廣東工業大學

课程设计报告

课程名称 机器人学

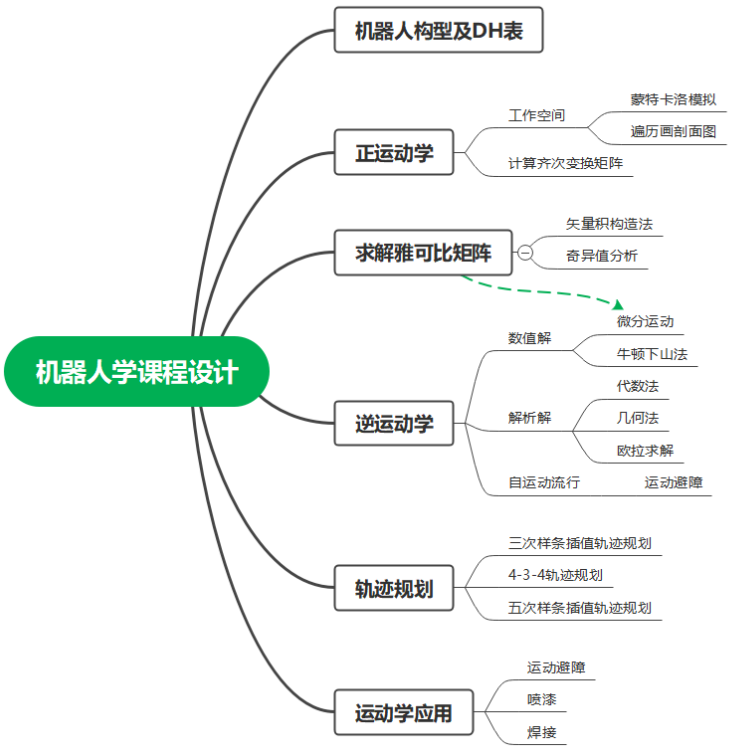
学　　院 粤港机器人学院

专业班级 19 机器人 1 班

2021 年 12 月 31 日

目录

项目总览与分工


机器人学课程设计
机器人构型及DH表
正运动学
工作空间
蒙特卡洛模拟
遍历画削面图
计算齐次变换矩阵
求解雅可比矩阵
矢量积构造法
奇异值分析
逆运动学
数值解
微分运动
牛顿下山法
解析解
代数法
几何法
欧拉求解
自运动流行
运动避障
轨迹规划
三次样条插值轨迹规划
4-3-4轨迹规划
五次样条插值轨迹规划
运动学应用
运动避障
喷漆
焊接

1.七自由度机器人设计

如下图为我设定的 7 自由度机器人的基本构型，其中 3 号关节为平移伸缩关节，其余关节为旋转关节，连杆之间长度依次标注如下图。

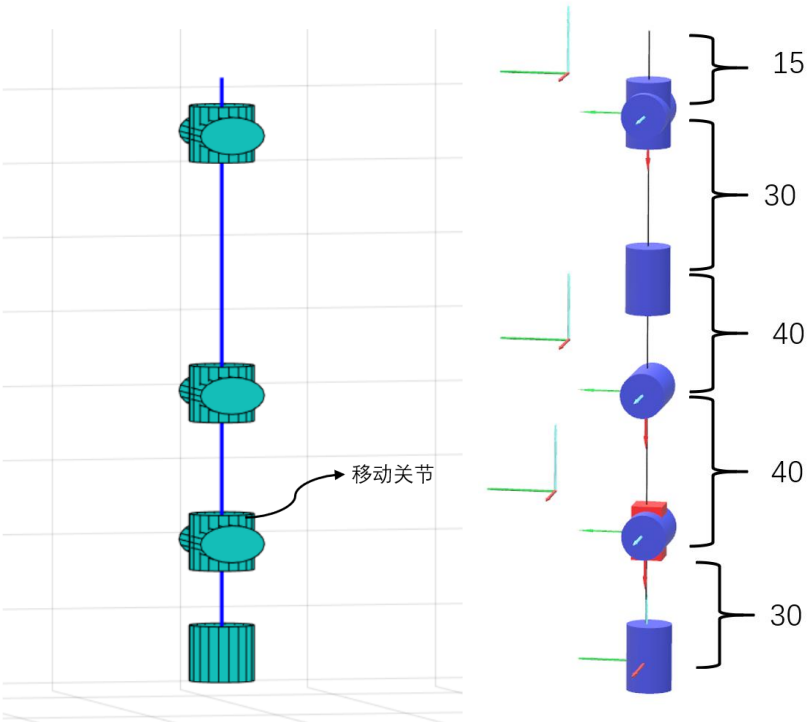


图 1　7 自由度机器人构型

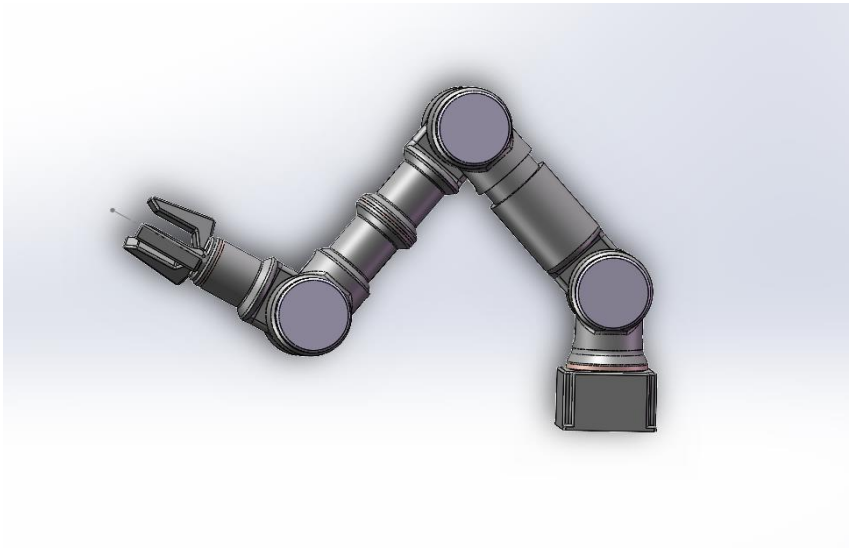


图 2 SolidWorks 机械臂模型

2.正运动学分析与工作空间设定

2.1：建立 DH 坐标系

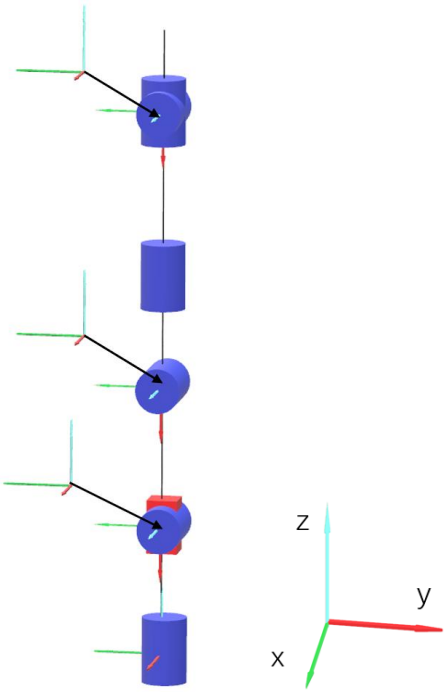


图 2 机器人关节坐标系建立

SDH 参数表

i	θ	d	a	α	Qlim
1	θ_1	30	0	-90	$-180\sim180$
2	θ_2	0	0	90	$-140\sim140$
3	0	$40+d_3$	0	-90	$-10\sim50$
4	θ_4	0	0	90	$-160\sim160$
5	θ_5	70	0	-90	$-180\sim180$
6	θ_6	0	0	90	$-170\sim170$
7	θ_7	15	0	0	$-180\sim180$

2.3：建立齐次变换矩阵

机器人正运动学方法 D–H 表示法中，关节 n 到关节 n+1 的变换矩阵为：

$$ {}^{n}_{n+1}T = \begin{bmatrix} c\theta_n & -c\alpha_n s\theta_n & s\alpha_n s\theta_n & ac\theta_n \\ s\theta_n & c\alpha_n c\theta_n & -s\alpha_n c\theta_n & as\theta_n \\ 0 & s\alpha_n & c\alpha_n & d_n \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

根据上式，很容易可以写出${}^{0}_{1}T$，${}^{1}_{2}T$，${}^{2}_{3}T$，${}^{3}_{4}T$，${}^{4}_{5}T$，${}^{5}_{6}T$，${}^{6}_{7}T$共 7 个变换矩阵。

$$ {}^{0}_{1}T = \begin{bmatrix} cos(t1) & 0 & -sin(t1) & 0 \\ sin(t1) & 0 & cos(t1) & 0 \\ 0 & -1 & 0 & 30 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{1}_{2}T = \begin{bmatrix} cos(t2) & 0 & sin(t2) & 0 \\ sin(t2) & 0 & -cos(t2) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{2}_{3}T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 40+d3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{3}_{4}T = \begin{bmatrix} cos(t4) & 0 & sin(t4) & 0 \\ sin(t4) & 0 & -cos(t4) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{4}_{5}T = \begin{bmatrix} cos(t5) & 0 & -sin(t5) & 0 \\ sin(t5) & 0 & cos(t5) & 0 \\ 0 & -1 & 0 & 70 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{5}_{6}T = \begin{bmatrix} cos(t6) & 0 & sin(t6) & 0 \\ sin(t6) & 0 & -cos(t6) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{6}_{7}T = \begin{bmatrix} cos(t7) & -sin(t7) & 0 & 0 \\ sin(t7) & cos(t7) & 0 & 0 \\ 0 & 0 & 1 & 15 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

通过上述矩阵即可计算工作点到基坐标的变换矩阵${}^{0}_{7}T = {}^{0}_{1}T\ {}^{1}_{2}T\ {}^{2}_{3}T\ {}^{3}_{4}T\ {}^{4}_{5}T\ {}^{5}_{6}T\ {}^{6}_{7}T$

