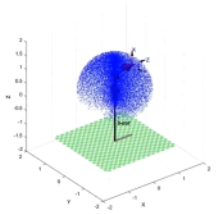
rtbdemo轨迹规划1

2024年12月8日 10:22

机器人工具箱 工作空间可视化

关节空间随机生成变量 —fkine→ 变换矩阵 —transl→ 三维坐标



rand函数：

X = rand returns a single uniformly distributed random number in the interval (0,1).

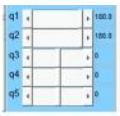
在[m, n]内随机生成一个数字：

$$m + rand * (m - n)$$

随机关节空间变量：

$$q = q_{min} + rand \times (q_{max} - q_{min})$$

默认关节范围：



关节限制是Link类的一个属性：Link.qlim

定义关节限制：

```
L(1).qlim = [-150, 150]/180*pi;
L(2).qlim = [-100, 90]/180*pi;
L(3).qlim = [-90, 90]/180*pi;
L(4).qlim = [-100, 100]/180*pi;
L(5).qlim = [-180, 180]/180*pi;
```

```
L(1) = Link('revolute','d',0.216,'a',0,'alpha',pi/2, ...
            'qlim', [-150, 150]/180*pi);
L(2) = Link('revolute','d',0,'a',0.5,'alpha',0,'offset',pi/2,...
            'qlim', [-100, 90]/180*pi);
```

然后你像查看相应的呃关节的一个关节限制的话

```
num = 30000;
```

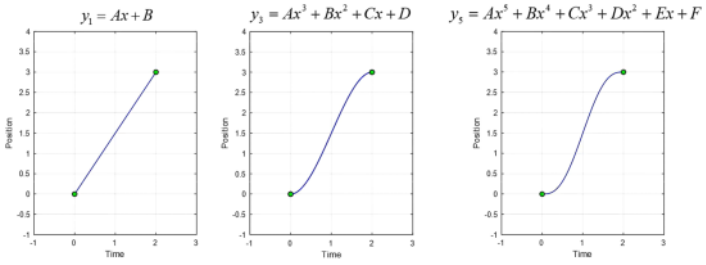
然后n u m呢是迭代次数

```
P = zeros(num, 3);

for i=1:num

    q1 = L(1).qlim(1) + rand * ( L(1).qlim(2) - L(1).qlim(1) );
    q2 = L(2).qlim(1) + rand * ( L(2).qlim(2) - L(2).qlim(1) );
    q3 = L(3).qlim(1) + rand * ( L(3).qlim(2) - L(3).qlim(1) );
    q4 = L(4).qlim(1) + rand * ( L(4).qlim(2) - L(4).qlim(1) );
    q5 = L(5).qlim(1) + rand * ( L(5).qlim(2) - L(5).qlim(1) );

    q = [q1 q2 q3 q4 q5];

    T = Five_dof.fkine(q);

    P(i, :) = transl(T);
end

plot3(P(:,1), P(:,2), P(:,3), 'b.', 'markersize', 1);
```

轨迹：时间、位置、速度、加速度

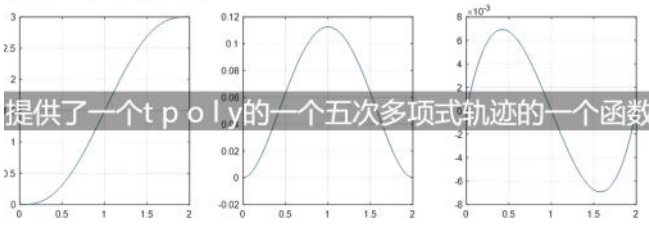
给定时间 0——2s，位置0——3

$y_1 = Ax + B$ $y_3 = Ax^3 + Bx^2 + Cx + D$ $y_5 = Ax^5 + Bx^4 + Cx^3 + Dx^2 + Ex + F$



五次多项式轨迹 tpoly:

```
t = linspace(0, 2, 51); ==0:0.04:2
[P, dP, ddP] = tpoly(0, 3, t);
```

