GHENT UNIVERSITY
ENGINEERING DESIGN
FALL 2025

Control as Optimization for Robotic Manipulators

1 Project Overview

In robotic systems, manipulator control often involves balancing multiple competing objectives. A manipulator must accurately follow a desired trajectory while remaining dexterous/flexible enough to adapt to new tasks.

In this project we will pose the manipulator control problem as an optimization problem that balances two objectives: minimizing effort (joint velocity magnitudes), and maximizing dexterity (to maintain flexibility). We will do this while ensuring that the manipulator tracks a desired end-effector velocity trajectory and respects joint velocity and position limits.

Redundant manipulators, i.e. robots with more joints than strictly needed (e.g. a 3-joint planar arm moving in a 2D space) are generally preferred for such applications, as they can use their extra joint freedom to enhance dexterity while still achieving the desired end-effector velocity. **Non-Redundant manipulators**, i.e. manipulators with exactly the number of joints needed for the task, lack this flexibility but may still benefit from small deviations from the desired trajectory to improve dexterity.

The tasks would be to formulate and solve this multi-objective control problem for both redundant and non-redundant manipulators. You will work with simple three-link and two-link planar manipulators with revolute joints, understand their necessary kinematic quantities, transcribe the control of these manipulators as an optimization problem and use different numerical optimization techniques to solve it and finally implement it inside a control loop.

2 Background Theory

In this section, we introduce the necessary kinematic concepts and quantities needed to formulate the manipulator control problem as an optimization problem, namely the manipulator Jacobian, manipulability measure and manipulability Jacobian. Helper functions to compute these quantities will be provided in the supplementary material.

2.1 Planar Manipulator Kinematics

For an n-link planar manipulator with joint angles $\mathbf{q} = [q_1, q_2, \ldots, q_n]^\top$, the end-effector position $\mathbf{p}$ is given by:

$$\mathbf{p} = \begin{bmatrix} x \\ y \end{bmatrix} = f(\mathbf{q}) \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}^2$ is the forward kinematics function that maps joint angles to end-effector position.

It is important to distinguish between two spaces: the **joint space** and the **task space**. The space of all possible joint configurations $\mathbf{q}$ is called the joint space, while the space of all possible end-effector positions $\mathbf{p}$ is called the task space or operational space.

The differential relationship between joint velocities and end-effector velocity is [1]:

$$\dot{\mathbf{p}} \triangleq \mathbf{v} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \tag{2}$$

where $\mathbf{J}(\mathbf{q}) \in \mathbb{R}^{2\times n}$ is the manipulator Jacobian matrix, given by:

$$\mathbf{J}(\mathbf{q}) = \begin{bmatrix} \frac{\partial x}{\partial q_1} & \frac{\partial x}{\partial q_2} & \cdots & \frac{\partial x}{\partial q_n} \\ \frac{\partial y}{\partial q_1} & \frac{\partial y}{\partial q_2} & \cdots & \frac{\partial y}{\partial q_n} \end{bmatrix} \tag{3}$$

The required joint velocities can be calculated using the inverse of the Jacobian (if square and invertible) or the Moore-Penrose pseudoinverse (if not square or not invertible):

$$\dot{\mathbf{q}} = \mathbf{J}^{+}(\mathbf{q})\mathbf{v} \tag{4}$$

Note that equation (4) also turns out to be the solution to the projection problem we saw in the exercise session!

$$\begin{aligned} \min_{\dot{\mathbf{q}}} \quad & E(\dot{\mathbf{q}}) \triangleq \frac{1}{2}||\dot{\mathbf{q}}||^2 \\ \text{s.t.} \quad & J(\mathbf{q})\dot{\mathbf{q}} = \mathbf{v} \end{aligned} \tag{5}$$

The interpretation of the above optimization problem is that among all joint velocities that achieve the desired end-effector velocity, we choose the one with the smallest norm or **minimum effort** ($E(\dot{\mathbf{q}})$).

We will call the control strategy that uses the inverse velocity kinematics in (4) as the Inverse Velocity Kinematics Control ($\mathbf{IVKC}$) strategy.

2.2 Dexterity: Manipulability Measure and Jacobian

The **manipulability measure** quantifies how dexterous the manipulator is at a given configuration $\mathbf{q}$. A common definition is:

$$m(\mathbf{q}) = \sqrt{\det(\mathbf{J}(\mathbf{q})\mathbf{J}(\mathbf{q})^T)} \tag{6}$$

where, $\det$ is the determinant.

