



# Introduction to Robotics

- ▶ Final Project Assignment.
  - ▶ Assignment No. **14**.
- ▶ Presented By: **Zohaib Salahuddin**.
- ▶ Presented To: *Prof. Gianluca Antonelli*

# TASK NO. 1

- ▶ The control objective is position only **without exploiting the redundancy**.
- ▶ The required path is a horizontal circumference lying on **xy** axis with **radius of 10 cm** in length. The end effector occupies the upper point of the circumference looking from the z axis.
- ▶ The trajectory has a duration of **8s** and a *trapezoidal velocity profile characterized by a proper cruise velocity*.
- ▶ The robot stops for an additional **3s** once back in the initial end effector position.



# Denavit Hartenberg Parameters.

- ▶ Kinova Jaco2 7- DOFs with Spherical Wrist
- ▶ Provided Functions Used for Kinova to DH conversion and vice versa

joint	$a$ (m)	$\alpha$ (°)	$d$ (m)	$\theta$ (°)
1	0	90°	0.2755	$\theta_1$
2	0	90°	0	$\theta_2$
3	0	90°	-0.410	$\theta_3$
4	0	90°	-0.0098	$\theta_4$
5	0	90°	-0.3111	$\theta_5$
6	0	90°	0	$\theta_6$
7	0	0°	0.2638	$\theta_7$

TABLE I: DH parameter

# 1. Getting the 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 the 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 result is saved as the **initial displacement and the rotation of the end effector**.

## 2. Calculating the Center and the desired position for each iteration

- ▶ The initial displacement vector obtained by homogenous transformation matrix.
- ▶ The required location in the **xy frame** is obtained according the parametric equations of the circular trajectory.
- ▶ The center point is obtained by subtracting radius from the initial position of the end effector in the y direction.
- ▶  $[x\_init, y\_init, z\_init] - [0 \text{ radius } 0] + [r \sin(s/r), r \cos(s/r), 0]$ , gives the desired position in the xy-plane.

### 3. Trapezoidal Velocity Profile

- ▶ With a sampling rate of 1000, time instances are fixed from 0 to 8s.
- ▶ According to the **parametric equations**, we need the arc length to vary from **0 to 2\*pi** for a circular trajectory in 8s.
- ▶ We use the trapezoidal velocity to get the arc length, the velocity and the acceleration.
- ▶ The cruise velocity is fixed according the following equation:

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

### 3. Trapezoidal Velocity Profile

- ▶ With a sampling rate of 1000, time instances are fixed from 0 to 8s.
- ▶ According to the **parametric equations**, we need the arc length to vary from **0 to 2\*pi** for a circular trajectory in 8s.
- ▶ We use the trapezoidal velocity to get the arc length, the velocity and the acceleration.
- ▶ The cruise velocity is fixed according the following equation:

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



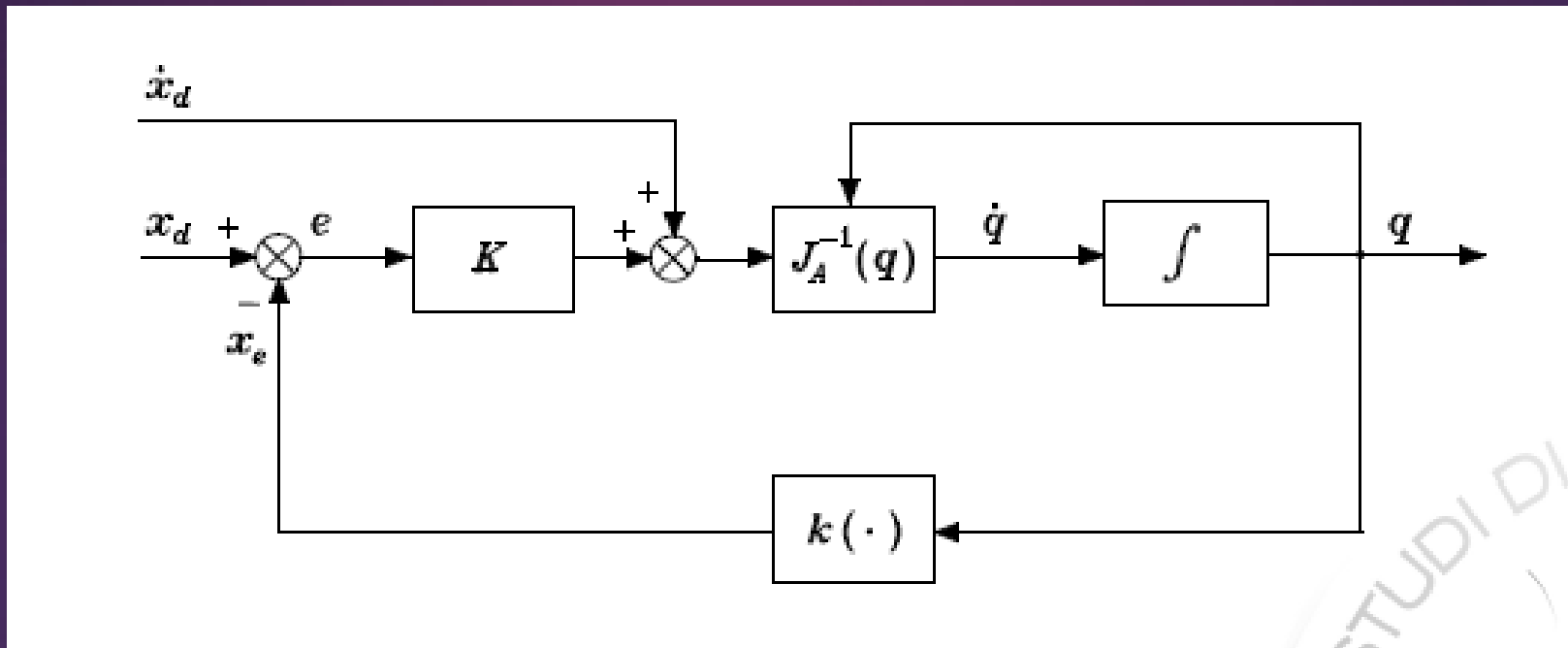
## 4. Desired Position and Linear Velocity.

