Energy-Aware Path Choice with Computed-Torque Tracking

Robotics Project

2025-2026

Ahmad Abu Zainab - 6320
Lebanese University - Faculty of Engineering III
Robotics

Table of Contents

I. Introduction

The goal of this project is to track different end-effector paths to compute torque (via inverse dynamics), tracking error and total energy consumed at each path. For this project, we will be using a simplified model of the SCARA robot configuration and resolved rate control with PID for control as resolved rate control doesn't involve forward dynamics as other control methods do. [1]

II. Theory and Implementation

For this project we will use a simplified version of the SCARA robot model in which we make a few assumptions to simplify our work

1. No joint limits (This doesn't really change much as we can simply take a path that doesnt require us to exceed usual SCARA robot joint limits)
2. A movement of Δq of the robot configuration is always possible and unimpeded (If we command the robot to move to a certain position in the q-space it will always perform that movement)

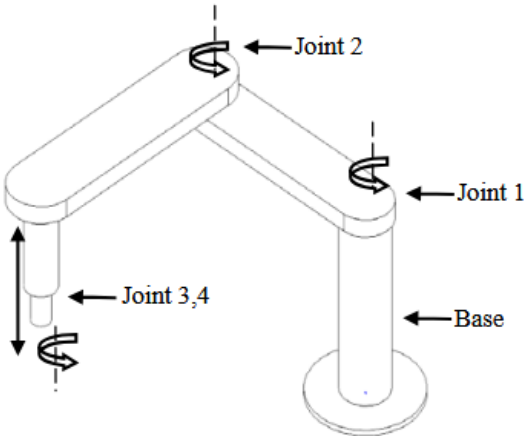
II.1. Kinematics

DH Parameters

First we define the robot parameters as follows

Figure 1 — Configuration of the SCARA Robot [2]

(a) The structure of the SCARA robot



(b) The coordinate systems of the SCARA robot

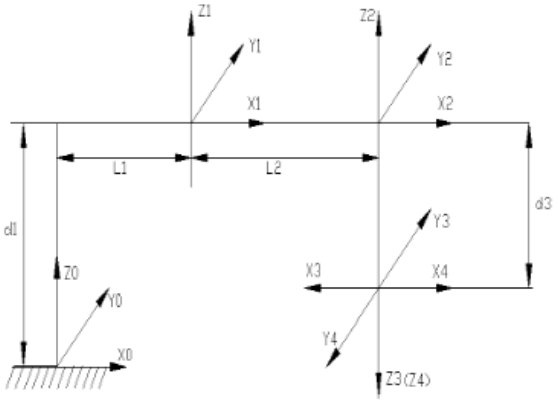


Table 1 — DH Parameters of the SCARA Robot

Joints	d_i	a_{i-1}	α_{i-1}	θ_i
1	1	0.5	0	θ_1
2	0	0.5	0	θ_2
3	$-d_3$	0	0	0
4	0	0	0	θ_4

We apply the non-proximal DH parameter transformation matrix. [3]

$$ {}^{i-1}_{i}T = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Here we can lock the orientation of the end-effector ($\theta_4 = 0$) as it's not really relevant to our study, so our $\boldsymbol{q}$ vector becomes.

$$ \boldsymbol{q} = [\theta_1 \;\; \theta_2 \;\; d_3]^{\mathsf{T}} $$

Finally we obtain the final transformation matrix:

$$ {}^{3}_{0}T = \begin{bmatrix} c_{124} & -s_{124} & 0 & 0.5c_1 + 0.5c_{12} \\ s_{124} & c_{124} & 0 & 0.5s_1 + 0.5s_{12} \\ 0 & 0 & 1 & 1 - d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Forward Kinematics

Using the transformation matrix we can get the position of the end-effector $\boldsymbol{x}$

$$ \boldsymbol{x} = f(\boldsymbol{q}) = \begin{bmatrix} 0.5\cos\theta_1 + 0.5\cos\theta_1 + \theta_2 \\ 0.5\sin\theta_1 + 0.5\sin\theta_1 + \theta_2 \\ 1 - d_3 \end{bmatrix} $$

