TÉCNICO LISBOA

Robotic Manipulation

Masters Degree in Mechanical Engineering

Project - Part 2 [EN]

Authors:

Francisco Pinto (ISTID 089888) francisco.v.pinto@tecnico.ulisboa.pt
Henrique Branquinho (ISTID 087203) henrique.branquinho@tecnico.ulisboa.pt

Group 11

2024/2025 – 4th Quarter

Contents

1 Introduction

This second report focuses on the dynamics and control of the Mako surgical robot. To support this analysis, preliminary calculations were conducted to estimate the mass and center of mass of each robotic arm.

Subsequently, a Simulink dynamics model was developed using the Newton-Euler method, allowing us to test the worst-case inertia scenarios for each joint.

Following this, a decentralized PD joint controller was designed and implemented. Finally, a centralized inverse controller was also developed.

The performance of both control techniques were compared through a constant input and through a time depending trajectory.

2 Manipulator Parameters

To study the dynamics of the Mako surgical robot, we started by defining mass, center of mass, and inertia.

2.1 Simple Solid Shapes Diagram

Each link of the manipulator is modeled as a cylinder with dimensions summarized in Table 1. These dimensions were selected to reflect the expected total dimensions of the robot (1m length).

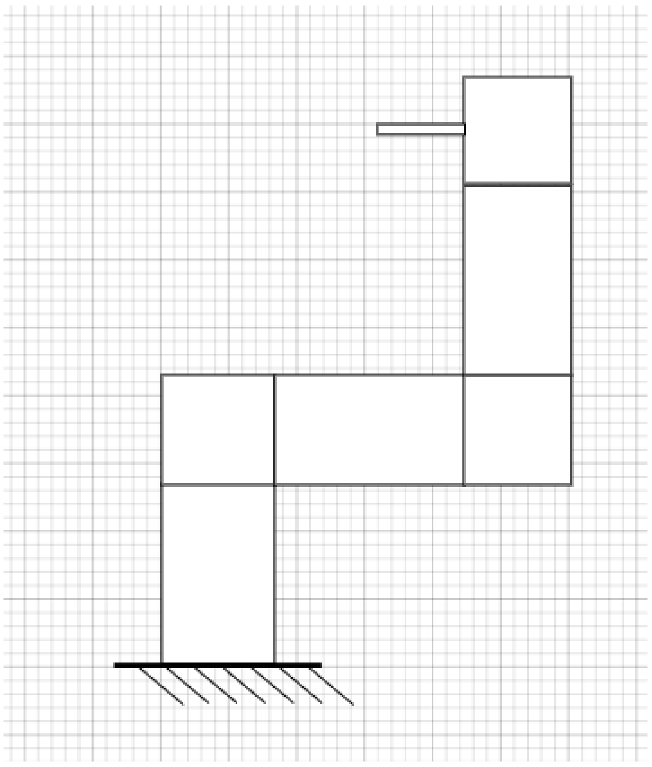


Figure 1: Schematic diagram of the manipulator modeled as simple cylindrical links. *Note: dimensions in this figure are illustrative and not to scale.*

Link	Radius (m)	Height (m)
1	0.25	0.5
2	0.2	0.2
3	0.125	0.4
4	0.1	0.2
5	0.09	0.5
6	0.08	0.15
e	0	0

Table 1: Dimensions of cylindrical components

The total mass M_{total} is fixed at 30 kg, and the volumes of each cylindrical component are computed as:

$$V_i = \pi r_i^2 h_i$$

These volumes are then used to distribute the total mass proportionally across the links. The last link (tool) was ignored in the mass distribution due to its negligible size, which leads to an extremely small mass compared to the other links.

2.2 Mass

Each link of the manipulator is modeled as having a concentrated mass located at its geometric center. These values were chosen to reflect the relative size and material distribution of the segments in the Mako surgical manipulator.

m_1	m_2	m_3	m_4	m_5	m_6	m_e	**Total**
17.8537	4.5706	3.5707	1.1426	2.3138	0.5485	0	**30**

Table 2: Updated mass values of the components

2.3 Center of Mass

For each link, the center of mass location is given in the local coordinate frame. These vectors are used to correctly account for gravitational loading and to compute dynamic forces.

$\mathbf{Com}_1$	$\mathbf{Com}_2$	$\mathbf{Com}_3$	$\mathbf{Com}_4$	$\mathbf{Com}_5$	$\mathbf{Com}_6$	$\mathbf{Com}_e$
$\begin{bmatrix}0\\0\\0\end{bmatrix}$	$\begin{bmatrix}0\\0\\0\end{bmatrix}$	$\begin{bmatrix}0\\0.25\\0\end{bmatrix}$	$\begin{bmatrix}0\\0\\0\end{bmatrix}$	$\begin{bmatrix}0\\0.5\\0\end{bmatrix}$	$\begin{bmatrix}0\\0\\0.2\end{bmatrix}$	$\begin{bmatrix}0\\0\\0\end{bmatrix}$

Table 3: Centers of mass of the components

2.4 Inertia Tensors

The inertia tensor of each link is computed assuming solid cylinders. These tensors are essential for deriving the equations of motion in the manipulator's dynamic model.

The formulas for the inertia tensor of a solid cylinder of mass m, radius r, and height h aligned along the z-axis are:

$$\begin{aligned} I_{xx} &= I_{yy} = \frac{1}{12}m(3r^2 + h^2), \\ I_{zz} &= \frac{1}{2}mr^2. \end{aligned}$$

Using these expressions, the inertia tensors for each link are:

$$I_1 = \begin{bmatrix} \frac{1}{12}m_1\left(3 \cdot 0.25^2 + 0.5^2\right) & 0 & 0 \\ 0 & \frac{1}{12}m_1\left(3 \cdot 0.25^2 + 0.5^2\right) & 0 \\ 0 & 0 & \frac{1}{2}m_1 \cdot 0.25^2 \end{bmatrix},$$

$$I_2 = \begin{bmatrix} \frac{1}{12}m_2\left(3 \cdot 0.2^2 + 0.2^2\right) & 0 & 0 \\ 0 & \frac{1}{12}m_2\left(3 \cdot 0.2^2 + 0.2^2\right) & 0 \\ 0 & 0 & \frac{1}{2}m_2 \cdot 0.2^2 \end{bmatrix},$$

$$I_3 = \begin{bmatrix} \frac{1}{12}m_3\left(3 \cdot 0.125^2 + 0.4^2\right) & 0 & 0 \\ 0 & \frac{1}{12}m_3\left(3 \cdot 0.125^2 + 0.4^2\right) & 0 \\ 0 & 0 & \frac{1}{2}m_3 \cdot 0.125^2 \end{bmatrix},$$

