Introduction to Robotics

Final Project Assignment 2019

ABDULLAH THABIT

Contents

- Task (1,2,3,4): problem statement
  - Steps
  - Results

Task 1: Problem Statement

- The Control objective is **the position-only** without exploiting the redundancy. The required path is an **horizontal square** lying on the **xy axis** with side of **10 cm** length.
- The end-effector occupies the **upper-left corner** of the square looking from the z axis.
- Each side needs to be executed with a trajectory of duration **2 s** and a **trapezoidal velocity profile** characterized by a proper cruise velocity.
- The robot stops for **200ms at each corner** and **3s once back** in the initial end effector position.

Task 1: Steps

- KinovaJaco2 7-DOFs with Spherical Wrist
- Provided Functions Used for Kinovato DH conversion and vice versa

joint	a (m)	α (°)	d (m)	θ (°)
1	0	90°	0.2755	θ_1
2	0	90°	0	θ_2
3	0	90°	−0.410	θ_3
4	0	90°	−0.0098	θ_4
5	0	90°	−0.3111	θ_5
6	0	90°	0	θ_6
7	0	0°	0.2638	θ_7

1. Getting Initial Position

- From the home configuration and the DH notation is completed.
- To get the rotation matrix as well as displacement of the end effector, DH table values are plugged in the homogenous transformation matrices.

$$\begin{bmatrix} c_{\theta i} & -s_{\theta i}c_{\alpha i} & s_{\theta i}s_{\alpha i} & a_i c_{\theta i} \\ s_{\theta i} & c_{\theta i}c_{\alpha i} & -c_{\theta i}s_{\alpha i} & a_i s_{\theta i} \\ 0 & s_{\alpha i} & c_{\alpha i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

1. Getting Initial Position

- The corresponding homogenous transformation matrices are multiplied starting for the base frame to the end effector.
- The multiplication result, a matrix, gives the displacement as well as rotation of the end effector.
- This is result is saved as the **initial displacement and the rotation of the end effector.**

2. Calculating the desired position for each iteration

- The initial displacement vector obtained by homogenous transformation matrix. (upper left corner of the square)
- The location of the other 3 corners are calculated with respect to the 1st corner, giving the length of line segments (10 cm) between corners.

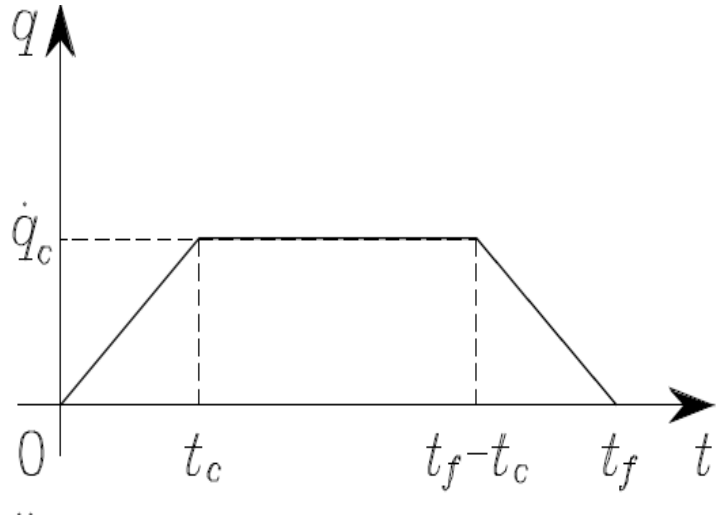
2. Calculating the desired position for each iteration

- With a sampling time Ts = 1ms, time instances are devided based on where they fall between corners :
  - First line segment … (i*Ts < 2.2s)
  - Second line segment … ((i*Ts)> 2.2 && (i*Ts) <= 4.4)
  - Third line segment … ((i*Ts)>4.4 && (i*Ts)<=6.6)
  - Forth line segment … ((i*Ts)>6.6 && (i*Ts)<=11.6)

3. Trapezoidal Velocity Profile

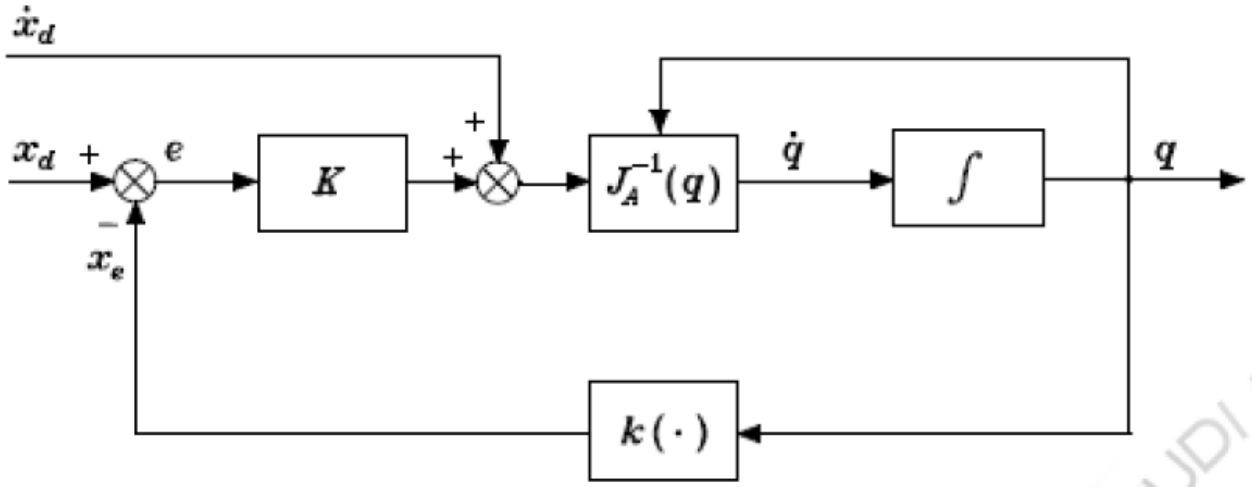
- At each line segment, the position and linear velocity of time instances were calculated using a trapezoidal velocity profile, with final time fixed to 2s
- The cruise velocity is fixed according the following equation: (~=0.08 [m/s])