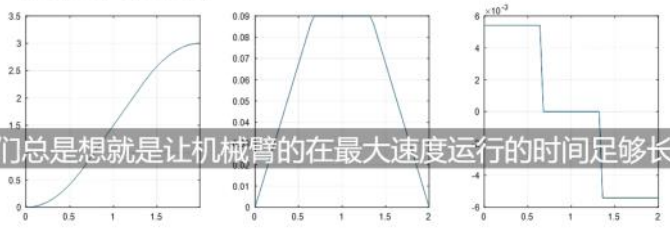


提供了一个t p o l y的一个五次多项式轨迹的一个函数

指定初末速度：[P, dP, ddP] = tpoly(0, 3, 51, 0.02, 0.01);

混合曲线轨迹 lspb:

```
t = linspace(0, 2, 51);
[P, dP, ddP] = lspb(0, 3, t);
```

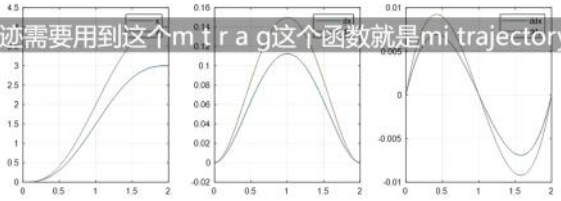


我们总是想就是让机械臂的在最大速度运行的时间足够长

指定最大速度：[P, dP, ddP] = lspb(0, 3, 51, 0.1);

多维轨迹 mtraj: (0, 0) ⟶ (3, 4)

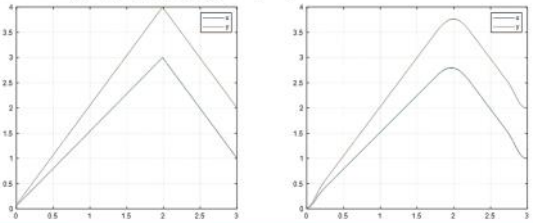
```
t = linspace(0, 2, 51);
[P, dP, ddP] = mtraj(@tpoly, [0, 0], [3, 4], t);
```



二维轨迹需要用到这个m t r a g这个函数就是mi trajectory

多雏多段轨迹 mstraj: (0, 0) —2s→ (3, 4) —1s→ (1, 2)

```
TRAJ = mstraj(WP, QDMAX, TSEG, Q0, DT, TACC, OPTIONS)
wp = [0, 0; 3, 4; 1, 2];
P1 = mstraj(wp, [ ], [2, 1], [ ], 0.04, 0);
P2 = mstraj(wp, [ ], [2, 1], [ ], 0.04, 0.5);
```



这时候就要用到多维多端轨迹m s t r a g这个函数嗯

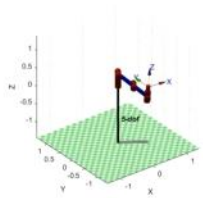
81.运动轨迹规划2

2024年12月8日 10:40

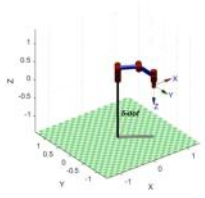
机器人工具箱 轨迹规划（2）

给定位置：（0.7，-0.5，0） ⟶ （0.7，0.5，0.5）

```
T1 = transl(0.7, -0.5, 0);
T2 = transl(0.7, 0.5,0.5);

q1 = Five_dof.ikunc(T1);
q2 = Five_dof.ikunc(T2);

Five_dof.plot(q1);
pause;
Five_dof.plot(q2);
```



（1）



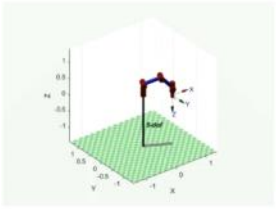
（2）

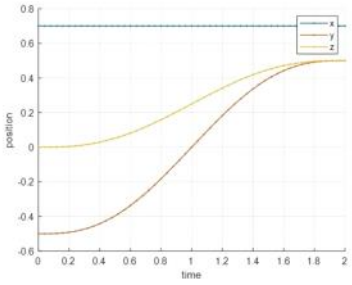
增加姿态：trotx(180)

P1/P2 —mtraj→ Traj —transl→ T —ikunc→ Qtraj

轨迹：'trail', 'b'

保存动画： 'movie', 'trail.gif'





运行and保存一个动画

```
% Five_dof.plot(Qtraj, 'trail', 'b');
% Five_dof.plot(Qtraj, 'movie', 'trail.gif');
```