$$I_4 = \begin{bmatrix} \frac{1}{12}m_4\left(3 \cdot 0.1^2 + 0.2^2\right) & 0 & 0 \\ 0 & \frac{1}{12}m_4\left(3 \cdot 0.1^2 + 0.2^2\right) & 0 \\ 0 & 0 & \frac{1}{2}m_4 \cdot 0.1^2 \end{bmatrix},$$

$$I_5 = \begin{bmatrix} \frac{1}{12}m_5\left(3 \cdot 0.09^2 + 0.5^2\right) & 0 & 0 \\ 0 & \frac{1}{12}m_5\left(3 \cdot 0.09^2 + 0.5^2\right) & 0 \\ 0 & 0 & \frac{1}{2}m_5 \cdot 0.09^2 \end{bmatrix},$$

$$I_6 = \begin{bmatrix} \frac{1}{12}m_6\left(3 \cdot 0.08^2 + 0.15^2\right) & 0 & 0 \\ 0 & \frac{1}{12}m_6\left(3 \cdot 0.08^2 + 0.15^2\right) & 0 \\ 0 & 0 & \frac{1}{2}m_6 \cdot 0.08^2 \end{bmatrix}.$$

3 Dynamics of the robot - Newton Euler Method

3.1 Newton–Euler Method

Now that we have determined all the necessary parameters, we can proceed to derive the equations of motion using The Newton–Euler method. This method calculates inverse dynamics through two recursive sweeps over the manipulator:

A forward sweep to obtain link velocities and accelerations.

A backward sweep to compute forces, moments, and joint torques.

Forward recursion (kinematics):

$$\begin{aligned}
\boldsymbol{\omega}_i^i &= R_i^{i-1}\,\boldsymbol{\omega}_{i-1}^{i-1} + \dot{q}_i\,\mathbf{z}_0, &(1)\\
\dot{\boldsymbol{\omega}}_i^i &= R_i^{i-1}\,\dot{\boldsymbol{\omega}}_{i-1}^{i-1} + \ddot{q}_i\,\mathbf{z}_0 + \boldsymbol{\omega}_i^i \times (\dot{q}_i\,\mathbf{z}_0), &(2)\\
\mathbf{a}_i^i &= R_i^{i-1}\,\mathbf{a}_{i-1}^{i-1} + \dot{\boldsymbol{\omega}}_i^i \times \mathbf{r}_i^{i-1} + \boldsymbol{\omega}_i^i \times (\boldsymbol{\omega}_i^i \times \mathbf{r}_i^{i-1}), &(3)\\
\mathbf{a}_{C_i}^i &= \mathbf{a}_i^i + \dot{\boldsymbol{\omega}}_i^i \times \mathbf{r}_{C_i}^i + \boldsymbol{\omega}_i^i \times (\boldsymbol{\omega}_i^i \times \mathbf{r}_{C_i}^i) &(4)
\end{aligned}$$

Backward recursion (dynamics):

$$\begin{aligned}
\mathbf{f}_i^i &= R_{i+1}^i\,\mathbf{f}_{i+1}^{i+1} + m_i\,\mathbf{a}_{C_i}^i, &(5)\\
\boldsymbol{\mu}_i^i &= R_{i+1}^i\,\boldsymbol{\mu}_{i+1}^{i+1} + \mathbf{r}_i^{i-1} \times (R_{i+1}^i\,\mathbf{f}_{i+1}^{i+1}) - \mathbf{r}_{C_i}^i \times (m_i\,\mathbf{a}_{C_i}^i) + I_i\,\dot{\boldsymbol{\omega}}_i^i + \boldsymbol{\omega}_i^i \times (I_i\,\boldsymbol{\omega}_i^i), &(6)\\
\tau_i &= \begin{cases} \mathbf{f}_i^{iT}\,R_i^{i-1}\,\mathbf{z}_0, & \text{prismatic joint,}\\ \boldsymbol{\mu}_i^{iT}\,R_i^{i-1}\,\mathbf{z}_0, & \text{revolute joint.} \end{cases} &(7)
\end{aligned}$$

Equation of motion: After assembling all τ_i, the overall joint-space dynamics can be written as

$$\tau = B(q)\,\ddot{q} \;+\; C(q,\dot{q})\,\dot{q} \;+\; g(q),$$

where B is the inertia matrix, C contains Coriolis/centrifugal terms, and g is the gravity vector.

3.2 Implementation in MATLAB

The Newton–Euler algorithm was implemented in MATLAB. The robot's dynamics were modeled with the following considerations:

- The manipulator consists of 5 revolute joints modeled using the Denavit–Hartenberg convention.
- Link properties (mass, center of mass, and inertia tensor) were obtained from a solid cylinder approximation.
- The recursive Newton–Euler method was applied to calculate joint torques using symbolic joint velocities and accelerations.

3.3 Dynamic Parameters and Modeling Assumptions

For simulation purposes, the following modeling assumptions were made:

- The manipulator base is inertial, with gravity introduced as $\mathbf{g}_0 = [0, 0, g]^T$. Our origin referential has the z axis pointing downwards.
- The sixth joint is not considered. The dynamics model only considers the first five joints.
- Friction, actuator dynamics, and external disturbances were not included in the model.
- All links were modeled as uniform solid cylinders, which simplifies inertia computation but may introduce approximation errors compared to the real link structures.

3.4 Torque Computation and Model Extraction

Symbolic expressions were used to compute the torque vector $\boldsymbol{\tau}$ for the first five joints:

$$\boldsymbol{\tau} = B(\mathbf{q})\,\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}})\,\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q})$$

where:

- $B(\mathbf{q})$ is the inertia matrix, computed as the Jacobian of $\boldsymbol{\tau}$ with respect to joint accelerations.
- $\mathbf{g}(\mathbf{q})$ is extracted by zeroing out all joint velocities and accelerations.
- $C(\mathbf{q}, \dot{\mathbf{q}})\,\dot{\mathbf{q}}$ is obtained by subtracting the other components from $\boldsymbol{\tau}$.

This formulation enables full decoupling of the dynamic effects. This generated symbolic model is particularly advantageous for control design, since each component is independently accessible and differentiable.

3.5 Simulink Implementation of the Dynamic Model

To validate the dynamic model and perform simulation-based analysis, the symbolic expressions for $B(\mathbf{q})$, $\mathbf{g}(\mathbf{q})$, and $\phi(\mathbf{q}, \dot{\mathbf{q}})$ were compiled into a Simulink block named Mako_Dynamics. This block takes as inputs the joint positions $\mathbf{q}$, gravity $\mathbf{g}$ and velocities $\dot{\mathbf{q}}$.