$$\frac{|q_f - q_i|}{t_f} < |\dot{q}_c| \leq \frac{2|q_f - q_i|}{t_f}$$

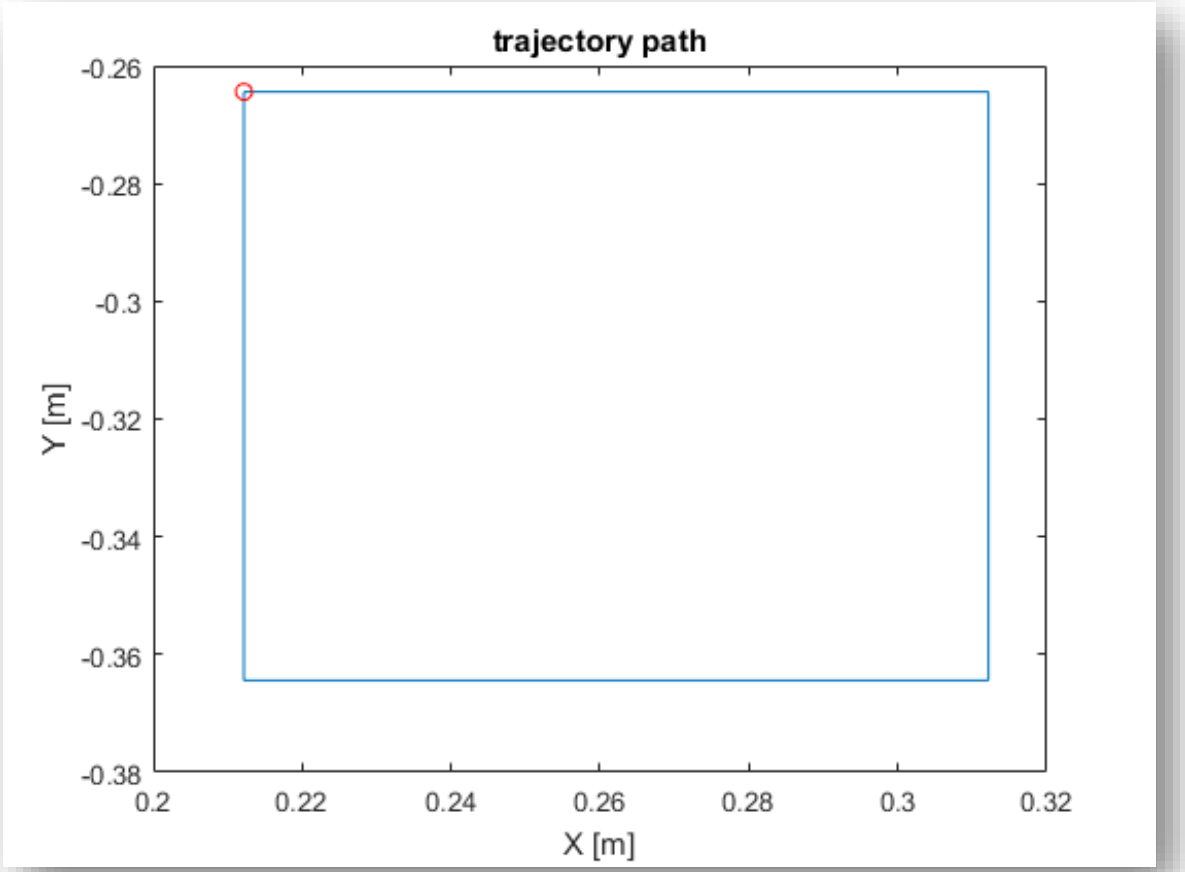


Error and Feedback

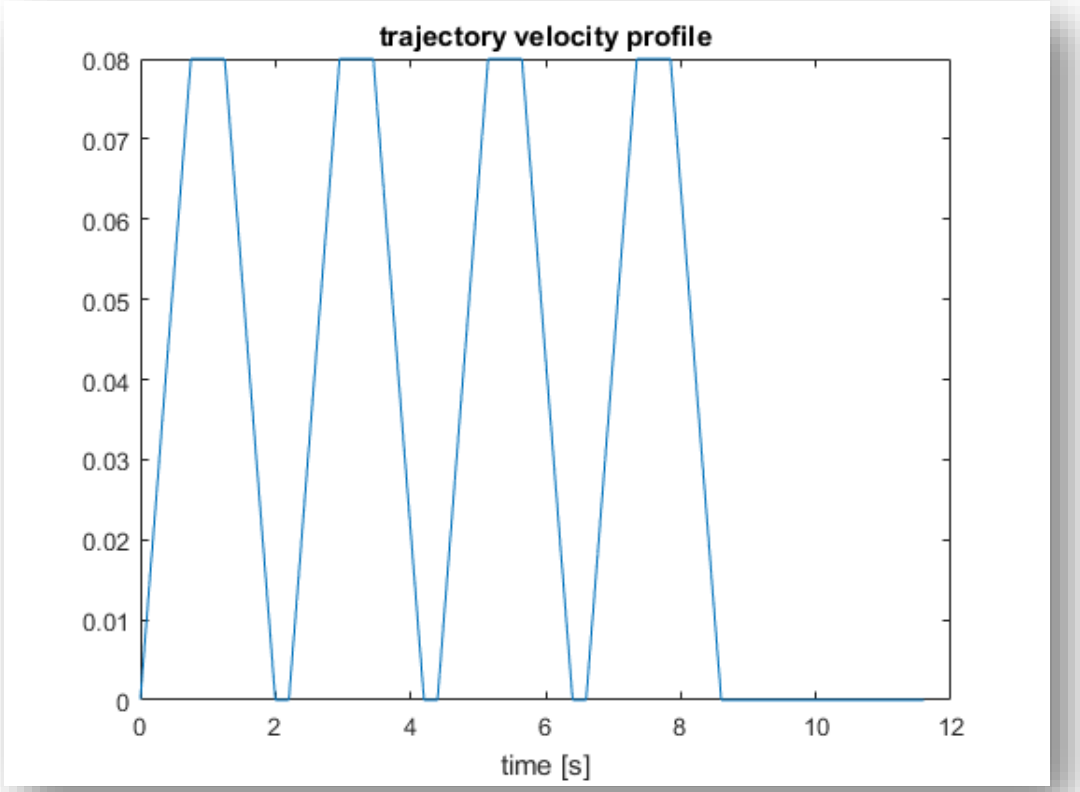
- Only the positional error and the desired linear velocity is taken into consideration in this task. The following feedback is implemented to the new position at each iteration.
- The joint angles are then fed to the V_rep



