TÉCNICO LISBOA

Optimization and Algorithms

MEEC

Project Report

Authors:

David Marafuz Gaspar (106541) david.marafuz.gaspar@tecnico.ulisboa.pt
Pedro Gaspar Mónico (106626) pedro.monico@tecnico.ulisboa.pt
Francisco Palma (105949) francisco.c.palma@tecnico.ulisboa.pt
Pedro Salazar Leite (106812) pedro.s.leite@tecnico.ulisboa.pt

Group 33

2025/2026 – 1st Semester, P1

Contents

1 Task 1 [Numerical Task] 2

2 Task 2 [Theoretical task] 5

3 Task 3 [Theoretical task] 6

4 Task 4 [Numerical task] 8

5 Task 5 [Theoretical task] 9

6 Task 6 [Theoretical task] 10

7 Task 7 [Numerical task] 11

8 Task 8 [Theorical task] 12

9 Task 9 [Numerical task] 13

10 Task 10 [Numerical task] 15

11 Task 11 [Numerical task] 18

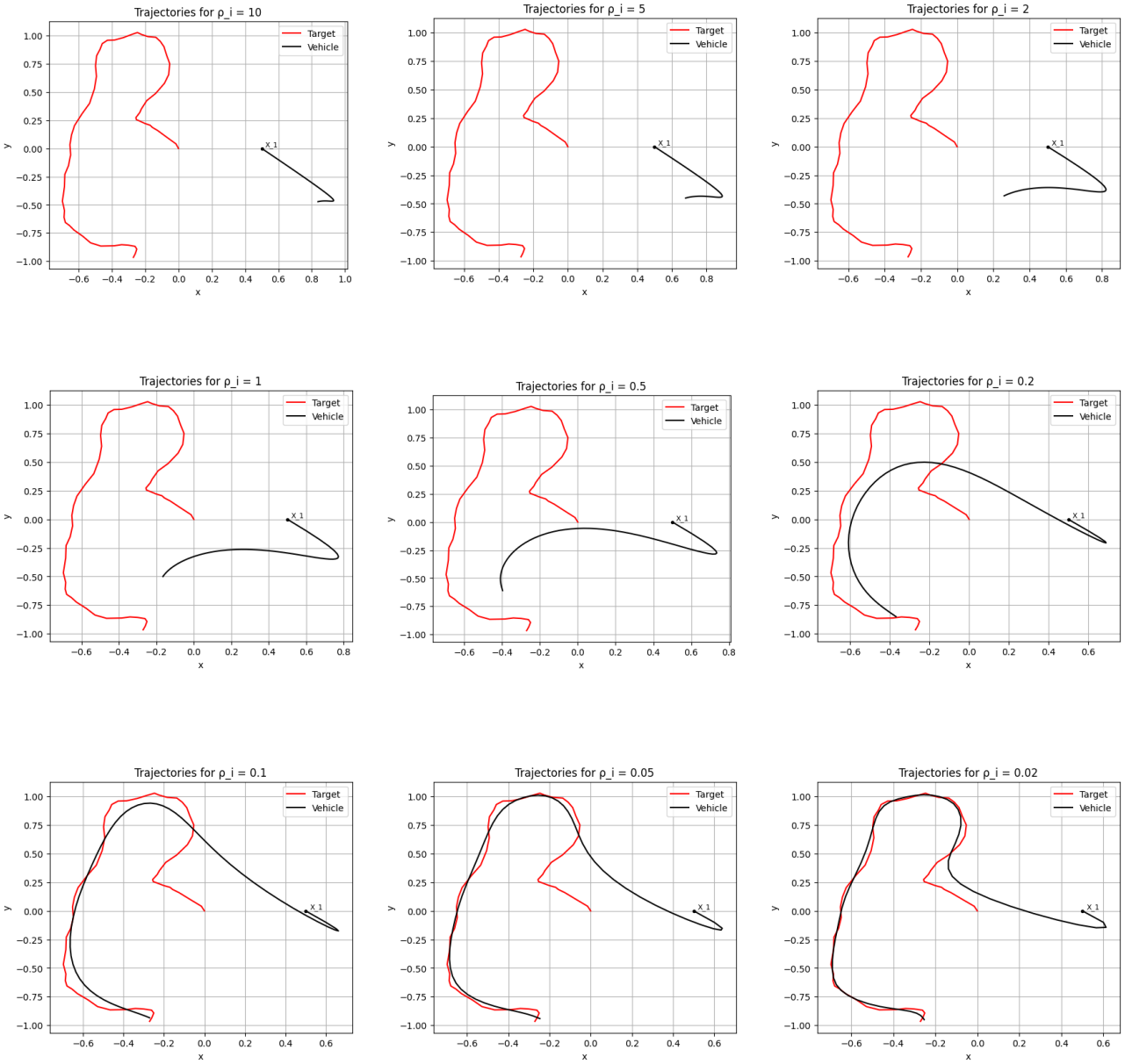
1 Task 1 [Numerical Task]