The Simulink implementation integrates these dynamics in a loop to compute joint accelerations, which are then integrated to obtain velocities and positions.

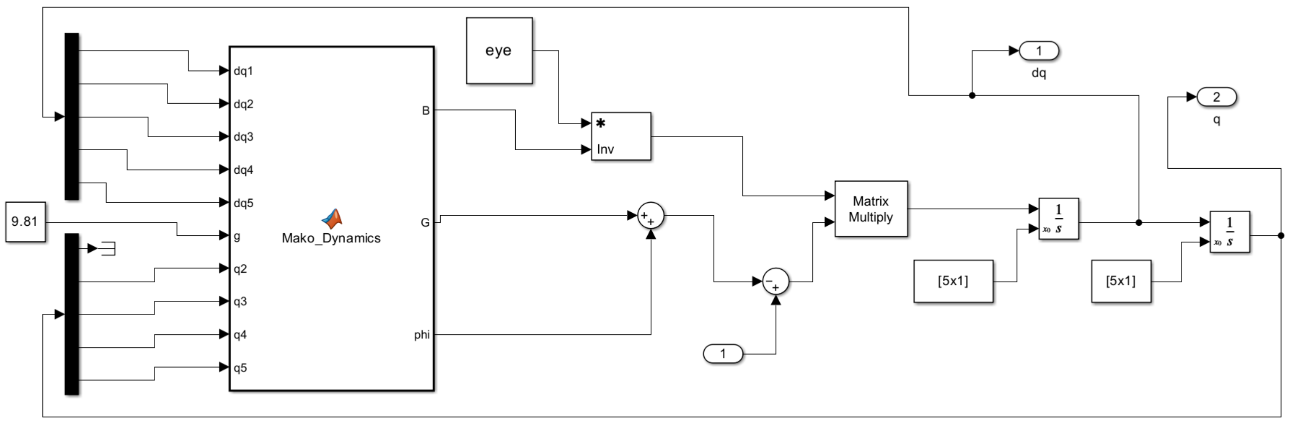


Figure 2: Simulink implementation of the dynamic model using the Mako_Dynamics block

This implementation enables simulation of the manipulator's forward dynamics under applied torques and provides a solid foundation for the testing of model-based control strategies.

3.6 Zero-Gravity Single-Joint Tests

To validate the dynamics and numerical integration under idealized conditions a Simulink model were constructed:

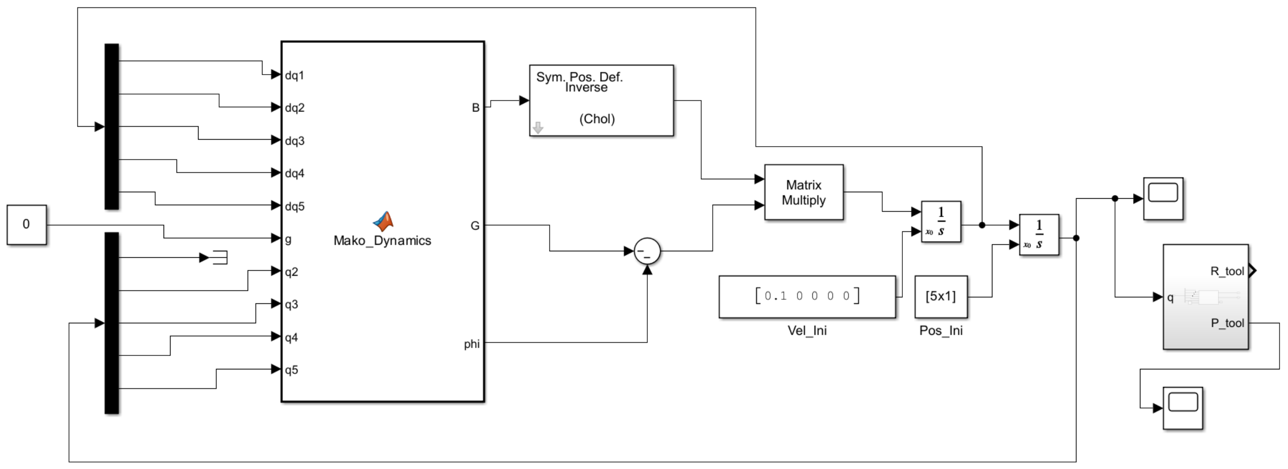


Figure 3: Simulink setup for zero-gravity, single-joint initial velocity test ($\mathbf{g}_0 = \mathbf{0}$).

In this configuration, the manipulator's base gravity vector is set to zero, and only an initial joint velocity is applied to Joint 1.

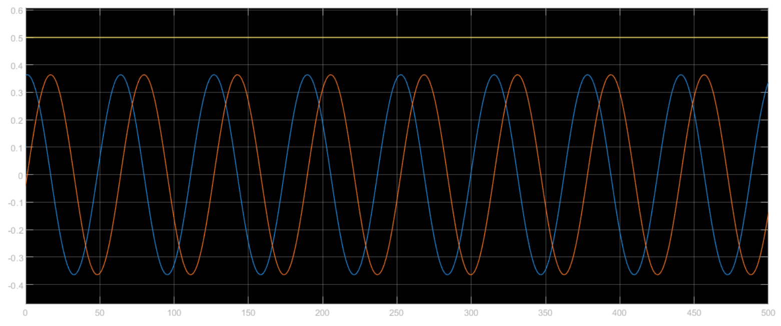


Figure 4: End-effector trajectory in the $x-y$ plane for joint-1 excitation.

Figure 4 plots the end-effector position $(x(t), y(t))$. The path is a circle, confirming that a single-joint velocity outputs a circular motion when the arm is massless.

All the joints were tested, and all tests have the expected result, the end-effector follows a circular trajectory. This confirms that the model is implemented correctly and that the numerical integration remains stable in the absence of external forces.

4 Decentralized PD Control Design and Implementation

4.1 Worst-Case Inertia Calculation

To design decentralized joint controllers, the worst-case value of each diagonal element $B_{ii}(q)$ of the inertia matrix was computed. This accounts for the maximum dynamic load each joint might experience.