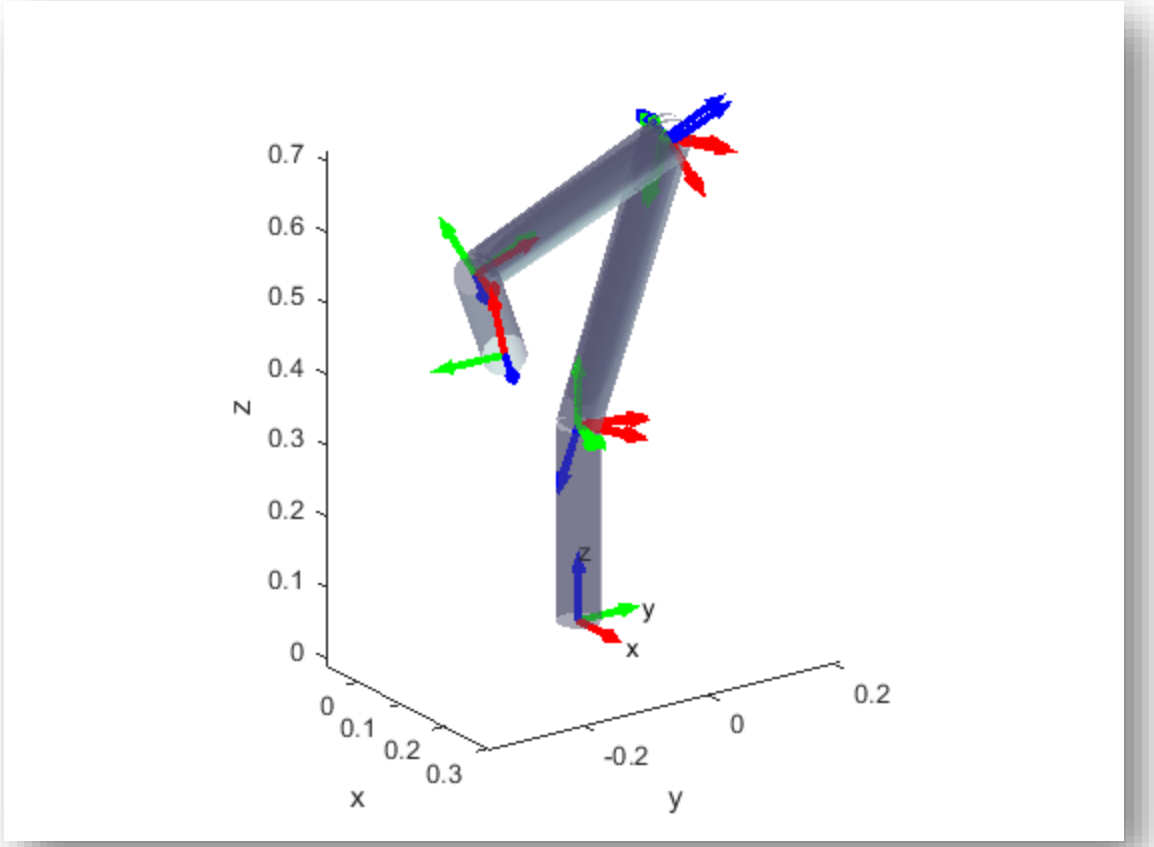
Task 1: Results



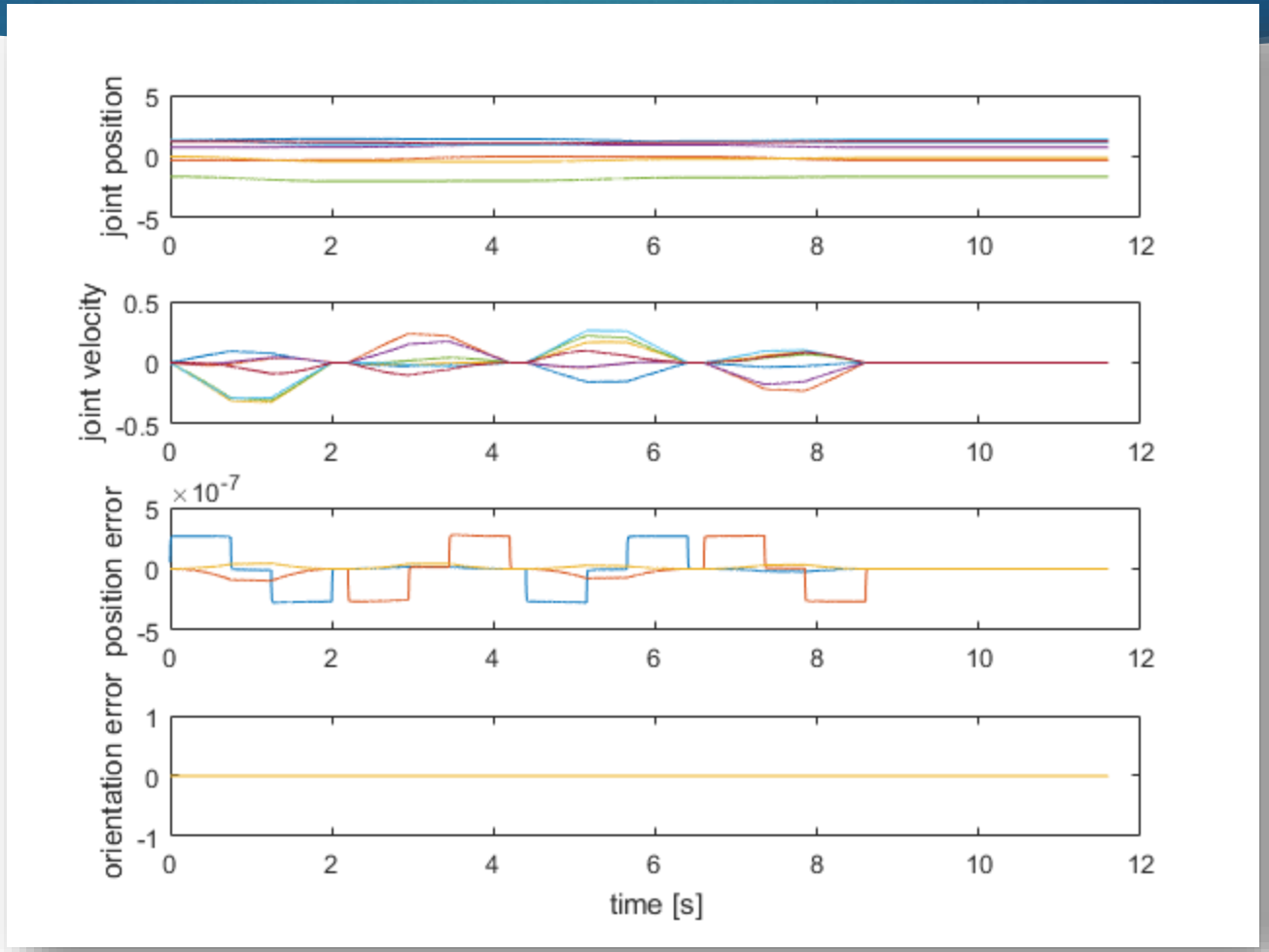
Task 1: Results



Task 1: Results



Task 1: Results



Task 1 Demo >>

Task 2: Problem Statement

- In a second run, the control objective is given by **both the position and the orientation**. While the position needs to move according to the indications above, the orientation needs to be controlled such that **it is kept constant at the initial value.**

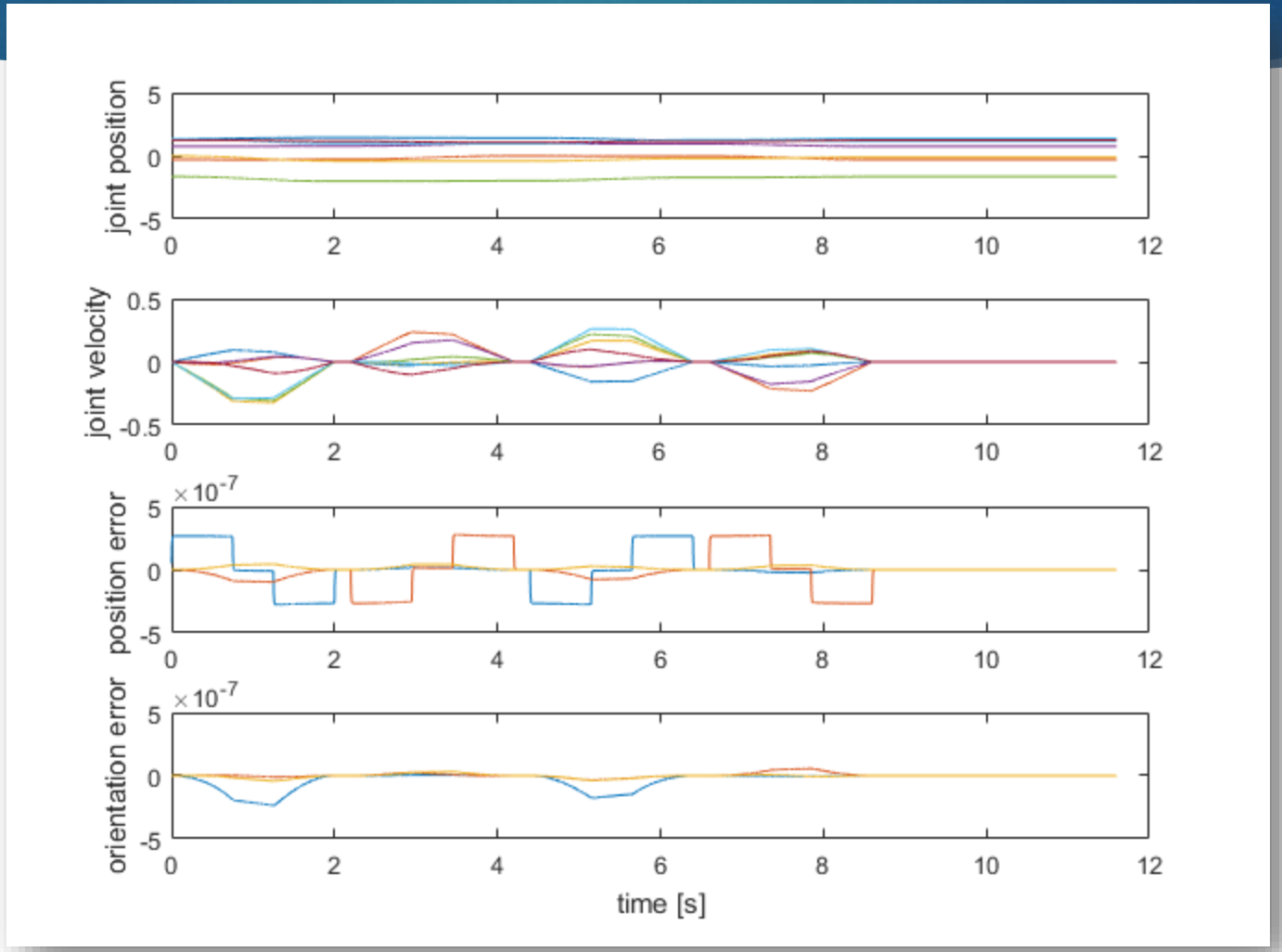
Task 2: Steps

- The initial rotation matrix is calculated from the DH table specified by first calculating the base to end effector homogenous transformation matrix.
- The transformation matrix gives the rotation matrix.
- From the rotation matrix, we get the current quaternion orientation.
- We set the current quaternion orientation as the desired quaternion position.

Adding Quaternion Error in the feedback

- The quaternion error is added in the feed back by subtracting it from the current quaternion orientation.
- The current quaternion orientation is calculated during each iteration.
- The difference as an error is added in the error vector as the quaternion error which was previously set to 0.

Task 2: Results



Task 2 Demo >>

Task 3: Problem Statement

- In a third run the end-effector orientation needs to be changed such that, when running the first side, it is rotated of **−20◦ around the zee axis** according to a trapezoidal velocity profile of proper angular cruise velocity.
- During the second side the inverse rotation is commanded such that, in the second corner, **the orientation is equal to the initial one**.
- And so on …

Task 3: Steps

- For each line segment, the rotation direction is different, therefore, different trapezoidal angular velocity profiles were obtained.
- The cruise velocity is set according the constraints that were showed before.
- The outputs of the trapezoidal function is the angle as well as the angular velocity.

