Fundamentels of Robotics

Assignment 4 - Trajectory Planning. Innopolis University, Fall 2020

Name: Hany Hamed
Group Number: BS18-Robotics
GitHub Repository: here (It is public after the submission deadline) (Please read the README, it has a lot of visualization)

Contents

FoR, Innopolis University, Fall 2020
Author: Hany Hamed

Section 1: Robot's FK, IK & Jacobian

The robot is RRR, it is a simple and easy manipulator that only controls the position [x,y,z] (3 DoF).

- Forward Kinematics (FK): (indexing from 0)

  $T = T_{base}\ R_z(q_0)\ T_z(l_0)\ R_y(q_1)\ T_x(l_1)\ R_y(q_2)\ T_x(l_2)@T_{tool}$

- Inverse Kinematics (IK): As from previous assignments, (It is implemented in the code [IK.py])

  $T = T_{base}\ T_z(l_0)\ T_{123}\ T_{tool}$

  $T_o = T_z(l_0)^{-1}\ T_{base}^{-1}\ T\ T_{tool}^{-1}$

  x, y and z symbolic equations can be equal to the computed values from the last column (position vector) from T_o matrix $(x, y, z = T_o[:, :3])$

  $x = (l_1c_1 + l_2c_{1+2})c_0$

  $y = (l_1c_1 + l_2c_{1+2})s_0$

  $-z = l_1s_1 + l_2s_{1+2}$

  And we can find easily $q_0,\ q_1,\ q_2$

- Jacobian: Skew theory and Numerical Derivative method have been implemented. With the same code as last assignment but just changed the transition matrices as following

  $$T = A_0A_1A_2A_3$$

  Such that:

  $A_0 = T_{base} = T_0^w$

  $A_1 = R_z(q_0)T_z(l_0) = T_1^0$

  $A_2 = R_y(q_1)T_x(l_1) = T_2^1$

  $A_3 = R_y(q_2)T_x(l_2)T_{tool} = T_3^2$

  Note: $T_j^i = T_k^iT_j^k$

  The jacobian is in the following form:

  $\mathcal{J} = [\vec{\mathcal{J}}_0\ \vec{\mathcal{J}}_1\ \vec{\mathcal{J}}_2]$

  As all the joints are revolute joints, we will use the following in order to compute each column $\vec{\mathcal{J}}_i$ in the jacobian matrix:

  $$\vec{\mathcal{J}}_i = \begin{bmatrix} \vec{U}_i\ \times\ (\vec{O}_n - \vec{O}_i) \\ \vec{U}_i \end{bmatrix}$$

  such that: n=3 and i=0,1,2

  Then we need to get the $\vec{O}\ \ \vec{U}$ vectors:

- $\vec{O_i}$ is the positional vector from the transformation matrix from the world to i^th frame (T_i^w)
- $\vec{U_i}$ is the column vector corresponds to the axis of the rotation of the joint from the rotation matrix that is composed into the transformation matrix from the world to i^th frame (T_i^w) (e.g. rotation axis is z-axis, take 3rd column

Section2: 5th order Polynomial profile

In this task, we have 6 constraints, then we have to work with 5th order polynomial. The constraints are as following:

- $q(0) = q_0 = (0.5, -0.6, 0)$
- $\dot{q}(0) = \dot{q}_0 = (0, 0, 0)$
- $\ddot{q}(0) = \ddot{q}_0 = (0, 0, 0)$
- $q(2) = q_f = (1.57, 0.5, -2.0)$
- $\dot{q}(2) = \dot{q}_f = (0, 0, 0)$
- $\ddot{q}(2) = \ddot{q}_f = (0, 0, 0)$

The 5th order polynomial is as following:

$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5$

$\dot{q}(t) = a_1 + 2a_2 t + 3a_3 t^2 + 4a_4 t^3 + 5a_5 t^4$

$\ddot{q}(t) = 2a_2 + 6a_3 t + 12a_4 t^2 + 20a_5 t^3$

Then, we can make substitute in the previous equations, the values of the constraints, and solve the 6 equations for 6 equations and get the equations for the trajectory in the form of 5th order polynomial. We can rewrite it in a matrix form Ax=b, such that,

$$\text{x (Unknown)} = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix}$$

$$\text{b (Known)} = \begin{bmatrix} q_0 \\ \dot{q}_0 \\ \ddot{q}_0 \\ q_f \\ \dot{q}_f \\ \ddot{q}_f \end{bmatrix}$$

$$\text{A (Known)}= \begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 & t_0^4 & t_0^5 \\ 0 & 1 & 2t_0 & 3t_0^2 & 4t_0^3 & 5t_0^4 \\ 0 & 0 & 2 & 6t_0 & 12t_0^2 & 20t_0^3 \\ 1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5 \\ 0 & 1 & 2t_f & 3t_f^2 & 4t_f^3 & 5t_f^4 \\ 0 & 0 & 2 & 6t_f & 12t_f^2 & 20t_f^3 \end{bmatrix}$$

We need to get x (the matrix of coefficients) $x = A^{-1}b$, then substitute the coefficients in the equations in order to get the trajectory equations.