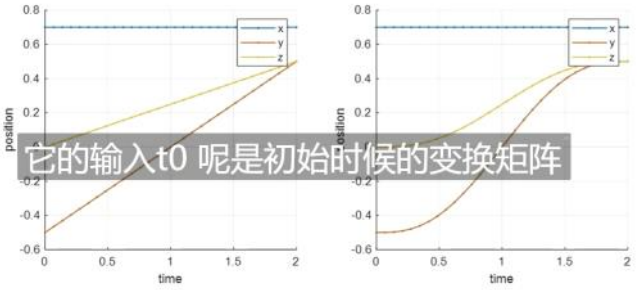
%%绘制XYZ的曲线图

```
hold on

plot(t, Traj(:, 1), '.-', 'linewidth', 1);
plot(t, Traj(:, 2), '.-', 'linewidth', 1);
plot(t, Traj(:, 3), '.-', 'linewidth', 1);

grid on
legend('x', 'y', 'z');
xlabel('time');
ylabel('position')
```

位姿插值 trinterp：

```
trinterp(T0, T1, M) as above but M is a positive integer and return a
sequence (4x4xM) of homogeneous transforms linearly interpolating between
T0 and T1 in M steps.
```

T = trinterp(T1, T2, 51);　　T = trinterp(T1, T2, tpoly (0, 2, 51) / 2);



它的输入t0 呢是初始时候的变换矩阵

线性差值

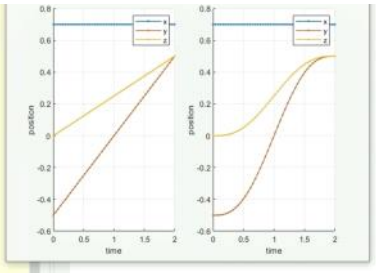
```
T_liner = trinterp(T1, T2, 51);
P_liner = transl(T_liner);
t1 = linspace(0, 2, 51);

subplot(1, 2, 1);
hold on
plot(t1, P_liner(:, 1), '.-', 'linewidth', 1);
plot(t1, P_liner(:, 2), '.-', 'linewidth', 1);
```

5次多项式差值

笛卡尔轨迹ctraj:

```
TC = ctraj(T0, T1, N) is a Cartesian trajectory (4x4xN) from pose T0 to T1
with N points that follow a trapezoidal velocity profile along the path.
The Cartesian trajectory is a homogeneous transform sequence and the last
subscript being the point index, that is, T(:,:,i) is the i'th point along
the path.
```

```
t1 = linspace(0, 2, 51);

subplot(1, 2, 1);
hold on
plot(t1, P_liner(:, 1), '.-', 'linewidth', 1);
plot(t1, P_liner(:, 2), '.-', 'linewidth', 1);
plot(t1, P_liner(:, 3), '.-', 'linewidth', 1);

grid on
legend('x', 'y', 'z');
xlabel('time');
ylabel('position')

T_tpoly = trinterp(T1, T2, tpoly(0, 2, 50) / 2);
P_tpoly = transl(T_tpoly);
t2 = linspace(0, 2, 50);clc

subplot(1, 2, 2);
hold on
plot(t2, P_tpoly(:, 1), '.-', 'linewidth', 1);
plot(t2, P_tpoly(:, 2), '.-', 'linewidth', 1);
plot(t2, P_tpoly(:, 3), '.-', 'linewidth', 1);

grid on
```

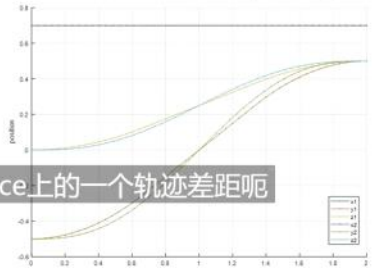


哎这时候就可以运行啊

TC = **ctraj**(T0, T1, N) is a Cartesian trajectory (4x4xN) from pose T0 to T1 with N points that follow a trapezoidal velocity profile along the path. The Cartesian trajectory is a homogeneous transform sequence and the last subscript being the point index, that is, T(:,:,i) is the i'th point along the path.