Symbolic expressions for the diagonal elements of $B(q)$ were first extracted and converted into MATLAB functions. The joint space was bounded within standard limits:

$$q_i \in [-\pi, \pi] \quad \text{for } i = 1, \ldots, 5$$

Using a global optimization routine (`MultiStart` with `fmincon`), each $B_{ii}(q)$ was maximized independently. The resulting values define the worst-case inertia loads:

$$B_{ii}^{\max} = \max_{q \in \mathcal{Q}} B_{ii}(q) \tag{8}$$

This method ensures robustness by accounting for the highest expected inertia along each joint axis. The joint configurations that led to these maxima were also stored for analysis.

4.2 Gain Design

For each joint i, the controller was modeled as a second-order system. Given a target natural frequency $\omega_n = 5$ rad/s and damping ratio $\xi = 0.8$, the proportional and derivative gains were computed as:

$$K_{p,i} = B_{ii}^{\max} \cdot \omega_n^2 \tag{9}$$

$$K_{d,i} = 2 \cdot B_{ii}^{\max} \cdot \xi \cdot \omega_n \tag{10}$$

These values ensure the closed-loop response is stable, well-damped, and responsive even under worst-case dynamic conditions.

4.3 Simulink Implementation

The decentralized PD controllers were implemented for all 5 joints independently in Simulink. Each controller receives position and velocity error inputs and applies feedback control using the corresponding K_p and K_d gains. The structure assumes independent joint control, with no dynamic coupling between axes.

An Simulink model is shown below, which includes gravity compensation through a dedicated symbolic block `g_block`, alongside standard PD feedback:

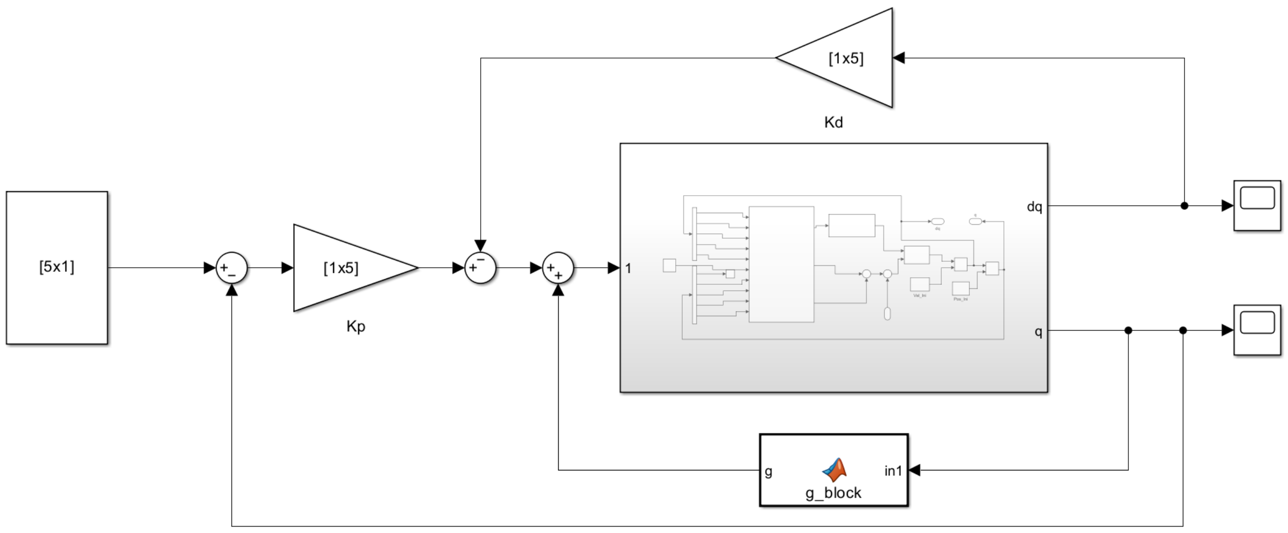


Figure 5: Complete Simulink layout of the decentralized control system with gravity compensation

We organized the controller in separate blocks: One part uses a feedforward term to cancel out gravity, and another uses a PD loop to fix position and velocity errors. This setup means we can drop our symbolic dynamics straight into Simulink and easily change or modify the controller later.

4.4 Simulation Results

To evaluate the controller effectiveness, the joint position and velocity profiles were monitored over time.

The joint configuration used to evaluate the gravity compensation block was:

$$\mathbf{q} = \begin{bmatrix} 0.35 \\ 0.5 \\ 0.8 \\ 0.75 \\ 0.5 \end{bmatrix}$$

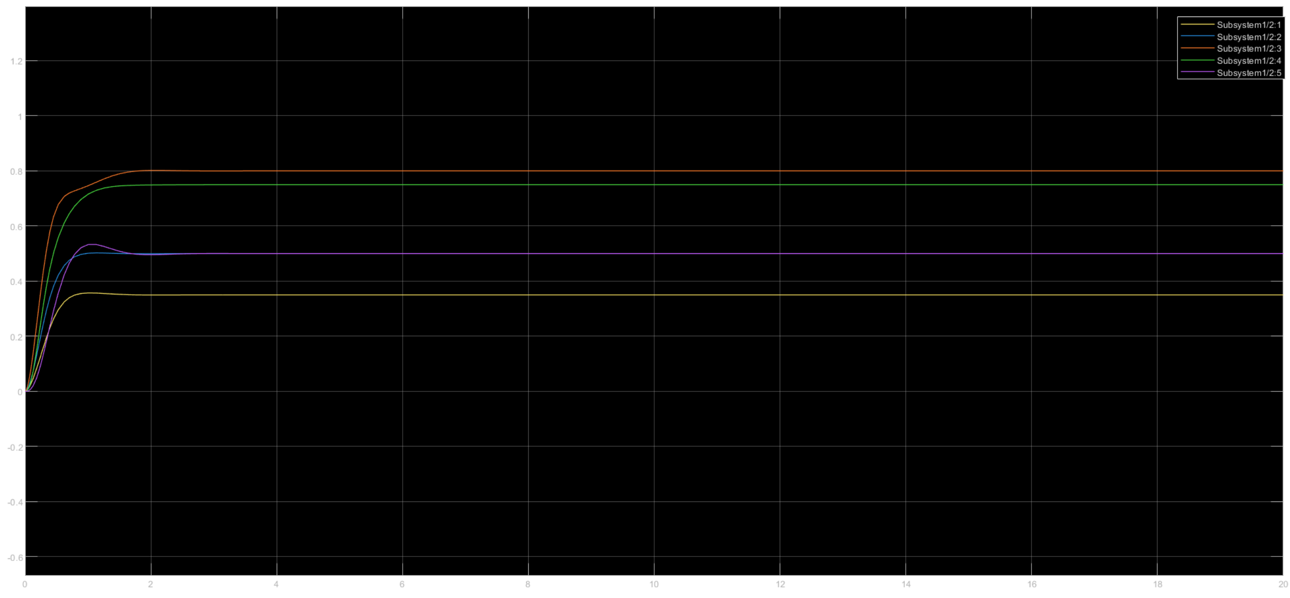


Figure 6: Simulated joint positions $\mathbf{q}(t)$ from the robotic system

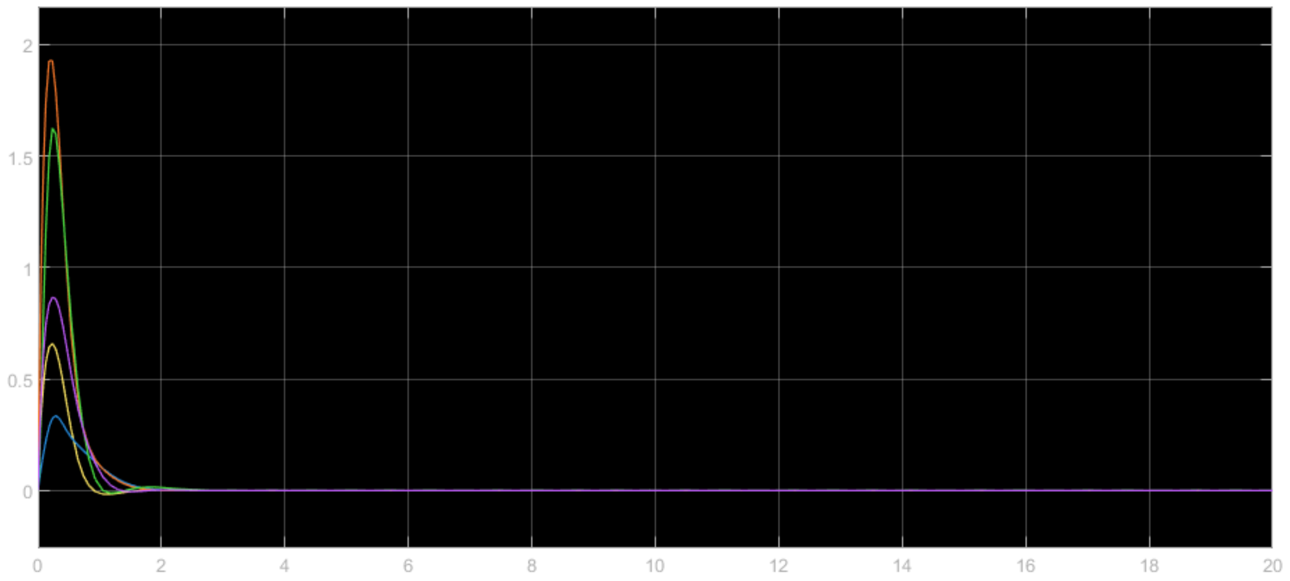


Figure 7: Simulated joint velocities $\dot{\mathbf{q}}(t)$ from the robotic system

Overall, the controllers achieved precise tracking with settling times under 2s for all joints. These results confirm the effectiveness of the worst-case inertia–based gain design and validate the decentralized PD approach under realistic dynamic conditions.

5 Centralized Inverse Dynamics Control

5.1 Control Architecture

A centralized inverse dynamics controller was designed to improve trajectory tracking by compensating for the full nonlinear dynamics of the robot. Unlike decentralized PD controllers, this approach accounts for inter-joint coupling, centrifugal forces, and gravity.

The control law is defined as:

$$\boldsymbol{\tau} = B(\mathbf{q})\,\ddot{\mathbf{q}}_d + C(\mathbf{q},\dot{\mathbf{q}})\,(\dot{\mathbf{q}}_d - \dot{\mathbf{q}}) + K_p(\mathbf{q}_d - \mathbf{q}) + K_d(\dot{\mathbf{q}}_d - \dot{\mathbf{q}}) + g(\mathbf{q})$$

This structure linearizes the closed-loop dynamics and reduces tracking errors.

5.2 Simulink Implementation

The inverse dynamics control law was implemented in Simulink, where:

- The desired accelerations $\ddot{\mathbf{q}}_d$, velocities $\dot{\mathbf{q}}_d$, and positions $\mathbf{q}_d$ are input.
- Errors in position and velocity are processed via gains K_p and K_d.
- The resulting control input $\mathbf{u}$ is mapped through the inertia matrix $B(\mathbf{q})$ and combined with the nonlinear terms $\boldsymbol{\phi} = C(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + g(\mathbf{q})$.

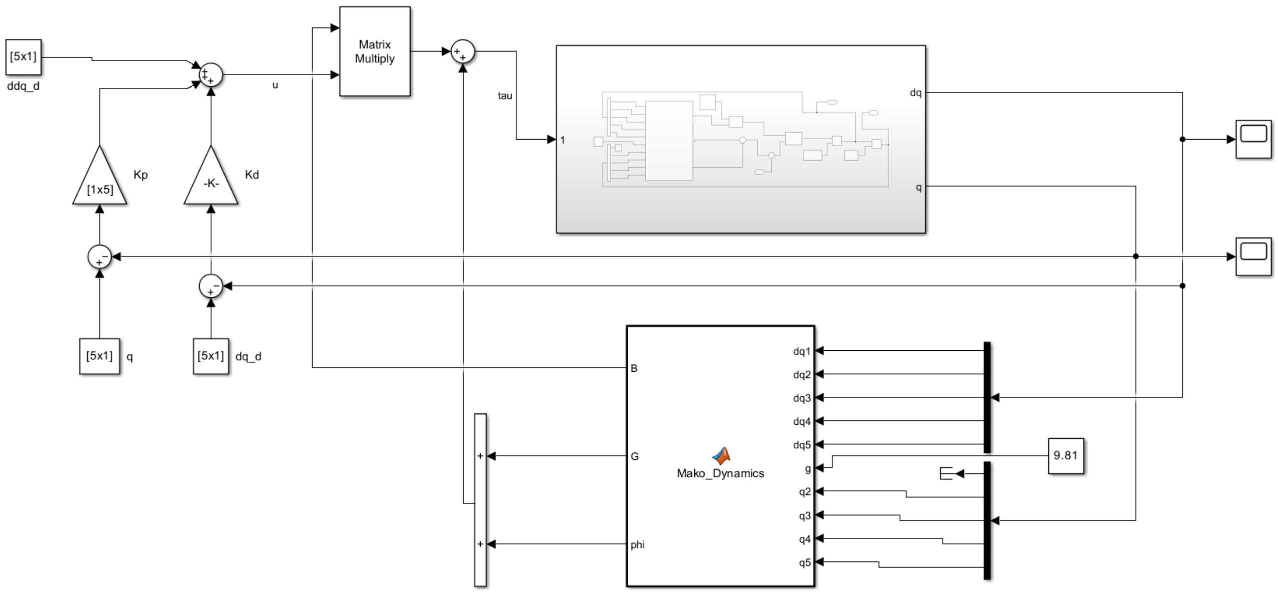


Figure 8: Simulink implementation of centralized inverse dynamics control

5.3 Joint Torques

While the outputs of the robot dynamics block in Simulink are limited to joint positions $\mathbf{q}(t)$ and velocities $\dot{\mathbf{q}}(t)$, the joint torques $\boldsymbol{\tau}(t)$ play a central role in the simulation. These torques are computed by the controller and serve as the inputs to the robotic system.

