

机器人学课程设计

姓名：朱楷文

学号：520030910178

班级：F2003001

1 机械臂构建

1.1 实现思路

Puma560 机械臂的 D-H 参数如表 1 所示, 其中 θ_i , $i = 1, 2, \dots, 6$ 为关节变量, a_2, d_3, a_3, d_4 为定值. 本设计中, 各关节变量的取值范围均为 $(-90^\circ, 90^\circ)$. 据此建立各连杆并将其连接, 即可得到机械臂.

表 1: Puma560 机械臂的 D-H 参数表

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	-90°	0	0	θ_2
3	0	a_2	d_3	θ_3
4	-90°	a_3	d_4	θ_4
5	90°	0	0	θ_5
6	-90°	0	0	θ_6

1.2 结果

构建出的机械臂如图 1 所示, 其各个关节变量均设为 0, a_2, d_3, a_3, d_4 分别设为 100, 20, 10, 100.

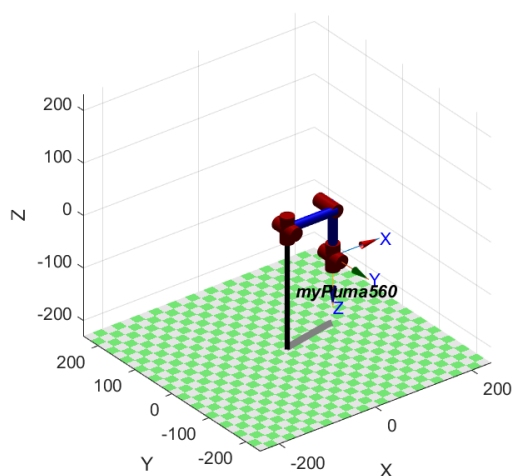


图 1: 机械臂示意图

1.3 代码

构建机械臂的函数代码如下.

```
1 function my_p560 = create_robot(a2,d3,a3,d4)
2     % set the D-H parameters
3     qlim = [-pi/2, pi/2];
4     theta(1) = 0; d(1) = 0;    a(1) = 0;    alpha(1) = 0;
5     theta(2) = 0; d(2) = 0;    a(2) = 0;    alpha(2) = -pi/2;
6     theta(3) = 0; d(3) = d3;   a(3) = a2;   alpha(3) = 0;
7     theta(4) = 0; d(4) = d4;   a(4) = a3;   alpha(4) = -pi/2;
8     theta(5) = 0; d(5) = 0;    a(5) = 0;    alpha(5) = pi/2;
9     theta(6) = 0; d(6) = 0;    a(6) = 0;    alpha(6) = -pi/2;
10    % create the links
11    L(1) = Link([theta(1), d(1), a(1), alpha(1), 0], 'modified');
12    L(2) = Link([theta(2), d(2), a(2), alpha(2), 0], 'modified');
13    L(3) = Link([theta(3), d(3), a(3), alpha(3), 0], 'modified');
14    L(4) = Link([theta(4), d(4), a(4), alpha(4), 0], 'modified');
15    L(5) = Link([theta(5), d(5), a(5), alpha(5), 0], 'modified');
16    L(6) = Link([theta(6), d(6), a(6), alpha(6), 0], 'modified');
17    L(1).qlim = qlim;
18    L(2).qlim = qlim;
19    L(3).qlim = qlim;
20    L(4).qlim = qlim;
21    L(5).qlim = qlim;
22    L(6).qlim = qlim;
23    % create the robot
24    my_p560 = SerialLink(L, 'name', 'myPuma560');
25 end
```

2 工作空间可视化

2.1 实现思路

机械臂的工作空间是末端执行器可达到的空间范围, 即, 对于工作空间中的一个点 \mathbf{p} , 在关节空间中存在一个点 \mathbf{q} , 使得 $\mathbf{f}(\mathbf{q}) = \mathbf{p}$, 其中 \mathbf{f} 为正运动学函数. 因此, 对关节空间中的每个点应用正运动学函数, 即可得到工作空间.

实际操作中, 为了可视化工作空间, 由于不可能遍历关节空间中的所有点, 我们在其中进行均匀的随机采样, 对采样得到的每个点应用正运动学函数, 绘制出得到的位置. 若样本足够代表整个关节空间, 即可实现工作空间的可视化. 本设计选择样本容量为 20000.