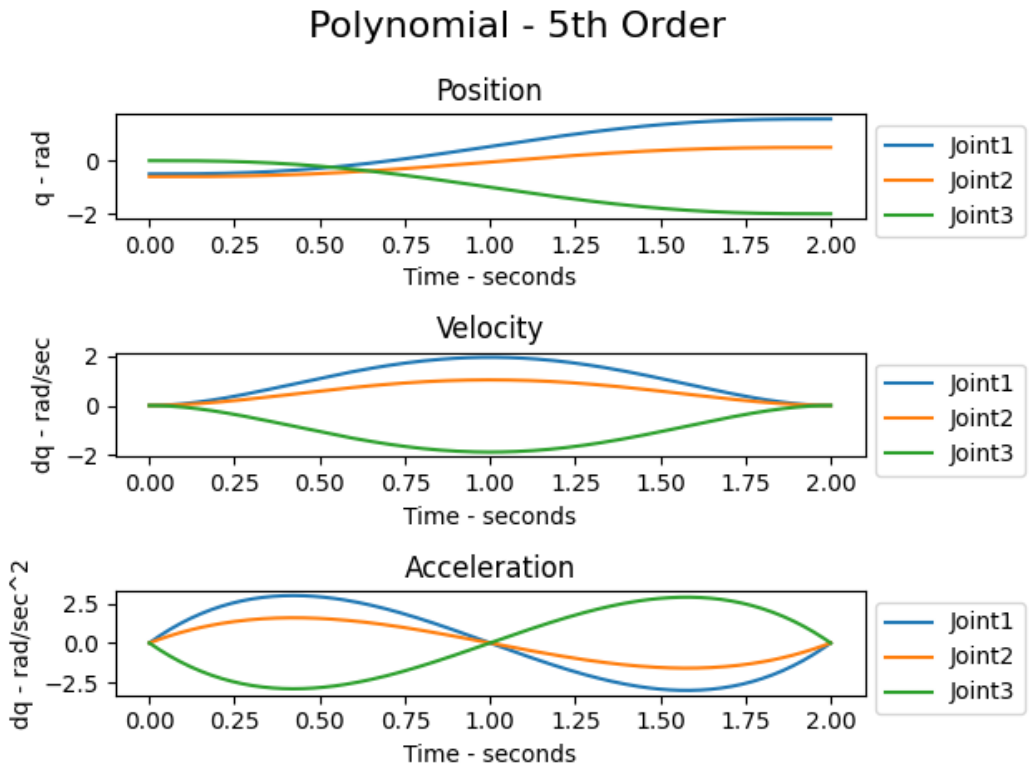

Figure: Polynomial 5th order (Task2 configuration)

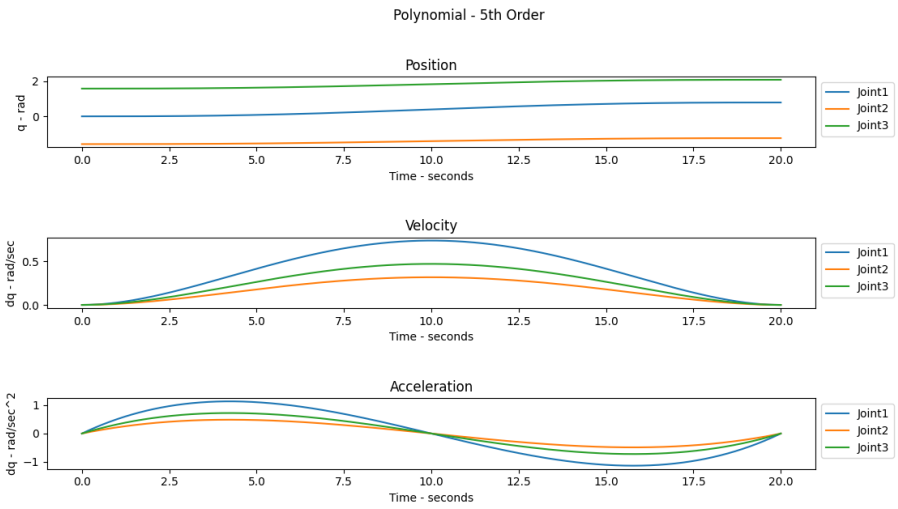

Figure: Polynomial 5th order (different configuration)

Different configurations and details are in the README in the repository with gifs

Section 3: Trapezoidal Velocity Profile

Given $q_0, q_f, f, \dot{q}_{max}, \ddot{q}_{max}$, we need to find $q(t), \dot{q}(t), \ddot{q}(t)$ for each reference point.