The torque vector $\boldsymbol{\tau}(t)$ is not an output of the robot itself, but rather the actuation command required to produce the desired motion. It dynamically compensates the inertia, centrifugal forces, gravity, and control errors. Monitoring these torques is essential to evaluate:

- The efficiency and aggressiveness of the controller,
- Whether the torque demands remain within actuator limits,
- The comparative load across joints for different control strategies.

5.4 Simulation Results

The joint configuration used to evaluate was the same as in the Decentralized PD:

$$\mathbf{q} = \begin{bmatrix} 0.35 \\ 0.5 \\ 0.8 \\ 0.75 \\ 0.5 \end{bmatrix}$$

The joint positions and velocities obtained were:

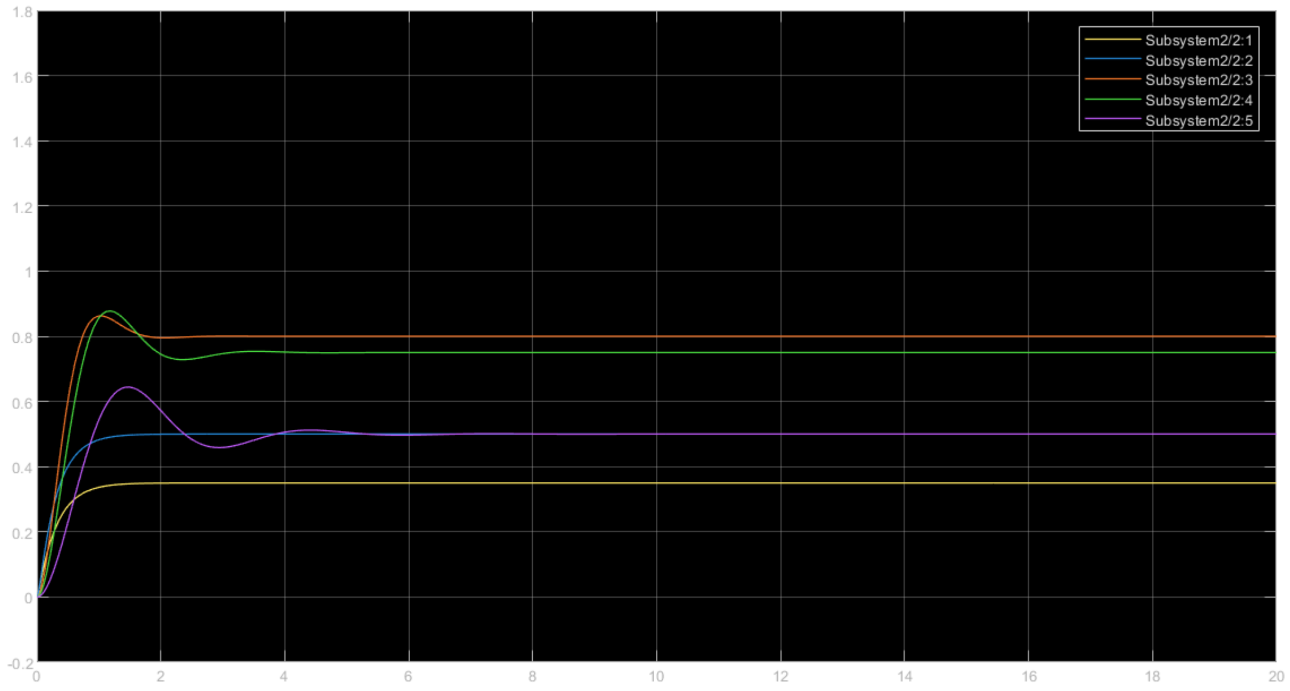


Figure 9: Simulated joint positions $\mathbf{q}(t)$ from the robotic system

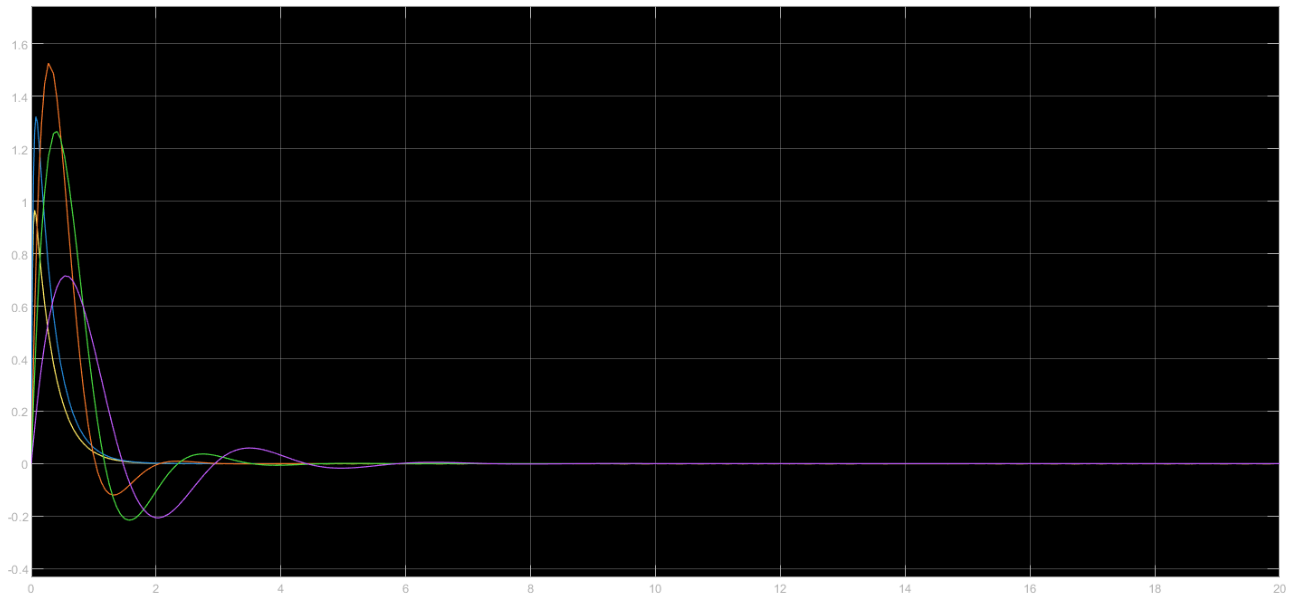


Figure 10: Simulated joint velocities $\dot{\mathbf{q}}(t)$ from the robotic system

Overall, the centralized inverse dynamics controller demonstrated settling times within 4s (Joint 5 within 6s). The centralized controller takes a bit longer to settle compared to the decentralized PD approach. This simpler setup usually means quicker settling, even if it's not as optimal overall.