A geometric interpretation is that $m(\mathbf{q})$ represents the volume of the velocity ellipse[2] that can be achieved by unit joint velocities. As illustrated in Figure 1, the planar manipulator in configuration A can move its end-effector freely in both x and y direction. In contrast, in the fully extended configuration B, motion in the x direction is no longer possible. Configuration A therefore exhibits higher manipulability than configuration B

[1] Note that $\mathbf{p}$ and $\mathbf{q}$ are both functions of time. Although for brevity we omit the explicit time dependence, it is implied.

[2] Ellipsoid in 3D case and ellipse in 2D case

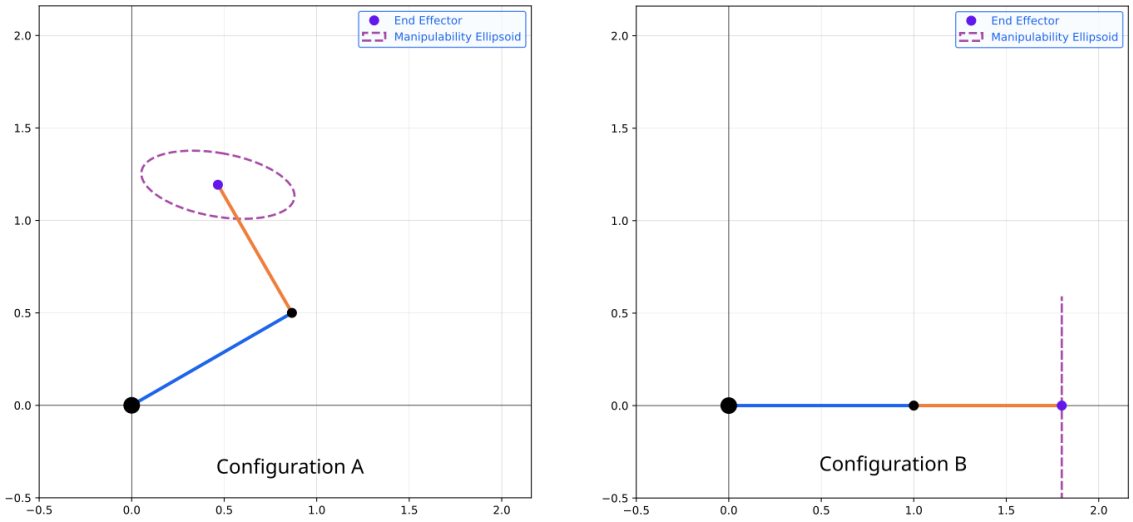

Figure 1: *Dexterity of a planar manipulator in two different configurations. Configuration A has high manipulability, while configuration B has low manipulability due to being fully extended.*

The ellipsoid is given by:

$$\mathbf{x} \in \mathbb{R}^2 : \mathbf{x}^T (\mathbf{J}\mathbf{J}^T)^{-1} \mathbf{x} \leq 1 \tag{7}$$

Similar to the mapping from joint velocities to end-effector velocities, a mapping from joint velocities to the rate of change of manipulability $M(\mathbf{q}, \dot{\mathbf{q}})$ can be obtained by differentiating (6):

$$M(\mathbf{q}, \dot{\mathbf{q}}) \triangleq \dot{m} = J_m^T(\mathbf{q})\dot{\mathbf{q}} \tag{8}$$

where $J_m(\mathbf{q})$ is called the **manipulability Jacobian**. Note that if the configuration $\mathbf{q}$ is known, then the rate of change of manipulability is only dependent on the joint velocities $\dot{\mathbf{q}}$. In that case we can drop the explicit dependence on $\mathbf{q}$, and it becomes a **linear function** of $\dot{\mathbf{q}}$. As we will see later, this will be the case for us and we will therefore write $M(\dot{\mathbf{q}})$.

3 Problem Formulation: Redundant Manipulator Control

Consider a point-to-point motion task wherein a three-link planar manipulator's end-effector must move from an initial position $\mathbf{p}_0$ to a goal position $\mathbf{p}_{\text{goal}}$. To achieve this task we firstly develop a high-level planner that generates a desired end-effector velocity trajectory to move the end-effector from its current position to the goal position. We then track this desired end-effector velocity while moving in directions that minimize effort and maximize dexterity.

3.1 Trajectory Generation

For trajectory generation, we use a simple proportional controller to generate a desired end-effector velocity that drives the end-effector toward the goal position.

