Design of third-order uniform acceleration and deceleration trajectory based on Simulink StateFlow

Xudong Yang, Song Wang *

College of Mechanical Engineering, Guizhou University, Guiyang, China

*Corresponding author e-mail: 1169954689@qq.com

Abstract. In order to improve the movement accuracy of the robot, let the movement of the robot trajectory smoother, we design a third-order uniform acceleration / deceleration trajectory based on the Jerk value. Through the establishment of the mathematical model of the trajectory, combined with the logic control algorithm, constituted the core of the trajectory control. Finally, the StateFlow toolbox in Simulink as the carrier. Using the state module in StateFlow combined with the logic control conversion module, the third-order uniform acceleration and deceleration trajectory design is realized.

1. Introduction

Trajectory planning is an important part of the robot motion control system. In the course of the robot's work, the motion trajectory of the robot can be divided into multiple point-to-point motion trajectories.

In order to realize the accuracy, stability and smoothness of the robot during the movement, In the reference [1], a motion profile based on exponential function is proposed. The trapezoidal velocity profile is generated by the exponential function, resulting in smooth motion of the robot. In the reference [4], the robot trajectory is taken as the research object, and two kinds of point-to-point motion trajectory planning methods are proposed. In the first method, the motion parameters Jerk are neglected, only to consider the effect of velocity and acceleration on the motion trajectory. This method will cause the actuator to generate a discontinuous driving force, robots in the process of movement will have an impact, thus affecting the control accuracy.In order to solve the above problem, the author adds the third motion parameter Jerk to the motion trajectory, the acceleration profile is trapezoidal, the velocity profile is S-shaped, the movement trajectory will be more smooth, the implementing agency will have a continuous driving force in any case, reduce the impact of movement and expand the service life of the organization [1]。

Matlab as a good calculation and simulation software, it is widely used in robot technology. StateFlow as a toolbox in MATLAB Simulink, has a strong logic control function. Using StateFlow in the state module and flow chart module to establish the control model, and ultimately complete the system logic control.This paper combines the trajectory planning of robot motion with StateFlow state control to form a new trajectory control method. Compared with the traditional C language control procedures, this method is more simple and practical.

The rest of the article will be divided into four parts: The first part will introduce the mathematical model of the third order uniform acceleration and deceleration trajectory based on the Jerk value limit. The second part will introduce the algorithm of trajectory control; The third part constructs the logic control module of StateFlow, and combines the motion trajectory mathematical model to realize the motion trajectory control of the robot. The fourth part is the conclusion.

2. Design of Third - order Acceleration and Deceleration Motion Trajectory

The third-order acceleration and deceleration trajectory refers to the addition of the Jerk value in addition to the speed and acceleration. The Jerk value is set to a square curve, which can form a trapezoidal acceleration curve and an S-shaped velocity curve, resulting in a smoother motion trajectory. The initial position of the motion trajectory is x_0, the initial velocity is v_0, the initial acceleration is a_0, jerk is j, and the basic principle of the trajectory is as follows:

$$\mathrm{x(t)} = p^3(t, x_0, v_0, a_0, j) = x_0 + v_0 t + \frac{1}{2}at^2 + \frac{1}{6}jt^3 \tag{1}$$

$$\mathrm{v(t)} = p^3(t, x_0, v_0, a, j) = v_0 + at + \frac{1}{2}jt^2 \tag{2}$$

$$\mathrm{a(t)} = p^3(t, x_0, v_0, a, j) = a + jt \tag{3}$$

On the basis of the above equation of motion, through Jerk's different changes, the whole point-to-point motion process can be divided into seven segments, The value of Jerk is from positive maximum to zero to negative maximum, and through this change produces a continuously varying acceleration and velocity, resulting in a smooth trajectory, as shown in Figure 1.

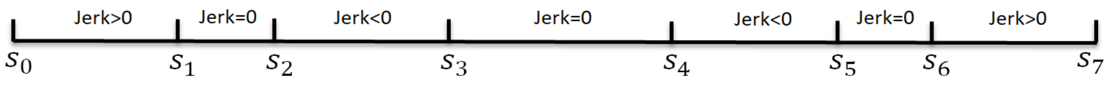


Figure 1 Jerk value change diagram

In the course of motion, set the starting point S_0 coordinates is Pos_start, the starting time is T_input, the coordinates of the end point S_7 are Pos_target, the arrival time is $\mathrm{t_{out}}$, the system running time is T_run, the time required for the running process is T_exe. Through the above motion parameters, we can calculate the maximum speed, maximum acceleration and Jerk value:

$$\mathrm{Vel\_max} = \frac{\mathrm{Pos\_target - Pos\_start}}{\mathrm{T_{exe}\,(1-n)}} \tag{4}$$

$$\mathrm{Acc\_max} = \frac{\mathrm{Vel\_max}}{\mathrm{T_{exe}\,(n-n^2)}} \tag{5}$$

$$\mathrm{Jerk} = \frac{\mathrm{Acc\_max}}{\mathrm{T_{exe}}\, n^2} \tag{6}$$

Where n is the time scale factor.