We aim to solve the following optimization problem

$$\begin{aligned} \min_{x,\,u} \quad & \sum_{t=1}^{T} \|Ex(t) - q(t)\|_2 \; + \; \rho_i \sum_{t=1}^{T-1} \|u(t)\|_2^2 \\ \text{subject to} \quad & x(1) = x_{\text{initial}}, \\ & x(t+1) = Ax(t) + Bu(t), \quad t = 1, \ldots, T-1. \end{aligned} \tag{1}$$

for $\rho_i = \{10,\ 5,\ 2,\ 1,\ 0.5,\ 0.2,\ 0.1,\ 0.05,\ 0.02,\ 0.01,\ 0.005,\ 0.002\}$.

Solution:



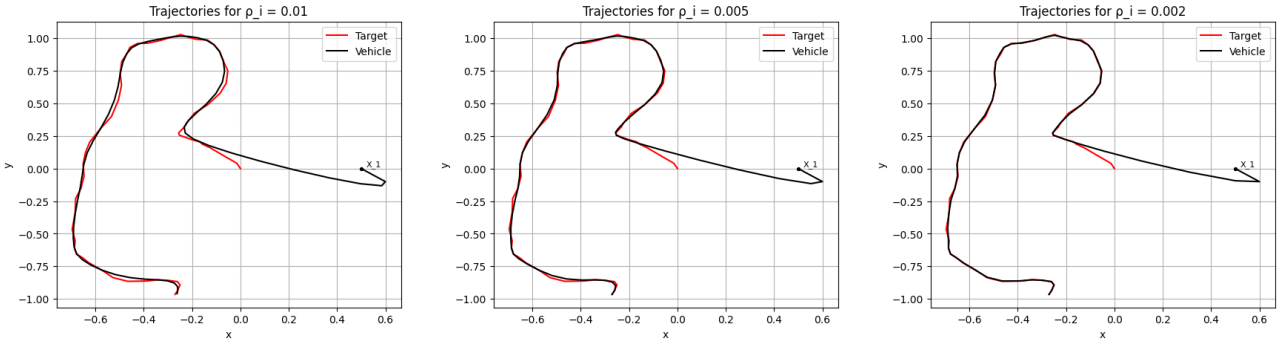


Figure 4: Each subfigure shows the trajectories of the target (red) and our vehicle (black) obtained by solving the optimization problem 1 for the corresponding ρ_i.

For each ρ_i, the corresponding tracking error ($\text{TE} = \sum_{t=1}^{T} \|Ex(t) - q(t)\|_2$) and control effort ($\text{CE} = \sum_{t=1}^{T-1} \|u(t)\|_2^2$) were measured to illustrate the impact of varying the weights.

ρ_i	10	5	2	1	0.5	0.2	0.1	0.05	0.02	0.01	0.005	0.002
TE_i	88.84	83.56	69.84	55.75	43.40	23.73	12.00	9.24	6.45	4.53	3.44	2.60
CE_i	0.28	1.07	5.77	15.91	33.68	99.65	179.38	216.86	310.13	441.80	594.65	860.26

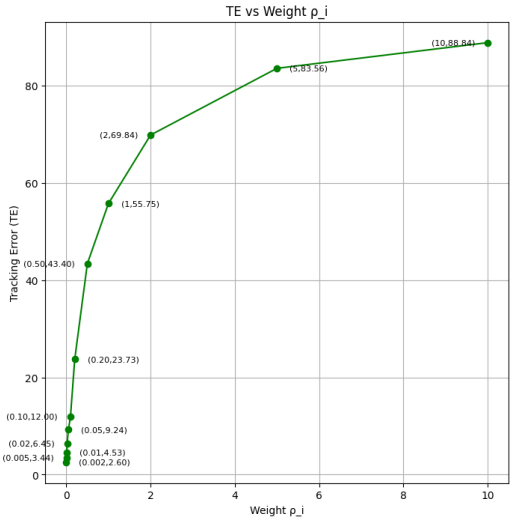


Figure 5: Tracking Error (TE) for each ρ_i.