- The desired position and velocity is calculated according to the following parametric equations:

$$p(s) = c + R \begin{bmatrix} \rho \cos(s/\rho) \\ \rho \sin(s/\rho) \\ 0 \end{bmatrix}$$
$$\dot{p} = R \begin{bmatrix} -\dot{s} \sin(s/\rho) \\ \dot{s} \cos(s/\rho) \\ 0 \end{bmatrix}$$

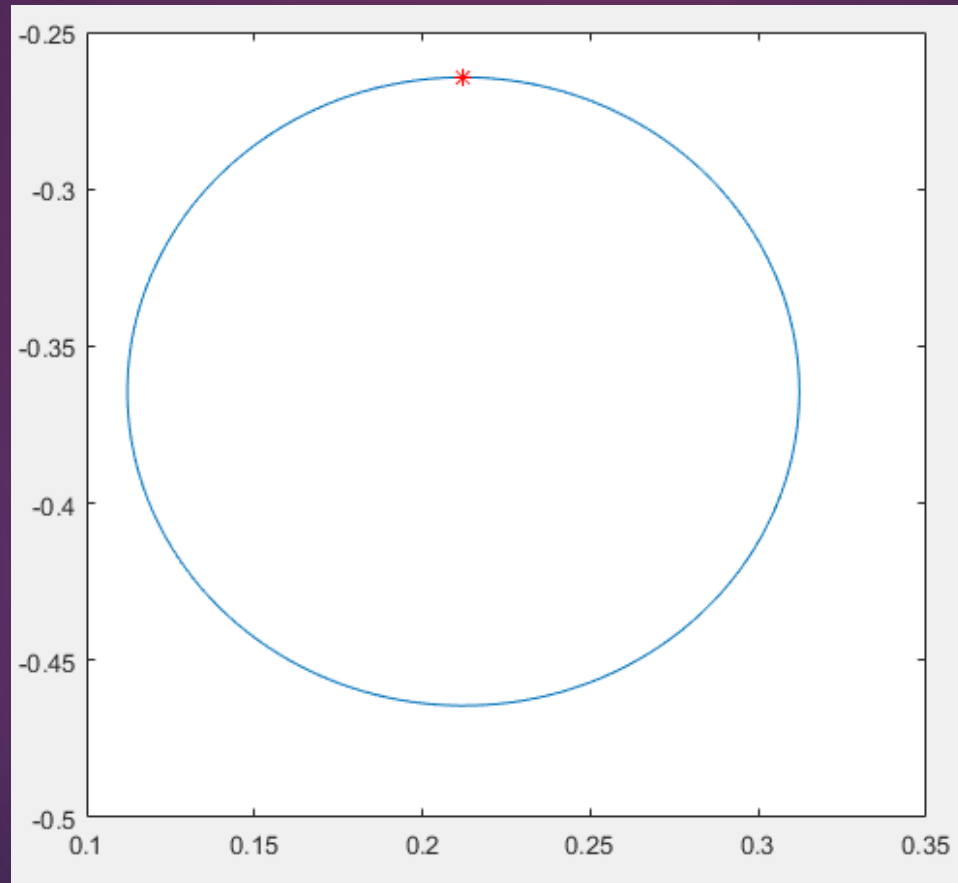
## 5. Error and Feedback.

- Only the positional error and the desired linear velocity is taken into consideration in this case. The following feedback is implemented to the new position at each iteration.



