代码如下：

```
function s = QuinticTimeScaling(Tf, t)
  s = 10 * (t / Tf) ^ 3 - 15 * (t / Tf) ^ 4 + 6 * (t / Tf) ^ 5;
end
```

- 功能：实现了一个轨迹生成函数，用于生成一个满足特定条件的五次多项式轨迹。
- 输入：总运动时间Tf和当前时间t
- 输出：路径参数s

该五次多项式由以下公式(2)给出：

$$s = 10(\frac{t}{Tf})^3 - 15(\frac{t}{Tf})^4 + 6(\frac{t}{Tf})^5 \tag{2}$$

这个函数原理同CubicTimeScaling，输出值s会随着时间t的增加而变化，从0逐渐增加到1。

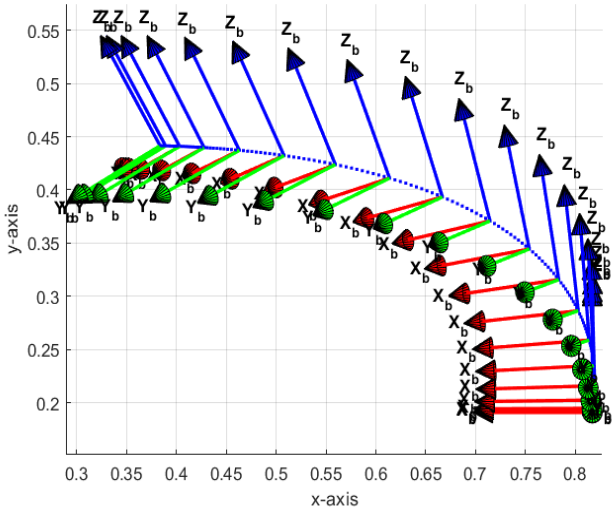
1.1.2 step2:Traj Plot

代码及注释见下：

```
% 计算结果轨迹数据的大小
n = size(result_traj,2);

% 创建大小为1×n的数组x、y和z，并初始化为0
x = zeros(1,n);
y = x;
z = x;

% 遍历结果轨迹数据中的每个元素，提取x、y、z坐标并存储到相应的数组中
for i = 1:n
    x(1,i) = result_traj{i}(1,4);
    y(1,i) = result_traj{i}(2,4);
    z(1,i) = result_traj{i}(3,4);
end

% 创建一个名为figure(1)的图形窗口，并在其上绘制三维散点图
figure(1);
hold on;
plot3(x,y,z,'b.');

% 打开网格线显示
grid on;

% 每隔10个数据点，获取当前点的旋转矩阵Rot和位置向量Pos，
% 并在图形中绘制当前点的坐标系
for i = 1:10:n
    Rot = result_traj{i}(1:3,1:3);
    Pos = [x(i); y(i); z(i)];
    draw_coordinate_system(0.1,Rot,Pos,'rgb','{b}');
end

% 设置坐标轴刻度尺寸一致
axis equal;

% 设置x、y和z轴的标签
xlabel('x-axis');
ylabel('y-axis');
zlabel('z-axis');

```

```
40 % 结束绘图
41 hold off;
```

按报告初始化后，得到轨迹：



1.2 nextState状态更新

具体代码见下：

```
1 function [next_conf] = nextState(current_conf, control, dt)
2 % configuration of the robot arm: theta1 ~ theta6
3 next_conf = current_conf + control'*dt; % control is a column vector,
4 %                                 current_conf and next_conf are row vector.
5 end
```

功能：状态转移函数，用于计算机器人臂的下一个状态（next_conf）。

输入：

- current_conf：表示当前机器人臂的配置，包含了6个关节角度（theta1 ~ theta6）。它是一个行向量。
- control：表示控制信号，它是一个列向量。这个控制信号影响机器人臂的运动。具体来说，它可能代表每个关节的速度或加速度。
- dt：表示时间步长，即离散化的时间间隔。

输出：下一个状态（next_conf）

具体来说，函数内部通过以下公式(3)来计算下一个状态（next_conf）：

$$next\_conf = current\_conf + control' * dt \tag{3}$$

其中，control'表示control的转置，是为了保持与current_conf相同的维度。

原理：根据**数值积分方法**来计算下一个状态。在欧拉法中，我们假设在很小的时间间隔 dt 内，系统的变化速度保持恒定。因此，我们可以将控制信号与时间步长相乘，得到一个增量，并将其加到当前状态上，从而得到下一个状态。

1.3 controller.m控制器

输入：

- X：这是当前实际末端执行器配置。在文档中写为 Tse。
- Xd：这是当前末端执行器参考配置，在文档中写为 Tse,d。
- Xdnext：这是参考轨迹在下一个时间步的末端执行器参考配置，在文档中写为Tse,d,next，它在之后的Delta t 时间里。
- Kp 和 Ki - 这是比例（P）和积分（I）增益矩阵，用于PI控制器。
- dt：这是参考轨迹配置之间的时间步长Delta t。
- thetalist：这是对应的关节角度列表。