For implementation purposes, we will discretize time into steps of size Δt. At each instant $\mathbf{t_k} = k\,\Delta t$, the controller observes the error $\mathbf{e_k}$ between the current end-effector position $\mathbf{p_k}$ and the goal position $\mathbf{p}_{\text{goal}}$ and generates a desired end-effector velocity $\mathbf{v_{d,k}}$ using a proportional control law:

$$\mathbf{v_{d,k}} = K_p\,\mathbf{e_k} \tag{9}$$

where $K_p > 0$ is a proportional gain (typically a scalar or diagonal matrix).

3.2 Optimization Problem Description

At each discrete time step k, the desired end-effector velocity $\mathbf{v_{d,k}}$ **must** be achieved while minimizing the required effort $E(\dot{\mathbf{q}})$ and maximizing the rate of change of manipulability $M(\dot{\mathbf{q}})$. To encapsulate these competing objectives into a single objective we can use their weighted sum, with the weights $\mathbf{w_E}$ and $\mathbf{w_M}$ for effort and manipulability respectively.

We would also like to ensure that the joint velocities remain within specified limits:

$$\dot{\mathbf{q}}_{\min} \le \dot{\mathbf{q}} \le \dot{\mathbf{q}}_{\max} \tag{10}$$

To ensure the manipulator operates within safe joint limits, we add implicit velocity-based constraints. These constraints progressively reduce the maximum allowed joint velocity as the joint approaches its positional limit. The constraint activates when the joint enters an *influence zone* and becomes increasingly restrictive to prevent any motion that would violate physical joint boundaries.

For each joint j, two constraints are therefore applied:

$$\begin{aligned} \dot{\mathbf{q}}_j &\le \frac{1}{K_{\text{safe}}}\frac{\mathbf{d}_j^+ - \mathbf{s}_j}{\mathbf{i}_j - \mathbf{s}_j} \quad \text{if } \mathbf{d}_j^+ < \mathbf{i}_j \\ \dot{\mathbf{q}}_j &\ge -\frac{1}{K_{\text{safe}}}\frac{\mathbf{d}_j^- - \mathbf{s}_j}{\mathbf{i}_j - \mathbf{s}_j} \quad \text{if } \mathbf{d}_j^- < \mathbf{i}_j \end{aligned} \tag{11}$$

where $\mathbf{d}_j^+ = \mathbf{q}_j^+ - \mathbf{q}_j$ and $\mathbf{d}_j^- = \mathbf{q}_j - \mathbf{q}_j^-$ are distances to upper and lower joint limits, $\mathbf{i}_j$ is the influence distance, $\mathbf{s}_j$ is the stopping distance that a joint should never reach, and K_{safe} is the safety gain controlling velocity reduction.

3.3 Control Algorithm

The complete control algorithm is summarized in Algorithm 1, which we will call Effort Minimizing and Dexterity Maximizing Control ($\mathbf{EMDM}$) Algorithm. At each time step, the current end-effector position is computed, the position error is determined, and the desired end-effector velocity is generated. The necessary kinematic quantities like Jacobian and manipulability Jacobian are then computed, which are fed to the optimization problem to obtain the optimal joint velocity command. Finally, the configuration is updated for the next iteration.

4 Problem Formulation: Non-Redundant Manipulator Control

As was discussed in Section (1), non-redundant manipulators cannot achieve both the desired end-effector velocity and improved manipulability simultaneously. To allow some flexibility, we introduce a slack vari-

Algorithm 1 Effort Minimizing and Dexterity Maximizing Control ($\mathbf{EMDM}$)

Input: Initial joint configuration $\mathbf{q}_0$, goal position $\mathbf{p}_{\text{goal}}$, sampling time Δt, servo gain K_p, weights $\mathbf{w_E}$ and $\mathbf{w_M}$, joint velocity and position limits: $\dot{\mathbf{q}}_{\min}$, $\dot{\mathbf{q}}_{\max}$, $\mathbf{q}_{\min}$, $\mathbf{q}_{\max}$, safety parameters: $\mathbf{i_j}$, $\mathbf{s_j}$, K_{safe}
Output: Sequence of configurations $\{\mathbf{q}_k\}$ and end-effector positions $\{\mathbf{p}_k\}$
Initialize: $\mathbf{q} \leftarrow \mathbf{q}_0$
while target not reached **do**
  Compute current end-effector position $\mathbf{p}_k = f(\mathbf{q}_k)$
  Compute position error $\mathbf{e}_k = \mathbf{p}_{\text{goal}} - \mathbf{p}_k$
  Compute desired end-effector velocity $\dot{\mathbf{v}}_{\mathbf{d,k}} = K_p \mathbf{e}_k$
  Compute Jacobian $J(\mathbf{q}_k)$ and manipulability Jacobian $J_m(\mathbf{q}_k)$
  Solve the optimization problem described in section (3.2) to find $\dot{\mathbf{q}}_k^*$
  Update configuration: $\mathbf{q}_{k+1} = \mathbf{q}_k + \dot{\mathbf{q}}_k^* \Delta t$
end while

able $\boldsymbol{\delta}$, which will allow small deviations from the desired end-effector velocity. We therefore have:

$$\mathbf{v_{d,k}} - \boldsymbol{\delta} = J(\mathbf{q})\dot{\mathbf{q}} \tag{12}$$