First, we need to define some variables:

$\Delta t = \frac{1}{f}$

$\Delta q = |(q_f - q_0)|$

$direction = sign(q_f - q_0)$

t_1 & τ are vectors

The algorithm is as following:

1. Decide the case for each joint, using the following: $\dot{q}{\cdot}max = \sqrt{\Delta q * \ddot{q}_m ax}$
   if $\dot{q}`_{max} \leq \dot{q}_{max}$ then:
   $t_1 = \sqrt{\Delta q / \ddot{q}_{max}}$, $\tau = 0$ and it is triangular case
   if $\dot{q}`_{max} > \dot{q}_{max}$ then:
   $t_1 = \dot{q}_m ax / \ddot{q}_{max}$, $\tau = \Delta q / \dot{q}_{max}$ and it is Trapezoidal case
2. Then, we need to synchronize all the profiles, in order to start and finish at the same time by stretching the profiles:

- if any of the profiles is trapezoidal, then all of them will be trapezoidal
- if all of them are triangular, then all of them will be triangular
- $t`_1 = max(t)$ over all the computed profiles
- $\tau` = max(\tau - t) + t`_1$
- Re-plan again all the profiles (get $\dot{q}“_{max}$ and $\ddot{q}“_{max}$)
- If the selected profile for all is trapezoidal, then $\dot{q}“_{max} = \Delta q / \tau`$ and $\ddot{q}“_{max} = \dfrac{\Delta q}{\tau` t`_1}$
- If the selected profile for all is triangular, then $\dot{q}“_{max} = \Delta q / t`_1$ and $\ddot{q}“_{max} = \dfrac{\Delta q}{t`^2_1}$

3. Then, we take into consideration the discretization in the controller (Frequency of the controller).

   - Calculate n and m:
     $n = ceil(\dfrac{t`_1}{\Delta t})$
     $n = ceil(\dfrac{\tau` - t`_1}{\Delta t})$
   - Calculate $t“_1,\ \tau“$:
     $t“_1 = n * dt$
     $\tau“ = m * dt$
   - Calculate $\dot{q}“‘_{max},\ \ddot{q}“‘_{max}$
     - If the selected profile for all is trapezoidal, then $\dot{q}“‘_{max} = \Delta q / \tau“$ and $\ddot{q}“‘_{max} = \dfrac{\Delta q}{\tau“ t“_1}$
     - If the selected profile for all is triangular, then $\dot{q}“‘_{max} = \Delta q / t“_1$ and $\ddot{q}“‘_{max} = \dfrac{\Delta q}{t“^2_1}$

4. Then, now we have the total calculated time.

   - If Trapezoidal: total time $= t“_1 + \tau“$
   - If Triangular: total time $= 2t“_1$

5. Then, we can get the position, velocity and acceleration for each join according to the time period during the simulation as following: *Each joint has different initial, goal positions and different limits for velocities and accelerations, but all of them have the same profile and the same time periods*

   - if Trapezoidal profile:
     - $pos_0 = q_0,\ vel_0 = 0,\ acc_0 = 0$

  - $i \in [1, \frac{total\ time}{\Delta t}]$
  - $t = i * \Delta t$
  - $if(0 \leq t\ \&\ t < t\text{“}_1)$ Then, acceleration period:
    * $acc_i = \ddot{q}\text{“‘}_{max} * direction$
    * $vel_i = acc_i * t$
    * $pos_i = pos_{i-1} + vel_i * \Delta t)$
  - $if(t\text{“}_1 \leq t\ \&\ t < \tau\text{“})$ Then, Zero acceleration period:
    * $acc_i = 0$
    * $vel_i = \dot{q}\text{“‘}_{max} * direction$
    * $pos_i = pos_{i-1} + vel_i * \Delta t)$
  - $if(\tau\text{“} \leq t\ \&\ t <= total\ time)$ Then, Declaration period:
    * $acc_i = -\ddot{q}\text{“‘}_{max} * direction$
    * $vel_i = direction * \dot{q}\text{“‘}_{max} + acc_i * (t - \tau\text{“})$
    * $pos_i = pos_{i-1} + vel_i * \Delta t)$
  - else. Then, they have reached to the goal:
    * $acc_i = 0$
    * $vel_i = 0$
    * $pos_i = pos_{i-1}$