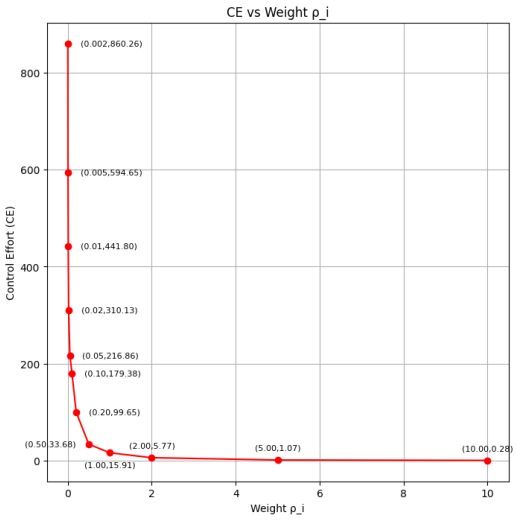


Figure 6: Control Effort (CE) vfor each ρ_i.

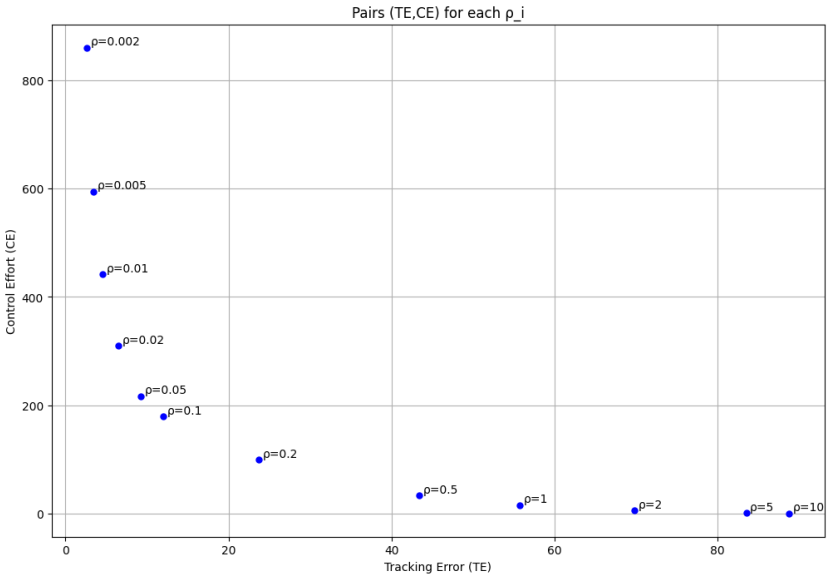


Figure 7: Pairs (TE(ρ_i), CE(ρ_i)) for each ρ_i.

Observing the plots of Tracking Error (TE) and Control Effort (CE) as a function of ρ_i, it can be noticed that, as ρ_i decreases, the tracking error also decreases, indicating that the vehicle follows the target more accurately. This behavior is expected since ρ_i represents the weight given to the control effort in the cost function, lower values of ρ_i reduce the penalty for large control inputs, allowing the controller to act more aggressively to minimize the tracking error. This can also be confirmed by examining the trajectory plots. For very small values of ρ_i, such as 0.002, the vehicle trajectory almost perfectly matches the target, whereas for large values, as ρ_i increases, the trajectory significantly deviates from the target. At the same time, the control effort increases significantly as ρ_i decreases, confirming the trade-off between minimizing tracking error and minimizing control effort. However, beyond a certain point, further decreasing ρ_i results in a small improvement in tracking error while substantially increasing the control effort. This highlights the importance of selecting ρ_i appropriately,to avoid unnecessarily high control efforts that may be costly or impractical.

2 Task 2 [Theoretical task]

Let TE(ρ) and CE*(ρ) denote the tracking error and control effort obtained after minimizing the cost function 1 for a given $\rho > 0$. Consider now two values of ρ, say, ρ_a and ρ_b, and suppose that TE*(ρ_a) $\leq$ TE*(ρ_b). Prove that:*

$$\mathrm{CE}^*(\rho_a) \geq \mathrm{CE}^*(\rho_b).$$

Proof:

Let TE(ρ) and CE(ρ) denote the tracking error and control effort obtained after minimizing the cost function

$$J_\rho(x, u) = \mathrm{TE}(\rho) + \rho\,\mathrm{CE}(\rho)$$

for a given $\rho > 0$.