2.4：计算机器人工作空间

2.4.1 蒙特卡洛法求解工作空间

将机器人各关节限制量差分计算，在 1->0 随机取值乘上差分量，将机器人关节角度离散化，再通过正运动学计算空间坐标，对末端执行器的坐标随机化。

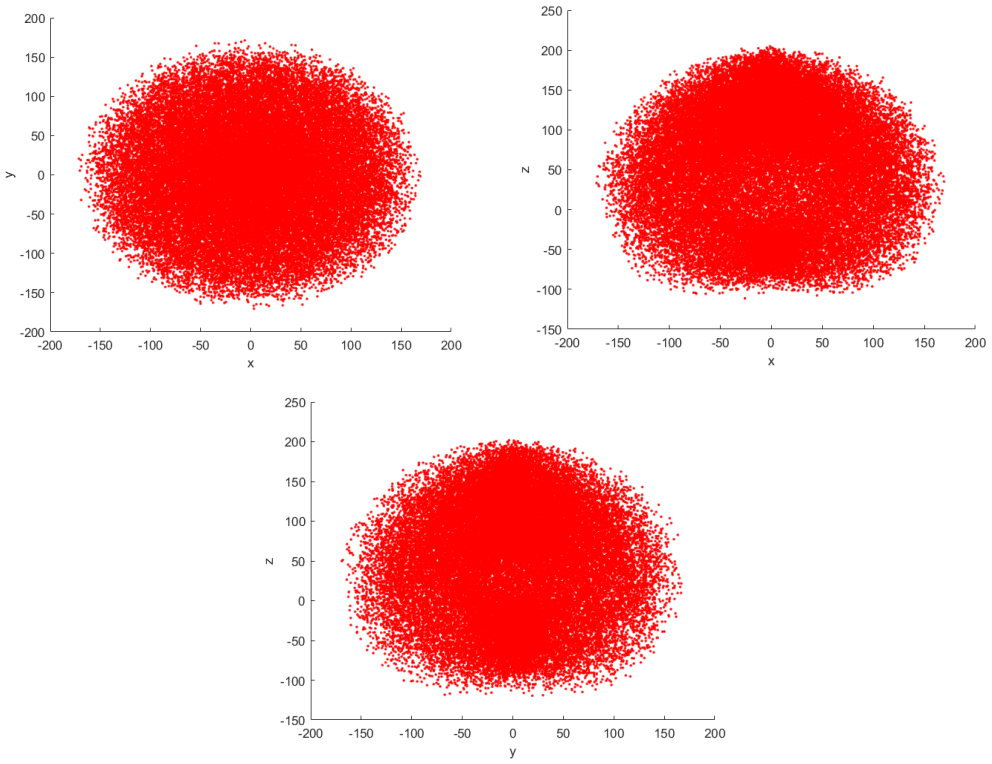


图 3 机器人工作空间 X-Y X-Z Y-Z(蒙特卡洛法 50000 个点)

2.4.2 遍历法求解工作空间剖面

通过将机器人除第一个关节外的各个关节的工作范围进行等分，将各个关节的等分点进行组合，使用正运动学计算空间坐标，得到侧面剖视图。

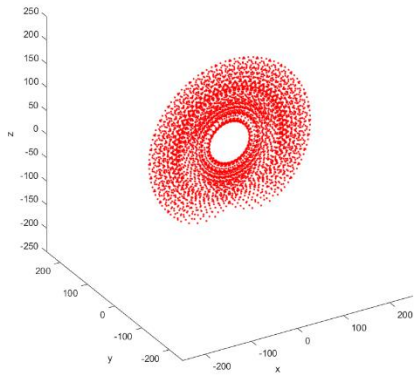


图 4 机器人工作空间剖面图

3.雅克比矩阵求解

3.1 矢量积法求解雅克比矩阵

$$\dot{x} = J\dot{q} \qquad J = \begin{bmatrix} \frac{dx_1}{dq_1} & \cdots & \frac{dx_1}{dq_n} \\ \vdots & \ddots & \vdots \\ \frac{dx_m}{dq_1} & \cdots & \frac{dx_m}{dq_n} \end{bmatrix}$$

雅克比矩阵为 6 X N 的矩阵 N 为关节数，求解雅可比矩阵的方式我们选择矢量积法。

基本思路：通过求取各个关节微分运动队末端微分运动的表达式，利用速度矢量相加原理构造雅可比矩阵。

对于移动关节 i，有：

$$\begin{bmatrix} \nu \\ \omega \end{bmatrix} = \begin{bmatrix} z_i \\ 0 \end{bmatrix} \dot{q}_i, J_i = \begin{bmatrix} z_i \\ 0 \end{bmatrix}$$

对于转动关节 i，有：

$$\begin{bmatrix} \nu \\ \omega \end{bmatrix} = \begin{bmatrix} z_i \times {}^i p_n^0 \\ 0 \end{bmatrix} \dot{q}_i, J_i = \begin{bmatrix} z_i \times {}^i p_n^0 \\ 0 \end{bmatrix} = \begin{bmatrix} z_i \times ({}_i^0 R\, {}^i p_n) \\ 0 \end{bmatrix}$$

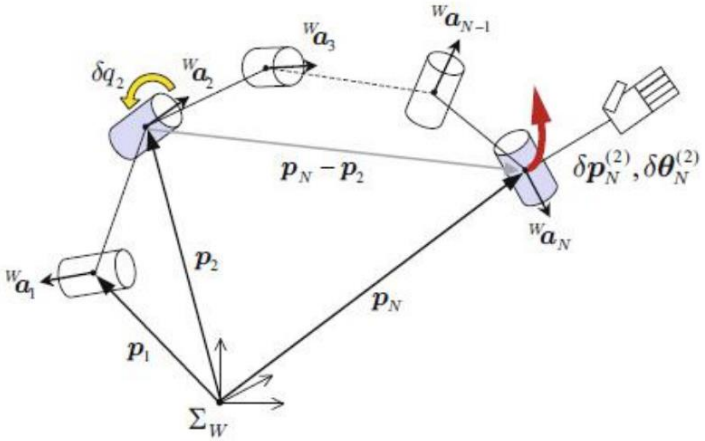


图 5 矢量积法求解雅克比矩阵示意图

3.2 机械臂奇异值分析

机械手的奇异位型主要分为两种：（1）边界奇异位型，出现在机械手工作空间的边界，可通过阻止机械手运动到可达空间的边界来避免出现这种情况；（2）内部奇异位型，出现在机械手工作空间内，通常是由于两个或两个以上的关节轴重合导致，或者是由特定的末端

执行器构型所导致，这种情况很危险，遇到的可能性较大。
我们设计的机械臂存在两种奇异构型，如下：

3.2.1 腕部奇异位型

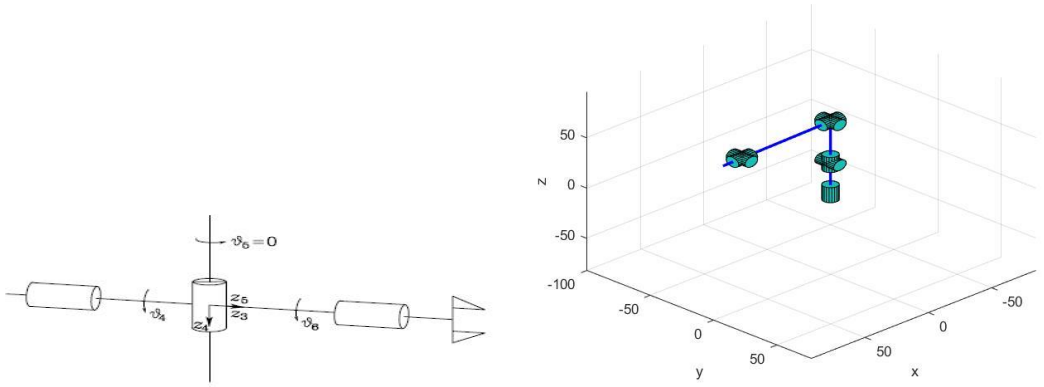


图 6 腕部奇异位型示意图

在本组设计的 7 自由度机械臂中，当机械臂的第六个关节处于 0 度或 180 度时，第五和第七轴的轴线处于同一直线上，此时会出现腕关节奇异点。

3.2.2 手臂奇异位型

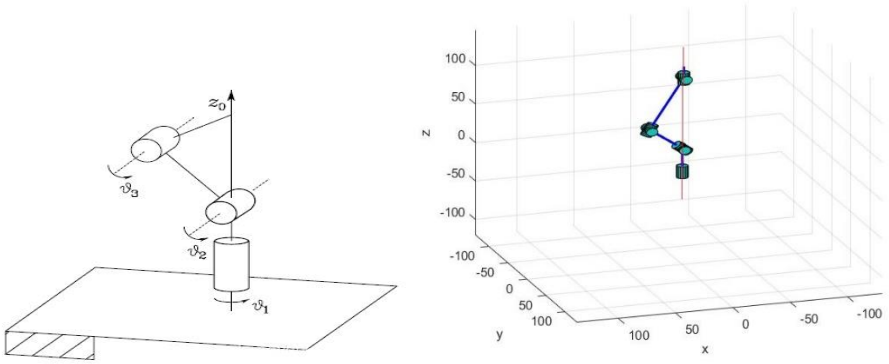


图 7 手臂奇异位型示意图

当机械臂的第五、六、七轴的交点位于第一轴的正上方时，机械臂处于臂奇点位置。

4.逆运动学求解

4.1 自运动流形

7 自由度的机器人，对于三维空间来说多了一个冗余的自由度，此时就会有使得 m×n 雅可比矩阵冗余—— m < n (三维机器人时，m=6)，会导致：

$$\exists q', Jq' = 0$$

也就是说对于一个位姿，其逆运动学解算会得到无限个解，对于冗余的自由度给定不同的值都会有一组不同的解，这些解能被划分为有限的有界的不相交的光滑流形，这些流形中的每一个就称为自运动流形

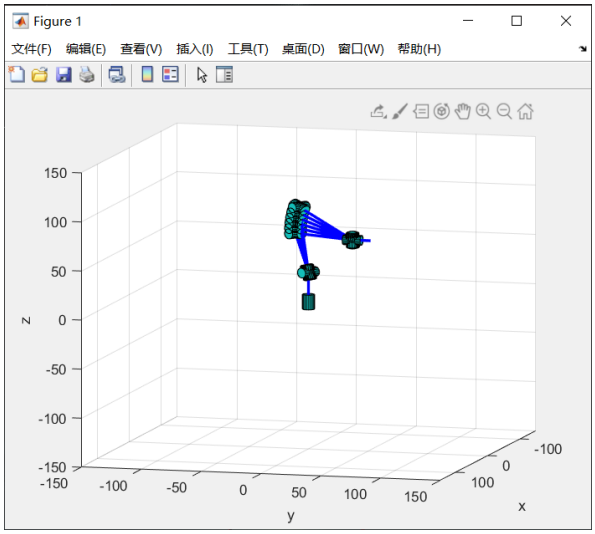


图 8 机械臂自运动流形示意图

自运动流形应用：

轨迹规划的避障，在规划好一段轨迹后，运动途中可能会有障碍物，对于六轴机械臂只能旋转其他的解空间去避障或是重新规划路线，一般来说六轴机械臂的解空间差距较大，有时并不能很好的相互进行平滑转换，这时七自由度机械臂的冗余自由度就可以轻松做到避障