2.2 结果

构建参数为 $a_2 = 100, d_3 = 20, a_3 = 10, d_4 = 100$ 的机械臂, 采样并计算后得到的工作空间如图 2 所示, 其中每个红点表示采样并计算后得到的工作空间中的一个点. 工作空间在 X-Y 平面, Y-Z 平面, Z-X 平面上的投影如图 3 所示.

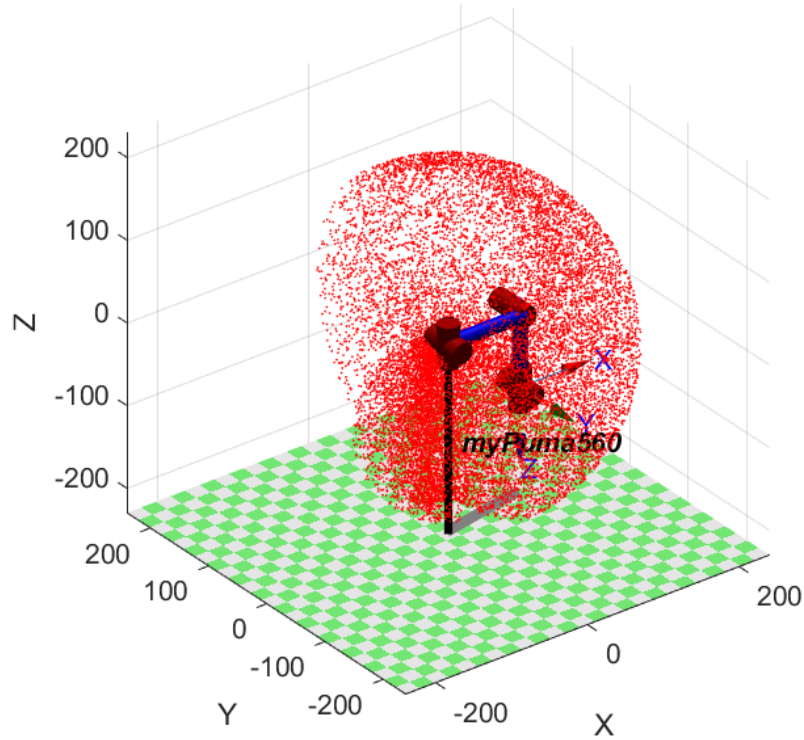


图 2: 机械臂的工作空间

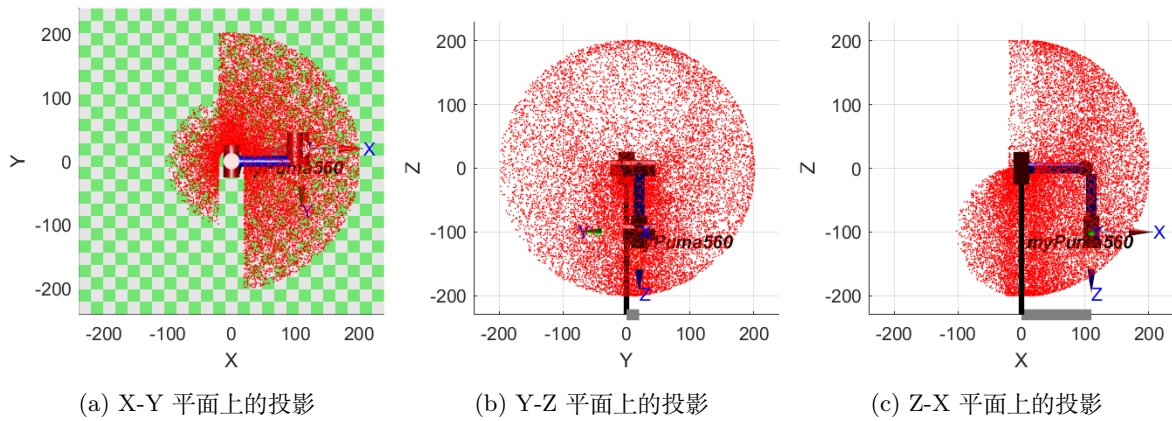


图 3: 工作空间在各平面上的投影

2.3 代码

可视化工作空间的代码如下.

```

1 clear; close all; clc;
2
3 % create the robot with a2=100,d3=20,a3=10,d4=100
4 my_p560 = create_robot(100,20,10,100);
5
6 % plot the robot
7 my_p560.plot(zeros(1,6));
8 hold on;
9
10 % sample parameters randomly to get the workspace
11 N = 20000; % num of samples
12 for i = 1:N
13     T = my_p560.fkine(-pi/2 + rand(1, 6) * pi);
14     pos = transl(T);
15     x(i) = pos(1);
16     y(i) = pos(2);
17     z(i) = pos(3);
18 end
19
20 % plot the positions
21 scatter3(x,y,z, 1, 'r', 'filled');