6 Task-Space Trajectory Tracking and Controller Comparison

6.1 Task and Simulation Architecture and Parabolic Trajectory

The objective was to evaluate and compare the performance of the decentralized PD and centralized inverse dynamics controller in executing a task-space trajectory. To this end, we set up a Simulink model with both control architectures and drove them with a simple, symmetric parabolic trajectory on the interval $[0, T]$.

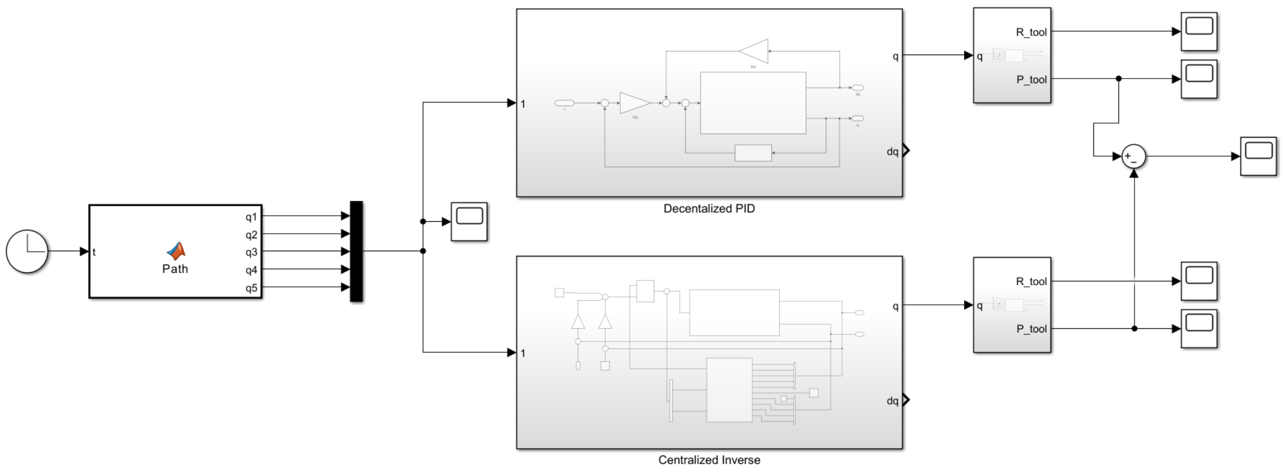


Figure 11: Simulink comparison setup for decentralized PD and centralized inverse dynamics controllers

6.1.1 Parabolic Trajectory Generation

We generate a single-parameter motion profile using a simple parabola:

$$s(t) \;=\; 4\,\frac{t}{T}\Bigl(1-\frac{t}{T}\Bigr),$$

so that

$$s(0)=0,\quad s\bigl(\tfrac{T}{2}\bigr)=1,\quad s(T)=0.$$

The desired joint trajectory is then

$$\mathbf{q}_d(t)=\mathbf{q}_{\text{start}}+s(t)\,\bigl(\mathbf{q}_{\text{peak}}-\mathbf{q}_{\text{start}}\bigr).$$

Because $s(t)$ is a simple quadratic, the profile is:

- **Symmetric:** identical rise and fall phases, so acceleration and deceleration loads match.
- **Single-peak:** reaches maximum exactly at $t=T/2$, giving predictable timing.

This trajectory is easy to implement in real time.

6.2 Error Between Decentralized PD and Centralized Inverse

To quantify the performance deviation of the decentralized PD controller from the ideal centralized inverse kinematics, we define the point-to-point error vector

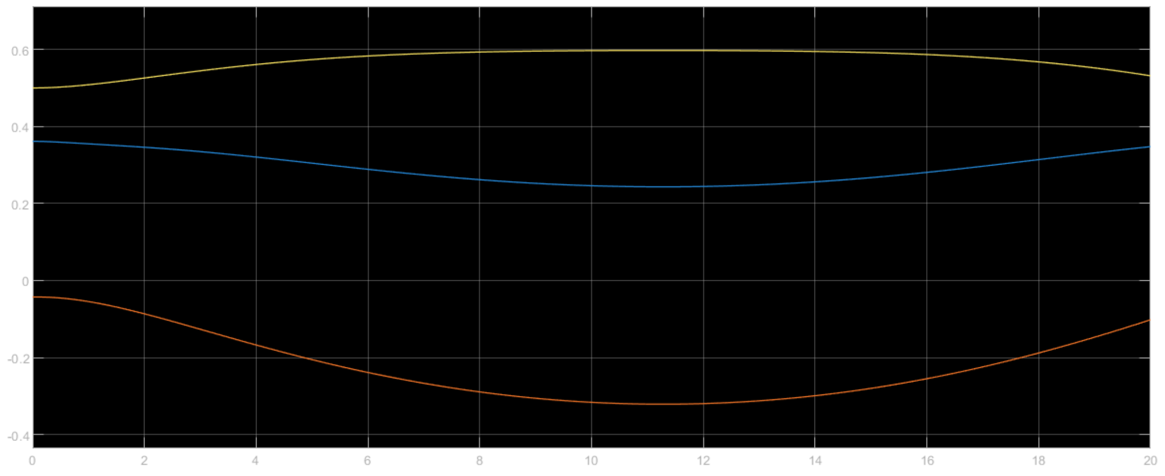
$$\mathbf{e}(t) \;=\; \mathbf{p}_{\text{PD}}(t) \;-\; \mathbf{p}_{\text{inv}}(t),$$