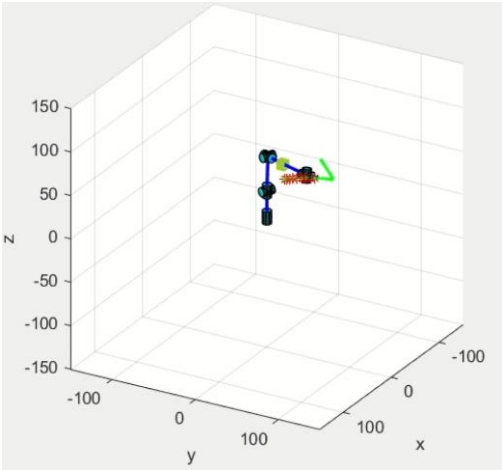


图 9 机械臂运动干涉

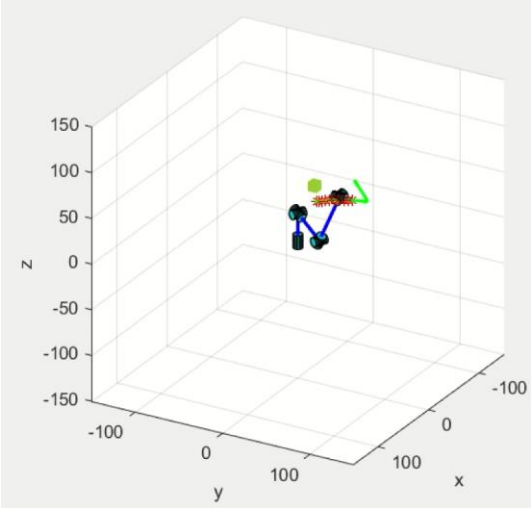


图 10 其他解空间运动过程

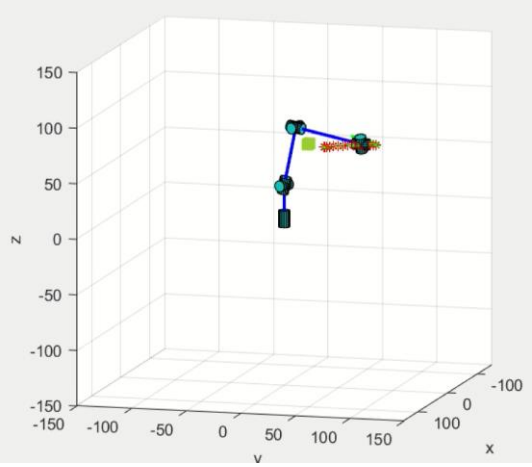


图 11 冗余自由度避障

4.2 代数法+几何法+欧拉角求解析解

机器人的逆运动学是根据其末端位置和姿态获得各关节的运动量，即已知，求解$\theta_i(\mathrm{i}=1\sim\mathrm{n})$。对于本文所研究的7自由度机器人，就是要通过12 个方程求解7 个未知数。的旋转矩阵分量所包含的九个方程中有三个是相互独立的，所以实际上有6个方程解7个未知数，无法求解，所以需要人为给一个冗余自由度设定一个值，去求解其他六个，本文将滑动关节设为0。

由于有六个方程求解较为困难。Pieper研究了三个相邻坐标系的旋转轴相交于一点的操作臂（本文为后三个关节），使得机器人的位置由前四个关节决定，后三个关节决定姿态，可以将计算简化。本文将使用Pieper 法求解机器人的逆运动学方程。

4.2.1 代数+几何计算前三个旋转关节角度

通过已知位置坐标，求解前四个关节的变换矩阵，由于只需要考虑位置，可以忽视变换矩阵

中的姿态部分

$$ {}^{0}_{1}T\ {}^{1}_{2}T\ {}^{2}_{3}T\ {}^{3}_{4}T\ {}^{4}_{5}T\ {}^{5}_{6}T\ = \begin{bmatrix} | & | & | & x \\ | & | & | & y \\ | & | & | & z \\ | & | & | & 1 \end{bmatrix} $$

$$ \Longrightarrow {}^{1}_{2}T\ {}^{2}_{3}T\ {}^{3}_{4}T\ {}^{4}_{5}T\ {}^{5}_{6}T\ = {}^{0}_{1}T^{-1} \begin{bmatrix} | & | & | & x \\ | & | & | & y \\ | & | & | & z \\ | & | & | & 1 \end{bmatrix} $$

$$ \Longrightarrow \begin{bmatrix} | & | & | & S_2 * (L2\ +\ d3) + (C_2 * S_4\ +\ C_4 * S_2) * (L3\ +\ L4) \\ | & | & | & -C_2 * (L2\ +\ d3) + (C_2 * C_4 + S_2 * S_4) * (L3\ +\ L4) \\ | & | & | & 0 \\ | & | & | & 1 \end{bmatrix} = \begin{bmatrix} | & | & | & x * C_1 + y * S_1 \\ | & | & | & L1 - z \\ | & | & | & y * C_1 - x * S_1 \\ | & | & | & 1 \end{bmatrix} $$

可得到θ_1求解公式：

$$ y * C_1 - x * S_1 = 0 $$

$$ \Longrightarrow \frac{y}{x} = tan_1 \Longrightarrow \theta_1 = tan^{-1}(\frac{y}{x}) $$

又有：

$$ \begin{cases} S_2 * (L2\ +\ d3) + (C_2 * S_4\ +\ C_4 * S_2) * (L3\ +\ L4) = x * C_1 + y * S_1 \\ -C_2 * (L2\ +\ d3) + (C_2 * C_4 + S_2 * S_4) * (L3\ +\ L4) = L1 - z \end{cases} $$

方法一：通过求解该方程组，解得θ_2和θ_4

思想：通过 $\sin(a)^2 + \cos(a)^2 = 1$进行简化

$$ \begin{cases} S_2 * (k1 + k3) + C_2 * k4\ = a \\ -C_2 * (k1 + k3) + S_2 * k4 = b \end{cases} $$

$$ \begin{cases} a = x * C_1 + y * S_1 \\ b = L1 - z \\ k1 = L2 + d3 \\ k2 = L3 + L4 \\ k3 = C_4 * k2 \\ k4 = S_4 * k2 \end{cases} $$

方法二：先通过几何法求解θ_4，再代入上诉方程求解θ_2

思想：$\cos A = (b^2 + c^2 - a^2)/2bc$

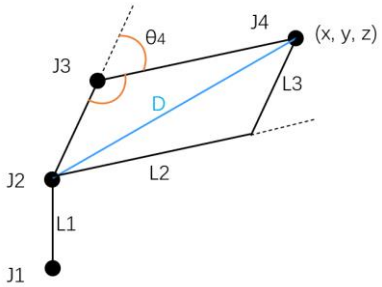


图 12 几何法求解示意图

$$ D = \sqrt{x^2 + y^2 + (z - L1)^2} $$

$$ \theta_4 = \pm(\pi - \text{acos}\,(\frac{L2^2 + L3^2 - D^2}{2L1L2})) $$

将θ_4回代可得θ_2的求解函数：

$$\theta_2 = \text{atan } 2(S_2, C_2)$$

$$\begin{cases} S_2 = (w1 * b/-w2 - a)/(w1 * w1/-w2 - w2) \\ C_2 = (w2 * a1/w1 - a2)/(-w2 * w2/w1 - w1) \end{cases}$$

$$\begin{cases} a = x * C_1 + y * S_1 \\ b = L1 - z \\ w1 = (L3 + L4) * S_3 \\ w2 = L2 + (L3 + L4) * C_3 \end{cases}$$

由于计算过程用的都是反三角函数，理论上每个角度都有两个解：

$$\theta_1 = \begin{cases} \theta_1 \\ \theta_1 + \pi \end{cases}$$

$$\theta_2 = \begin{cases} \theta_2 \\ \theta_2 + \pi \end{cases}$$

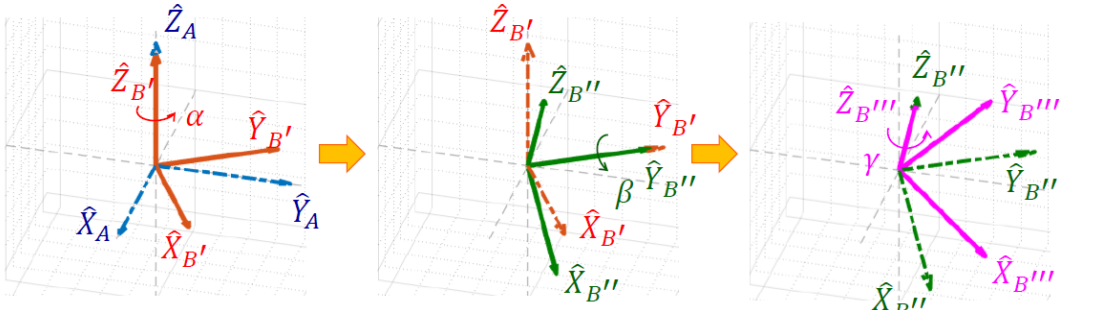
$$\theta_4 = \begin{cases} \theta_4 \\ -\theta_4 \end{cases}$$

那么两两组合一共就是 8 组解，经验算其中 4 组解是错误的组合，剩下的四组解最后只会使机械臂到达两种位姿，所以我们取其中两组角度转动较小的解：

$$\theta = \begin{cases} [\theta_1, \theta_2, -\theta_4] \\ [\theta_1, \theta_2, \theta_4] \end{cases}$$

4.2.2 欧拉法计算后三个旋转关节角度

根据${}^4_7R$可以得到θ_5，θ_6，θ_7的值。此部分采用的是Z−Y−Z Euler Angles的方法，将后三轴中D−H 表达法坐标系对应到欧拉角相对坐标系旋转中。若坐标系B 相对于坐标系A 重合，然后坐标系B 依次沿着自身的欧拉相对坐标系旋转Z−Y−Z轴旋转角度α，β，γ。



$${}^A_BR_{Z'Y'Z'}(\alpha, \beta, \gamma) = R_{Z'}(\alpha)R_{Y'}(\beta)R_{Z'}(\gamma)$$

图 13 欧拉角变换示意图

$$\begin{bmatrix} c\alpha c\beta c\gamma - s\alpha s\gamma & -c\alpha c\beta s\gamma - s\alpha c\gamma & c\alpha s\beta \\ s\alpha c\beta c\gamma + c\alpha s\gamma & -s\alpha c\beta s\gamma + c\alpha c\gamma & s\alpha s\beta \\ -s\beta c\gamma & s\beta s\gamma & c\beta \end{bmatrix}$$

目标：求出${}^{4}_{7}R$，目前已知前四个关节的角度，就可以求出${}^{0}_{4}R$，则：

$$ {}^{4}_{7}R = {}^{0}_{4}R^{-1}\ {}^{0}_{7}R $$

$$ {}^{A}_{B}R_{Z'Y'Z'}(\alpha,\beta,\gamma) = \begin{bmatrix} c\alpha c\beta c\gamma - s\alpha s\gamma & -c\alpha c\beta s\gamma - s\alpha c\gamma & c\alpha s\beta \\ s\alpha c\beta c\gamma + c\alpha s\gamma & -s\alpha c\beta s\gamma + c\alpha c\gamma & s\alpha s\beta \\ -s\beta c\gamma & s\beta s\gamma & c\beta \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} $$

因中间的∠β在后续计算会用到sin (β)，为防止其值为 0，所以要先求∠β并进行分类讨论：在$\beta = pi$和$\beta = 0$时，腕关节处于奇异位置，此时前后两个关节合为一个自由度，只需将其中一个设为 0 去求解另一个即可。