```

3 避障路径规划

3.1 实现思路

欲控制机械臂避开障碍物从起点到达终点, 只需观察出安全的末端执行器路径, 选取合适的锚点, 利用逆运动学计算出各锚点对应的关节变量, 在这些关节变量之间进行平滑的插值, 最后令机械臂按照插值结果运动即可.

本设计中, 障碍物为长方体, 中心为 (100, 0, 50), 大小为 (200, 30, 150), 路径的起点为 (100, 100, 10), 终点为 (100, -100, 10). 因此取锚点为: 起点 (100, 100, 10), 起点正上方 (100, 100, 200), 终点正上方 (100, -100, 200), 终点 (100, -100, 10). 选取机械臂参数为 $a_2 = 200, d_3 = 20, a_3 = 10, d_4 = 200$, 基座位置为 (-100, 0, 0). 但是, 在 Matlab 中调用函数 `ikine` 求解逆运动学时出现了无法收敛的错误. 推测这是由于传入的末端执行器朝向无法达到. 可达的朝向是难以获知的, 而事实上, 这里我们只关心末端执行器的位置而并不关心其朝向, 而位置仅由前三个关节变量决定. 因此考虑推导其前三个关节变量的解析解, 后三个关节变量设为 0 即可.

给定末端位置 (x, y, z) , 下面求解其逆运动学.

机械臂前 4 个关节的齐次变换矩阵计算如下, 其中 θ_4 设为 0.

$${}^0T_1 = \begin{pmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad {}^1T_2 = \begin{pmatrix} c_2 & -s_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s_2 & -c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$${}^2T_3 = \begin{pmatrix} c_3 & -s_3 & 0 & 200 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 20 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad {}^3T_4 = \begin{pmatrix} 1 & 0 & 0 & 10 \\ 0 & 0 & 1 & 200 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

进而可得

$${}^0T_2 = {}^0T_1 T_2 = \begin{pmatrix} c_1 c_2 & -c_1 s_2 & -s_1 & 0 \\ s_1 c_2 & -s_1 s_2 & c_1 & 0 \\ -s_2 & -c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$${}^0T_3 = {}^0T_2 T_3 = \begin{pmatrix} c_1 c_{23} & -c_1 s_{23} & -s_1 & 200c_1 c_2 - 20s_1 \\ s_1 c_{23} & -s_1 s_{23} & c_1 & 200s_1 c_2 + 20c_1 \\ -s_{23} & -c_{23} & 0 & -200s_2 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$${}^0T_4 = {}^0T_3 T_4 = \begin{pmatrix} c_1 c_{23} & -c_1 s_{23} & -s_1 & 10c_1 c_{23} - 200c_1 s_{23} + 200c_1 c_2 - 20s_1 \\ s_1 c_{23} & -s_1 s_{23} & c_1 & 10s_1 c_{23} - 200s_1 s_{23} + 200s_1 c_2 + 20c_1 \\ -s_{23} & -c_{23} & 0 & -10s_{23} - 200c_{23} - 200s_2 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

由于后三个关节变量不影响末端执行器位置, 因此末端位置为

$$\mathbf{p} = \begin{pmatrix} 10c_1 c_{23} - 200c_1 s_{23} + 200c_1 c_2 - 20s_1 \\ 10s_1 c_{23} - 200s_1 s_{23} + 200s_1 c_2 + 20c_1 \\ -10s_{23} - 200c_{23} - 200s_2 \end{pmatrix}.$$