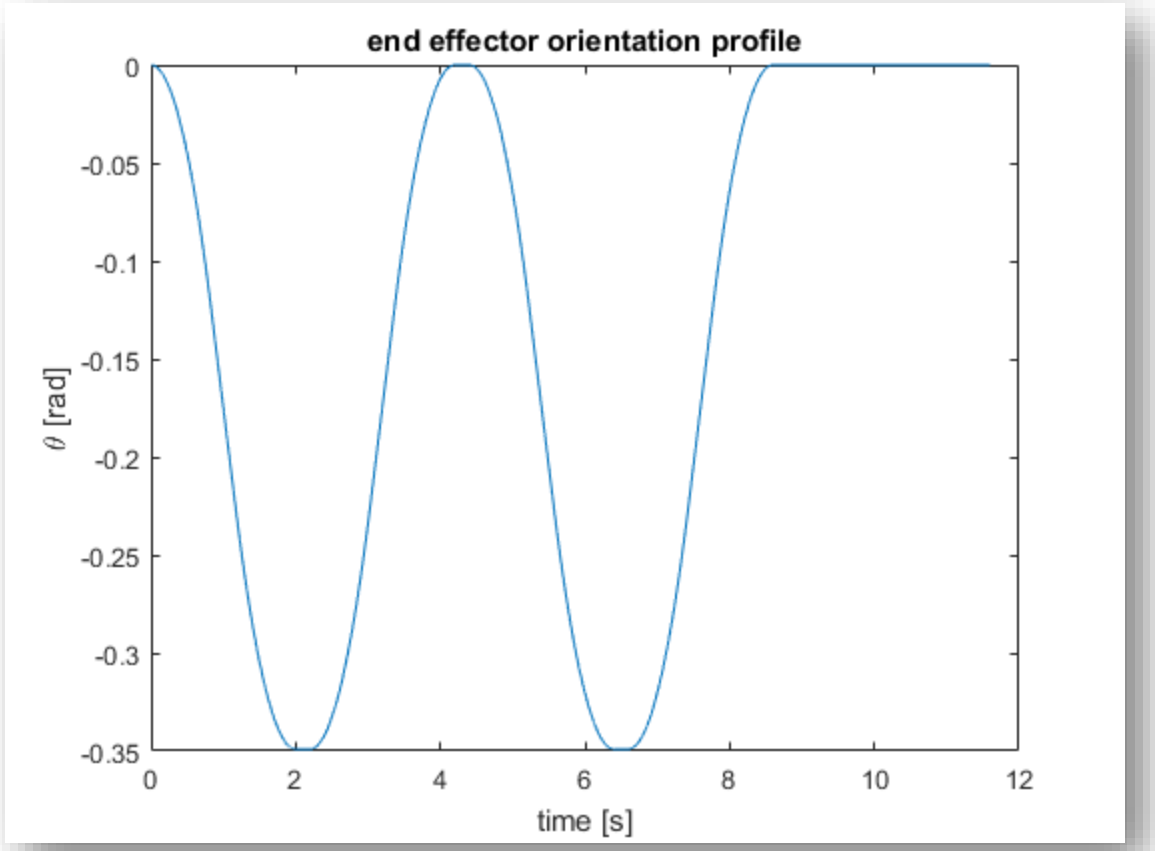
Calculating The Desired Rotation

- The desired rotation is calculated by calculating the rotation matrix around the Z –Axis.
- The value of theta obtained from the trapezoidal function is used to calculate the rotation.
- In order to get the current desired rotation, the rotation matrix calculated is multiplied with the initial rotation matrix. The multiplication is from the left since we want the rotation according to the initial base frame.

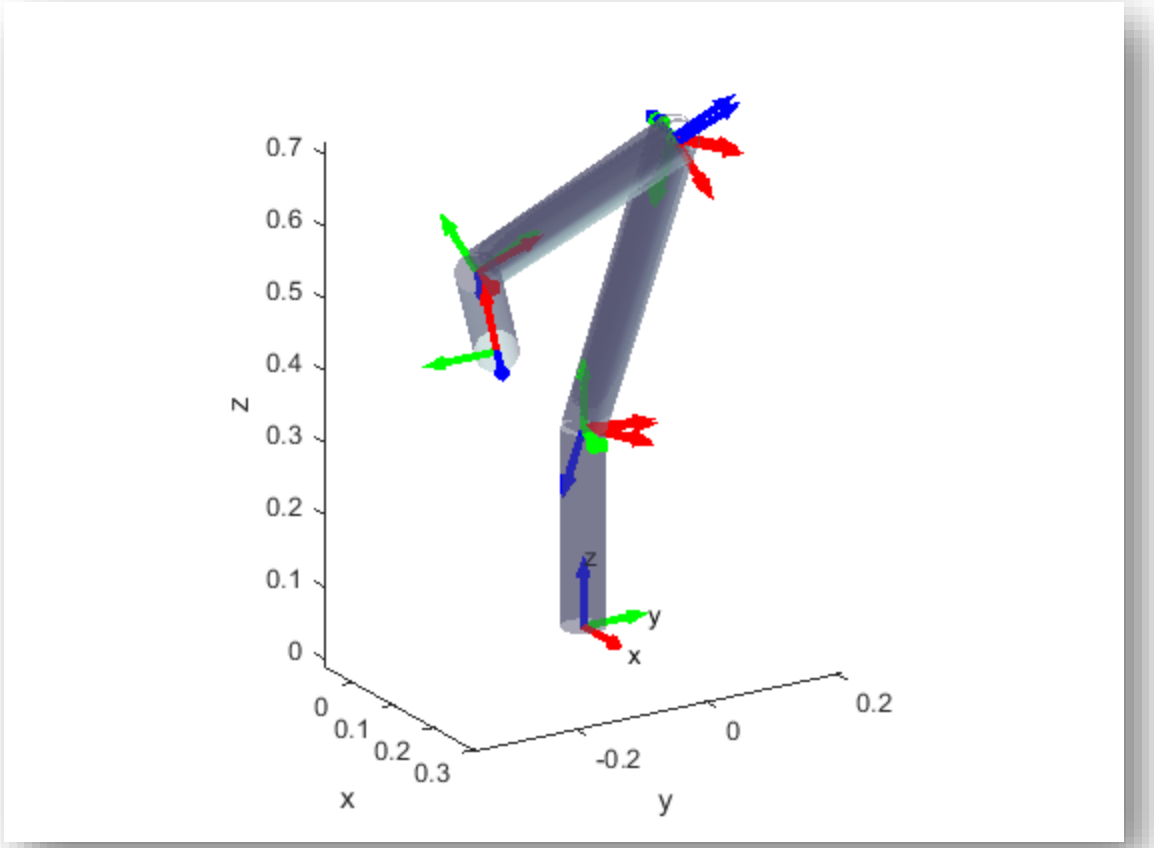
Quaternion Error

- The desired rotation is then converted into quaternion.
- Once we have converted in into quaternion, we calculate the difference between the current and desired quaternion orientation.
- The Angular Velocity is only taken around the z-axis of the end effector as this is the axis around which we want the rotation.
- We feed back these parameters.