We also augment the optimization objective discussed in section (3.2) to penalize large deviations from the desired end-effector velocity, by considering the following term:

$$\mathbf{E}_{\text{slack}} = \mathbf{w_S} \frac{1}{2} \|\boldsymbol{\delta}\|^2 \tag{13}$$

where $\mathbf{w_S}$ is a weight that penalizes large deviations. A common practice is to set this weight adaptively, proportional to $1/\mathbf{e_k}$. We will therefore set it as follows: $\mathbf{w_S} = \frac{\mathbf{c_S}}{\mathbf{e_k}}$, where $\mathbf{c_S}$ will be our tuning parameter. This ensures that as the end-effector approaches the goal position, the slack variable is penalized more heavily, encouraging precise tracking.

Finally, we add constraints to ensure that the slack variable remains within reasonable bounds:

$$\boldsymbol{\delta}_{\min} \leq \boldsymbol{\delta} \leq \boldsymbol{\delta}_{\max} \tag{14}$$

We keep the trajectory generation and control algorithm the same as in section (3), with the only difference thus being the optimization problem solved at each time step.

5 Implementation Notes

To distinguish between constrained and unconstrained versions of the problem, we prepend the algorithm names with $\mathbf{C}-$ and $\mathbf{UC}-$ respectively. Here, "constrained" refers **specifically** to the inclusion of joint velocity and joint position limit constraints. For example, the constrained version of the $\mathbf{EMDM}$ algorithm is called $\mathbf{C-EMDM}$, while its unconstrained counterpart is $\mathbf{UC-EMDM}$. A skeleton code for the control algorithm will be provided. You will only need to fill in the missing parts ☺.

6 Project Tasks

The project consists of two main tasks. Task 1 involves implementing and comparing different control algorithms for a 3-link redundant planar manipulator, while Task 2 focuses on a 2-link non-redundant planar

manipulator. Each task requires you to implement the respective control algorithms, analyze their performance, and explore the trade-offs between effort minimization and dexterity maximization.

Task 1: Control of a 3-link Redundant Manipulator

With the set of parameters provided in Table 1, workout the following:

1. Implement and compare $\mathbf{UC-EMDM}$ and $\mathbf{UC-IVKC}$[a].
   - For the $\mathbf{UC-EMDM}$ controller, formulate and solve the optimization problem at each time step using an appropriate custom solver. If the custom solver does not converge to a locally optimal solution, employ an off-the-shelf solver as a fallback and document the convergence behavior and characterize the numerical challenges faced in your report.
   - Animate the controller trajectories and plot the effort, manipulability $m(\mathbf{q})$, joint velocities and positions over time for both controllers.
   - Discuss the differences in performance between the two controllers in terms of trajectory tracking and manipulability (Create overlaid plots for easy comparison).
   - For the $\mathbf{UC-EMDM}$ controller, try playing with the weights $\mathbf{w_E}$ and $\mathbf{w_M}$ to explore the trade-off between effort minimization and dexterity maximization. Plot the Pareto front between effort $E(\dot{\mathbf{q}})$ and manipulability rate $M(\dot{\mathbf{q}})$ at four time steps spanning the trajectory: $t = 0$, $t = \lfloor T/3 \rfloor$, $t = \lfloor 2T/3 \rfloor$, and $t = T$, where T is the final time step reached by the controller and $\lfloor \cdot \rfloor$ denotes the floor operation to ensure integer indices.
2. Implement and compare $\mathbf{C-EMDM}$ and $\mathbf{C-IVKC}$, following the same steps and analysis as in the previous part.