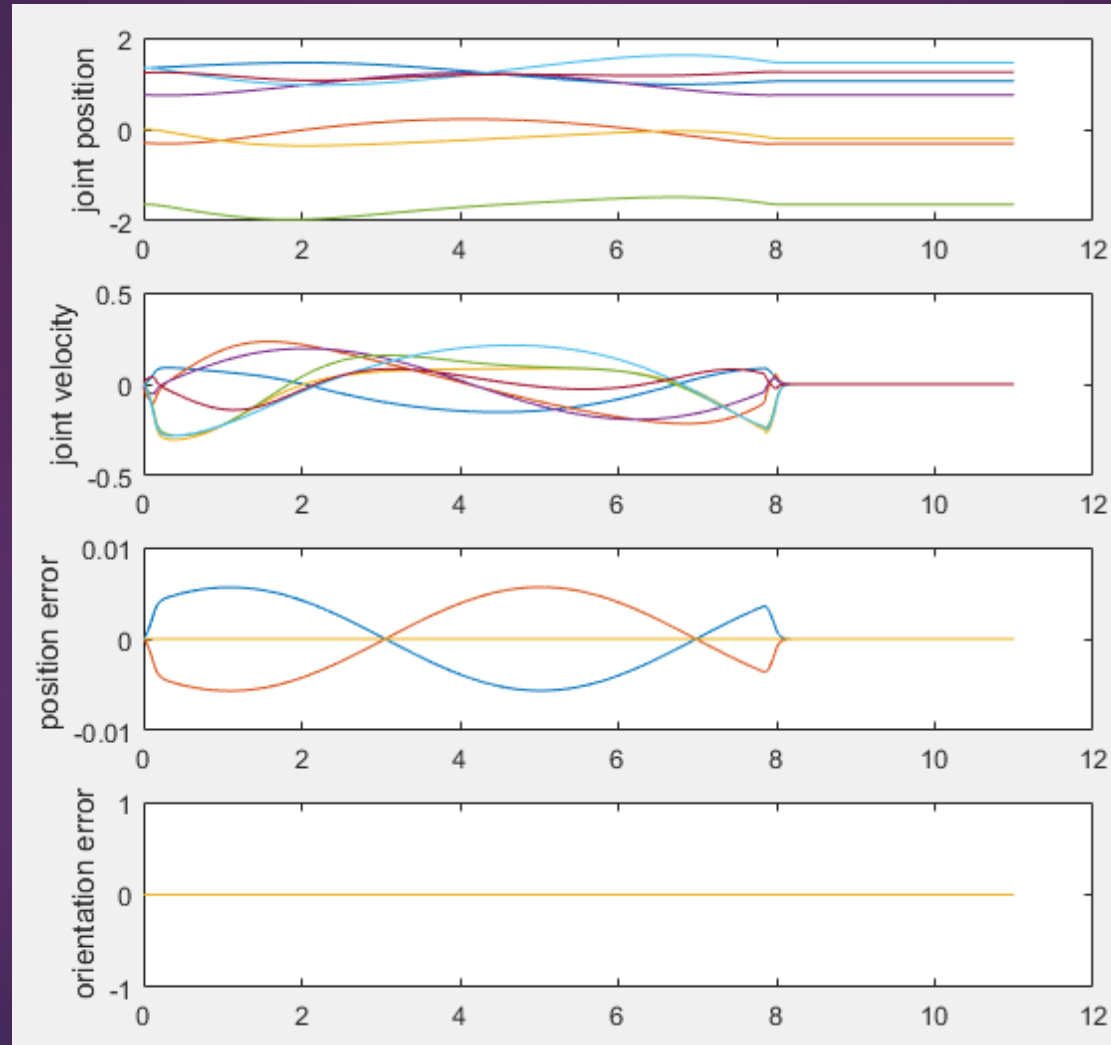
The joint angles are then fed to the  $V_{rep}$  using the given function

# RESULTS – Upper Circumference.



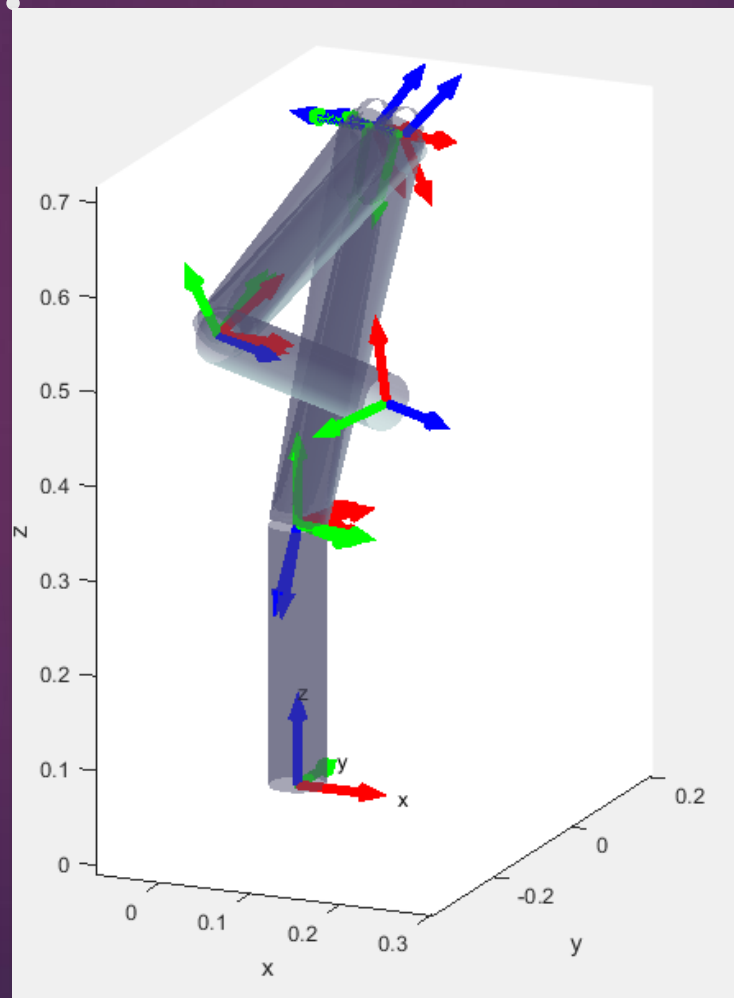
X and Y axis

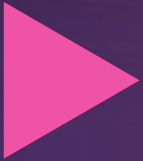
# RESULTS – Graphs.