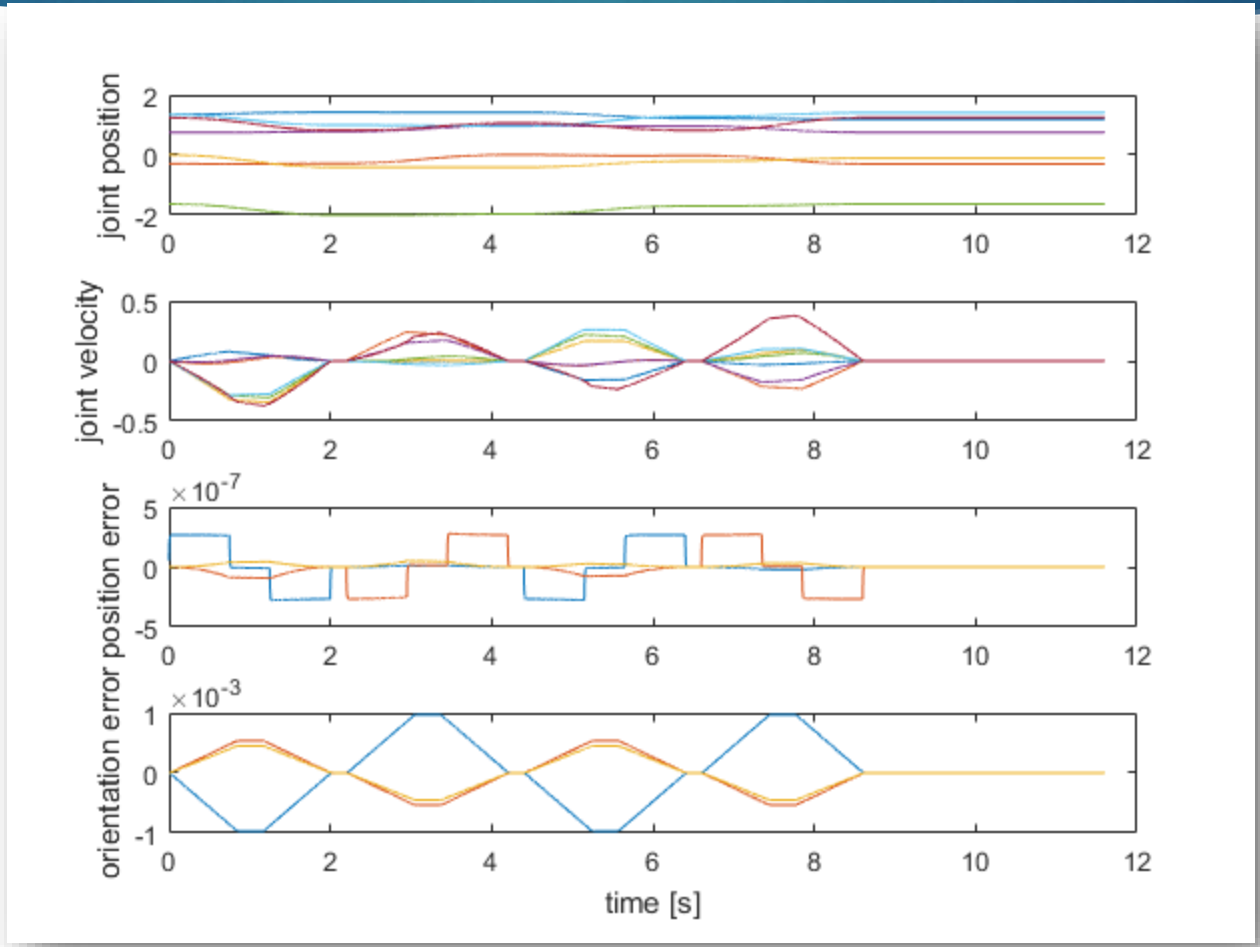
Task 3: Results



Task 3: Results



Task 3: Results



Task 3 Demo >>

Task 4: Problem Statement

- Finally, with respect to the last movement, **the redundancy needs to be exploited** by maximizing the manipulability.

Task4: Steps

- We calculate the manipulability measure by using the manipulability Jacobian provided. It is then multiplied by K_a.

Internal motion characterization

$$\dot{\boldsymbol{q}}_a = k_a \left(\frac{\partial w(\boldsymbol{q})}{\partial \boldsymbol{q}} \right)^{\mathrm{T}}$$

manipulability measurement

$$w(\boldsymbol{q}) = \sqrt{\det\left(\boldsymbol{J}(\boldsymbol{q})\boldsymbol{J}^{\mathrm{T}}(\boldsymbol{q})\right)}$$