where

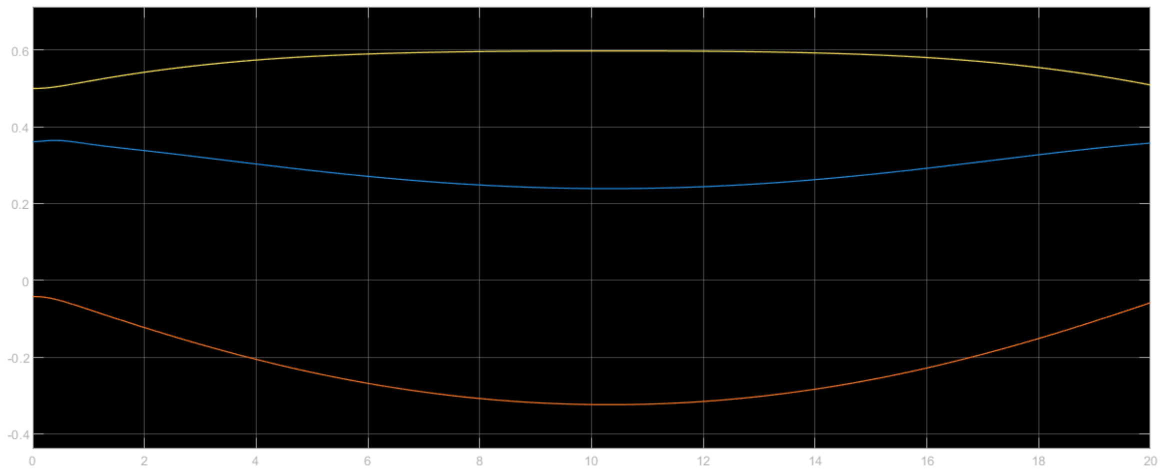
- $\mathbf{p}_{\text{PD}}(t)$ is the end-effector position computed by the decentralized PD loop at time t,
- $\mathbf{p}_{\text{inv}}(t)$ is the position predicted by the exact centralized inverse solution.

6.3 Path outputs

Figure 12 shows the joint-space trajectories generated by the two control strategies along the parabolic blend.



(a) Centralized Inverse controller output



(b) Decentralized PD controller output

Figure 12: Comparison of the end effector trajectories for the 5-DOF robot following the parabolic path using (a) centralized inverse kinematics and (b) decentralized PD controllers.

Notice that while both schemes track the desired parabolic trajectory, the decentralized PD trajectories exhibit slightly less smooth transitions.

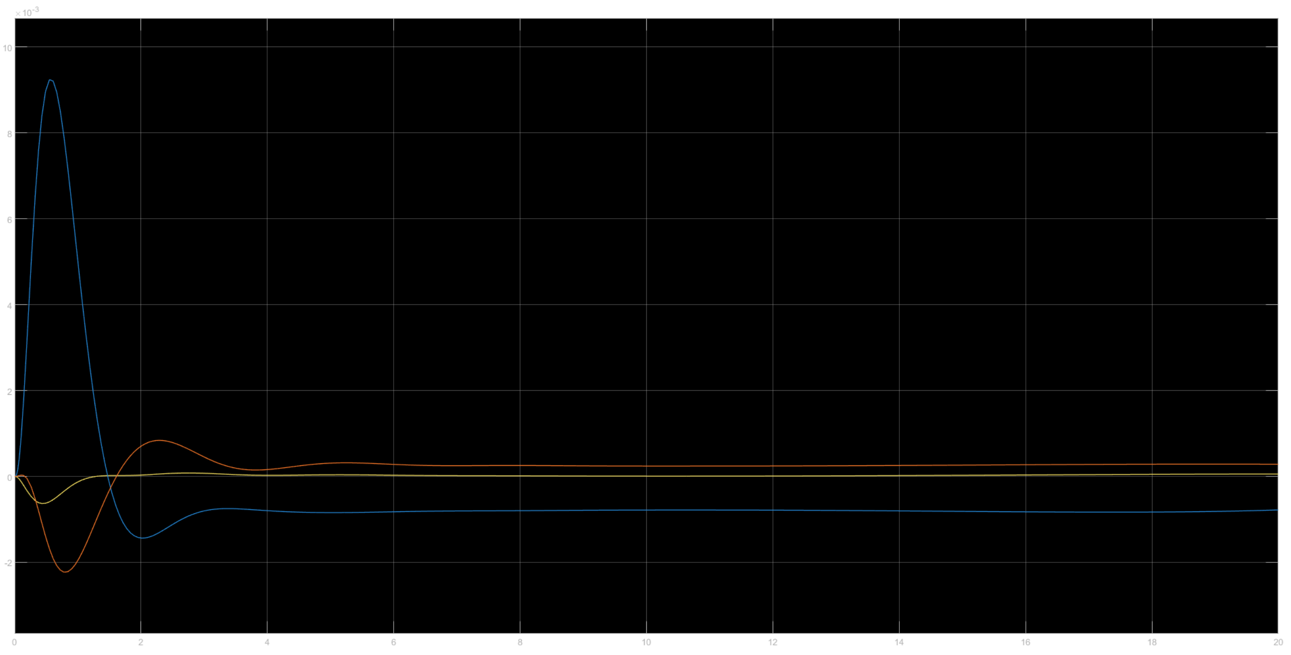


Figure 13: Difference between Decentralized PD and Centralized Inverse

When analyzing the difference between the two end effector trajectories of both the controllers, we can verify that the differences are withing the 10^{-3} m scale, what we can consider good results for our experiment.

7 Conclusion

By developing the two control models—a decentralized PD controller and a centralized inverse dynamics controller—we were able to directly compare their performance on the Mako surgical manipulator.

Both approaches used the same approximation to compute link masses and inertia from their volumes. The decentralized PD controller is straightforward to implement and runs with minimal computation. The centralized inverse-dynamics controller, gives much tighter and smoother trajectory tracking.

This study demonstrates that, for applications demanding high precision and fluid response non of the approaches are good, we are facing differences in the 10^{-2} m range, that in a surgical field can be the difference between life and death.

GitHub Link

You can find the code on GitHub at:
https://github.com/FranciscoVPinto/Lab_G11_Phase2

References

Craig, J. J. (2018). *Introduction to robotics: Mechanics and control.* Pearson.

Khalil, W., & Dombre, E. (2004). *Modeling, identification and control of robots.* Kogan Page Science.

Schweikard, A., & Ernst, F. (2015). *Medical robotics.* Springer International Publishing.

Siciliano, B., Sciavicco, L., Villani, L., & Oriolo, G. (2009). *Robotics: Modelling, planning and control.* Springer Verlag.

Spong, M. W., Hutchinson, S., & Vidyasagar, M. (2005). *Robot modeling and control.* Wiley.