If $\beta \neq 0°$

$\beta = Atan2(\sqrt{{r_{31}}^2 + {r_{32}}^2}, r_{33})$

$\alpha = Atan2(r_{23}/s\beta, r_{13}/s\beta)$

$\gamma = Atan2(r_{32}/s\beta, -r_{31}/s\beta)$

If $\beta = 180°$

$\alpha = 0°$

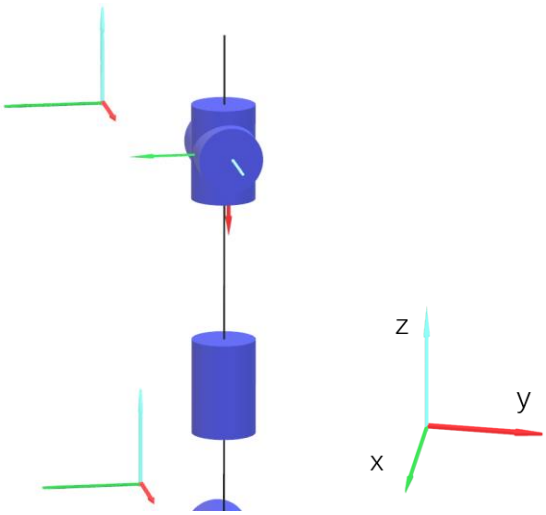
$\gamma = Atan2(r_{12}, -r_{11})$

If $\beta = 0°$

$\alpha = 0°$

$\gamma = Atan2(-r_{12}, r_{11})$

我们的后三轴坐标系定义如右图：



可见关节 5 的 Y 轴与关节 6 的 Z 轴方向相同，关节 6 的 Y 轴与关节 7 的 Z 轴方向相同，所以计算出来的α，β，γ 即为θ_5，θ_6，θ_7

4.3 微分运动

微分运动指机构（例如机器人）的微小运动，可以用它来推导不同部件之间的速度关系。当机器人关节做微量运动时，机器人手坐标系也会产生微量运动。而前面博客讲机器人雅可比矩阵已经讲到，这是机器人关节速度与末端机械手（末端坐标系）速度之间的映射，由于微分运动除以 dt 即可得到速度，因此可以说雅可比矩阵是机器人关节微分运动与末端微分运动之间的映射。当计算得到机器人末端坐标系的微分运动时，通过雅可比矩阵，则能计算得到各关节的微分运动，能解决逆运动学及速度控制的问题。

$$\begin{bmatrix} d_x \\ d_y \\ d_z \\ \delta_x \\ \delta_y \\ \delta_z \end{bmatrix} = \begin{bmatrix} 机器人 \\ 雅克比矩阵 \end{bmatrix} \begin{bmatrix} d_{\theta_1} \\ d_{\theta_2} \\ d_{\theta_3} \\ d_{\theta_4} \\ d_{\theta_5} \\ d_{\theta_6} \\ d_{\theta_7} \end{bmatrix}$$

$$D = Jd_\theta$$

D 表示六轴机器人末端机械手坐标的微分运动，前三项表示机械手坐标沿 x，y，z 轴的微分平移，后三项表示绕这三个轴的微分旋转；d_θ表示各关节的微分运动。

微分旋转:

$$\mathrm{Rot}(x,\delta x) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & -\delta x & 0 \\ 0 & \delta x & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \mathrm{Rot}(y,\delta y) = \begin{bmatrix} 1 & 0 & \delta y & 0 \\ 0 & 1 & 0 & 0 \\ -\delta y & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \mathrm{Rot}(z,\delta z) = \begin{bmatrix} 1 & -\delta z & 0 & 0 \\ \delta z & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Rot(x,\delta_x)Rot(y,\delta_y)Rot(z,\delta_z) = \begin{bmatrix} 0 & -\delta_z & \delta_y & 0 \\ \delta_z & 0 & -\delta_x & 0 \\ -\delta_y & \delta_x & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

微分算子：

$$\Delta = \begin{bmatrix} 0 & -\delta_z & \delta_y & d_x \\ \delta_z & 0 & -\delta_x & d_y \\ -\delta_y & \delta_x & 0 & d_z \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

则末端的微小变动dT可以表示为：

$$dT = \Delta T \quad \text{针对基坐标}$$

$$dT = T\,\Delta^T \quad \text{针对 T 坐标}$$

4.4 牛顿下山法原理

牛顿法：

$f(x)=0$，将左边的式子用泰勒展开 $f(x)=f(x_k)+f'(x_k)(x-x_k)+\cdots=0$ 省略高

阶就有：$f(x_k)+f'(x_k)(x-x_k)=0$，所以 $x=x_k-\dfrac{f(x_k)}{f'(x_k)}$

牛顿下山法是基于牛顿法的改进：

对于上式有$x = x_k - \lambda \frac{f(x_k)}{f'(x_k)}$

$$\begin{cases} \lambda = 1 & |f(x_k)| > |f(x_{k+1+})| \\ \lambda = \dfrac{\lambda}{2} & |f(x_k)| < |f(x_{k+1+})| \end{cases}$$

4.5 数值解

4.5.1 数值迭代步骤

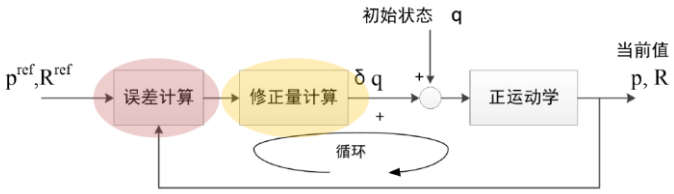


图 14 数值迭代流程图

第一步 给定机器人末端（End effector）目标位姿态（p^{ref}，R^{ref}）

第二步 定义机器人各关节的初始关节角q

第三步 由正运动学计算机器人末端的当前位姿

第四步 计算机器人末端位姿的误差（p^{err}，R^{err}）=（$p^{ref}-p$, $R^T R^{ref}$）

第五步 当误差（p^{err}，R^{err}）足够小时停止运算；

第六步 当误差（p^{err}，R^{err}）大于设定值时计算关节角的修正量δq

第七步 $q = q + \delta q$，返回第三步

4.5.2 误差计算

位置误差：当前位置与目标位置的绝对距离

$$P^{err} = \Delta P_x^2 + \Delta P_y^2 + \Delta P_z^2$$

姿态误差：使用轴角表达式描述 X，Y，Z 轴的角度差，也就是将原来绕 X，Y，Z 轴的旋转表达为绕一特殊轴的旋转

$$R^{err} = R^T R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

$$\omega^{err} = \begin{cases} \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T & (R^{err} = E\text{时}) \\ \dfrac{\theta}{2\sin\theta} \begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix} & (R^{err} \neq E\text{时}) \end{cases}$$

最终误差 err = $P^{err} + \omega^{err}$

4.5.3 修正量计算

将计算得到的误差作为广义位姿修正量，结合牛顿下山法计算各关节的修正量

$$\delta q = \lambda J^{-1} \begin{bmatrix} \delta p \\ \delta \omega \end{bmatrix} \quad (\lambda \in (0,1]) \begin{cases} \delta p = P^{err} \\ \delta \omega = \omega^{err} \end{cases}$$

广义位姿 $\rightarrow$

$$P = F(q)$$

$$f(q) = F(q) - P = 0$$

$$q_{k+1} = q_k - \lambda \frac{f(q_k)}{f'(q_k)}$$

$$q_{k+1} - q_k = -\lambda \frac{f(q_k)}{f'(q_k)}$$

$$\delta q = -\lambda \frac{-\begin{bmatrix} \delta p \\ \delta \omega \end{bmatrix}}{J}$$

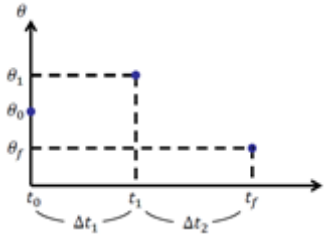
$$\delta q = \lambda J^{-1} \begin{bmatrix} \delta p \\ \delta \omega \end{bmatrix}$$

5.轨迹规划

5.1　基于三次非均匀样条插值的轨迹规划

5.1.1 算法原理

先考虑，起始点->过渡点->终点的简单轨迹



先将时间段进行平移变换，方便后续计算：

$$\begin{cases} \psi_1 = t_1 - t_0 \ \epsilon[0 \ \Delta t_1] \\ \psi_2 = t_f - t_1 \ \epsilon[0 \ \Delta t_2] \end{cases}$$

则其两段轨迹可以表示为$\psi_1 \ \psi_2$的函数：

$$\begin{cases} \theta_I(\psi_1) = a_{10} + a_{11}\psi_1 + a_{12}{\psi_1}^2 + a_{13}{\psi_1}^3 \\ \theta_{II}(\psi_2) = a_{20} + a_{21}\psi_2 + a_{22}{\psi_2}^2 + a_{23}{\psi_2}^3 \end{cases}$$

对于其中 8 个未知参数，需要 8 个方程求解：

位置信息：

1.初始位置；2.中间点位置；3.中间点位置连续；4.结束位置

$$\theta_0 = a_{10}$$
$$\theta_1 = a_{10} + a_{11}\Delta t_1 + a_{12}\Delta t_1^2 + a_{13}\Delta t_1^3$$
$$\theta_1 = a_{20}$$
$$\theta_f = a_{20} + a_{21}\Delta t_2 + a_{22}\Delta t_2^2 + a_{23}\Delta t_2^3$$

速度信息：

1.初始速度； 3.结束速度；初末速度一般为 0

$$\dot{\theta}_0 = 0 = a_{11}$$
$$\dot{\theta}_f = 0 = a_{21} + 2a_{22}\Delta t_2 + 3a_{23}\Delta t_2^2$$

中间点连续性：

1.速度连续；2.加速度连续

$$\dot{\theta}_1 = a_{11} + 2a_{12}\Delta t_1 + 3a_{13}\Delta t_1^2 = a_{21}$$
$$\ddot{\theta}_1 = 2a_{12} + 6a_{13}\Delta t_1 = 2a_{22}$$

通过上述式子可解得：

$$a_{10} = \theta_0 \qquad a_{20} = \theta_1$$
$$a_{11} = 0 \qquad a_{21} = \frac{3\theta_f - 3\theta_0}{4\Delta t}$$
$$a_{12} = \frac{12\theta_1 - 3\theta_f - 9\theta_0}{4\Delta t^2} \qquad a_{22} = \frac{-12\theta_1 + 6\theta_f + 6\theta_0}{4\Delta t^2}$$
$$a_{13} = \frac{-8\theta_1 + 3\theta_f + 5\theta_0}{4\Delta t^3} \qquad a_{23} = \frac{8\theta_1 - 5\theta_f - 3\theta_0}{4\Delta t^3}$$