[a]Note that for $\mathbf{UC-IVKC}$ one simply needs to replace Step 9 in Algorithm 1 by the equation 4

Table 1: *Problem Parameters for 3-link Manipulator*

Parameter	**Value**
Manipulator link lengths	$l_1 = 0.2, l_2 = 0.6, l_3 = 0.3$ m
Joint limits	$q \in [-2.0, 2.0],\ i = 1, 2, 3$ rad
Joint velocity limits	$\dot{q}_i \in [-1.0, 1.0],\ i = 1, 2, 3$ rad/s
Initial configuration	$\mathbf{q}_0 = [0, \pi/6, \pi/6]^T$
Goal position	$\mathbf{p}_{\text{goal}} = [-0.7, 0.5]^T$ m
Sampling time	$\Delta t = 0.05$ s
Proportional gain	$K_p = 1.0$
Weights	$\mathbf{w_E} = 0.1$, $\mathbf{w_M} = 5.0$, $\mathbf{c_S} = 10.0$
Slack limits	$\boldsymbol{\delta}_{\min} = [-1, -1]^T$, $\boldsymbol{\delta}_{\max} = [1, 1]^T$
Safety parameters	$\mathbf{i_j} = 0.785398$ rad, $\mathbf{s_j} = 0.0872665$ rad, $K_{\text{safe}} = 1.0$

Task 2: Control of a 2-link Non-Redundant Manipulator

With the set of parameters provided in Table 2, workout the following:

1. Implement and compare $\mathbf{C-EMDM}$ and $\mathbf{C-IVKC}$.
   - For the $\mathbf{C-EMDM}$ controller, formulate and try to solve the optimization problem at each time step using a gradient-based AND a gradient-free solver, and compare their behavior in terms of convergence speed and solution quality.
   - Animate the controller trajectories and plot the effort, manipulability $m(\mathbf{q})$, joint velocities and positions over time for both controllers (Create overlaid plots for easy comparison).
   - Discuss the differences in performance between the two controllers in terms of trajectory tracking, manipulability and time needed to reach the goal.
2. Implement and compare the performance of $\mathbf{UC-EMDM}$ and $\mathbf{UC-IVKC}$.

Table 2: *Problem Parameters for 2-link Manipulator*

Parameter	**Value**
Manipulator link lengths	$l_1 = 0.5, l_2 = 0.4$ m
Joint limits	$q_1 \in [-2, 2],\ q_2 \in [-2, 2]$ rad
Joint velocity limits	$\dot{q}_i \in [-1.5, 1.5],\ i = 1, 2$ rad/s
Initial configuration	$\mathbf{q}_0 = [0, \frac{\pi}{6}]^T$
Goal position	$\mathbf{p}_{\text{goal}} = [-0.3, -0.6]^T$ m
Sampling time	$\Delta t = 0.05$ s
Proportional gain	$K_p = 1.0$
Weights	$\mathbf{w_E} = 0.1$, $\mathbf{w_M} = 5.0$, $\mathbf{c_S} = 10.0$
Slack limits	$\boldsymbol{\delta}_{\min} = [-1, -1]^T$, $\boldsymbol{\delta}_{\max} = [1, 1]^T$
Safety parameters	$\mathbf{i_j} = 0.785398$ rad, $\mathbf{s_j} = 0.0872665$ rad, $K_{\text{safe}} = 1.0$

7 Supplementary Material

The supplementary material for this project can be found at https://github.ugent.be/abhole/emdm. This include:

- Skeleton code for the control algorithm described in Algorithm 1, which can used as the starting point for the project implementation. (/scripts/control.py)
- All helper functions, e.g. for calculating the kinematic quantities described in Sections (2.1) and (2.2) and animating the trajectory of the manipulator along with its manipulability ellipsoid can be found inside (/emdm/utils.py)

Note that students are required to use this provided code as the foundation for their implementation. Any questions concerning the project and the supplementary material may be directed to ajinkya.bhole@ugent.be

8 Deliverable Guidelines

The project consists of two deliverables: a report and a final pitch. It is mandatory to use either the Word or LaTeX IEEE, two column template for your reports. The templates can be found on Ufora.

- Please limit the final report to four pages maximum, ensuring it clearly outlines the optimization problem formulations for each task and provides direct answers to all questions.
- The goal of the last pitch is to present and motivate your solution to your peers and the project supervisors. The scope of the final pitch should be less technical than the final report, focussing on a short description of the objective, a brief high-level overview of the methods used and an analysis and discussion of the final results. Each group has a 10-minute time limit for the final pitch, followed by a 5-minute Q&A session.

9 Timeline Overview

The following deadlines are important to keep in mind for the project:

- Project subscription on **22/10/2025**
- Final report is due on **14/12/2025** at 23:59
- Final project pitch is due on **15/12/2025** (Details about the venue and time will be communicated later on.)