Inverse Kinematics

While inverse kinematics are not needed in this study we can easily deduce it from the forward kinematics

$$ \boldsymbol{q} = \begin{bmatrix} \operatorname{atan2}(x, y) - \operatorname{atan2}\left(\frac{1}{2} + \frac{\gamma}{2}, \frac{\delta}{2}\right) \\ \pm\operatorname{acos}(2(x^2 + y^2) - 1) \\ 1 - z \end{bmatrix} \text{ where } \gamma = 2x^2 + 2y^2 - 1 \text{ and } \delta = \sqrt{1 - \gamma^2} $$

II.2. Jacobian

We can easily obtain the Jacobian and inverse Jacobian using direct diffrentiation

$$\boldsymbol{J} = \begin{bmatrix} \frac{\partial f_1}{\partial q_1} & \frac{\partial f_1}{\partial q_2} & \cdots & \frac{\partial f_1}{\partial q_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial q_1} & \frac{\partial f_m}{\partial q_2} & \cdots & \frac{\partial f_m}{\partial q_n} \end{bmatrix} = \begin{bmatrix} -0.5\sin(\theta_1) - 0.5\sin(\theta_1+\theta_2) & -0.5\sin(\theta_1+\theta_2) & 0 \\ 0.5\cos(\theta_1) + 0.5\cos(\theta_1+\theta_2) & 0.5\cos(\theta_1+\theta_2) & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

$$\boldsymbol{J}^{-1} = \begin{bmatrix} \frac{2\cos(\theta_1+\theta_2)}{\sin(\theta_2)} & \frac{2\sin(\theta_1+\theta_2)}{\sin(\theta_2)} & 0 \\ -\frac{2\cos(\theta_1)+2\cos(\theta_1+\theta_2)}{\sin(\theta_2)} & -\frac{2\sin(\theta_1)+2\sin(\theta_1+\theta_2)}{\sin(\theta_2)} & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

II.3. Selected Trajectory

For this project we selected 3 (reachable) points arbitrarily

$$\boldsymbol{p_1} = f\left(-\frac{\pi}{6}, -\frac{\pi}{2}, 0\right) = \begin{bmatrix} 0.1830127 \\ -0.6830127 \\ 1 \end{bmatrix} \quad \boldsymbol{p_2} = f\left(\frac{\pi}{6}, \frac{\pi}{2}, 1\right) = \begin{bmatrix} 0.1830127 \\ 0.6830127 \\ 0 \end{bmatrix}$$

$$\boldsymbol{p}_i = \begin{bmatrix} 0.5 \\ 0 \\ 0.75 \end{bmatrix}$$

Then we define 3 paths($t \in [0,1]$): a straight line from $\boldsymbol{p}_1$ to $\boldsymbol{p}_2$, a straight line from $\boldsymbol{p}_1$ to an intermediate point $\boldsymbol{p}_i$ then to $\boldsymbol{p}_2$, and a cubic spline from $\boldsymbol{p}_1$ to $\boldsymbol{p}_2$ passing through $\boldsymbol{p}_i$ ($\boldsymbol{P}_3'$ at $\boldsymbol{p_i} = f$ otherwise 0).

$$\boldsymbol{P}_1(t) = (\boldsymbol{p}_2 - \boldsymbol{p}_1)t + \boldsymbol{p}_1 \quad \boldsymbol{P}_2(t) = \begin{cases} 2(\boldsymbol{p}_i - \boldsymbol{p}_1)t + \boldsymbol{p}_1 & \text{if } t \le 0.5 \\ 2(\boldsymbol{p}_2 - \boldsymbol{p}_i)(t-0.5) + \boldsymbol{p}_i & \text{if } t > 0.5 \end{cases}$$

$$\boldsymbol{P}_3(t) = \begin{cases} (2\boldsymbol{p}_1 - 2\boldsymbol{p}_i + f)2^3t^3 + (3\boldsymbol{p}_i - 3\boldsymbol{p}_1 - f)2^2t^2 + \boldsymbol{p}_1 & \text{if } t \le 0.5 \\ (2\boldsymbol{p}_i - 2\boldsymbol{p}_2 + f)2^3(t-0.5)^3 + (3\boldsymbol{p}_2 - 3\boldsymbol{p}_i - 2f)2^2(t-0.5)^2 + 2f(t-0.5) + \boldsymbol{p}_i & \text{if } t > 0.5 \end{cases}$$