For each ρ_i, let (x_i^*, u_i^*) be the control and state trajectory pair that achieves the minimum of J_{ρ_i}, i.e.,

$$(x_i^*, u_i^*) = \arg\min_{x,u} J_{\rho_i}(x, u).$$

Consider now two values ρ_a and ρ_b with $\mathrm{TE}(\rho_a) \leq \mathrm{TE}(\rho_b)$. By the definition of a global minimizer, the optimal value for ρ_b must satisfy

$$J_{\rho_b}(x_b^*, u_b^*) \leq J_{\rho_b}(x_a^*, u_a^*),$$

where (x_b^*, u_b^*) is the minimizer for ρ_b and (x_a^*, u_a^*) is the minimizer for ρ_a.

Expanding this inequality

$$\mathrm{TE}(\rho_b) + \rho_b\,\mathrm{CE}(\rho_b) \leq \mathrm{TE}(\rho_a) + \rho_b\,\mathrm{CE}(\rho_a),$$

which can be rearranged as

$$\mathrm{TE}(\rho_b) - \mathrm{TE}(\rho_a) \leq \rho_b\left(\mathrm{CE}(\rho_a) - \mathrm{CE}(\rho_b)\right).$$

Since $\rho_b > 0$ and $\mathrm{TE}(\rho_a) \leq \mathrm{TE}(\rho_b)$ by assumption, it follows that

$$\mathrm{CE}(\rho_a) - \mathrm{CE}(\rho_b) \geq 0 \quad \Longrightarrow \quad \mathrm{CE}(\rho_a) \geq \mathrm{CE}(\rho_b).$$

The results confirm the expected trade-off, reducing tracking error comes at the cost of increased control effort.

3 Task 3 [Theoretical task]

Show that the optimization problem 1 has a unique solution.

Hint: rewrite 1 as an unconstrained optimization problem that depends only on the variable u.

Proof:
Expanding the dynamics equation for the first few time steps t, we get:

$$\begin{aligned} x(1) &= x_{\text{init}}, \\ x(2) &= Ax_{\text{init}} + Bu(1), \\ x(3) &= A^2x_{\text{init}} + ABu(1) + Bu(2), \\ x(4) &= A^3x_{\text{init}} + A^2Bu(1) + ABu(2) + Bu(3), \\ &\vdots \end{aligned}$$

By inspection, for $1 \leq t \leq T$ we can rewrite $u(t)$ as a function of only $u(t)$:

$$x(t) = A^{t-1}x_{\text{init}} + \sum_{k=1}^{t-1} A^{t-1-k}B\,u(k). \tag{2}$$

We can define the cost function $f(u)$ as

$$f(u) := \sum_{t=1}^{T} \left\| \mathbb{E}\Big(A^{t-1}x_{\text{init}} + \sum_{k=1}^{t-1} A^{t-k}B\,u(k)\Big) - q(t) \right\|_{\infty} + \rho \sum_{t=1}^{T-1} \|u(t)\|_2^2.$$

The unconstrained optimization problem can then be written as

$$\min_{u} f(u).$$

To demonstrate that the optimization problem admits a unique solution, it is sufficient to establish that the cost function $f(u)$ is **strongly convex**.

Convexity Analysis

We can analyze the convexity properties of the cost function $f(u)$ by decomposing it into individual components.

$$f(u) = \sum_{t=1}^{T} g_t(u) + \rho \sum_{t=1}^{T-1} h_t(u),$$

where

$$g_t(u) = \left\| \mathbb{E}\Big(A^{t-1}x_{\text{init}} + \sum_{k=1}^{t-1} A^{t-k}B\,u(k)\Big) - q(t) \right\|_{\infty}, \qquad h_t(u) = \|u(t)\|_2^2.$$

Since the sum of convex functions remains convex, $f(u)$ is convex if each $g_t(u)$ and $h_t(u)$ is convex. If, in addition, at least one of these terms is strongly convex while the others are convex, $f(u)$ becomes strongly convex, which guarantees the existence of a unique global minimizer. In the following, we analyze both $g_t(u)$ and $h_t(u)$ to formally establish their convexity properties.

The function $g_t(u)$ can be expressed as the composition of two functions:

$$g_t(u) = \left\| \underbrace{\mathbb{E}\big[A^{t-1}x_{\text{initial}} - q(t)\big]}_{C_t} + \sum_{k=1}^{t-1} \mathbb{E}\big[A^{t-k}B\,u(k)\big] \right\|_\infty = (\ell_\infty \circ z_t)(u)$$

where

$$z_t(u) = \sum_{k=1}^{t-1} \mathbb{E}\big[A^{t-k}B\,u(k)\big] + C_t$$

Note that $z_t(u)$ is an affine mapping, as it can be written in the standard form $z_t(u) = s^\top u + b$, where $u = [\,u(1)^\top, \ldots, u(t-1)^\top\,]^\top$ is the control vector, $s^\top = \mathbb{E}[A^{t-1-k}B]$ the linear coefficients, and $b = C_t$ is a constant vector. Therefore, $g_t(u)$, being the composition of the convex ℓ_∞ norm with an affine mapping, is convex by definition.