输出：

- control：这是命令的末端执行器扭转，用末端执行器的帧 {e} 表示。
- Xerr：用于绘图的 Xerr 历史记录。

控制器的具体流程如下：

首先是Vd和Vb。Vd是参考轨迹的导数，用于前馈控制，而Vb则是在末端执行器坐标系中的控制输入，是通过前馈控制与比例-积分反馈控制的组合来计算得出的。接着，定义Blist，它是一个关于机器人臂各关节扭矩的列表。这个列表依赖于机器人臂的具体设计参数，之后计算Jb，这是机器人臂的雅可比矩阵。它描述了每个关节角度变化对末端执行器位置和方向的影响。最后通过伪逆（pseudo-inverse）的方式，利用Vb和Jb来计算控制输入control。这个控制输入就是要施加到机器人臂各个关节上的扭矩和力的集合，以便于实现末端执行器从当前位置 X 移动到目标位置 Xd。

我们编写的controller通过PI反馈和前馈控制，生成了控制机器人臂的输入。既允许机器人臂跟踪复杂的路径，同时又提供了良好的误差补偿和鲁棒性。

1.4 main.m主函数

主函数的具体流程如下：

1. 运行trajectoryGenerator.m脚本以生成一个期望的轨迹。result_traj将会存储这个生成的轨迹。
2. 定义机器人的物理参数，然后生成了两个列表Slist和Blist，描述了机器人关节和连杆的配置。同时还生成了末端执行器在基坐标系下的初始位姿M。
3. 初始化PI控制器的比例和积分增益矩阵Kp和Ki，设定了模拟的时间步长dt，并创建了几个用于存储结果的矩阵result_conf, result_Xerr, result_control。
4. 设定初始的末端执行器的位姿X、初始的参考位姿Xd和下一个参考位姿Xdnext，并且初始化了关节角thetalist和current_conf。
5. 在这个循环中，对每一个时间步进行以下操作：

   调用之前定义的controller函数来计算当前的控制输入control，并获得当前的位姿误差Xerr。使用nextState函数来计算机器人的下一个配置next_conf。将这一步的位姿误差、配置和控制输入分别存储到result_Xerr、result_conf和result_control中。根据新的配置，计算末端执行器在空间中的新位置X。更新参考轨迹Xd和Xdnext，以及当前的配置current_conf。

6. 最后，代码会打印出一条消息，表示配置的计算已经完成。接着，它将result_conf和result_Xerr存储到CSV文件中，这样可以方便之后进行数据分析和可视化。最后再打印一条消息，表示CSV文件写入已经完成。

2 总体思路

我们经过研究作业要求后制定了以下实验思路：

1. 首先，研究轨迹生成函数ScrewTrajectory的使用方法。该函数能够生成给定预定的轨迹持续时间和步数，并采用五次多项式规划方法。函数的输出为轨迹中每个相邻点之间的齐次变换矩阵。

2. 接下来，设计位形计算函数。该函数接收当前机器人的各个关节角、各个关节角的控制量以及采样时间dt作为输入，并能够计算出下一个机器人的形态。

3. 然后，设计控制器。采用前馈控制加PI控制的方法。首先根据当前的关节角计算出末端执行器的当前位形，然后利用规划的轨迹得到期望的位形。接下来设计PI控制器，根据误差计算出控制量，并作用于关节角以获得下一个形态的关节角。这个过程通过迭代实现控制器的连续调节。

4. 在主函数中，首先确定起点和终点的位姿。然后调用轨迹生成函数，并设置PI控制器的增益。设计一个循环，以时间间隔δt对机器人进行更新，并记录所有状态量。最后将输出结果保存为.csv文件。

5. 最后，将生成的.csv文件导入CoppeliaSim，以实现机器臂的运动。

3 仿真结果

仿真结果参见随附件视频。

4 问题发现与解决

1. **仿真时位形无变化**

   解决：初步怀疑是代码问题，先检查经过代码仔细排查后，发现在流程控制部分没有更新对应的thetalist

   ```
   % 读取当前的关节角度
   thetalist = current_conf';  %更新thetalist
   % 调用PI控制器
   [control, Xerr] = controller(X, Xd, Xdnext, Kp, Ki, dt, thetalist);
   ```

2. **位形变化速度异常**

   解决：在仿真模拟过程中，出现过机器人位形以较快速度变化的情况，在多次排查后，我们发现是采样时间`dt`设置的问题，我们将dt从0.01设置成了0.1，极大的改善了该情况

3. **模型异常抖动/不收敛下的参数调整**

   解决：经过多次调参，我们发现当参数设置过大时，模型期望位形难以收敛，因此需要控制模型参数设置。

   同时，相关参数N，TF对最终模型收敛和表现情况影响不大。

4. **未到正确位形**

   解决：在每次迭代时，只改变那个关节角，也就是6*1的矩阵，所以我们的所有计算的T，都必须在初始形态下，利用原始的旋量矩阵，因此不能用X去迭代，只能用Tseini

   ```
   % 原来是X = FKinSpace(X,Slist,thetalist);
   X = FKinSpace(Tseini,Slistthetalist);
   ```

5 总结

在本次实验中，我们通过编写函数实现了各控制步骤的算法，包括轨迹规划及生成，位形计算，还有PI控制器反馈控制，实现了对UR5机械臂的路路径规划和轨迹控制，同时，还将机器人的关节位形保存下来，在 CoppeliaSim实现仿真。

通过此次实验，我们学习了机器人控制的详细步骤和流程。通过实践，对机械臂型机器人的控制有了更进一步的认识与掌握，为我们后续学习机器人控制，轨迹规划打下了坚实的基础。