II.4. Dynamics

While the dynamics for the SCARA robot are readily available they are often large and cumbersome to implement, so for this project we can derive our own from a simplified model of the SCARA robot by assuming the mass of the robot is concentrated at the joints. So for joint 1 the mass is $m_1 = 5\text{kg}$, joint 2 $m_2 = 4\text{kg}$, and joint 3 $m_3 = 2\text{kg}$.

We can the compute the K.E. and P.E. for each joint

$$T = \sum_{i=1}^{m}\left(\frac{1}{2}m_i\sum_{j=1}^{n}\left(\frac{\mathrm{d}q_j}{\mathrm{d}t}\right)^2\right)$$

$$U = \sum_{i=1}^{m} g \cdot z_i \cdot m_i$$

$$L = T - U$$

Finally we plug in L in the Euler–Lagrange equations [4]

$$\varphi_i = \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i}$$

Noting that φ_i is the generalized force where

$$\varphi_i = \begin{cases} \tau_i \text{ if the joint is revolute} \\ f_i \text{ if the joint is prismatic} \end{cases}$$

Computing the resultant torques is tedious, so we can use a package like `sympy` to compute the torques for us [5]. Finally we obtain the following dynamic equation

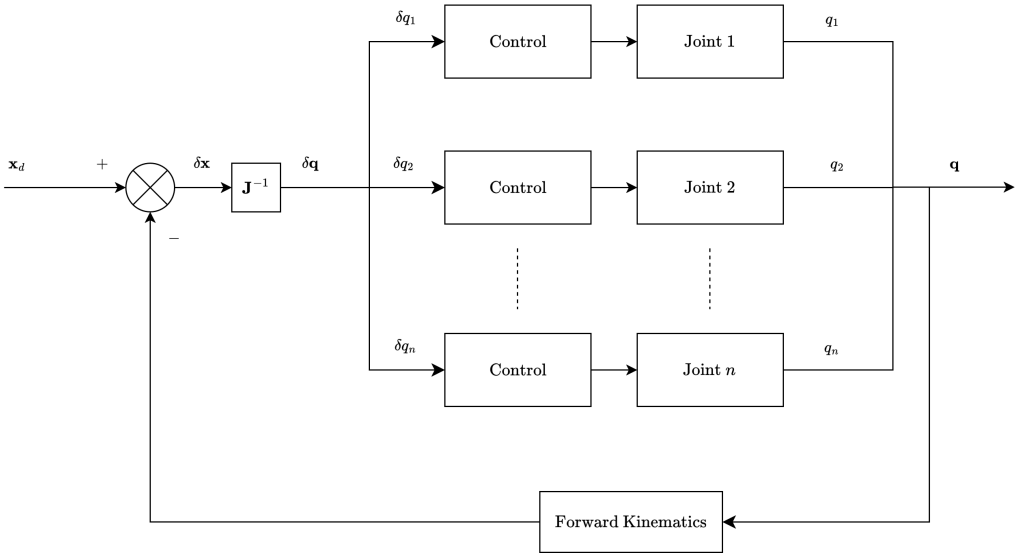
$$\varphi = \begin{bmatrix} -3\sin(\theta_2)\dot{\theta}_1\dot{\theta}_2 - 1.5\sin(\theta_2)\dot{\theta}_2^2 + 3\cos(\theta_2)\ddot{\theta}_1 + 1.5\cos(\theta_2)\ddot{\theta}_2 + 4.25\ddot{\theta}_1 + 1.5\ddot{\theta}_2 \\ 1.5\sin(\theta_2)\dot{\theta}_1^2 + 1.5\cos(\theta_2(t))\ddot{\theta}_1 + 1.5\ddot{\theta}_1 + 1.5\ddot{\theta}_2 \\ 2\ddot{d}_3 - 19.6 \end{bmatrix}$$