于是可以列出方程组

$$x = 10c_1 c_{23} - 200c_1 s_{23} + 200c_1 c_2 - 20s_1 \quad (1a)$$

$$y = 10s_1 c_{23} - 200s_1 s_{23} + 200s_1 c_2 + 20c_1 \quad (1b)$$

$$z = -10s_{23} - 200c_{23} - 200s_2 \quad (1c)$$

设

$$A = 10c_{23} - 200s_{23} + 200c_2, \quad (2)$$

则式 (1a) 和式 (1b) 可写成

$$x = Ac_1 - 20s_1 \quad (3a)$$

$$y = As_1 + 20c_1 \quad (3b)$$

$(3a)^2 + (3b)^2$, 得

$$A^2 = x^2 + y^2 - 400, \quad (4)$$

$(2)^2 + (1c)^2$, 整理可得

$$\frac{A^2 + z^2}{100} = 801 + 40(c_3 - 20s_3), \quad (5)$$

将 (4) 代入上式, 并设

$$\rho^2 = x^2 + y^2 + z^2, \quad (6)$$

整理可得,

$$c_3 - 20s_3 = \frac{\rho^2 - 80500}{4000}, \quad (7)$$

设

$$B = \frac{\rho^2 - 80500}{4000}, \quad (8)$$

将其代入式 (7), 并应用辅助角公式, 可得

$$\sqrt{401} \sin(\theta_3 + \phi_1) = B, \quad (9)$$

其中 $\phi_1 = \arctan(\frac{1}{-20}) + \pi$. 由此即可解出 θ_3 .

将解得的 θ_3 代回式 (1c), 整理可得

$$(B + 20)s_2 + (s_3 + 20c_3)c_2 = \frac{z}{-10}. \quad (10)$$

设

$$C = B + 20, \quad (11)$$

$$D = s_3 + 20c_3. \quad (12)$$

注意到, $C = (\rho^2 - 500)/4000$, 而本设计选取的路径上的点显然都满足到基座的距离大于 $\sqrt{500}$, 因此可以保证 $C > 0$. 从而, 对式 (10) 应用辅助角公式, 并将式 (11) 和式 (12) 代入, 可得

$$\sqrt{C^2 + D^2} \sin(\theta_2 + \phi_2) = \frac{z}{-10}, \quad (13)$$

其中 $\phi_2 = \arctan(D/C)$. 由此即可解出 θ_2 .

(3b) $\times A - (3a) \times 20$, 结合式 (4), 整理可得

$$s_1 = \frac{yA - 20x}{x^2 + y^2}. \quad (14)$$

将解得的 θ_2, θ_3 代入式 (2) 即可得到 A 的值. 考虑到 $\theta_1 \in (-\pi/2, \pi/2)$, 由式 (14) 立刻得到

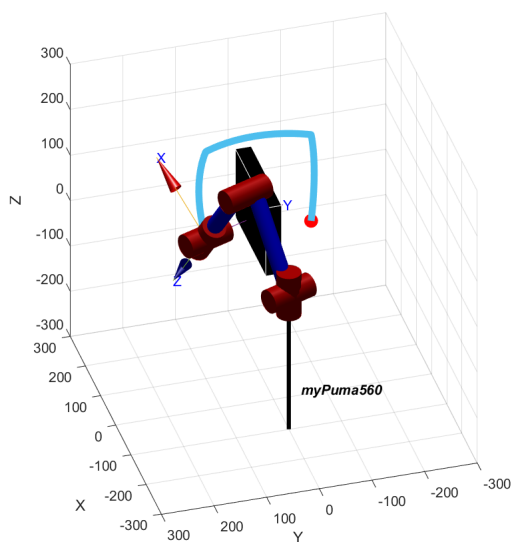
$$\theta_1 = \arcsin\left(\frac{yA - 20x}{x^2 + y^2}\right). \quad (15)$$

至此, $\theta_1, \theta_2, \theta_3$ 的解析解已全部可以给出. 需要说明的是, θ_2, θ_3 的值分别由式 (13) 和式 (9) 给出, 但这两个式子各自可能存在两个解. 因此, 在计算出可能的解后, 应将结果代回正运动学公式 (1) 进行验证. 此外, 还需要考虑 θ_2, θ_3 的值应在 $(-\pi/2, \pi/2)$ 范围内. 具体的实施细节见下文列出的 `my_ikine` 函数代码.