T = ctraj(T1, T2, 51);

T = ctraj(T1, T2, tpoly (0, 2, 50) / 2);



就表示在condition space上的一个轨迹差距呃

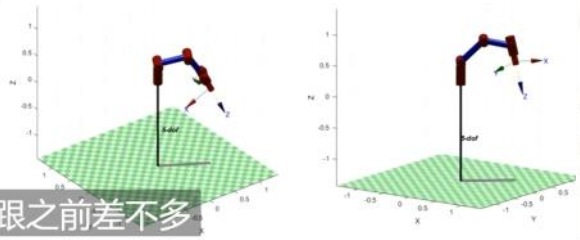
rtbdemo轨迹规划3

2024年12月8日 10:51

轨迹规划（3）

给定位置和姿态：（0.7，-0.5，0）* troty(150) ⟶ （0.7，0.5，0.5）* trotx(200)

```
T1 = transl(0.7, -0.5, 0) * troty(150);
T2 = transl(0.7, 0.5,0.5) * trotx(200);

q1 = Five_dof.ikunc(T1);
q2 = Five_dof.ikunc(T2);

Five_dof.plot(q1);
pause;
Five_dof.plot(q2);
```

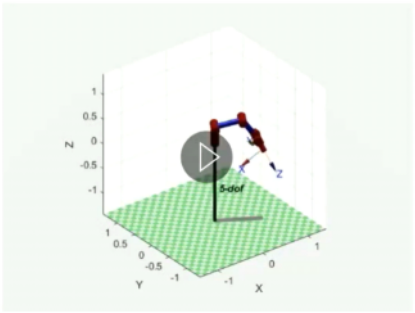


运行程序呢跟之前差不多

```
%%
T1 = transl(0.7, -0.5, 0) * troty(150);
T2 = transl(0.7, 0.5, 0.5) * trotx(200);

q1 = Five_dof.ikunc(T1);
q2 = Five_dof.ikunc(T2);

Five_dof.plot(q1);
pause
Five_dof.plot(q2);
```

T1、T2 ⟶(tr2rpy) rpy1、rpy2 ⟶(mtraj) rpy_traj ⟶(rpy2tr) T_traj_rot

```
rpy1 = tr2rpy(T1) / pi * 180;
rpy2 = tr2rpy(T2) / pi * 180;

t = linspace(0, 2, 51);
rpy_traj = mtraj(@tpoly, rpy1, rpy2, t);
T_traj_rot = rpy2tr(rpy_traj);

P1 = transl(T1);
P2 = transl(T2);
P_traj = mtraj(@tpoly, P1', P2', t);
T_traj_transl = transl(P_traj);

n = length(t);
T_traj = zeros(4, 4, n);
for i = 1:n
   T_traj(:, :, i) = T_traj_transl(:, :, i) * T_traj_rot(:, :, i);
end
```



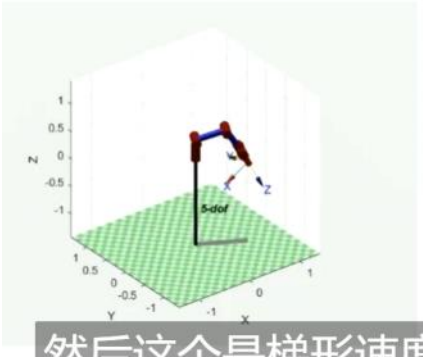
其实可以类比于之前位移的规划对吧

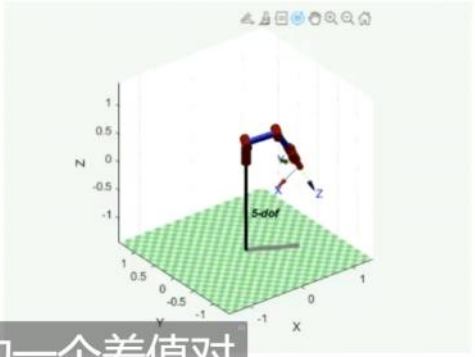
trinterp:

```
t = linspace(0, 2, 51);
T_traj = trinterp(T1, T2, t/2);
q = Five_dof.ikunc(T_traj);

Five_dof.plot(q, 'trail', 'r')
```

ctraj:

```
t = linspace(0, 2, 51);
T_traj = ctraj(T1, T2, t/2);
q = Five_dof.ikunc(T_traj);

Five_dof.plot(q, 'trail', 'r')
```