II.5. Control

As mentioned in the introduction, we will use resolved rate control with PID control. The PID parameters are as the following

$$K_p = 10.0 \quad K_i = 0.5 \quad K_d = 0.05$$

Figure 2 — Resolved Rate Control Scheme [1]



II.6. Implementation

The simulation was implemented in Python using libraries like `numpy` for computation and `matplotlib` for plots [6], [7]. `vpython` was used for the 3D visualization. The code is bundled in Python notebooks. The project contains 2 Python notebooks `Derivation.ipynb` and `Simulation.ipynb` .

- `Derivation.ipynb` contains the code used to derive the transformation matrix, Jacobian, dynamics, and the position of every joint along the robot for the 3D visualization.

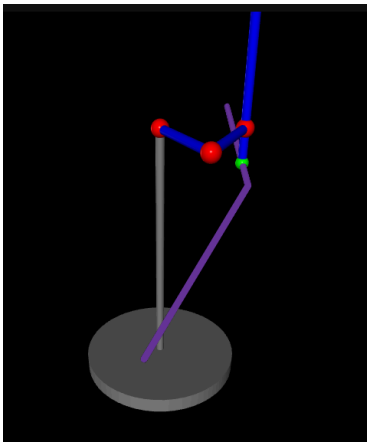
- `Simulation.ipynb` contains the code for the simulation and visualization and it's functions are a direct implementation of the ones derived in `Derivation.ipynb` .

To select the path in the code set the `path` variables to either `path1` , `path2` , or `path3` .

Visualization

Upon selecting the path and running all the cells below the cell that selects the path, you can see that the last cell contains a `vpython` window showing a robot visualization which can be rotated around by dragging it using the Right Mouse Button. Grey cylinders are fixed parts/ support of the robot, red spheres are joints, blue cylinders are links and the green sphere is the end-effector. The purple path is the desired path for the robot.