distance from joints mechanical limits

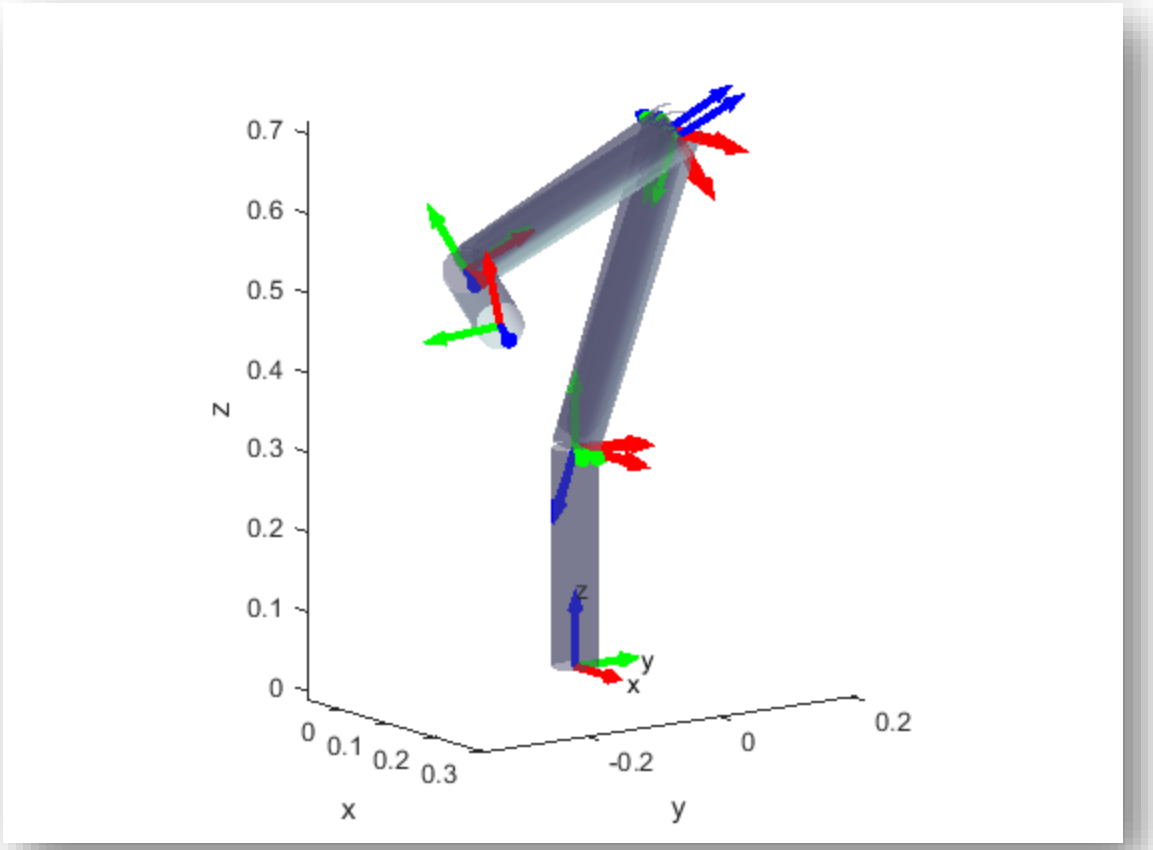
$$w(\boldsymbol{q}) = -\frac{1}{2n} \sum_{i=1}^{n} \left(\frac{q_i - \bar{q}_i}{q_{iM} - q_{im}} \right)^2$$

Feedback Modification

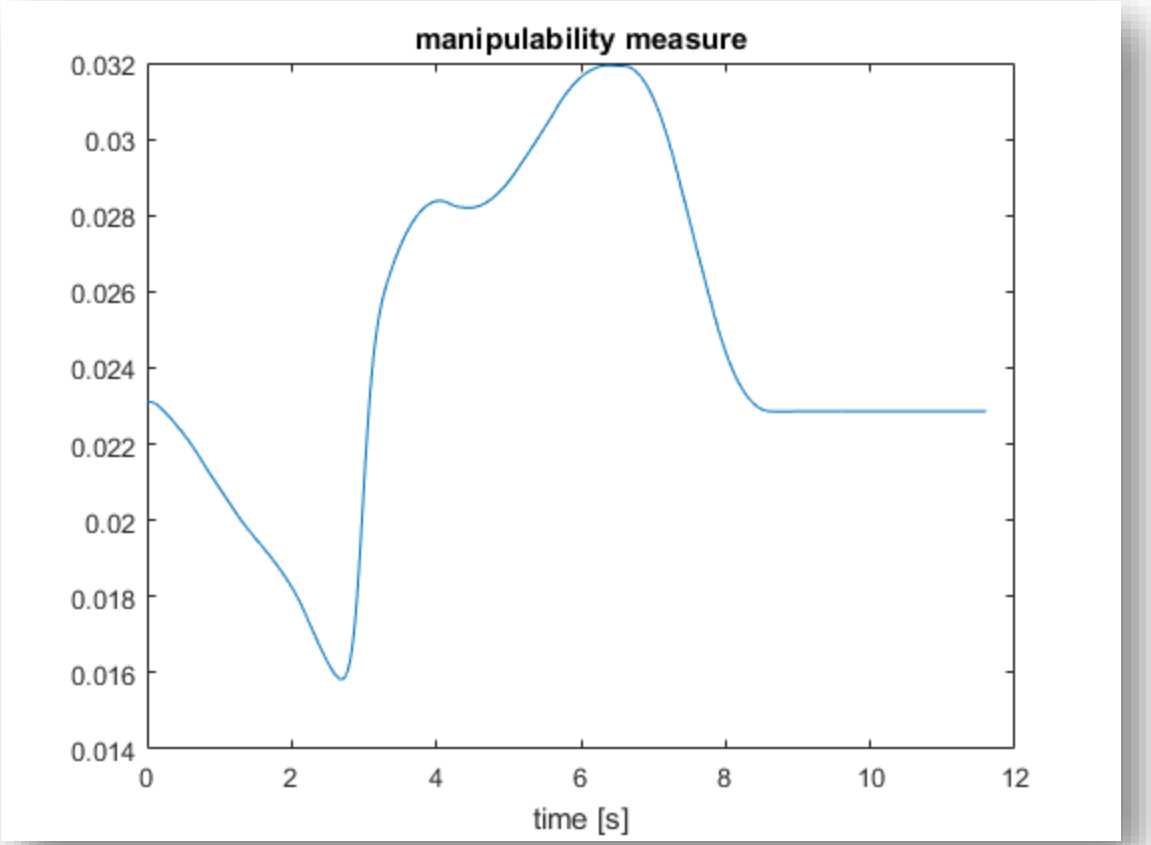
- The feedback loop is then modified to add the redundancy.