We now analyze the function $h_t(u)$. If $h_t(u)$ is strongly convex, its Hessian satisfies

$$\nabla^2 h(u) \succeq mI_2 \qquad \text{for all } u \in \mathbb{R}^2.$$

To verify this property for $h_t(u)$, we set $u(t) = [u_1\ u_2]^\top$ and compute the gradient and Hessian matrix:

$$h_t(u) = u_1^2 + u_2^2.$$

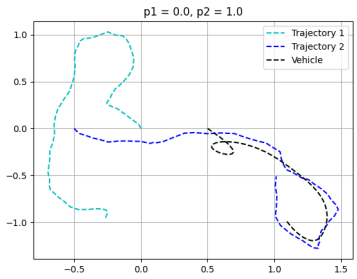
$$\nabla h_t(u) = \begin{bmatrix} 2u_1 \\ 2u_2 \end{bmatrix} = 2u, \qquad \nabla^2 h_t(u) = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} = 2I_2.$$

Since $h_t(u)$ is twice continuously differentiable and its Hessian satisfies $\nabla^2 h_t(u) = 2I_2 \succeq 2I_2$ for all $u \in \mathbb{R}^2$ and is therefore positive semidefinite, it follows that $h_t(u)$ is strongly convex (with modulus $m = 2$).

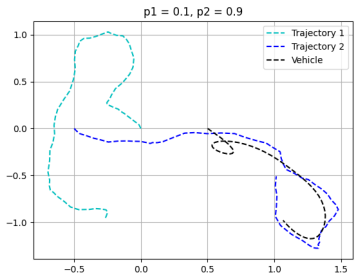
In summary, because $f(u)$ is composed of convex functions $g_t(u)$ and strongly convex functions $h_t(u)$, the cost function $f(u)$ is strongly convex. Consequently, the optimization problem admits a unique global minimizer, ensuring both existence and uniqueness of the solution.

4 Task 4 [Numerical task]

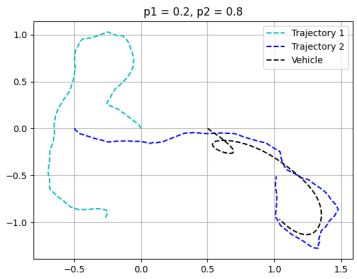
The results obtained are shown in the following graphs. We can see that, as expected, the vehicle will try to approach the trajectory that represents the greatest weight in the cost function, in this case represented as a probability, with extreme cases being instance 11 ($p_1 = 1, p_2 = 0$) and instance 1 ($p_1 = 0, p_2 = 1$), where the vehicle's trajectory will almost completely overlap trajectories 1 and 2, respectively. In intermediate cases, there is a division between the approximation to the two trajectories, and the vehicle tends to stay closer to the one with the higher probability.