- if Triangular profile:
  - $pos_0 = q_0,\ vel_0 = 0,\ acc_0 = 0$
  - $i \in [1, \frac{total\ time}{\Delta t}]$
  - $t = i * \Delta t$
  - $if(0 \leq t\ \&\ t < t\text{“}_1)$ Then, acceleration period:
    * $acc_i = \ddot{q}\text{“‘}_{max} * direction$
    * $vel_i = acc_i * t$
    * $pos_i = pos_{i-1} + vel_i * \Delta t)$
  - $if(\tau\text{“} \leq t\ \&\ t <= total\ time)$ Then, Declaration period:
    * $acc_i = -\ddot{q}\text{“‘}_{max} * direction$
    * $vel_i = direction * \dot{q}\text{“‘}_{max} + acc_i * (t - æ_1)$
    * $pos_i = pos_{i-1} + vel_i * \Delta t)$
  - else. Then, they have reached to the goal:
    * $acc_i = 0$
    * $vel_i = 0$
    * $pos_i = pos_{i-1}$

Section 4: PTP & LIN command

- PTP (point to point with trapezoidal profile for Joint trajectory planning): it uses Trapezoidal velocity profile with the same algorithm with no changes as it is described in the previous section.

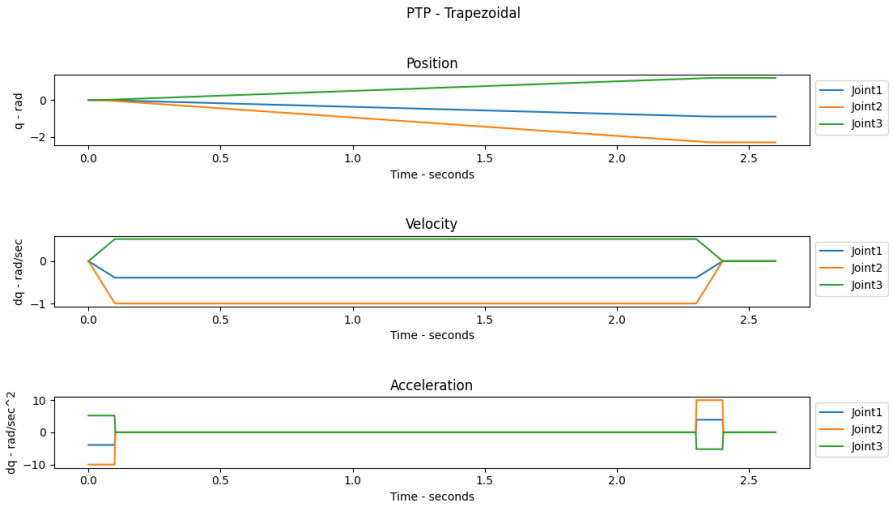

Figure: PTP command (Task3 configuration)

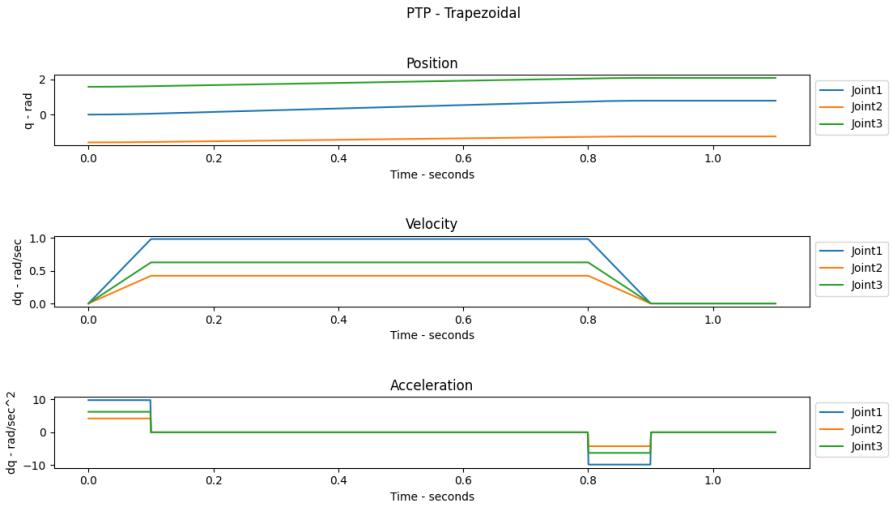

Figure: PTP command (Different Configuration)