The time for setting the Jerk value is T_jm, the time for the constant acceleration is T_am, the time for the constant velocity action is T_vm,, the time of the seven nodes is T_S1 − T_S7. Using the velocity, the acceleration and the Jerk value, we can compute the displacement Pos_S1 − Pos_S6 and the velocity Vel_S1 − Vel_S6 for the motion trajectory to reach each node.

After knowing the time distribution of each node and the displacement and velocity.Using the equations (4), (5), (6), we can compute the complete uniform acceleration and deceleration trajectory, and the equations of motion for each stage are as follows:

When the run time T_run <= T_S1, the operating trajectory parameters for this phase are:

$$\mathrm{Pos} = \mathrm{Pos\_start} + \frac{1}{6}\mathrm{Jerk}(\mathrm{T\_run} - \mathrm{T\_input})^3 \tag{7}$$

$$\mathrm{Vel} = \frac{1}{2}\mathrm{Jerk}(\mathrm{T\_run} - \mathrm{T\_input})^2 \tag{8}$$

$$\mathrm{Acc} = \mathrm{Jerk}\ (\mathrm{T\_run} - \mathrm{T\_input}) \tag{9}$$

$$\mathrm{Jer} = \ \mathrm{Jerk} \tag{10}$$

When the run time T_S1 <= T_run <= T_S2, the operating trajectory parameters for this phase are:

$$\mathrm{Pos} = \mathrm{Pos\_S1} + \mathrm{Vel\_S1}(\mathrm{T_{run}} - \mathrm{T_{S1}}) + \frac{1}{2}\ (\mathrm{T_{run}} - \mathrm{T_{S1}})^2 \tag{11}$$

$$\mathrm{Vel} = \mathrm{Vel\_S1} + \mathrm{Acc\_max}\ (\mathrm{T\_run} - \mathrm{T\_S1}) \tag{12}$$

$$\mathrm{Acc} = \mathrm{Acc\_max} \tag{13}$$

$$\mathrm{Jer} = 0 \tag{14}$$

When the run time T_S2<=T_run <= T_S3, the operating trajectory parameters for this phase are:

$$\mathrm{Pos} = \mathrm{Pos\_S2} + \mathrm{Vel\_S2}\ (\mathrm{T_{run}} - \mathrm{T_{S2}}) + \frac{1}{2}\mathrm{Acc\_max}(\mathrm{T_{run}} - \mathrm{T_{S2}})^2 - \frac{1}{6}\mathrm{Jerk}(\mathrm{T\_run} - \mathrm{T\_S2})^3 \tag{15}$$

$$\mathrm{Vel}\ = \mathrm{Vel\_S2} + \mathrm{Acc\_max}(\mathrm{T_{run}} - \mathrm{T_{S2}}) - \frac{1}{2}\mathrm{Jerk}(\mathrm{T_{run}} - \mathrm{T_{S2}})^2 \tag{16}$$

$$\mathrm{Acc} = \ \mathrm{Acc\_max} - \mathrm{Jerk}\ (\mathrm{T_{run}} - \mathrm{T_{S2}}) \tag{17}$$

$$\mathrm{Jer} = -\mathrm{Jerk} \tag{18}$$

When the run time T_S3<=T_run <= T_S4, the operating trajectory parameters for this phase are:

$$\mathrm{Pos} = \ \mathrm{Pos\_S3} + \mathrm{Vel\_max}(\mathrm{T_{run}} - \mathrm{T_{S3}}) \tag{19}$$

$$\mathrm{Vel} = \mathrm{Vel\_max} \tag{20}$$

$$\mathrm{Acc} = \ \ 0 \tag{21}$$

$$\mathrm{Jer} \ = \ \ 0 \tag{22}$$

When the run time T_S4<=T_run <= T_S5, the operating trajectory parameters for this phase are:

$$\mathrm{Pos} = \mathrm{Pos\_S4} + \mathrm{Vel\_max}(\mathrm{T\_run} - \mathrm{T\_S4}) - \frac{1}{6}\mathrm{Jerk}(\mathrm{T\_run} - \mathrm{T\_S4})^3 \tag{23}$$