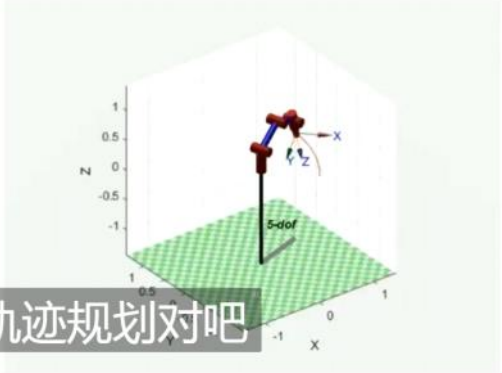


然后这个是梯形速度图像的一个差值对

`[Q,QD,QDD] = jtraj(Q0, QF, M)` is a joint space trajectory Q (MxN) where the joint coordinates vary from Q0 (1xN) to QF (1xN). A quintic (5th order) polynomial is used with default zero boundary conditions for velocity and acceleration. Time is assumed to vary from 0 to 1 in M steps. Joint velocity and acceleration can be optionally returned as QD (MxN) and QDD (MxN) respectively. The trajectory Q, QD and QDD are MxN matrices, with one row per time step, and one column per joint.

最好用的一集

```
q1 = Five_dof.ikunc(T1);
q2 = Five_dof.ikunc(T2);

t = linspace(0, 2, 51);
q_traj = jtraj(q1, q2, t);
```



然后再对这个关节变量做轨迹规划对吧

`[Q,QD,QDD] = jtraj(Q0, QF, T, QD0, QDF)` as above but specifies initial and final joint velocity for the trajectory and a time vector.

83.matlab三维mod

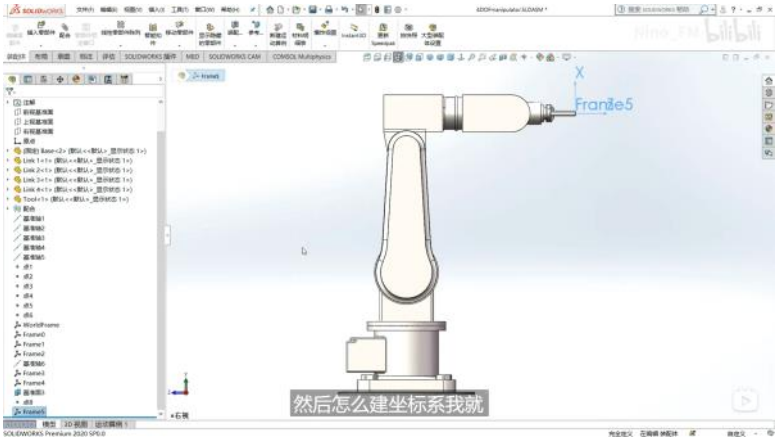
2024年12月8日 10:59

```
clear;
clc;

L(1)=Link('revolute','d',0.216,'a',0,'alpha',pi/2);
L(2)=Link('revolute','d',0,'a',0.5,'alpha',0,'offset',pi/2);
L(3)=Link('revolute','d',0,'a',sqrt(0.145^2+0.42746^2),'alpha',0,'offset',-atan(427.46/145));
L(4)=Link('revolute','d',0,'a',0,'alpha',pi/2,'offset',atan(427.46/145));
L(5)=Link('revolute','d',0.258,'a',0,'alpha',0);

Five_dof=SerialLink(L,'name','5-dof');
Five_dof.base=transl(0,0,0.28);

q0=[0 0 0 0 0];
v=[35 20];
w=[-1 1 -1 1 0 2];

Five_dof.plot3d(q0,'tilesize',0.1,'workspace',w,'path','E:\SW Data\Fdof','nowrist','view',v)

light('Position',[1 1 1],'color','w');
```