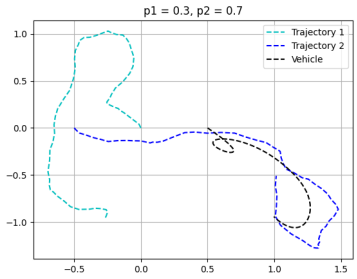
Instance 1



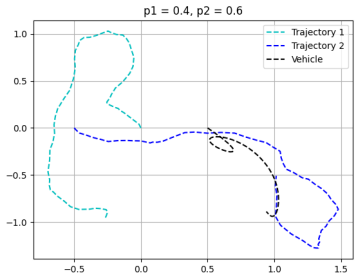
Instance 2



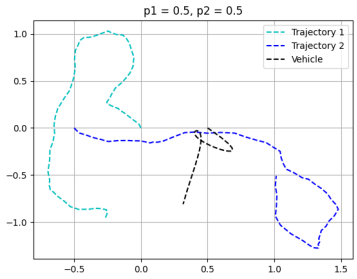
Instance 3



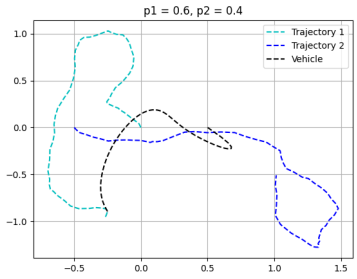
Instance 4



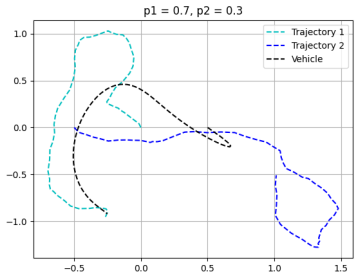
Instance 5



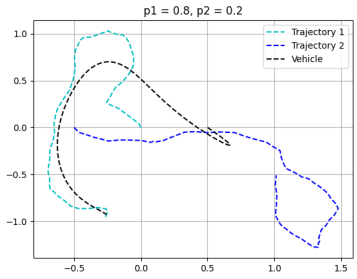
Instance 6



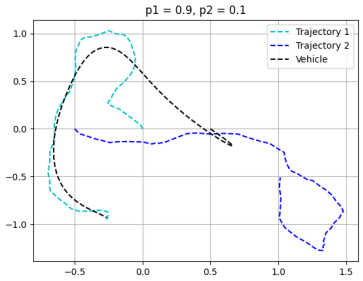
Instance 7



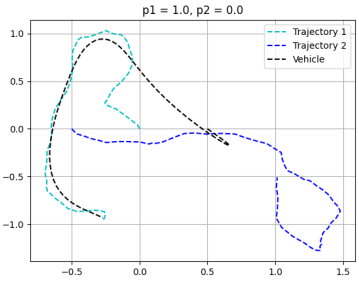
Instance 8



Instance 9



Instance 10



Instance 11

5 Task 5 [Theoretical task]

The optimization problem under consideration involves tracking a moving target where information of the what actual path the target is doing is revealed at midway point. Since we only have 2 possible two possible trajectories that the target can follow $q_1(t)$ or $q_2(t)$ they have the respective probabilities p_1 and p_2, where $p_1 + p_2 = 1$. Crucially, for the purpose of further tasks, the actual trajectory is revealed at time $t = 25$, creating an information structure that must be respected in the optimization formulation.

The incomplete formulation in Equation (5) can be expressed as:

$$
\begin{aligned}
\underset{x_1,u_1,x_2,u_2}{\text{minimize}} \quad & p_1 \left(\sum_{t=1}^{T} \|Ex_1(t) - q_1(t)\|_2 + \rho \sum_{t=1}^{T-1} \|u_1(t)\|_2^2 \right) \\
& + p_2 \left(\sum_{t=1}^{T} \|Ex_2(t) - q_2(t)\|_2 + \rho \sum_{t=1}^{T-1} \|u_2(t)\|_2^2 \right) \\
\text{subject to} \quad & x_1(1) = x_{\text{init}} \\
& x_1(t+1) = Ax_1(t) + Bu_1(t), \quad \text{for } 1 \le t \le T-1 \\
& x_2(1) = x_{\text{init}} \\
& x_2(t+1) = Ax_2(t) + Bu_2(t), \quad \text{for } 1 \le t \le T-1
\end{aligned}
$$

The main issue with this formulation is that the control variables $u_1(t)$ and $u_2(t)$ are treated independently throughout the entire time horizon. This implicitly assumes that the controller already knows which trajectory the target will follow, which contradicts the task specification, where the controller only learns the target's trajectory at time step $t = 25$. As a result, this decomposition allows the solver to drive x_1 towards q_1 and x_2 towards q_2 from the very beginning of the time horizon ($t = 1$).

To summarize, in the current formulation the controller can solve the minimization problem for each pair (x_n, u_n) from the very beginning. The new constraint should enforce that all the pairs (x_n, u_n) remain identical until the actual trajectory of the target is revealed and the time step $t = 25$. This ensures that the controller cannot "cheat" by using future information before it becomes available.

6 Task 6 [Theoretical task]

To resolve violation identified in Task 5, we must add constraints that enforce that until the trajectory information is revealed at $t = 25$, the control actions remain identical for all possible scenarios. For the general case with N possible trajectories, this condition is enforced by introducing the following constraints:

$$u_i(t) = u_j(t) \quad \text{for all } i, j \in \{1, \ldots, N\}, \quad 1 \leq t \leq 24$$

These constraints ensure that from $t = 1$ to $t = 24$, the controller applies the same control actions across all possible realizations, since the true trajectory is not yet known. Consequently, the corresponding state trajectories also coincide over this period. Given that $x_i(1) = x_{\text{init}}$ for all i, it follows from the state transition equations that

$$x_i(t) = x_j(t) \quad \text{for all } i, j \in \{1, \ldots, N\}, \quad 1 \leq t \leq 25$$

Thus, the system evolves identically across all scenarios until the information is revealed at $t = 25$. After this point, the controller can implement distinct control sequences $u_i(t)$ for each trajectory, allowing the system to adapt optimally once the uncertainty is resolved.