X – axis is in seconds

# RESULTS – Draw Robot Initial and Final Position.





# DEMO – Task 1

# TASK NO. 2

- ▶ *In the 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.***

# Calculating the initial rotation matrices

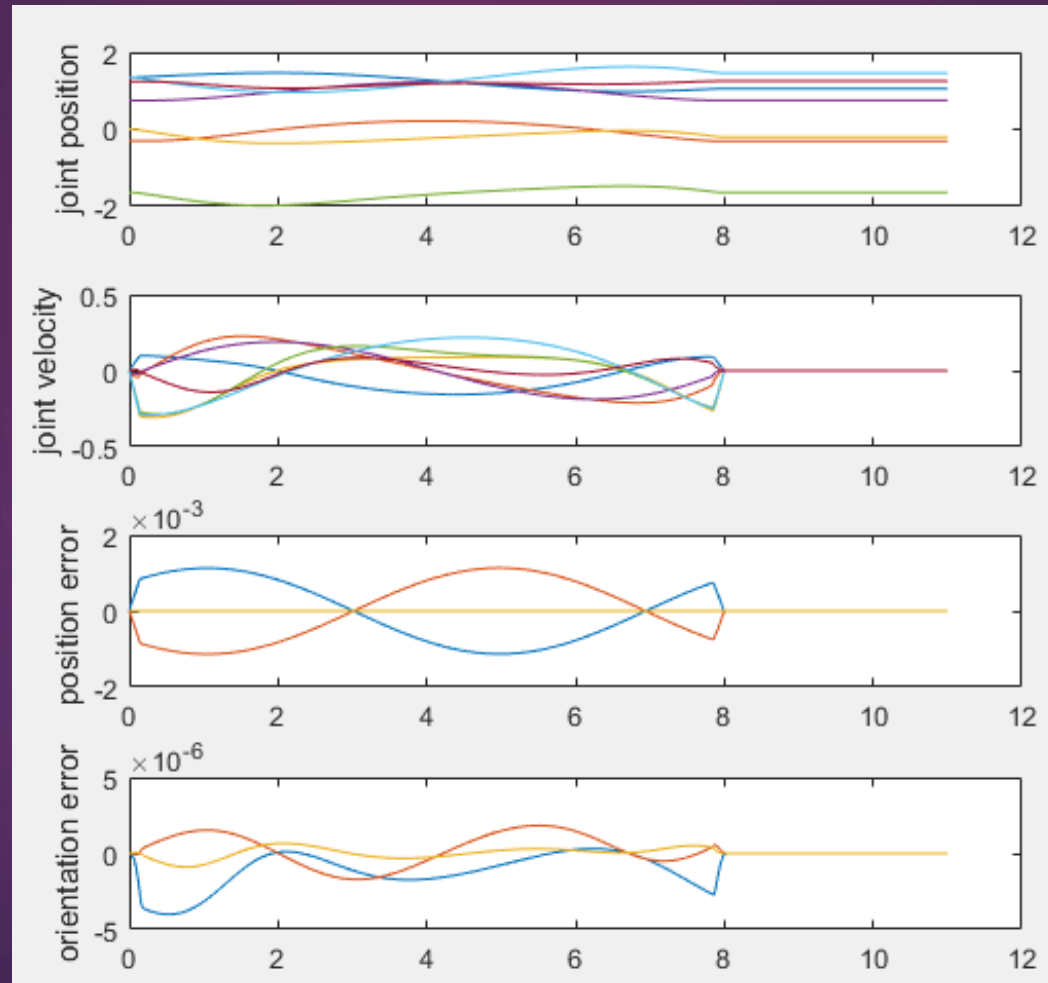
- ▶ The initial rotation matrix is calculated from the DH table specified by first calculating the base to end effector homogenous transformation matrix.
- ▶ The transformation matrix gives the rotation matrix.
- ▶ From the rotation matrix, we get the current quaternion orientation.
- ▶ We set the current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