- LIN (Linear motion between two points but with trapezoidal profile for Cartesian trajectory planning): here there are two variations that were implemented (1st one is used):

(a) The algorithm is as following:

- Use trapezoidal profile to go from initial to goal (plan for Cartesian space variables)
- Iterate through the generated trajectory
- Use IK to get the corresponding joint angle values with each configuration (get q)
- Use Jacobian to calculate $\dot{q} = J^{-1}[:, :3]\dot{p}$, using pseudo inverse and then take the first 3 rows that relates x,y,z
- And now we have the trajectory for the joints

Good reference for Cartesian planning

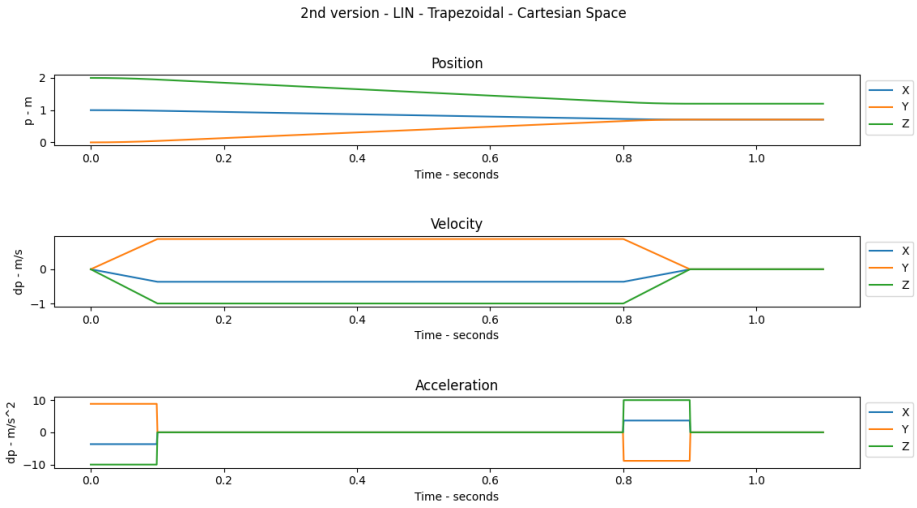

Figure: LIN Cartesian space (Task 4 configuration)

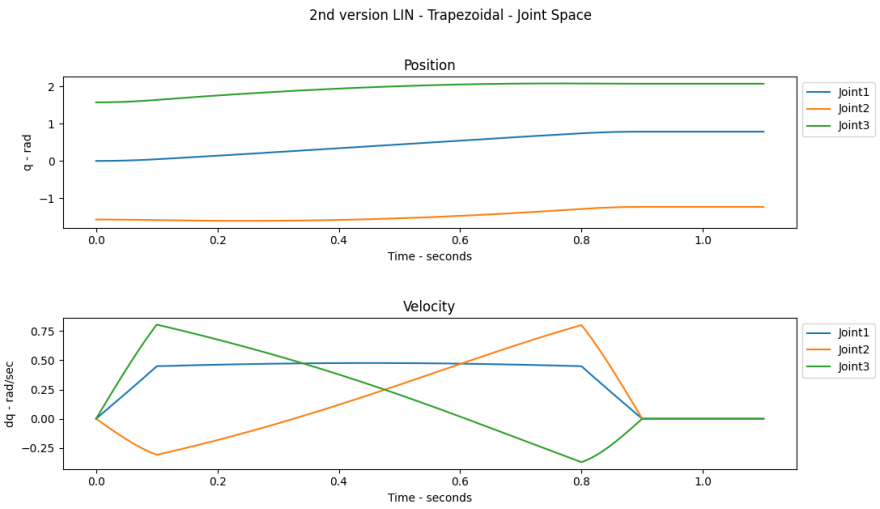

Figure: LIN Joint space (Task 4 configuration)

- The second method, as this was implemented first, then I understood that it is not correct. The algorithm is as following:
  - Create the equation of line between the two points
  - Sample different points on that line
  - Use IK to get the joint angles
  - Plan for each sampled joint angles and get the trajectory.
  - Use FK to get the corresponding points in the Cartesian space.
  - Use Jacobian as in the first method to get the velocities of the variables of the Cartesian space.

  The method doesn't lead to trapezoidal profile in Cartesian space variables, however, it makes the endeffector move on a line

Look on the README in the repository