$$\mathrm{Vel} = \mathrm{Vel\_max} - \frac{1}{2}\mathrm{Jerk}(\mathrm{T\_run} - \mathrm{T\_S4})^2 \tag{24}$$

$$\mathrm{Acc} = -\mathrm{Jerk}\ (\mathrm{T\_run} - \mathrm{T\_S4}) \tag{25}$$

$$\mathrm{Jer} = -\mathrm{Jerk} \tag{26}$$

When the run time T_S5<=T_run <= T_S6, the operating trajectory parameters for this phase are:

$$\mathrm{Pos} = \mathrm{Pos\_S5} + \mathrm{Vel\_S5}(\mathrm{T\_run} - \mathrm{T\_S5}) - \frac{1}{2}\mathrm{Acc\_max}(\mathrm{T\_run} - \mathrm{T\_S5})^2 \tag{27}$$

$$\mathrm{Vel} = \mathrm{Vel\_S5} - \mathrm{Acc\_max}(\mathrm{T\_run} - \mathrm{T\_S5}) \tag{28}$$

$$\mathrm{Acc} = -\mathrm{Acc\_max} \tag{29}$$

$$\mathrm{Jer} = 0 \tag{30}$$

When the run time T_S6<=T_run <= T_S7, the operating trajectory parameters for this phase are:

$$\mathrm{Pos} = \mathrm{Pos\_S6} + \mathrm{Vel\_S6}(\mathrm{T_{run}} - \mathrm{T_{S6}}) - \frac{1}{2}\mathrm{Acc\_max}(\mathrm{T_{run}} - \mathrm{T_{S6}})^2 + \frac{1}{6}\,\mathrm{Jerk}(\mathrm{T\_run} - \mathrm{T\_S6})^3 \tag{31}$$

$$\mathrm{Vel} = \mathrm{Vel\_S6} - \mathrm{Acc\_max}\,(\mathrm{T\_run} - \mathrm{T\_S6}) + \frac{1}{2}\mathrm{Jerk}(\mathrm{T\_run} - \mathrm{T\_S6})^2 \tag{32}$$

$$\mathrm{Acc} = -\mathrm{Acc\_max} + \mathrm{Jerk}(\mathrm{T\_run} - \mathrm{T\_S6}) \tag{33}$$

$$\mathrm{Jer} = \mathrm{Jerk} \tag{34}$$

When the run time T_run >= T_S7, the operating trajectory parameters for this phase are:

$$\mathrm{Pos} = \mathrm{AxisPos\_target} \tag{35}$$

$$\mathrm{Vel} = 0 \tag{36}$$

$$\mathrm{Acc} = 0 \tag{37}$$

$$\mathrm{Jer} = 0 \tag{38}$$

3. Motion control logic algorithm

In order to achieve the robot's trajectory control, we designed the trajectory control flow shown in Figure 2. Enter the start and end coordinates of the input port. The system calculates the maximum speed, the maximum acceleration, and the Jerk value by the equations (4), (5) and (6). Based on these three motion parameters, combined with our established mathematical model, a third - order uniform acceleration and deceleration trajectory curve is generated. In the process of track output, the system will always monitor whether there is a new target location command input. If a new position coordinate is entered when the track has not reached the target position, the system will stop moving to the previous target position, The system will start with the new instruction time as the starting time, the location of the track as the starting coordinates, the new target location as a new destination coordinates to plan a new movement trajectory. When the system moves to the target position, it will check whether there is a new position coordinate input. If there is a new coordinate input, the system will plan a new motion trajectory. If there is no new instruction input, the track output will remain at the last coordinate position.

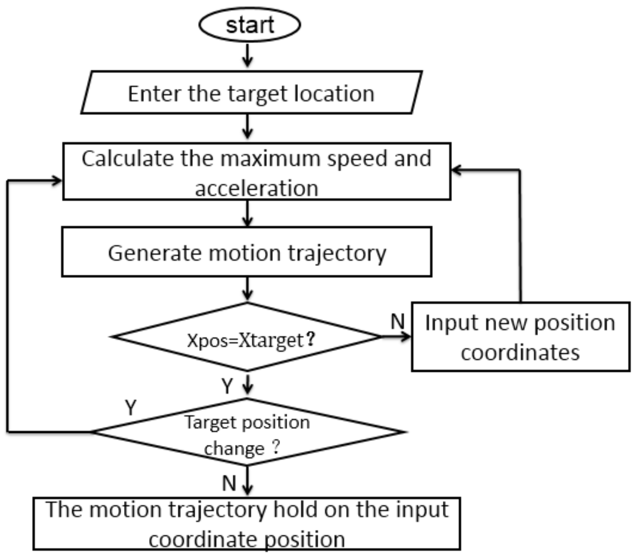

start
Enter the target location
Calculate the maximum speed and acceleration
Generate motion trajectory
Xpos=Xtarget?
N
Input new position coordinates
Y
Target position change ?
Y
N
The motion trajectory hold on the input coordinate position