# Adding Quaternion Error in the feed back.

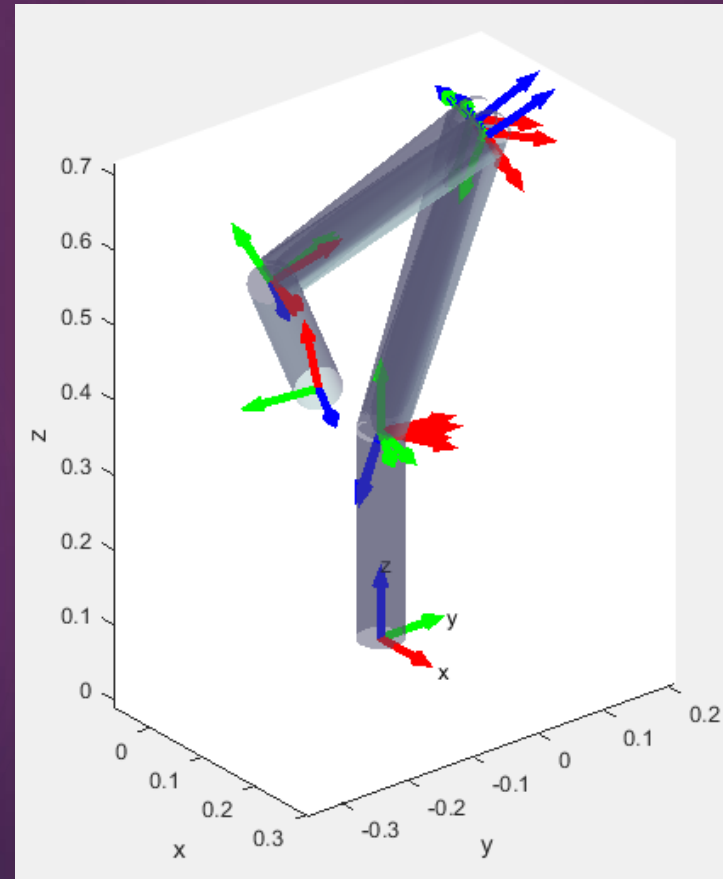
- ▶ The quaternion error is added in the feed back by subtracting it from the current quaternion orientation.
- ▶ The current quaternion orientation is calculated during each iteration.
- ▶ The difference as an error is added in the error vector as the quaternion error which was previously set to 0.

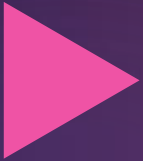
# RESULTS – Graphs.



X – axis is in seconds

# RESULTS – Draw Robot Initial and Final Position.





# DEMO – Task 2

# TASK NO. 3

- ▶ *In the third run, the end-effector orientation needs to be changed such that it implements a  **$-\pi$  rotation** around the **end effector z-axis**.*
- ▶ This should be done according to a trapezoidal velocity profile of proper angular cruise velocity during the whole previous movement.

# 1. Trapezoidal Angular Velocity

- ▶ Since the rotation is of  $-\pi$ , the cruise velocity is set according the constraints that were showed before.
- ▶ The outputs of the trapezoidal function which is the angle as well as the angular velocity.

## 2. Calculating the desired Rotation

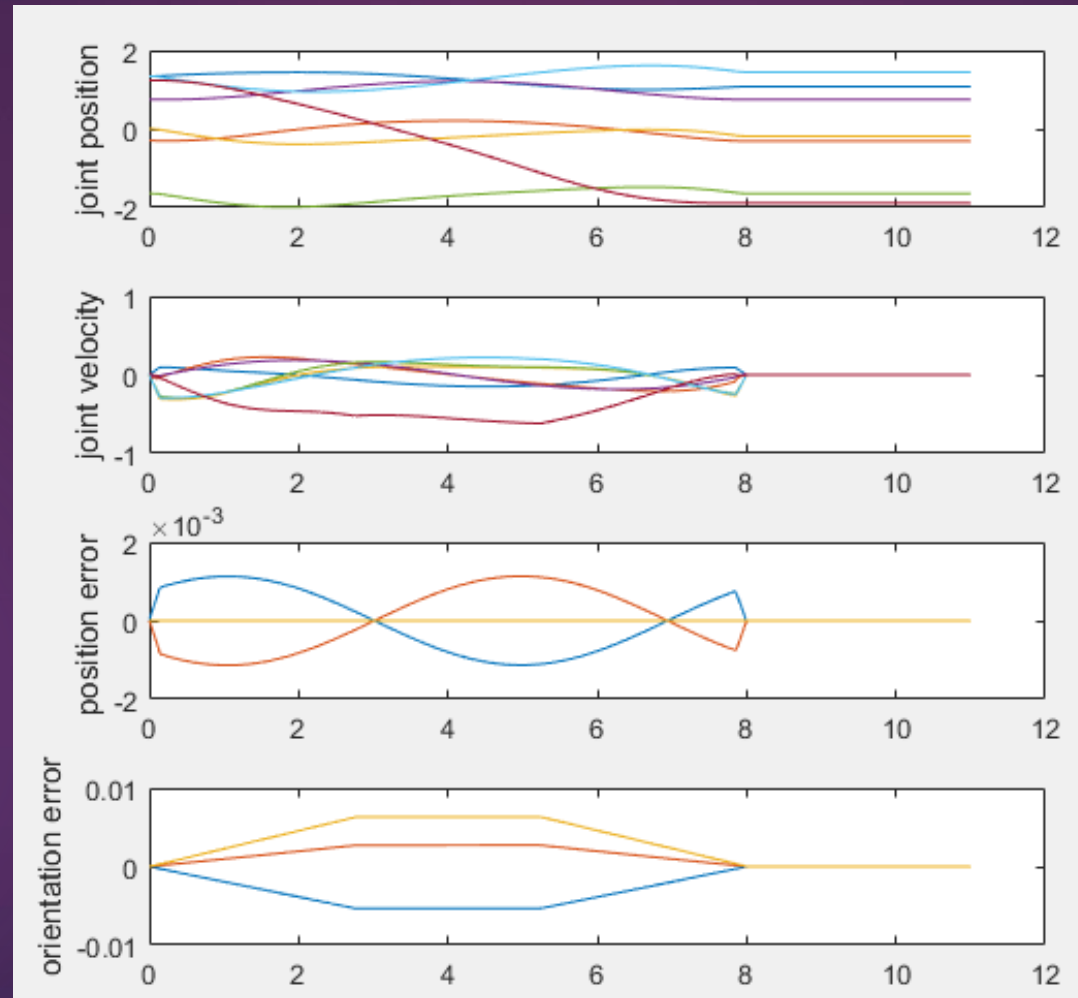
- ▶ The desired rotation is calculated by calculating the rotation matrix around the Z – Axis.
- ▶ The value of theta obtained from the trapezoidal function is used to calculate the rotation.
- ▶ In order to get the current desired rotation, the rotation matrix calculated is multiplied with the initial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