将其转化为矩阵形式：

$$\begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_1 \\ \theta_f \\ \dot{\theta}_0 \\ \dot{\theta}_f \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & \Delta t_1 & \Delta t_1^2 & \Delta t_1^3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \Delta t_2 & \Delta t_2^2 & \Delta t_2^3 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2\Delta t_2 & 3\Delta t_2^2 \\ 0 & 1 & 2\Delta t_1 & 3\Delta t_1^2 & 0 & -1 & 0 & 0 \\ 0 & 0 & 2 & 6\Delta t_1 & 0 & 0 & -2 & 0 \end{bmatrix} \begin{bmatrix} a_{10} \\ a_{11} \\ a_{12} \\ a_{13} \\ a_{20} \\ a_{21} \\ a_{22} \\ a_{23} \end{bmatrix}$$

$$A_{8\times1} = T_{8\times8}^{-1}\,\Theta_{8\times1}$$

泛化为 N+1 个点，其中有 N-1 个中间点，共 N 条曲线，4N 个未知参数，已知条件有：1.N+1 个点位置；2. N-1 个中间点的位置连续；3. N-1 个中间点速度与加速度连续，一共 4N-2 个等式，任选一组边界条件即可求解：

$$\begin{cases} S_1''(x_1) = S_N''(x_{N+1}) = 0 \quad \text{定义加速度} \\ S_1'(x_1) = u;\ S_N'(x_{N+1}) = v \quad \text{定义速度} \\ S_1(x_1) = S_N(x_{N+1});\ S_1'(x_1) = S_N'(x_{N+1});\ S_1''(x_1) = S_N''(x_{N+1}) \quad \text{周期运动连续性} \end{cases}$$

5.1.2 算法仿真

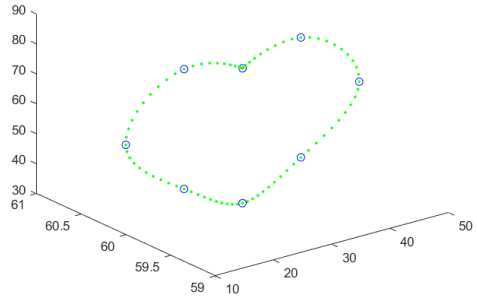


图 15 三次样条插值路径规划(蓝色点为特征点，绿色为插值点)

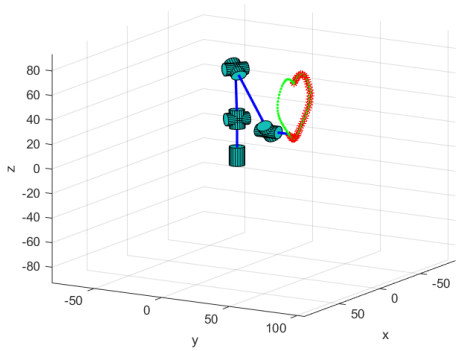
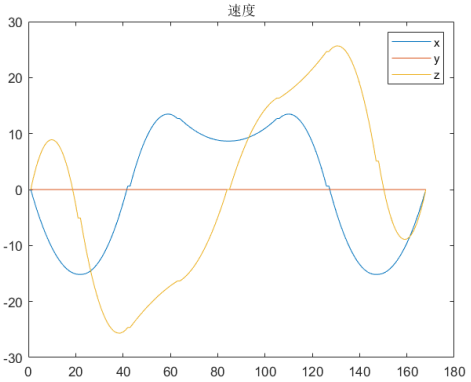
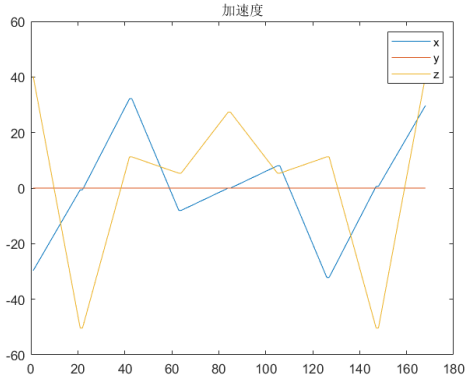


图 16 仿真控制示意图





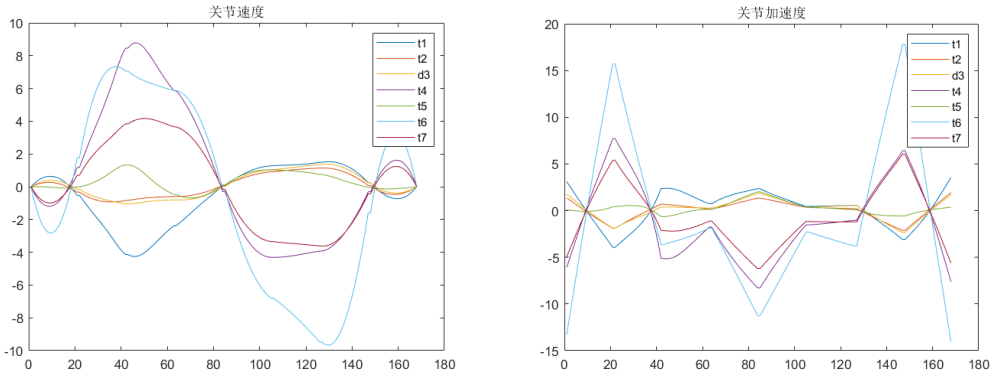


图 17 速度加速度变化曲线

5.2 4-3-4 轨迹规划

5.2.1 算法原理

采用 4-3-4 轨迹规划的方式：

$$\theta(t)_1 = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4$$
$$\theta(t)_2 = b_0 + b_1 t + b_2 t^2 + b_3 t^3$$
$$\theta(t)_3 = c_0 + c_1 t + c_2 t^2 + c_3 t^3 + c_4 t^4$$

假设：
全局时间变量为 t
第 j 个运动段的本地时间变量为τ_j
每一个运动段的初始时间为$\tau_{ji} = 0$，且给定每一运动段的终端本地时间τ_{ji}

为求解 14 个参数需要满足 14 个约束条件
（1）在时间τ_0 处，第一条 4 次多项式运动段产生的初值即为已知位置

$$\theta_1 = a_0$$

（2）在时间$\tau 0$ 处，已经给定第一运动段的初始速度

$$\dot{\theta}_1 = a_1$$

（3）在本地时间τ_0 处，已经给定第一运动段的初始加速度

$$\ddot{\theta}_1 = 2a_2$$

（4）在第一中间点位置θ_2 与第一运动段在本地时间τ_{1f} 时的末端位置相同

$$\theta_2 = a_0 + a_1(\tau_{1j}) + a_2(\tau_{1j})^2 + a_3(\tau_{1j})^3 + a_4(\tau_{1j})^4$$

（5）在第一中间点的位置与 3 次多项式在本地时间$\tau 2 = 0$ 时的初始位置相同

$$\theta_2 = b_0$$

（6）第一中间点速度连续

$$a_1 + 2a_2(\tau_{1j})^2 + 4a_3(\tau_{1j})^3 = b_1$$

（7）第一中间点加速度连续

$$2a_2 + 6a_3(\tau_{1j})^2 = 2b_2$$

（8）在第二中间点位置与第二段 3 次多项式在本地时间τ 2 f = 0 时的末端位置相同

$$\theta_3 = b_0 + b_1(\tau_{2j}) + b_2(\tau_{2j})^2 + b_3(\tau_{2j})^3$$

（9）在第二中间点位置与第三段 4 次多项式在本地时间τ 3 = 0 时的初始位置相同

$$\theta_3 = c_0$$

（10）第二中间点速度连续

$$b_1 + 2b_2(\tau_{2j})^2 + 3b_3(\tau_{2j})^2 = c_1$$

（11）第二中间点加速度连续

$$2b_2 + 6b_3(\tau_{2j})^2 = 2b_2$$

（12）由已知最后运动段在其本地时间τ_{3j}时的位置θ_f

$$\theta_f = c_0 + c_1(\tau_{3j}) + c_2(\tau_{3j})^2 + c_3(\tau_{3j})^3 + c_4(\tau_{3j})^4$$

（13）由已知最后运动段在其本地时间τ_{3j}时的速度$\dot{\theta}_f$

$$\dot{\theta}_f = c_1 + 2c_2(\tau_{3j}) + 3c_3(\tau_{3j})^2 + 4c_4(\tau_{3j})^3$$

（14）由已知最后运动段在其本地时间τ_{3j}时的加速度$\ddot{\theta}_f$

$$\ddot{\theta}_f = 2c_2(\tau_{3j}) + 6c_3(\tau_{3j}) + 12c_4(\tau_{3j})^2$$

当所需要的路径上经过的关键点的数量大于四个时，则中间的运动段均使用三次多项式表示。

矩阵表示：

$$\begin{bmatrix} \theta_1 \\ \dot{\theta}_1 \\ \ddot{\theta}_1 \\ \theta_2 \\ \theta_2 \\ 0 \\ 0 \\ \theta_3 \\ \theta_3 \\ 0 \\ 0 \\ \theta_4 \\ \dot{\theta}_4 \\ \ddot{\theta}_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & \tau_{1f}^{1} & \tau_{1f}^{2} & \tau_{1f}^{3} & \tau_{1f}^{4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2\tau_{1f} & 3\tau_{1f}^{2} & 4\tau_{1f}^{3} & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 6\tau_{1f} & 12\tau_{1f}^{2} & 0 & 0 & -2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & \tau_{2f} & \tau_{2f}^{2} & \tau_{2f}^{3} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2\tau_{2f} & 3\tau_{2f}^{2} & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 6\tau_{2f} & 0 & 0 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \tau_{3f} & \tau_{3f}^{2} & \tau_{3f}^{3} & \tau_{3f}^{4} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2\tau_{3f} & 3\tau_{3f}^{2} & 4\tau_{3f}^{3} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 6\tau_{3f} & 12\tau_{3f}^{2} \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ b_0 \\ b_1 \\ b_2 \\ b_3 \\ c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix}$$

5.2.2 算法仿真

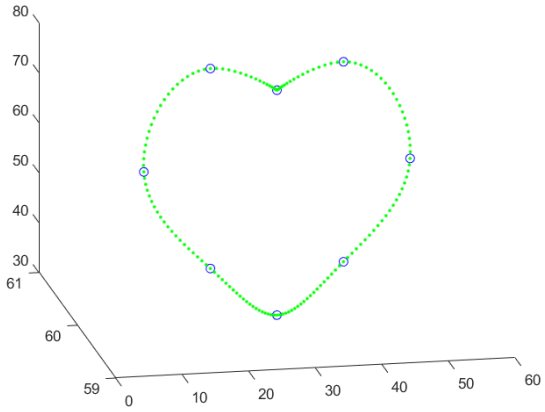


图 18 4-3-4 插值路径规划(蓝色点为特征点，绿色为插值点)

