KINOVA

Trajectory Generation

Last update: 2021-04-21

Gen3 lite Cursus

KINOVA

Table of contents

KINOVA

Objectives

During this lab, we will study trajectory generation for Gen3 lite. We will compute trajectories defining the position, velocity and acceleration of each actuator using two different methods. These trajectories will be discretized at a frequency of 1 kHz to be able to be sent to the robot. Our first method will generate trajectories in joint space, while the second method will generate them in cartesian space. Finally, our trajectories will be validated and compared experimentally on the robor.

About Gen3 lite

For this lab, we will need to know the joint position limits of the robot, seen in Table 1.

Joint #	Position limit (deg)
1	± 154.1
2	± 150.1
3	± 150.1
4	± 148.98
5	-144.97, +145.0
6	± 148.98

TABLE 1: Gen3 lite Joint position limit

For more information on Gen3 lite, you can always consult the User Guide.

Theory

Trajectory generation in joint space is usually relatively simple. Each joint can be treated as an isolated system which can be solved based on constraints like initial and final conditions, or limits. Duration is a parameter often shared between the joints, but since it is an external constraint, each system can still be solved independently.

Cartesian trajectories on the other hand are more complex since the six degrees of freedom are all functions of all the joint trajectories. In order to put some order into this, the proposed strategy is performed in two steps: first interpolating between cartesian poses, then moving to the angular space.

Cartesian pose interpolation

To move from an initial cartesian pose defined by a homogeneous transformation matrix T(0), to a final pose T(1), we can use screw theory to define a screw motion such that $T(1) = exp([\xi])T(0)$, where ξis a vector of $\mathbb{R}^6$. We can then describe a trajectory as follows:

$$T(s) = exp(s[\xi])T(0)$$

Where *s* is a parameter that we can choose to be linear relative to time.

Moving to the angular space

At first glance, converting a trajectory defined in cartesian space to joint space should be easy, we would only have to run the inverse kinematics for every point of the trajectory. However, in practice, this solution is not usable. As we have experienced in our lab sessions on iterative inverse kinematics, the computation can take a significant amount of time for a single point, ruling out the possibility of doing it for the multiple thousands of points necessary to describe a trajectory of a few seconds at a rate of 1 kHz.

Instead, it is preferable to use the differential inverse kinematics, which consists of starting at our initial position and looking for increments that will bring us close to our target. We know from our previous lab that this can be achieved using Newton's algorithm. In this case however, we want to take into account the actual initial position of our trajectory as well as the impact of time, so the form of the algorithm will be:

$$q_0 = q(t_0)$$

$$q(t_{k+1}) = q(t_k) + (t_{k+1} - t_k) J^s(q(t_k))^* V_k$$

Where

- q is a vector containing all the joint positions
- t_k is the time at instant k
- $J^s(q(t_k))^*$ is the pseudo-inverse of the geometric jacobian expressed in the base frame
- V_k is the kinematic screw

Trapezoidal velocity profiles

To interpolate between two positions, be it for a single joint or for cartesian points, a common method is called a trapezoidal velocity profile. As its name implies, it consists of defining a velocity profile in the shape of a trapeze, defined by a maximum acceleration, a maximum velocity, a duration and a displacement, as seen in Figure 1.