### 3. 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.

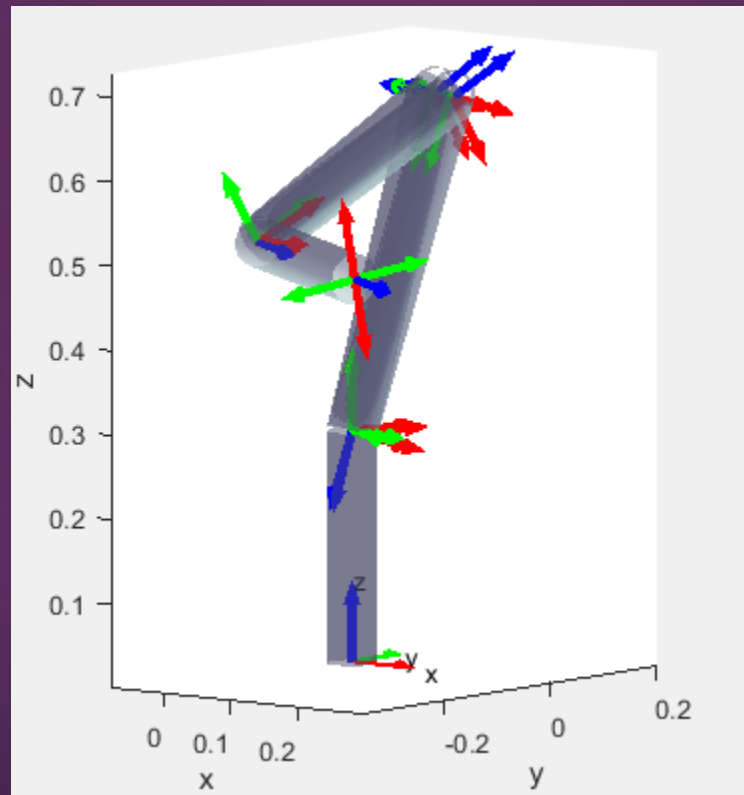


# RESULTS – Graphs.

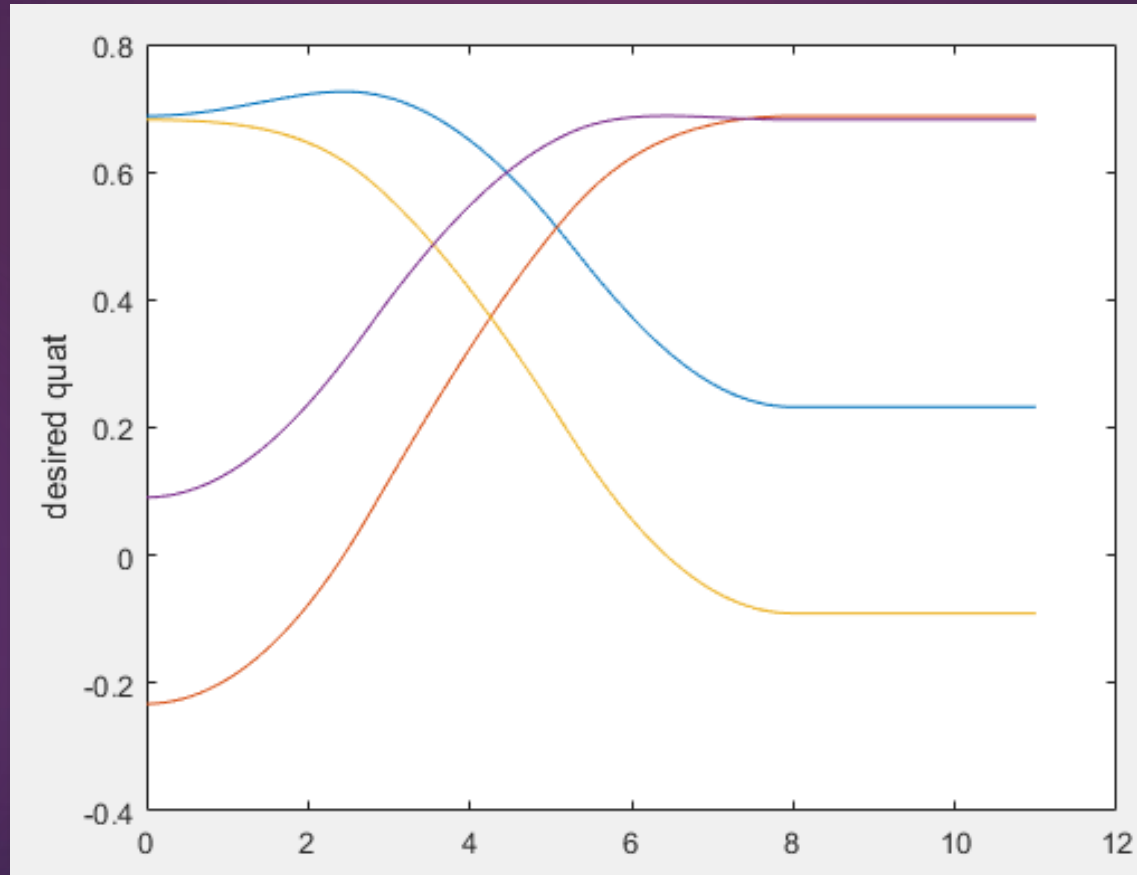


X – axis is in seconds

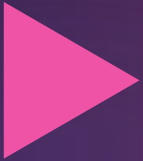
# RESULTS – Draw Robot Initial and Final Position.