The complete optimization problem with the necessary constraints for the case with 2 possible trajectories is:

$$\begin{aligned}
\underset{x_1,u_1,x_2,u_2}{\text{minimize}} \quad & p_1 \left(\sum_{t=1}^{T} \|Ex_1(t) - q_1(t)\|_2 + \rho \sum_{t=1}^{T-1} \|u_1(t)\|_2^2 \right) \\
& + p_2 \left(\sum_{t=1}^{T} \|Ex_2(t) - q_2(t)\|_2 + \rho \sum_{t=1}^{T-1} \|u_2(t)\|_2^2 \right) \\
\text{subject to} \quad & x_1(1) = x_{\text{init}} \\
& x_1(t+1) = Ax_1(t) + Bu_1(t), \quad \text{for } 1 \leq t \leq T-1 \\
& x_2(1) = x_{\text{init}} \\
& x_2(t+1) = Ax_2(t) + Bu_2(t), \quad \text{for } 1 \leq t \leq T-1 \\
& u_1(t) = u_2(t), \quad \text{for } 1 \leq t \leq 24
\end{aligned}$$

7 Task 7 [Numerical task]

We solved the minimization problem using the parameters $\rho = 0.1$, $p_1 = 0.6$, and $p_2 = 0.4$. The results align with our expectations based on the theoretical formulation. As imposed by the new constraint, the trajectories remain mathematically the same until the information about the target's actual path is revealed at $t = 25$. After this point, the optimization allows the trajectories to diverge, producing two distinct trajectories: one that tracks target 1 and another that tracks target 2.

Since the probability p_1 is larger than p_2, the solver prioritizes tracking trajectory 1 more closely in order to minimize the expected cost. This behavior is consistent with the probabilistic weighting in the objective function and confirms that the optimization correctly balances the likelihood of each scenario.

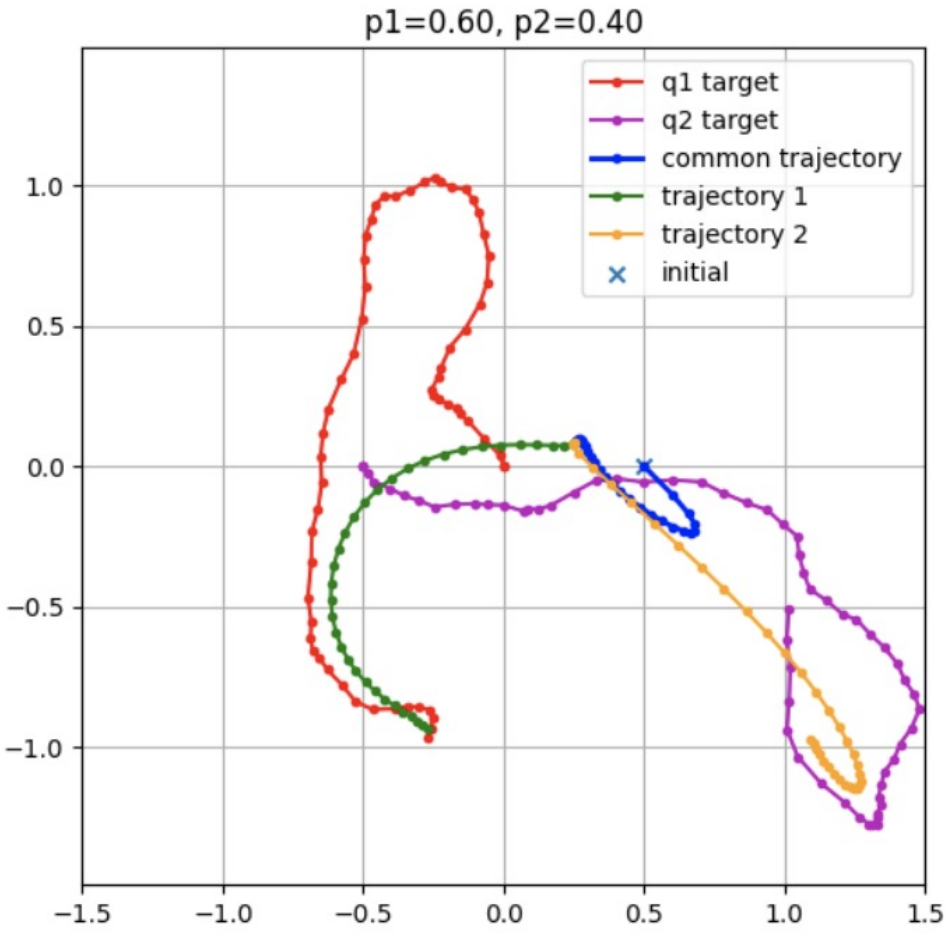


Figure 8: Optimized trajectories for $p_1 = 0.6$ and $p_2 = 0.4$.