Figure 3 — SCARA Robot Visualization in `vpython`



III. Results and Analysis

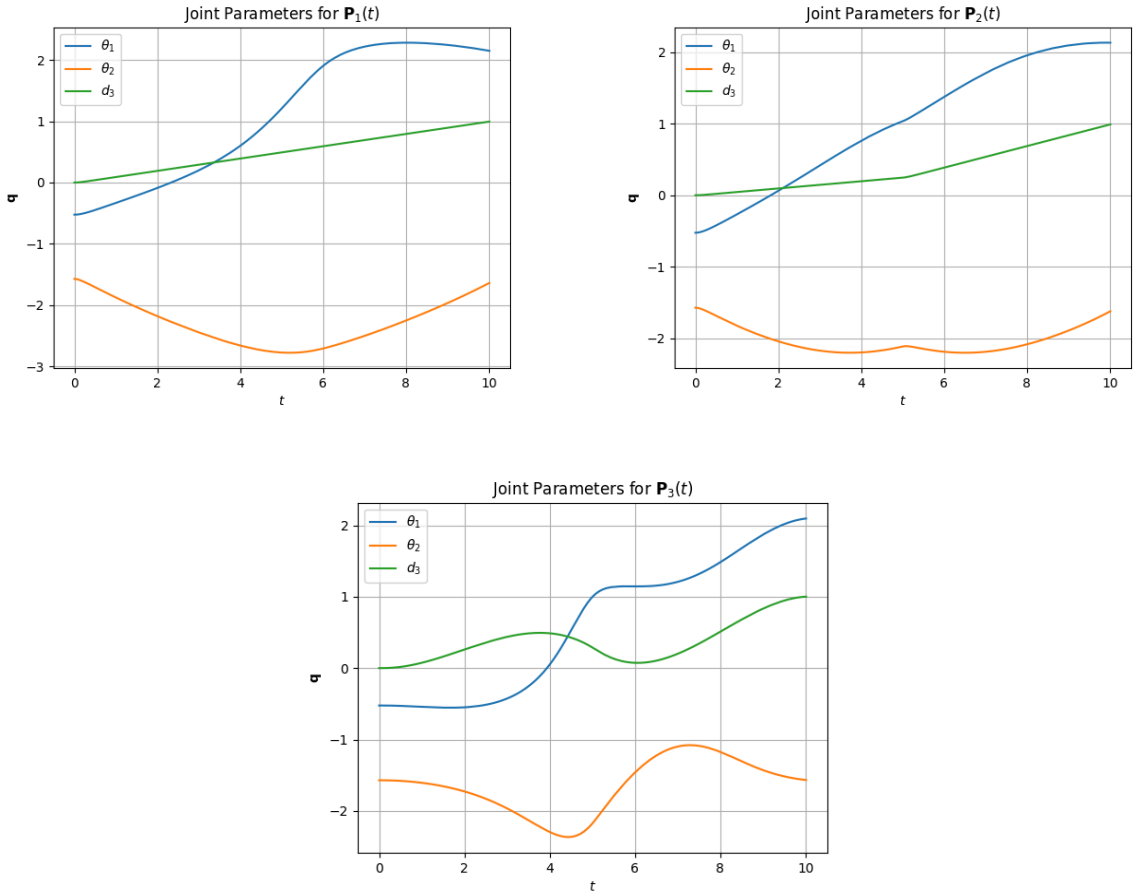
Running the simulation for all 3 paths we obtain the following measurements for energy

Table 2 — Resultant Energy for Each Path

Path Function	Energy Consumed
$\boldsymbol{P}_1(t)$	19.1803 J
$\boldsymbol{P}_2(t)$	18.8058 J
$\boldsymbol{P}_3(t)$	37.2540 J

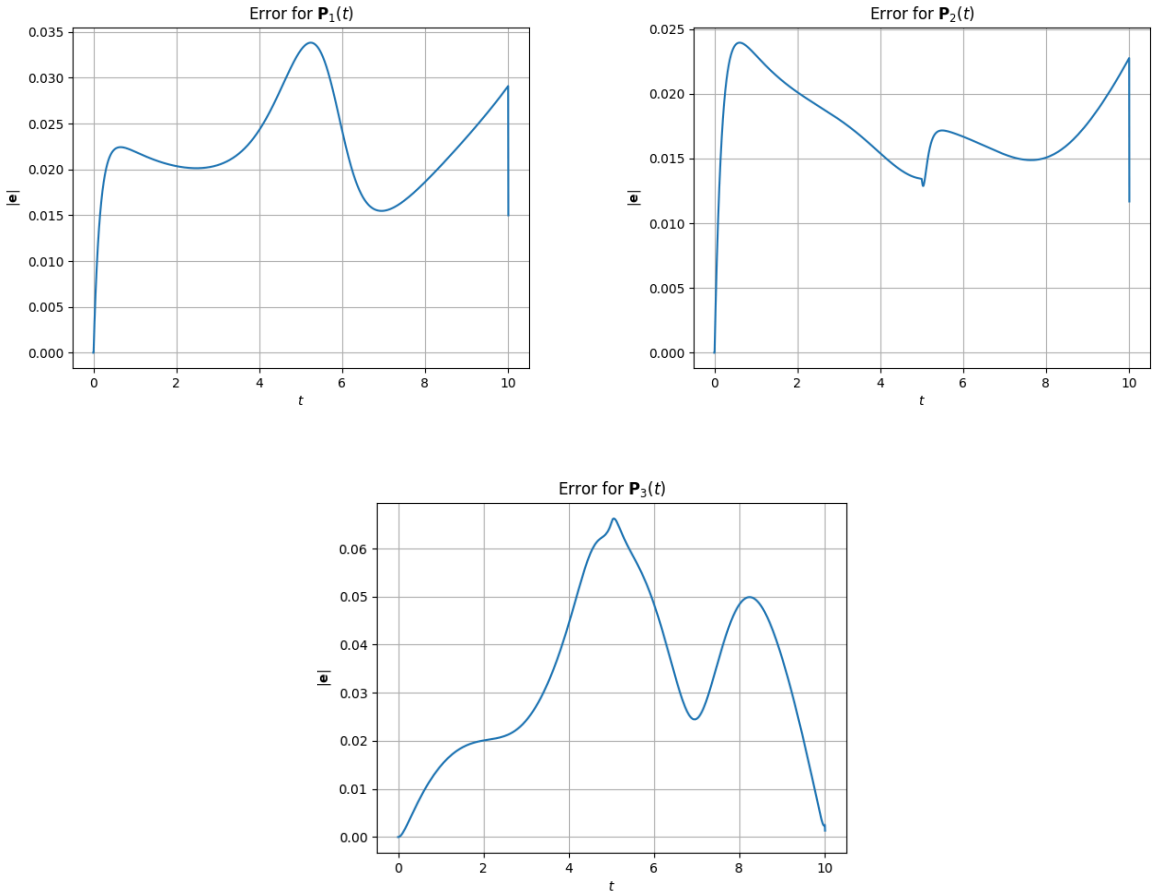
We notice that path 2 consumes the least energy despite not being the shortest path.