# Trapezoidal Angular Profile



X – axis is in seconds



# DEMO – Task 3

# TASK NO. 4

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

# Manipulability Measure.

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

Internal motion characterization

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

*manipulability measurement*

$$w(q) = \sqrt{\det(J(q)J^T(q))}$$

*distance from joints mechanical limits*

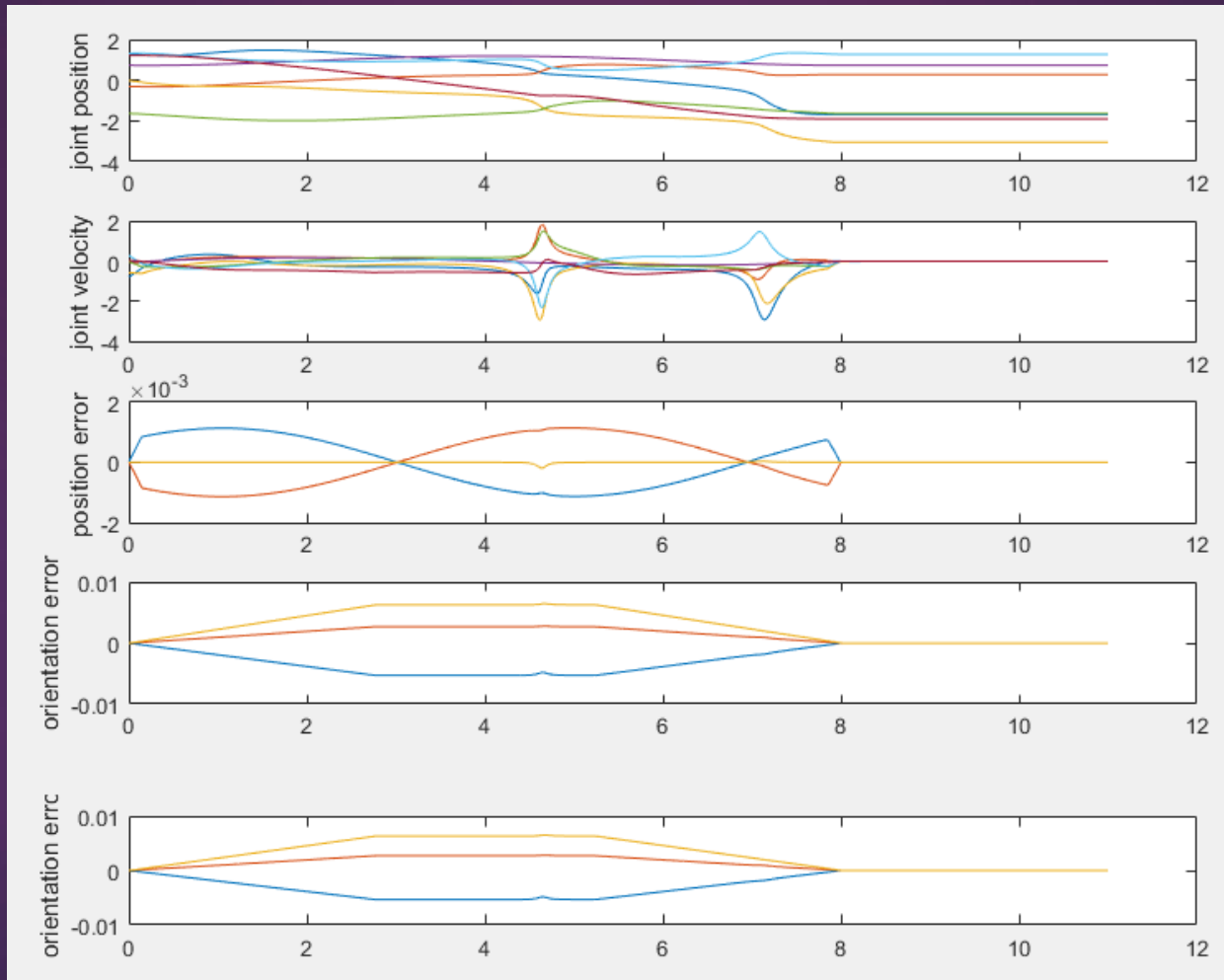
$$w(q) = -\frac{1}{2n} \sum_{i=1}^n \left( \frac{q_i - \bar{q}_i}{q_{iM} - q_{im}} \right)^2$$