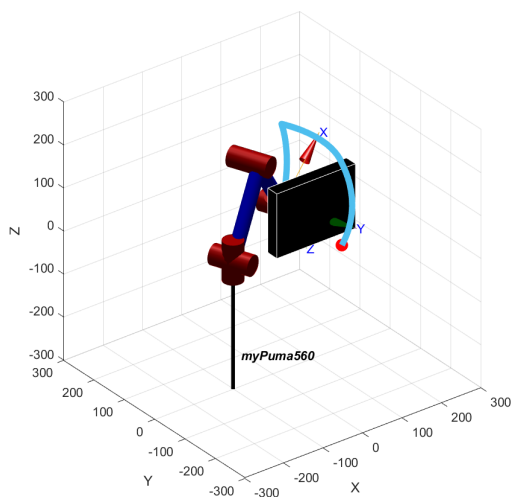
3.2 结果

按照以上思路, 计算出的机械臂从起点到终点的工作空间路径如图 4 所示, 各子图展示了机械臂在路径的起点、中点、终点时其不同视角下的姿态. 图中, 两个红色的点代表起点和终点, 蓝色的曲线代表路径.

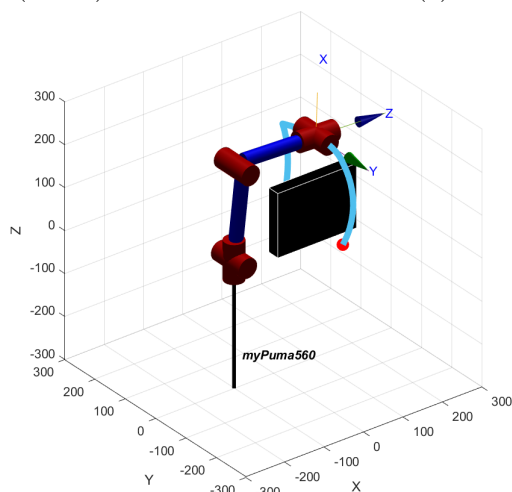
机械臂运动过程中, 其各关节角度的变化曲线如图 5 所示. 注意 $\theta_4, \theta_5, \theta_6$ 始终为 0, 因而其曲线重合.



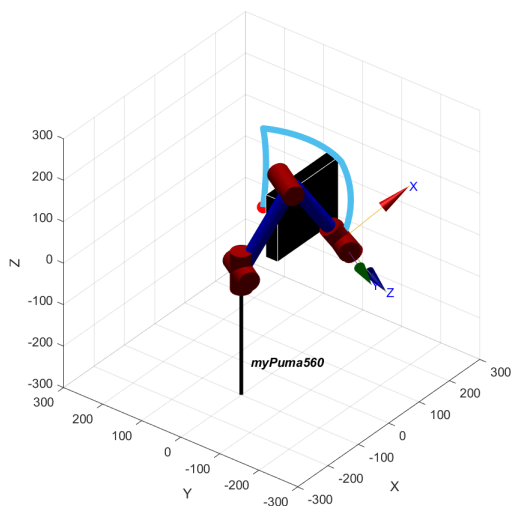
(a) 起点处 (视角 1)



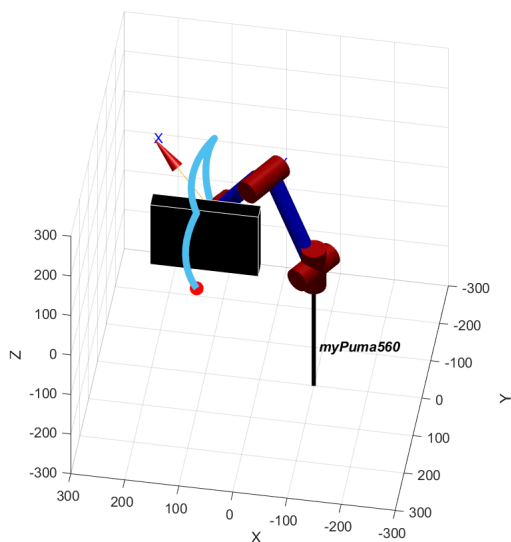
(b) 起点处 (视角 2)



(c) 中点处



(d) 终点处 (视角 1)



(e) 终点处 (视角 2)