Figure 4 — Tracking Error for Each Path



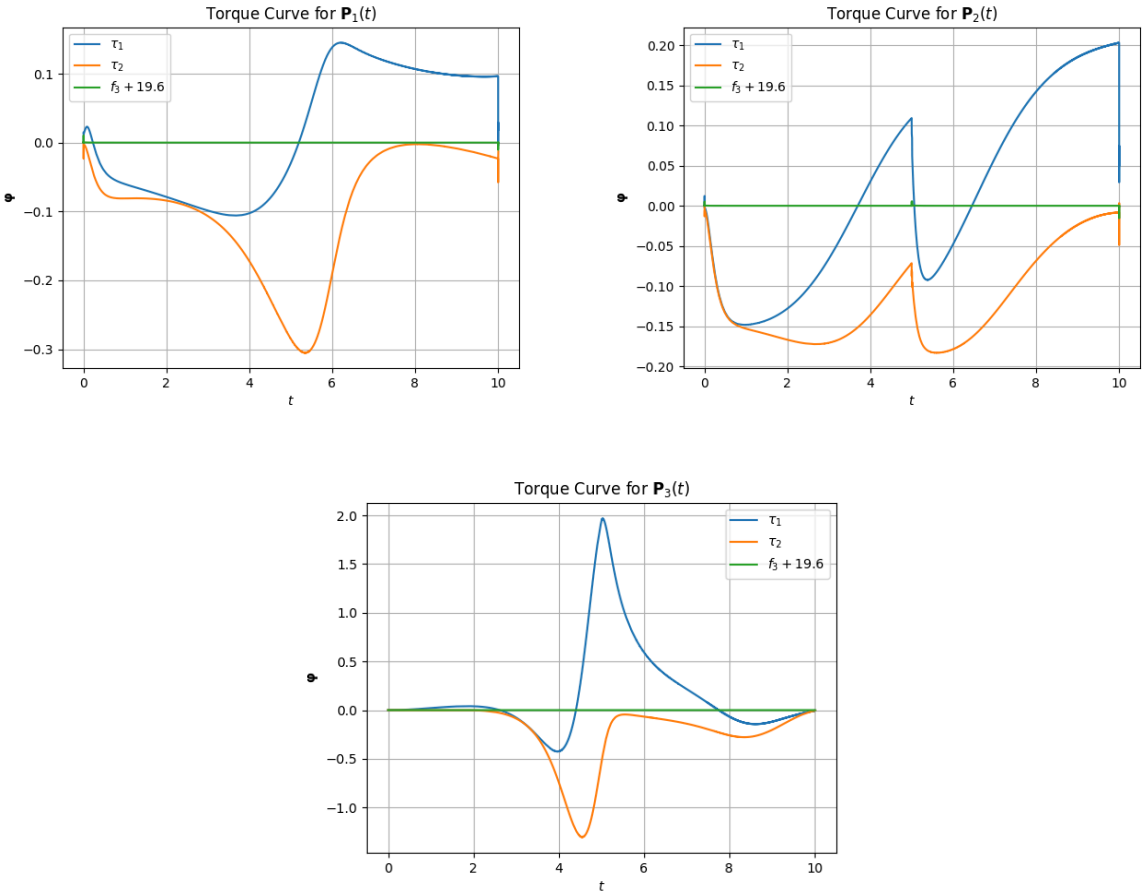
We can see from each plot what kind of discontinuity each path has. Path 1 is a straight line so it doesn't have any discontinuity and is C^{∞} continuos, path 2, on the other hand, clearly has a sharp edge at the halfway point implying it is only C^0 continuos. For path 3, we notice that there is an sudden change in velocity around the halfway point showing that it is C^1 continuos.