# Modification of Feed back.

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

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

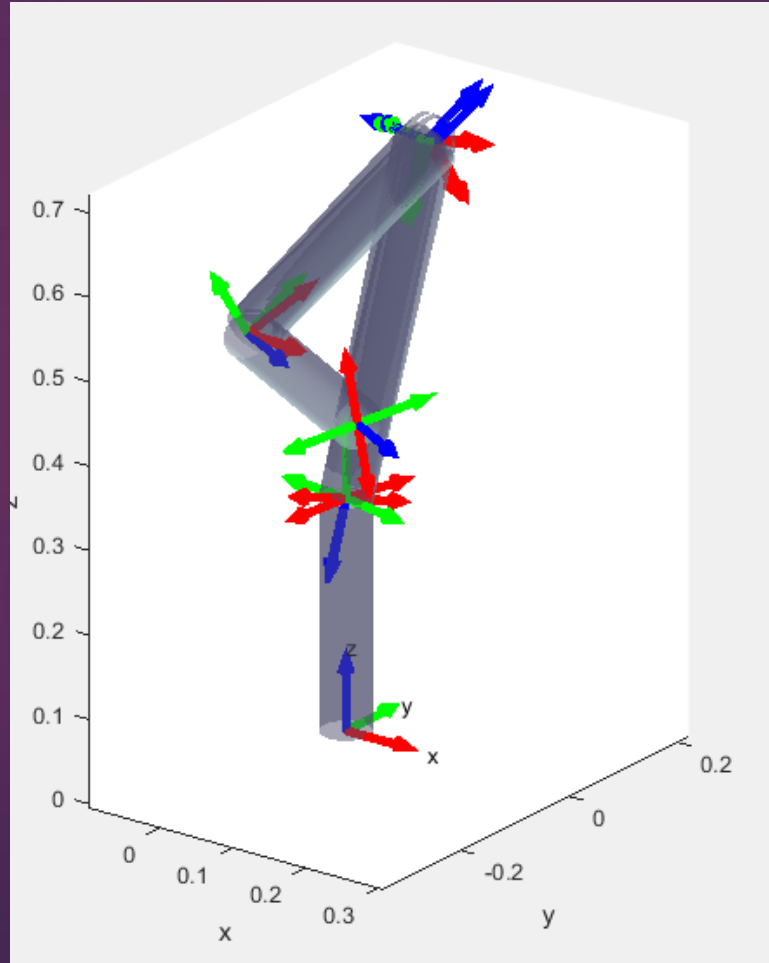
# RESULTS – Graphs.



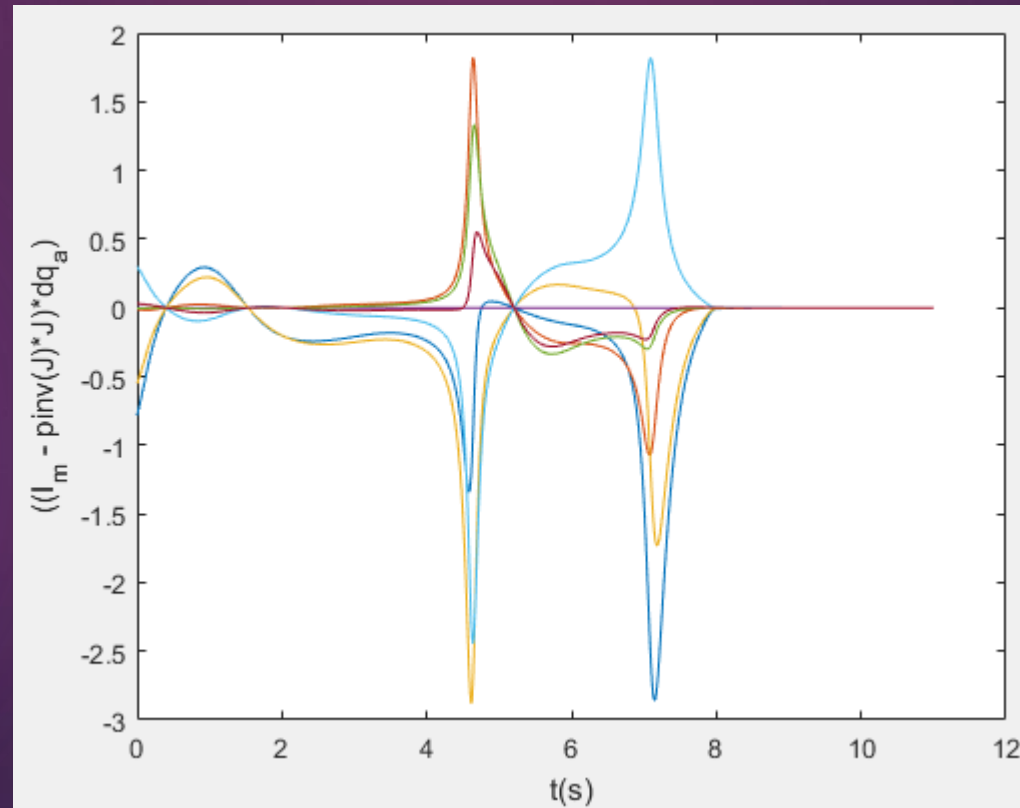
X – axis is in seconds

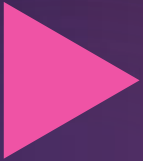


# RESULTS – Draw Robot Initial and Final Position.



# RESULTS – Jacobian Redundancy Compensation for Singular Avoidance





# DEMO – Task 4



**THANK YOU.**