Figure 2 The logic control flow of the trajectory control

4. Motion trajectory simulation experiment

The logic control and motion trajectory generation in the experiment will be done in the MATLAB Simulink environment. Through the Simulink Stateflow toolbox, using Stateflow state module and logic control module, combined with the mathematical model of motion trajectory and logic control algorithm, to establish a uniform acceleration and deceleration trajectory generator, as shown in Figure 3.

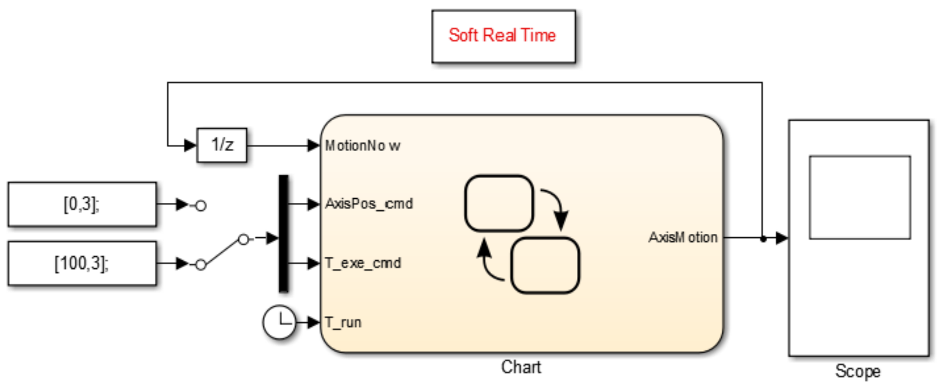
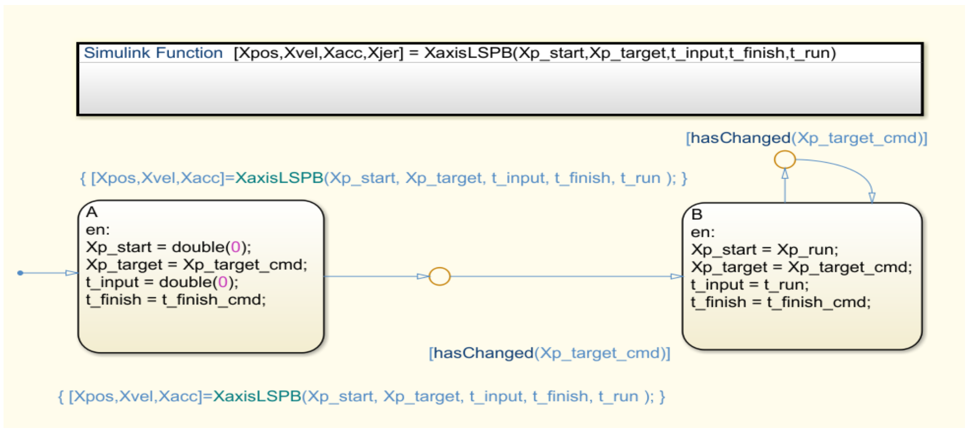


Figure 3 Motion trajectory generator based on Simulink State Flow

The most important part of the trajectory generator is the chart module. Through the state module and condition judging module, combined with the logic control algorithm, formed the core of the motion trajectory. As shown in Figure 4, the A module is the starting module, the initial motion parameters of the trajectory are set, and the parameters are sent to the Simulink Function function, produces the first trajectory to reach the set starting position. The system will detect whether the target command has changed. When entering a new target command, the system enters the B module, sets the new motion parameters, and then the new motion parameters through the function name to call the trajectory function, generate a new trajectory. During the motion, the B-state module will always detect whether there is a new instruction input. If there is a new instruction input, the system will return to the B module to update the motion parameters, the new motion parameters are sent to the trajectory function to update the trajectory.



```
Simulink Function  [Xpos,Xvel,Xacc,Xjer] = XaxisLSPB(Xp_start,Xp_target,t_input,t_finish,t_run)

[hasChanged(Xp_target_cmd)]

{ [Xpos,Xvel,Xacc]=XaxisLSPB(Xp_start, Xp_target, t_input, t_finish, t_run ); }

A
en:
Xp_start = double(0);
Xp_target = Xp_target_cmd;
t_input = double(0);
t_finish = t_finish_cmd;

B
en:
Xp_start = Xp_run;
Xp_target = Xp_target_cmd;
t_input = t_run;
t_finish = t_finish_cmd;

[hasChanged(Xp_target_cmd)]

{ [Xpos,Xvel,Xacc]=XaxisLSPB(Xp_start, Xp_target, t_input, t_finish, t_run ); }
```