图 4: 机械臂从起点到终点的工作空间路径及在运动中的姿态

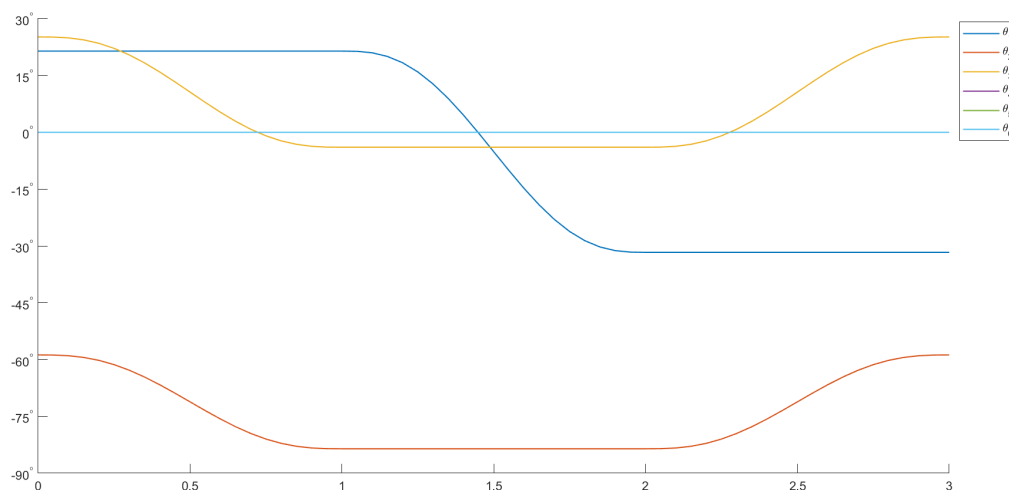


图 5: 机械臂各关节角度的变化

3.3 代码

计算逆运动学的代码如下.

```

1 function [q1,q2,q3] = my_ikine(x,y,z)
2     % determine whether two floating numbers are equal
3     eq = @(a,b) abs(a-b) < 1e-5;
4
5     function [x,y,z] = my_fkine(q1,q2,q3)
6         % calculate the forward kinematics
7         x = 10 * cos(q1) * cos(q2+q3) - 200 * cos(q1) * sin(q2+q3) ...
8             + 200 * cos(q1) * cos(q2) - 20 * sin(q1);
9         y = 10 * sin(q1) * cos(q2+q3) - 200 * sin(q1) * sin(q2+q3) ...
10            + 200 * sin(q1) * cos(q2) + 20 * cos(q1);
11        z = -10 * sin(q2+q3) - 200 * cos(q2+q3) - 200 * sin(q2);
12    end
13
14    function [q3_1, q3_2] = get_q3(x,y,z)
15        rho2 = x*x + y*y + z*z;
16        B = (rho2 - 80500) / 4000;
17        phi1 = atan(1/(-20)) + pi;
18        q3_p_phi1_m_pi = asin(-B/sqrt(401)); % q3 + phi1 - pi
19        if tan(q3_p_phi1_m_pi) < -20
20            q3_p_phi1_m_pi_2 = -pi - q3_p_phi1_m_pi;
21        else
22            q3_p_phi1_m_pi_2 = 1453;
23        end
24        q3_1 = q3_p_phi1_m_pi - phi1 + pi;
25        q3_2 = q3_p_phi1_m_pi_2 - phi1 + pi;
26    end

```



```

27
28 function [q2_1,q2_2] = get_q2(z,q3)
29     C = cos(q3) - 20 * sin(q3) + 20;
30     D = sin(q3) + 20 * cos(q3);
31     phi2 = atan(D / C);
32     if (-1 <= -z/10/sqrt(C*C + D*D)) && (-z/10/sqrt(C*C + D*D) <= 1)
33         q2_p_phi2 = asin(-z/10/sqrt(C*C + D*D)); % q2 + phi2
34         if q2_p_phi2 > 0
35             q2_p_phi2_2 = pi - q2_p_phi2;
36         else
37             q2_p_phi2_2 = -pi - q2_p_phi2;
38         end
39         q2_1 = q2_p_phi2 - phi2;
40         q2_2 = q2_p_phi2_2 - phi2;
41     else
42         q2_1 = 1453;
43         q2_2 = 1453;
44     end
45 end
46
47 function q1 = get_q1(x,y,q2,q3)
48     A = 10 * cos(q2+q3) - 200 * sin(q2+q3) + 200 * cos(q2);
49     q1 = asin((y*A - 20*x) / (x*x + y*y));
50 end
51
52 [q3_1,q3_2] = get_q3(x,y,z);
53 for q3 = [q3_1,q3_2]
54     if (q3 > -pi/2) && (q3 < pi/2)
55         [q2_1,q2_2] = get_q2(z,q3);
56         for q2 = [q2_1,q2_2]
57             if (q2 > -pi/2) && (q2 < pi/2)
58                 q1 = get_q1(x,y,q2,q3);
59                 [x1,y1,z1] = my_fkine(q1,q2,q3);
60                 if eq(x,x1) && eq(y,y1) && eq(z,z1)
61                     % [q1,q2,q3] is indeed a solution
62                     return
63                 end
64             end
65         end
66     end
67 end
68