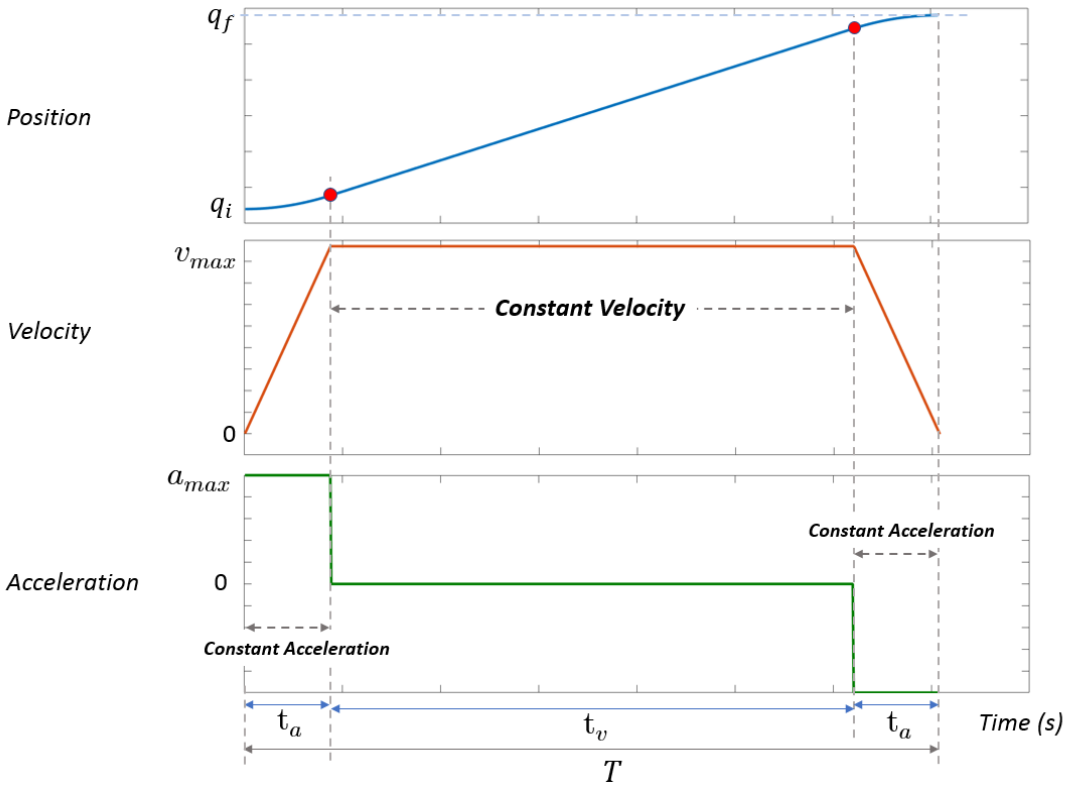

FIGURE 1: Trapezoidal velocity profile

To fully define a symmetrical trapezoidal velocity profile which respects the constraints $\mathbf{a_{max}}$, $\mathbf{q_i}$, $\mathbf{q_f}$, **T,** only two values ($\mathbf{t_a}$ and $\mathbf{v_{max}}$ seen in Figure 1) are necessary if the initial and final velocities are zero.

Manipulations

Part 1: Generating joint space trajectories

In this section, we will define a trajectory between two angular poses. First, we will do it using polynomial interpolation and then using trapezoidal velocity profiles.

1.1 Find the equations of to obtain the coefficients of a 5th order polynomial which respects initial and final position, velocity and acceleration constraints ($\mathbf{q_0}$, $\mathbf{v_0}$, $\mathbf{a_0}$, $\mathbf{q_f}$, $\mathbf{v_f}$, $\mathbf{a_f}$) as well as a duration (**t)**.

1.2 Write a function called **ang_traj_poly**(q0, v0, a0, qf, vf, af, t) which computes the trajectory interpolated between **q0** and **qf** using a 5th order polynomial which respects the initial and final conditions of position, velocity and acceleration. It must return **q, dq, ddq**, matrices of size *n x m*, where *n* is the number of joint variables (here 6) and *m* is the number of elements in *t*. The outputs **q, dq, ddq** respectively contain the evaluation of the polynomial at every instant listed in the vector **t**.

For our second method, we will define trajectories using trapezoidal velocity profiles.

1.3 Find the equations to obtain the values of $\mathbf{t_a}$ and $\mathbf{v_{max}}$ required to define a velocity profile with the constraints $\mathbf{a_{max}}$, $\mathbf{q_i}$, $\mathbf{q_f}$, **T.** Assume initial and final velocities and accelerations are zero.

1.4 Write a function called **ang_traj_trap**(q0, v0, qf, vf, t, amax) which computes the interpolation between **q0** and **qf** using trapezoidal velocity profiles and return **q, dq, ddq,** matrices defined in the same way as in **1.2**. Consider that the duration of the trajectory is given by the last element of **t**.

Part 2: Generating cartesian space trajectories

2.1 Write a function called **cart_traj**(q0, qf, t) which compute a trajectory between **q0** and **qf** (which were obtained from initial and final cartesian poses) using the Euler algorithm described in the previous page. Like our two previous functions, this functions returns **q, dq, ddq**, *n x m* matrices containing the evaluation of the position, velocity and acceleration of each joint for every instant defined in vector **t**.

Part 3: Validation with the robot

In this section, we will use the script **main_gen3lite_labTraj.m** which will display the position, velocity and acceleration of each joint for all the three methods we implemented above. The script then executes these trajectories on Gen3 lite using **PlayPreComputedTrajectory** from the MATLAB API. However, this function requires certain conditions to be met to run properly:

- The initial timestamp must be 0.0 seconds
- The time steps must be done in increments of 0.001s (Gen3 lite operates at 1 kHz)
- The current position of the arm when the trajectory launches must be the initial point of the trajectory.
- Position limits for each joint must be respected at all times within the trajectory.
- The maximum velocity for each joint must be below 50°/sec at all times within the trajectory
- The input tables must all contain the same number of joints and the same number of time steps
- Positions must be continuous
- The trajectory must not last more than 30 seconds

3.1 Write a function called **validate_precomputed_traj**(q, dq, ddq, t) which validates all the conditions mentioned above and add it to the implementation of all three of your trajectory generations.

3.2 Run the **main_gen3lite_labTraj.m** script. Compare the trajectories you obtained.

3.3 Explain one benefit and one downside of each of the trajectories we developed.

3.4 (Optional) Using the method of your choice, try to figure out what trajectory profiles are generated by Gen3 lite's firmware when performing its standard actions. **Hint:** You may need to code using Python or C++ instead of MATLAB to be able to call **RefreshFeedback()** at a sufficiently high rate.

Acknowledgements

This document was produced in collaboration with the following establishments

Including direct implication from the following people:

- Prof. David Saussié, Eng., M.A.Sc., Ph.D., Department of Electrical Engineering.
- Alexis Tang, M.Eng., Department of Electrical Engineering
- Prof. Jérôme Le Ny, Eng., M.A.Sc., Ph.D., Department of Electrical Engineering.
- Prof. Richard Gourdeau , B.A.Sc., M.A.Sc., Ph.D., Department of Electrical Engineering.