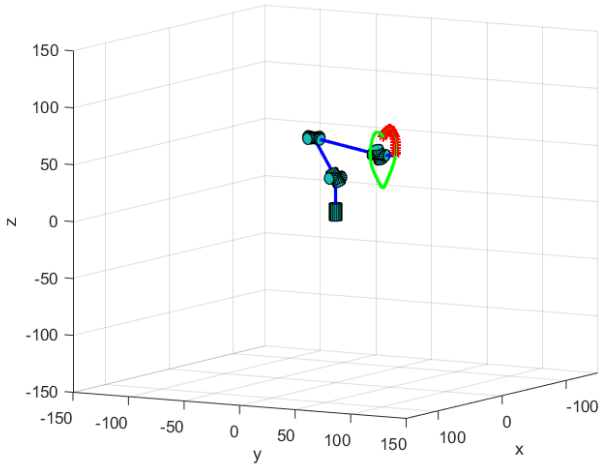
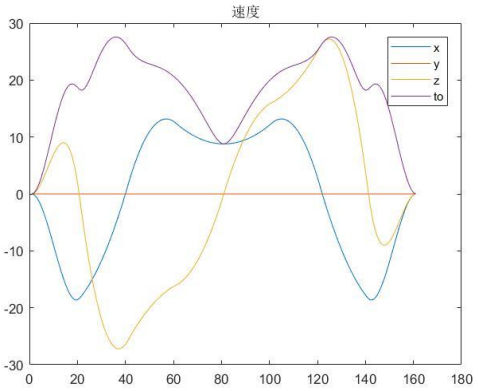
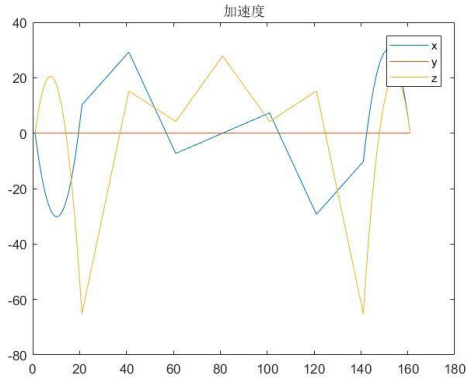


图 19 仿真控制示意图





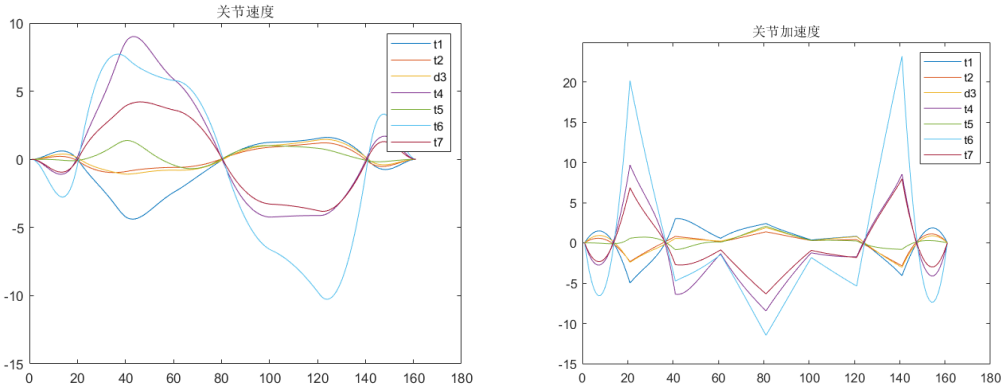


图 20 速度加速度变化曲线

5.3 基于五次非均匀样条插值的轨迹规划

5.3.1 算法原理

对于一段五次多项式曲线，有 6 个未知参数，需要 6 个等式。
同样将时间量进行平移变换：

$$t = t_f - t_0 \ \epsilon[0\ \Delta t]$$

$$\theta(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3 + c_4 t^4 + c_5 t^5$$
$$\theta'(t) = c_1 t + 2c_2 t + 3c_3 t^2 + 4c_4 t^3 + 5c_5 t^4$$
$$\theta''(t) = 2c_2 + 6c_3 t + 12c_4 t^2 + 20c_5 t^3$$

给定初始六个条件，根据五次多项式的公式：1.起末位移；2.起末速度；3.起末加速度。则会得到如下六个方程组：

$$\theta_0 = a_0$$
$$\theta_f = a_0 + a_1\Delta t + a_2\Delta t^2 + a_3\Delta t^3 + a_4\Delta t^4 + a_5\Delta t^5$$
$$\dot{\theta}_0 = a_1$$
$$\dot{\theta}_f = a_1 + 2a_2\Delta t + 3a_3\Delta t^2 + 4a_4\Delta t^3 + 5a_5\Delta t^4$$
$$\ddot{\theta}_0 = 2a_2$$
$$\ddot{\theta}_f = 2a_2 + 6a_3\Delta t + 12a_4\Delta t^2 + 20a_5\Delta t^3$$

写成矩阵形式：

$$\begin{bmatrix} 1 & t_s & t_s^2 & t_s^3 & t_s^4 & t_s^5 \\ 1 & t_e & t_s^2 & t_s^3 & t_s^4 & t_s^5 \\ 0 & 1 & 2ts & 3t_s^2 & 4t_s^3 & 5t_s^4 \\ 0 & 1 & 2t_e & 3t_e^2 & 4t_e^3 & 5t_e^4 \\ 0 & 0 & 2 & 6t_s & 12t_s^2 & 20t_s^3 \\ 0 & 0 & 2 & 6t_e & 12t_e^2 & 20t_e^3 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{bmatrix} = \begin{bmatrix} \theta_0 \\ \theta_f \\ \theta'_0 \\ \theta'_f \\ \theta''_0 \\ \theta''_f \end{bmatrix}$$

其中由于结果时间量的平移变换，$t_s = 0$；$t_e = \Delta t$，上述式子可简化为：

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & \Delta t & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2\Delta t & 3\Delta t & 4\Delta t & 5\Delta t \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 6\Delta t & 12\Delta t & 20\Delta t \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{bmatrix} = \begin{bmatrix} \theta_0 \\ \theta_f \\ \theta'_0 \\ \theta'_f \\ \theta''_0 \\ \theta''_f \end{bmatrix}$$

5.3.2 算法仿真

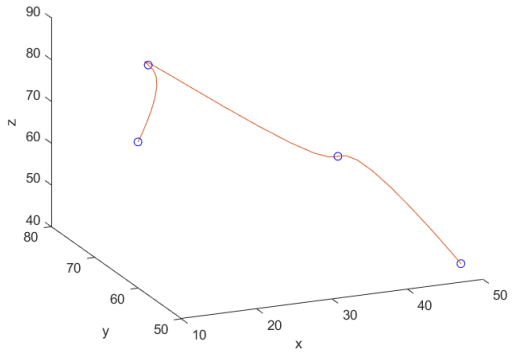


图 21 五次插值路径规划(蓝色点为特征点，红色为插值点)

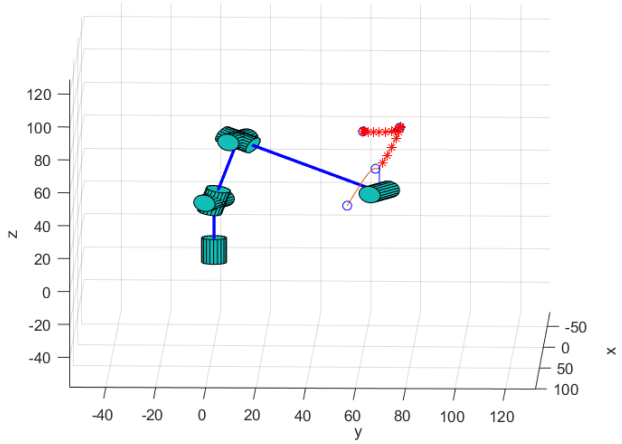


图 22 机械臂轨迹跟踪仿真

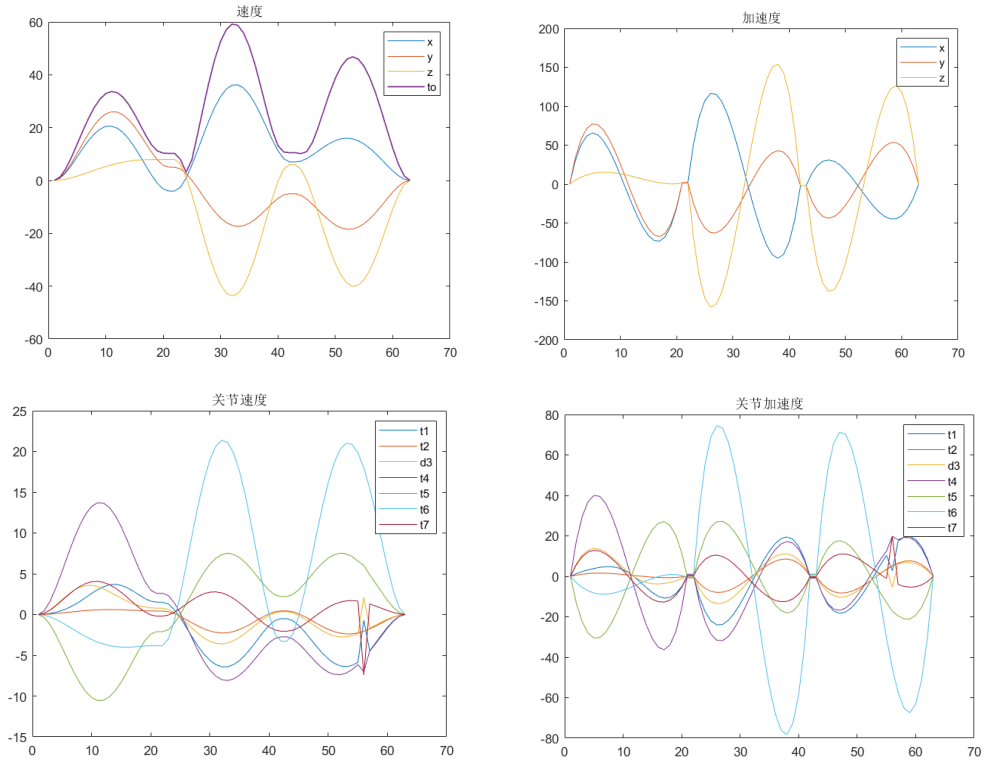


图 23 速度与加速度变化曲线

6.总结

在此次机器人学的期末课程设计方面，我们完成了 7 自由度机械臂设计与仿真，主要有以下几个任务如下：正运动学，求解雅可比矩阵（奇异值与奇异位型分析），逆运动学解析法，微分法+牛顿下山发画既定图案，三种轨迹规划与仿真。期间为了计算逆运动学解析解我们在这方面花费了不少时间，参考了许多资料，在参考文献中也有列出。 在本次期末课程设计中也有不足之处，由于时间分配不均，最后没有完成动力学模型的构建，也是有点小遗憾