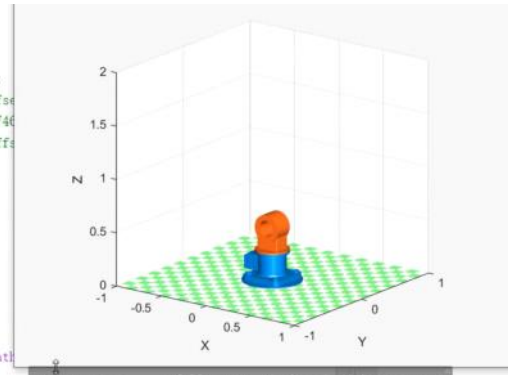
```
'path',P          Overide path to folder containing STL model files
'workspace', W    Size of robot 3D workspace, W = [xmn, xmx ymn ymx zmn zmx]
'floorlevel',L    Z-coordinate of floor (default -1)
'delay',D         Delay between frames for animation (s)
```

只能用于标准型的函数建模

pass后面接的是包含stl模型文件




```
clear;
clc;

L(1)=Link('revolute','d',0.216,'a',0,'alpha',pi/2);
% L(2)=Link('revolute','d',0,'a',0.5,'alpha',0,'offse
% L(3)=Link('revolute','d',0,'a',sqrt(0.145^2+0.42746
% L(4)=Link('revolute','d',0,'a',0,'alpha',pi/2,'offs
% L(5)=Link('revolute','d',0.258,'a',0,'alpha',0);

Five_dof=SerialLink(L,'name','5-dof');
Five_dof.base=transl(0,0,0.28);

q0=[0 0 0 0 0];
v=[35 20];
w=[-1 1 -1 1 0 2];

Five_dof.plot3d(q0,'tilesize',0.1,'workspace',w,'path

light('Position',[1 1 1],'color','w');
```



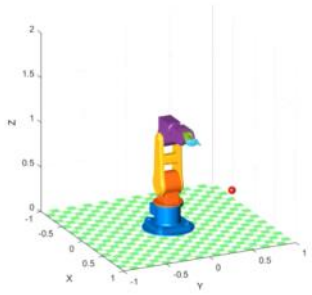
搬运仿真

plot_sphere(position, radius, color)

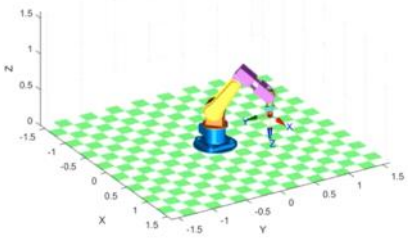
position = [0.5 0.5 0.5]

姿态

T = transl(position) * rpy2tr(180, 0, 0)



ikunc jtraj plot3d



```
%% pick

Positon = [0.5 0.5 0.5];
r = 0.04;
plot_sphere(Positon, r, 'r');

T1 = transl(Positon) * rpy2tr(180, 0, 0);

q1 = Five_dof.ikunc(T1);

q = jtraj(q0, q1, 60);

Five_dof.plot3d(q, 'view', v, 'fps', 60, 'nowrist');
```

搬运仿真

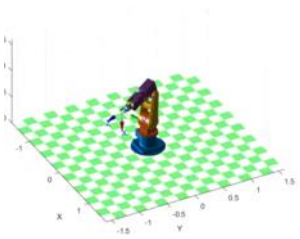
position2 = [0.5, -0.5, 1]

T2 = transl(position2) rpy2tr(90, 90, 0)

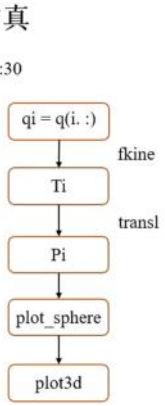
q2 = Five_dof.ikunc(T2);

q = jtraj(q1, q2, 30);

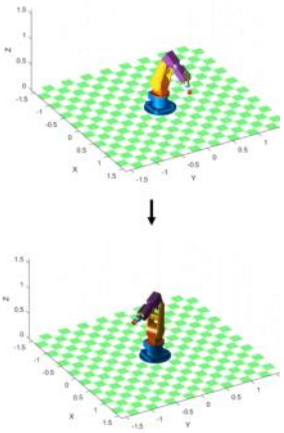
plot3d ?



for i = 1:30



end for



```
for i = 1:30
    qi = q(i, :);
    Ti = Five_dof.fkine(qi);
    Pi = transl(Ti);
    plot_sphere(Pi, r, 'r');
    Five_dof.plot3d(qi,'view', v, 'nowrist');
    cla
end
```

雅可比矩阵

2024年12月8日 11:18