```

```

69     % fail to solve
70     error("Solution not found!");
71 end

```

规划避障路径的代码如下.

```

1  clear; close all; clc;
2
3  % create the robot with a2=200,d3=20,a3=10,d4=200
4  my_p560 = create_robot(200,20,10,200);
5
6  % change the base
7  offset = 100;
8  my_p560.base = [-offset, 0, 0];
9
10 % plot the obstacle
11 center = [100, 0, 50];
12 size = [200, 30, 150];
13 origin = center - size/2;
14 vertex_index = [0,0,0; 0,0,1; 0,1,0; 0,1,1; 1,0,0; 1,0,1; 1,1,0; 1,1,1];
15 % compute the positions of eight vertices
16 vertex_pos = origin + vertex_index.*size;
17 % specify the six facets
18 facet=[1,2,4,3; 1,2,6,5; 1,3,7,5; 2,4,8,6; 3,4,8,7; 5,6,8,7];
19 patch('Vertices',vertex_pos, 'Faces',facet, 'EdgeColor','white');
20 hold on;
21 xlim([-300, 300]);
22 ylim([-300, 300]);
23 zlim([-300, 300]);
24
25 % compute the joint variables corresponding to the four anchor points
26 pini = [100, 100, 10]; % the start point
27 pmid1 = [100, 100, 200]; % the upper middle point
28 pmid2 = [100, -100, 200]; % the lower middle point
29 pend = [100, -100, 10]; % the end point
30 qini = zeros(1,6); qend = zeros(1,6);
31 qmid1 = zeros(1,6); qmid2 = zeros(1,6);
32 [qini(1),qini(2),qini(3)] = my_ikine(pini(1)+offset, pini(2), pini(3));
33 [qmid1(1),qmid1(2),qmid1(3)] = my_ikine(pmid1(1)+offset, pmid1(2), pmid1(3));
34 [qmid2(1),qmid2(2),qmid2(3)] = my_ikine(pmid2(1)+offset, pmid2(2), pmid2(3));
35 [qend(1),qend(2),qend(3)] = my_ikine(pend(1)+offset, pend(2), pend(3));
36
37 % interpolate the joint variables smoothly

```

```

38 t0 = [0 : 0.05 : 1]';
39 q1 = mtraj(@tpoly, qini, qmid1, t0);
40 q2 = mtraj(@tpoly, qmid1, qmid2, t0);
41 q3 = mtraj(@tpoly, qmid2, qend, t0);
42 q = [q1; q2(2:end,:); q3(2:end,:)];
43
44 % plot the path of the end effector
45 T = my_p560.fkine(q);
46 p = transl(T);
47 scatter3(pini(1), pini(2), pini(3), 100, 'r', 'filled');
48 scatter3(pend(1), pend(2), pend(3), 100, 'r', 'filled');
49 plot3(p(:,1), p(:,2), p(:,3), 'LineWidth',5);
50 my_p560.plot(q);
51
52 waitfor(gcf);
53
54 % plot the joint variables
55 hold on;
56 t = [0 : 0.05 : 3]';
57 for i = 1:6
58     plot(t, q(:,i), ...
59         'DisplayName',strcat('\theta_',num2str(i)), 'LineWidth',1);
60 end
61 legend;
62 ylim([-pi/2, pi/6]);
63 yticks(-pi/2 : pi/12 : pi/6);
64 yticklabels({'-90^\circ', '-75^\circ', '-60^\circ', '-45^\circ', ...
65             '-30^\circ', '-15^\circ', '0^\circ', '15^\circ', '30^\circ'});

```