我们在此次课程设计与本次机器人学课程中收获了许多，不仅对 matlab 更加的熟悉，也对机械臂有了更深的了解

7.参考文献

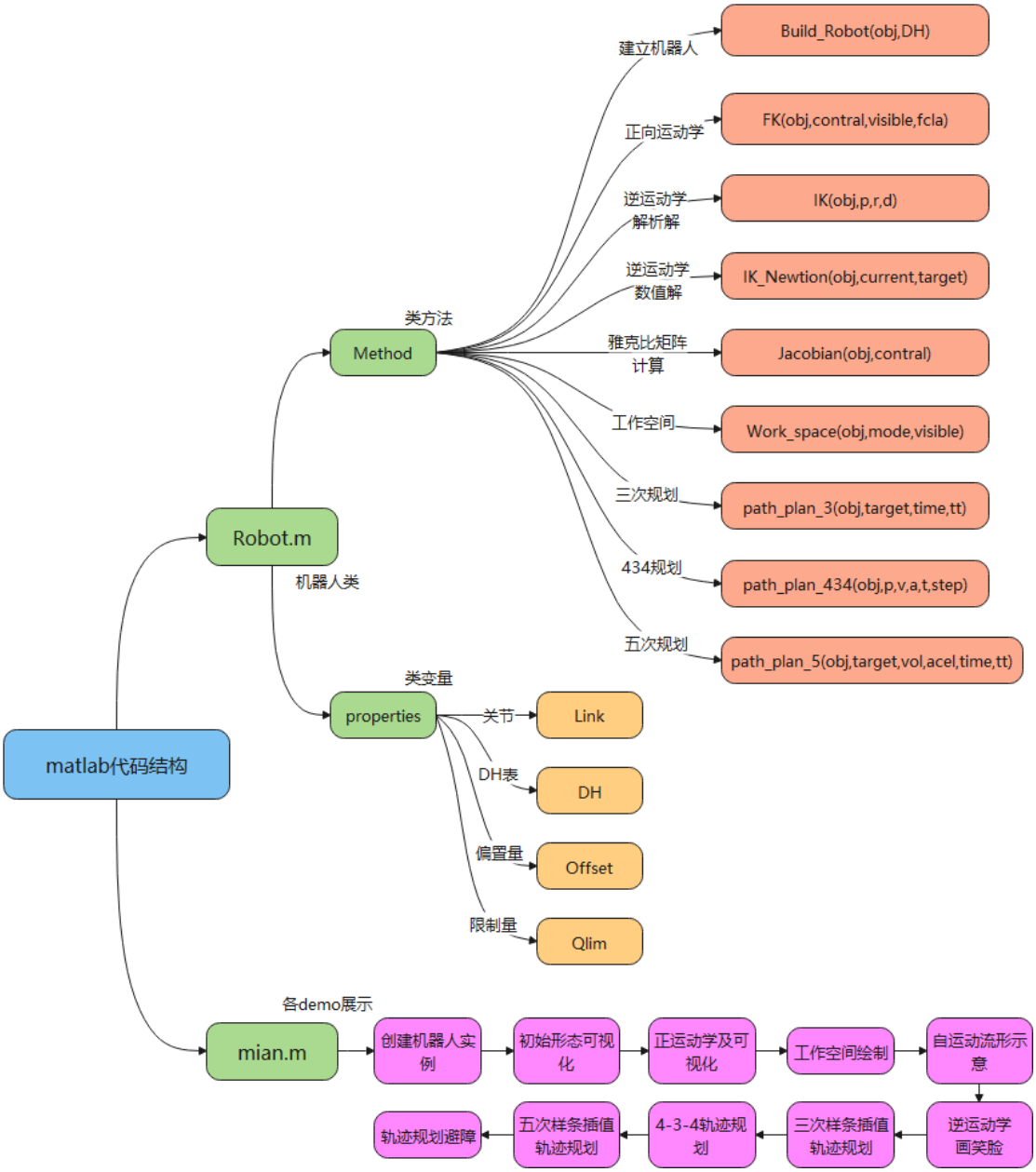
[1] https://www.guyuehome.com/5955

[2] https://blog.csdn.net/dmj9213/article/details/80624275

[3] https://blog.csdn.net/weixin_44229927/article/details/108056783

[4] https://www.guyuehome.com/5746

8.程序流程与代码附录



代码一 正运动学

```
%正向运动学解算
%参数：
%    控制量 ：各关节角度或伸缩量，1xN 的向量，N 为关节数
%    可视化：1（画图）0（不画图）
%    清除画面：1（清除图）0（不清除）
%返回值：
```

```
%   工作点相对于基座标变换矩阵
function ws=FK(obj,contral,visible,fcla)
     %关节转角赋值
     for i=2:obj.row+1
         if(obj.Link(i).Type==0)
            obj.Link(i).th = contral(i-1)*obj.ToRad + obj.Link(i).th;
         elseif(obj.Link(i).Type==1)
             obj.Link(i).dz = contral(i-1)  + obj.Link(i).dz;
         end
     end
    for i=1:obj.row+1
        obj = obj.Matrix_DH_Ln(i);
    end

     %计算工作点 R，P
     ws = [1 0 0 0;
           0 1 0 0;
           0 0 1 0;
           0 0 0 1];
    for i=1:obj.row+1
        ws=ws*obj.Link(i).A;
    end

    if(visible)
        radius    = 5;
        len       = 15;
        joint_col = 0;
        plot3(0,0,0,'ro');
        for i=2:obj.row+1
              obj.Link(i).A=obj.Link(i-1).A*obj.Link(i).A;
              obj.Link(i).p= obj.Link(i).A(:,4);
              obj.Link(i).n= obj.Link(i).A(:,1);
              obj.Link(i).o= obj.Link(i).A(:,2);
              obj.Link(i).a= obj.Link(i).A(:,3);
              obj.Link(i).R=[obj.Link(i).n(1:3),obj.Link(i).o(1:3),obj.Link(i).a(1:3)];
              obj.Connect3D(obj.Link(i-1).p,obj.Link(i).p,'b',2); hold on;
              obj.DrawCylinder(obj.Link(i-1).p, obj.Link(i-1).R * obj.Link(i).az, radius,len, joint_col); hold on;axis([-200,200,-200,200,-200,200]);
        end
        grid on;
```

```
        view(135,35);
        axis([-150,150,-150,150,-150,150]);
        xlabel('x');
        ylabel('y');
        zlabel('z');
        drawnow;

        if(fcla)
            cla;
        end
    end
end
```

代码二 逆运动学数值解

```
% @brief 数值法逆运动学求解
% @param obj 机器人类对象
% @param target 末端目标位姿（4X4）
% @param current 当前末端关节角度，初始值（1 X N）
% @return contral 保存解得的每个关节角和伸缩量（1 X N）
function contral = IK_Newtion(obj,current,target)

    contral = current; %保存当前状态
    %计算当前状态的雅可比矩阵
     J = Jacobian(obj,current);
     %计算误差
     current_T = obj.FK(current,0,0);
     err = Calculate_err(target,current_T);

     E = err' * err;
     if E < 1e-6   %结束迭代的条件
         return;
     end

     E_tmp=E;
     %开始迭代
     while 1

         tmp_th = zeros(1,obj.row);
         lambda = 1;
         while E - E_tmp <= 0
             dD = lambda * err;
             dth = pinv(J) * dD;

```

```
            %更新角度
            for i = 1:obj.row - 1
                if obj.Link(i).Type == 0
                    tmp_th(i) = current(i) + dth(i);
                    tmp_th(i) = obj.restrain_value(tmp_th(1,i),obj.Qlim(i,1),obj.Qlim(i,2));%限幅
                elseif obj.Link(i).Type == 1
                    tmp_th(i) = current(i) + dth(i);
                    tmp_th(i) = obj.restrain_value(tmp_th(1,i),obj.Qlim(i,1),obj.Qlim(i,2));%限幅
                end
            end

            %重新计算迭代后的误差
            current_T = obj.FK(tmp_th,0,0);
            err = Calculate_err(target,current_T);
            E_tmp = err' * err;

            %更新下山因子
            lambda = lambda/2.0;
        end

        current = tmp_th;
        E=E_tmp;
        J = Jacobian(obj,current);
        contral = current;%更新输出的逆解的角度

        if E_tmp < 1e-6
            break;
        end
    end
end
```

代码三 逆运动学解析解

```
%逆向运动解算
%参数:
%       p: 工作点 xyz 坐标，【x;y;z】
%       r: 工作点姿态
%       d: 移动关节的控制量
%返回值:
%       t: 各关节控制量 【2*7】两个解
function t=IK(obj,p,r,d)
```

```
    %杆长参数
     L1=30;
     L2=40+d;
     L3=40;
     L4=30;
     L5=15;
     L34=L3+L4;
     t=[];

     %pw（工作点） -> p7（关节 7）
     p=p-L5*r(:,3);

     %计算 theta1 和 theta3
     t1 = atan2(p(2),p(1));
     D = sqrt(p(1)^2 + p(2)^2+(p(3)-L1)^2);
     t3_1 = pi-acos( ((L2^2 + L34^2 - D^2)/(2*L2*L34) ));
     t3_2 = -(pi-acos( (L2^2 + L34^2 - D^2)/(2*L2*L34) ));
     t3=[t3_1 t3_2];
     of=obj.Offset;

     for j=1:2
         %计算 theta2
         k1 = -L2-L34*cos(t3(j));
         k2 = L34*sin(t3(j));
         a1 = L1-p(3);

         w1 = L34*sin(t3(j));
         w2 = L2+L34*cos(t3(j));
         a2 = p(1)*cos(t1) + p(2)*sin(t1);

         st2 = (w1*a1/k1 - a2)/(k2*w1/k1 - w2);
         ct2 = (w2*a1/k2 - a2)/(w2*k1/k2 - w1);
         t2 = atan2(st2,ct2);

         %代入 t1 t2 t3 d 用于计算 R14（关节 1-》关节 4 的旋转矩阵）
         obj.Link(2).th = t1 + of(1);
         obj.Link(3).th = t2 + of(2);
         obj.Link(4).dz = d + of(3);
         obj.Link(5).th = t3(j) + of(4);

         %计算 R14
         for i=1:obj.row+1
             obj = obj.Matrix_DH_Ln(i);
         end
```

```
53.         R14 = obj.Link(2).R*obj.Link(3).R*obj.Link(4).R*obj.L
    ink(5).R;
54.         R46 = (R14)\r;
55.
56.         %通过 z-y-z 的 euler 计算 t4 t5 t6
57.         [alf,bta ,gma] = obj.R2Euler(R46);
58.         t4=alf;
59.         t5=bta;
60.         t6=gma;
61.
62.         res=[t1, t2 ,d ,  t3(j), t4, t5, t6];
63.         t=cat(1,t,res);
64.     end
65. end
```