Figure 4 Motion Trajectory Control Based on Simulink Stateflow

In the trajectory generator, set the starting coordinate to 0mm, the end point coordinate is 100mm, the trajectory planning movement time is 3s. We obtained the displacement profile, velocity profile, acceleration profile and Jerk profile, as shown in Fig.5.

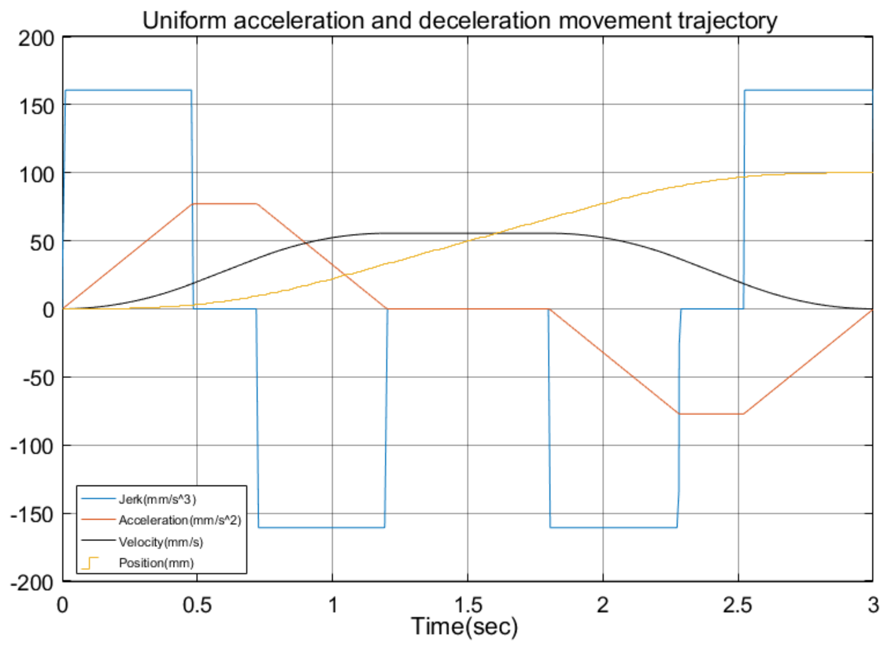


Figure 5 Third-order uniform acceleration and deceleration trajectory

5. Conclusion

In order to improve the smoothness of the trajectory and reduce the impact of the actuator, the third-order uniform acceleration / deceleration trajectory curve based on the Jerk value is designed. A third -order uniform acceleration and deceleration trajectory generator is designed by combining the mathematical model of motion trajectory with State Flow logic state control method in Matlab Simulink. Compared with the traditional C language programming, this method has many advantages:

(1) Simulink Stateflow visualization makes the design process clearer and simpler;

(2) There is no complicated program code, through a simple state module and determine the condition module settings, you can complete the complex logic control.

Acknowledgments
This work was financially supported by Yuan Ze Universtiy fund.

References

[1] Rymansaib Z., Iravani P., Sahinkaya M. N. Exponential trajectory generation for point to point motions[C]//Advanced Intelligent Mechatronics (AIM), 2013 IEEE/ASME International Conference on. IEEE, 2013: 906-911.
[2] Liu S. An on-line reference-trajectory generator for smooth motion of impulse-controlled industrial manipulators[C]//Advanced Motion Control, 2002. 7th International Workshop on. IEEE, 2002: 365-370.
[3] KröGer T, Tomiczek A, Wahl F M. Towards On-Line Trajectory Computation[C]// Ieee/rsj International Conference on Intelligent Robots and Systems. IEEE, 2007:736-741.
[4] Haschke R, Weitnauer E, Ritter H. On-line planning of time-optimal, jerk-limited trajectories[J]. 2008, 157(2):3248-3253.
[5] Kröger T, Wahl F M. Online Trajectory Generation: Basic Concepts for Instantaneous Reactions to Unforeseen Events[J]. IEEE Transactions on Robotics, 2010, 26(1):94-111.
[6] Information on https://www.mathworks.com/products/stateflow.html