Figure 5 — Tracking Error for Each Path



Looking at the tracking error for each path, we see that both paths 1 and 2 have relatively comparable errors while path 3 almost has double the error of path 1 and it's maximum is reached at the halfway point. We also notice that the error for path 1 "spikes" suddenly, observing the robot at that point reveals that the arm is close to the robot body. This sudden spike might be attributed to the fact that the joint has to rotate at a wider angle to keep up with our desired path.

Figure 6 — Force Quantities for Each Path



We note that we plot the curve of $f_3 + 19.6$ instead of f_3 as it remains relatively constant at that quantity as it constantly need to balance out the force exerted by the weight of the end-effector and the slight increases in height requested from it, which is why it sees very little fluctuations.

Observing the torque curve τ_2 for path 1 confirms the issue of being closer to the robot as the forces on that joint have to be higher to compensate for the wide change in angle. Perhaps this is why path 1 is not the optimal path despite being the shortest. We also notice a spike in torque in path 3 showing the sudden change of acceleration requested from the robot.

IV. Conclusion

In this project, we were able to analyze the performance of 3 different paths under the same control scheme. We concluded that path 2 (a linear path that remains away from the robot body) showed the best performance metrics and is most suitable under the same control scheme (Resolved Rate Control + PID), however our experiments are not encompassing enough to make a generalization for other robots, or even for different control schemes under the same configurations.

This project helps law a solid foundation to begin the analysis of energy consumption across multiple configurations and control laws, though the methodology for a more general solution would have to be different and consider a wider range of paths and control parameters.

For example, returning to Figure 6 we could investigate higher degrees of smoothness to avoid sudden torque spikes. Perhaps a C^2 path might yield a better result?

Bibliography

[1] D. E. Whitney, "Resolved motion rate control of manipulators and human prostheses," *IEEE Transactions on man-machine systems*, vol. 10, no. 2, pp. 47–53, 2007.

[2] C. Feng, G. Gao, and Y. Cao, "Kinematic modeling and verification for a SCARA robot," in *2016 3rd International Conference on Materials Engineering, Manufacturing Technology and Control*, 2016, pp. 918–921.

[3] J. Denavit and R. S. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," 1955.

[4] R. Ortega, A. Loria, P. J. Nicklasson, and H. Sira-Ramirez, "Euler-Lagrange systems," *Passivity-based Control of Euler-Lagrange Systems: Mechanical, Electrical and Electro-mechanical Applications*. Springer, pp. 15–37, 1998.

[5] A. Meurer *et al.*, "SymPy: symbolic computing in Python," *PeerJ Computer Science*, vol. 3, p. e103, Jan. 2017, doi: 10.7717/peerj-cs.103.

[6] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007, doi: 10.1109/MCSE.2007.55.

[7] C. R. Harris *et al.*, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sept. 2020, doi: 10.1038/s41586-020-2649-2.