代码四 三次多项式曲线轨迹规划

```
1. %三次样条插值轨迹规划
2. %   参数：targrt（N*6）【x y z t1(x) t2(y) t3(z)】 N 个点
3. %         time：各点时间
4. %   返回值：pos：各时间的位置
5. %           v：各时间的速度
6. %           a：各时间的加速度
7. function  [pos,v,a]=path_plan_3(obj,target,time,tt)
8.     [r,c]=size(target);
9.     dt = diff(time);   %时间差分
10.     via_num = r-2;
11.
12.     %初始化计算矩阵
13.     T=zeros(4+via_num*4,c);
14.     params=zeros(4+via_num*4,4+via_num*4);
15.     A=zeros((r-1)*4,c);
16.     %构造化计算矩阵
17.     T(1,:) = target(1,:);
18.     for i=1:via_num
19.         T(i*2,:) = target(i+1,:);
20.         T(i*2+1,:) = target(i+1,:);
21.     end
22.     T(2+2*via_num,:) = target(r,:);
23.
24.     for i=1:r-1
25.         params(1+2*(i-1):2*i,1+4*(i-
    1):4*i) = [1 0 0 0;1 dt(i) dt(i)^2 dt(i)^3];
26.     end
```

```
    params(3+via_num*2,2)=1;
    params(4+via_num*2,2+via_num*4:4+via_num*4)=[1 2*dt(r-
1) 3*dt(r-1)^2];
    for i=1:via_num
        k=5+via_num*2;
        params(k+2*(i-1):k+1+2*(i-1),2+4*(i-1):7+4*(i-
1)) = [1   2*dt(i) 3*dt(i)^2  0  -1  0;
                                                    0   2
     6*dt(1)    0   0 -2];
    end
    %计算参数矩阵
    A = (params)\T;

    k=1;
    for i=1:r-1
        %位置计算式
        SX = @(t) A(1+4*(i-1),1) + A(2+4*(i-
1),1)*t + A(3+4*(i-1),1)*t^2 + A(4+4*(i-1),1)*t^3;
        SY = @(t) A(1+4*(i-1),2) + A(2+4*(i-
1),2)*t + A(3+4*(i-1),2)*t^2 + A(4+4*(i-1),2)*t^3;
        SZ = @(t) A(1+4*(i-1),3) + A(2+4*(i-
1),3)*t + A(3+4*(i-1),3)*t^2 + A(4+4*(i-1),3)*t^3;
        ST1 = @(t) A(1+4*(i-1),4) + A(2+4*(i-
1),4)*t + A(3+4*(i-1),4)*t^2 + A(4+4*(i-1),4)*t^3;
        ST2 = @(t) A(1+4*(i-1),5) + A(2+4*(i-
1),5)*t + A(3+4*(i-1),5)*t^2 + A(4+4*(i-1),5)*t^3;
        ST3 = @(t) A(1+4*(i-1),6) + A(2+4*(i-
1),6)*t + A(3+4*(i-1),6)*t^2 + A(4+4*(i-1),6)*t^3;
        %速度计算式
        VX = @(t) A(2+4*(i-1),1) + 2*A(3+4*(i-
1),1)*t + 3*A(4+4*(i-1),1)*t^2;
        VY = @(t) A(2+4*(i-1),2) + 2*A(3+4*(i-
1),2)*t + 3*A(4+4*(i-1),2)*t^2;
        VZ = @(t) A(2+4*(i-1),3) + 2*A(3+4*(i-
1),3)*t + 3*A(4+4*(i-1),3)*t^2;
        VT1 = @(t) A(2+4*(i-1),4) + 2*A(3+4*(i-
1),4)*t + 3*A(4+4*(i-1),4)*t^2;
        VT2 = @(t) A(2+4*(i-1),5) + 2*A(3+4*(i-
1),5)*t + 3*A(4+4*(i-1),5)*t^2;
        VT3 = @(t) A(2+4*(i-1),6) + 2*A(3+4*(i-
1),6)*t + 3*A(4+4*(i-1),6)*t^2;
        %加速度计算式
        AX = @(t) 2*A(3+4*(i-1),1) + 6*A(4+4*(i-1),1)*t;
        AY = @(t) 2*A(3+4*(i-1),2) + 6*A(4+4*(i-1),2)*t;
```

```
56.          AZ = @(t) 2*A(3+4*(i-1),3) + 6*A(4+4*(i-1),3)*t;
57.          AT1 = @(t) 2*A(3+4*(i-1),4) + 6*A(4+4*(i-1),4)*t;
58.          AT2 = @(t) 2*A(3+4*(i-1),5) + 6*A(4+4*(i-1),5)*t;
59.          AT3 = @(t)  2*A(3+4*(i-1),6) + 6*A(4+4*(i-1),6)*t;
60.          %插值
61.          for t=0:tt:dt(i)
62.              pos(k,:)=[SX(t),SY(t),SZ(t),ST1(t),ST2(t),ST3(t)]
   ;
63.              v(k,:)=[VX(t),VY(t),VZ(t),VT1(t),VT2(t),VT3(t)];
64.              a(k,:)=[AX(t),AY(t),AZ(t),AT1(t),AT2(t),AT3(t)];
65.              k=k+1;
66.          end
67.      end
68. end
```

代码五 4-3-4 轨迹规划

```
1. % @brief 规划一个变量的轨迹，N 为轨迹的段数，输入需要经过的轨迹点数
   量为 N+1 个
2. % @param p 需要经过的轨迹点（1 X N+1）
3. % @param v 起始和末端点速度（1 X 2）
4. % @param a 起始和末端点加速度（1 X 2）
5. % @param t 每一段轨迹的时间（1 X N）
6. % @param step 采样时间
7. % @return point 按照采样时间取得的点集（一行）
8. function point = path_plan_434(obj,p,v,a,t,step)
9.
10.      tmp = size(p);
11.      N = tmp(1,2) - 1;
12.
13.      %构造 theta 矩阵
14.      theta = zeros(4*N+2,1);
15.      theta(1:3,1) = [p(1,1);v(1,1);a(1,1)];
16.      theta(4*N:4*N+2,1) = [p(1,N+1);v(1,2);a(1,2)];
17.
18.      index = 4;
19.      for i = 2:1:N
20.          theta(index,1) = p(1,i);
21.          theta(index+1,1) = p(1,i);
22.          index = index + 4;
23.      end
24.
```

```
    %构造 M 矩阵
    M = zeros(4*N+2,4*N+2);
    M(1:3,1:3) = [1,0,0;
                  0,1,0;
                  0,0,2];
    M(4:7,1:8) = [1 , t(1,1) , t(1,1)^2, t(1,1)^3, t(1,1)^4, 0 , 0 , 0;
                  0 , 0 , 0 ,0 , 0 , 1 , 0 , 0;
                  0 , 1 , 2*t(1,1) , 3*t(1,1)^2 , 4*t(1,1)^3 , 0 , -1 , 0;
                  0 , 0 , 2 , 6*t(1,1) , 12*t(1,1)^2 , 0 , 0 ,-2];
    index_x = 8;
    index_y = 6;
    for i = 1:1:N-2
        M(index_x:index_x+3,index_y:index_y+6) = [1 , t(1,i+1) , t(1,i+1)^2, t(1,i+1)^3, 0 , 0 , 0;
                                                  0 , 0 , 0 , 0 , 1 , 0 , 0;
                                                  0 , 1 , 2*t(1,i+1) , 3*t(1,i+1)^2 , 0 , -1 , 0;
                                                  0 , 0 , 2 , 6*t(1,i+1) , 0 , 0 ,-2];
        index_x = index_x + 4;
        index_y = index_y + 4;
    end

    M(4*N:4*N+2,4*N-2:4*N+2) = [1,t(1,N),t(1,N)^2,t(1,N)^3,t(1,N)^4;
                                0,1,2*t(1,N),3*t(1,N)^2,4*t(1,N)^3;
                                0,0,2,6*t(1,N),12*t(1,N)^2];


    %求解 C 矩阵
    C = M\theta;

    %采样
    Ct = zeros(5*N,1);
    Ct(1:5,1) = C(1:5,1);
    Ct(5*N-4:5*N,1) = C(4*N-2:4*N+2,1);
    for i = 0:1:N-3
        Ct(6 + 5*i:9 + 5*i,1) = C(6+4*i:9+4*i,1);
    end
```

```
59.
60.     total = sum(t);%总时间
61.     tt = 0:step:total;
62.     t_step = 0;
63.     index_begin = 1;
64.     for i = 0:1:N-1
65.         F_p = @(T) Ct(1+5*i,1) + Ct(2+5*i,1)*(T-
   t_step) + Ct(3+5*i,1)*(T-t_step)^2 + Ct(4+5*i,1)*(T-
   t_step)^3 + Ct(5+5*i,1)*(T-t_step)^4;
66.         F_v = @(T) Ct(2+5*i,1) + 2*Ct(3+5*i,1)*(T-
   t_step) + 3*Ct(4+5*i,1)*(T-t_step)^2 + 4*Ct(5+5*i,1)*(T-
   t_step)^3;
67.         F_a = @(T) 2*Ct(3+5*i,1) + 6*Ct(4+5*i,1)*(T-
   t_step) + 12*Ct(5+5*i,1)*(T-t_step)^2;
68.
69.         index_end = t(1,i+1)/step + index_begin;
70.         for j = index_begin:index_end
71.             point(1,j) = F_p(tt(1,j));
72.             point(2,j) = F_v(tt(1,j));
73.             point(3,j) = F_a(tt(1,j));
74.             point(4,j) = tt(1,j);
75.         end
76.
77.         index_begin = index_end;
78.         t_step = t_step + t(1,i+1);
79.     end
80. end
```

代码六 五次多项式轨迹规划

```
1. %五次样条插值轨迹规划
2. %   参数：targrt（N*6）【x y z t1(x) t2(y) t3(z)】 N个点
3. %         vol：各点速度
4. %         acel：各店加速度
5. %         time：各点时间
6. %   返回值：pos：各时间的位置
7. %           v：各时间的速度
8. %           a：各时间的加速度
9. function [pos,v,a]=path_plan_5(obj,target,vol,acel,time,tt)
10.     S = @(a,t) a(1) + a(2)*t + a(3)*t^2 + a(4)*t^3 + a(5)*t^4
    + a(6)*t^5;
```

```
    V = @(a,t) a(2) + 2*a(3)*t + 3*a(4)*t^2 + 4*a(5)*t^3 + 5*a(6)*t^4;
    A = @(a,t) 2*a(3) + 2*3*a(4)*t + 3*4*a(5)*t^2 + 4*5*a(6)*t^3;

    DT = diff(time);
    n=length(target);
    k=1;
    for i=1:n-1
        dt=DT(i);
        T=zeros(6,3);
        for j=1:3
            T(:,j)=[target(i,j),target(i+1,j),vol(i,j),vol(i+1,j),acel(i,j),acel(i+1,j)]';
        end
        params=[1,  0,  0,  0,  0,  0;
              1,dt,dt^2,dt^3,dt^4,dt^5
              0,  1,  0,  0,  0,  0;
              0,1,2*dt,3*dt^2,4*dt^3,5*dt^4;
              0,  0,  2,  0,  0,  0
              0,0,2,6*dt,12*dt^2,20*dt^3];
        C = (params)\T;
        for t=0:tt:dt
            pos(k,:)=[S(C(:,1),t),S(C(:,2),t),S(C(:,3),t)];
            v(k,:)=[V(C(:,1),t),V(C(:,2),t),V(C(:,3),t)];
            a(k,:)=[A(C(:,1),t),A(C(:,2),t),A(C(:,3),t)];
            k=k+1;
        end
    end
end
```