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_P^{\dagger}(\dot{\boldsymbol{p}}_d + \boldsymbol{K}_P \boldsymbol{e}_P) + (\boldsymbol{I} - \boldsymbol{J}_P^{\dagger}\boldsymbol{J}_P)\dot{\boldsymbol{q}}_a$$

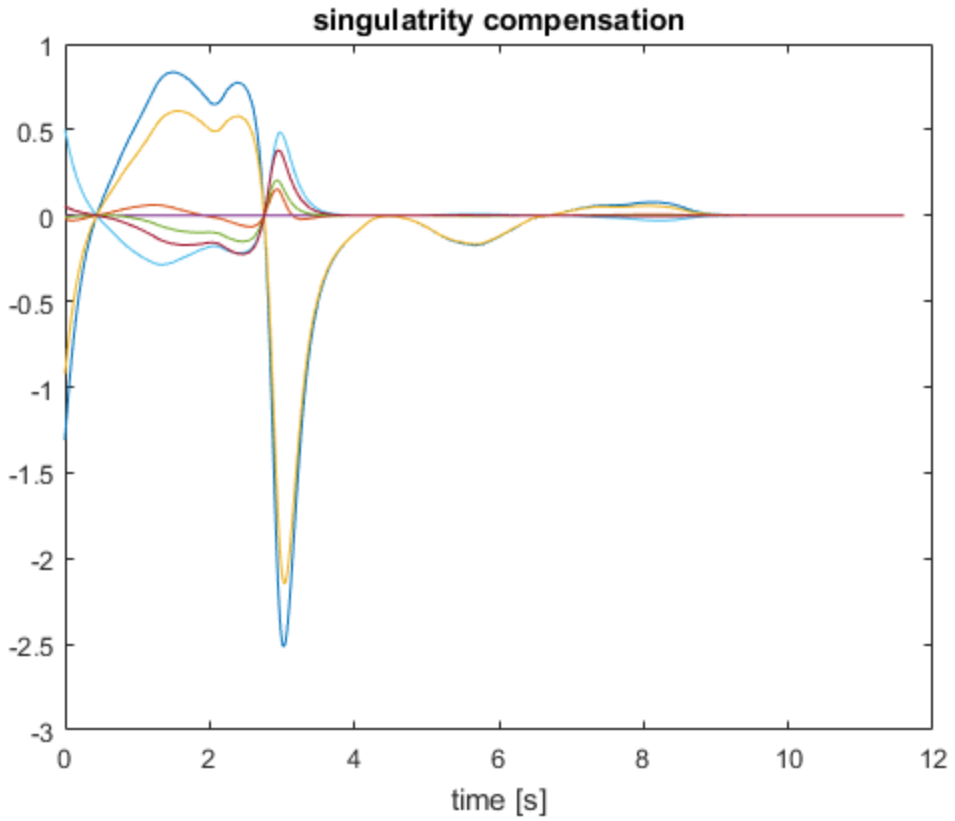
Task 4: Results



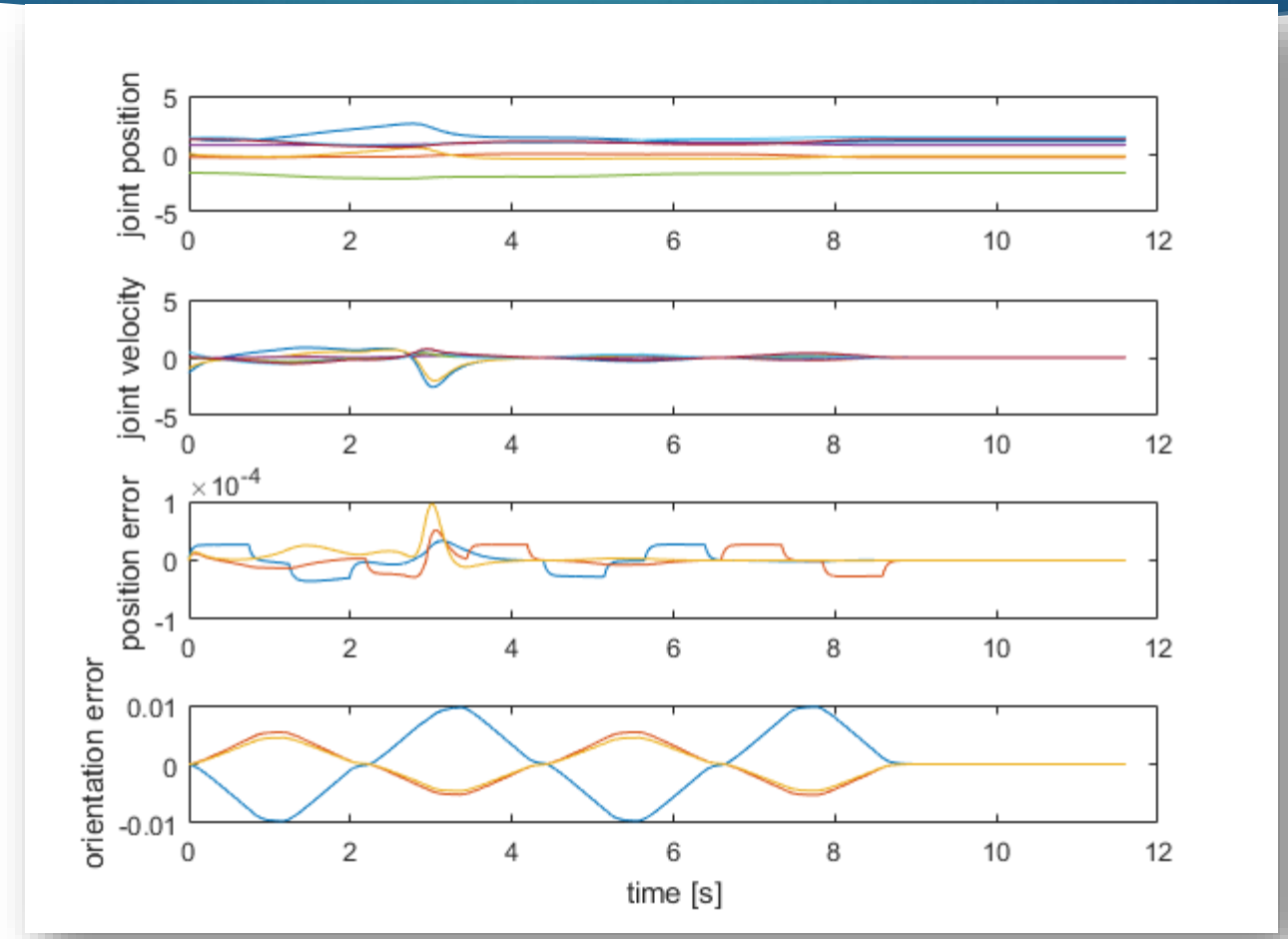
Task 4: Results



Task 4: Results



Task 4: Results



Task 4 Demo >>

THANK YOU!