8 Task 8 [Theorical task]

In this task, we want to study the convexity of the following cost function:

$$f(c, R) = (||c - x_n||_2 - R)^2$$

To demonstrate that the cost function of the problem is not convex, we can follow two paths. First, we could break down the function into simpler functions and try to use the properties of convex functions, or, the path we followed, find a counterexample that proves the non-convexity of the function.

The Jensen inequality states that, for a function f, two points x, y and for an arbitrary value $\alpha \in [0, 1]$, if the following inequality is satisfied, the function can be considered convex.

$$f\big((1-\alpha)x + \alpha y\big) \leq (1-\alpha)f(x) + \alpha f(y)$$

Basically, this tells us that the convex transformation of a mean is less than or equal to the mean applied after a convex transformation.

For our cost function, $c \in \Re^2$ is the centre of the circle, $R \in \Re$ is its radius, and x_n are the fixed data of the problem. Thus, we will have one fixed variable x_n and two variables, c and R, that can be freely chosen

Let us then consider the point $(0, 0)$ as our x_1. For the centres, we will choose the points $c_1 = (1, 0)$ and $c_2 = (0, 1)$, and the radius of the circle will be equal to 1 in both cases, $R_1 = R_2 = 1$. For the value of α, we choose 0.1. Thus,

$$f(c_1, R_1) = (||c_1 - x_1||_2 - R_1)^2 = (||(1, 0) - (0, 0)||_2 - 1)^2 = (1 - 1)^2 = 0$$
$$f(c_2, R_2) = (||c_2 - x_1||_2 - R_2)^2 = (||(0, 1) - (0, 0)||_2 - 1)^2 = (1 - 1)^2 = 0$$
$$f\big((1-\alpha)(c_1, R_1) + \alpha(c_2, R_2)\big) \leq (1-\alpha)f(c_1, R_1) + \alpha f(c_2, R_2) \Leftrightarrow$$
$$\Leftrightarrow f\big((1 - 0.1)((1, 0), 1) + 0.1((0, 1), 1)\big) \leq 0 \Leftrightarrow f\big(((0.9, 0), 0.9) + ((0, 0.1), 0.1)\big) \leq 0 \Leftrightarrow$$
$$\Leftrightarrow f\big((0.1, 0.9), 1\big) \leq 0 \Leftrightarrow (||(0.1, 0.9) - (0, 0)||_2 - 1)^2 \leq 0 \Leftrightarrow$$
$$\Leftrightarrow (\sqrt{0.1^2 + 0.9^2} - 1)^2 \leq 0 \Leftrightarrow 0.008923 \leq 0$$

Now, as we can see, inequality does not hold, so we have proven that our cost function and the problem are not convex.

9 Task 9 [Numerical task]

In this task, a sub-optimal method will be used in order to fit a set of noisy data $x_n \in \mathbb{R}^2$ to a circle. In this way, the optimization variables will be the radius of the circle $R \in \mathbb{R}$ and the coordinates of the center $c \in \mathbb{R}^2$. The method is considered sub-optimal because it does not attempt to directly optimize the cost function of the problem (seen in Task 8), which is problematic since it is not convex, as demonstrated in the past task. Instead, it is assumed that the data are not noisy and therefore behave exactly as if they fit perfectly on a circle (3).

$$\|x_n - c\|_2^2 = R^2, \quad n = 1, \ldots, N. \tag{3}$$

By expanding this equality, (7), where, in order to simplify the system, the variable $y = \|c\|_2^2 - R^2$ was introduced.

We start by expanding the squared Euclidean distance:

$$\|x_n - c\|_2^2 = (x_n - c)^\top (x_n - c) = \|x_n\|_2^2 - 2x_n^\top c + \|c\|_2^2. \tag{4}$$

Rearranging the terms gives:

$$\|x_n\|_2^2 - 2x_n^\top c + \|c\|_2^2 = R^2 \;\; \Rightarrow \;\; -2x_n^\top c + \left(\|c\|_2^2 - R^2\right) = -\|x_n\|_2^2. \tag{5}$$

To simplify the notation, we define the auxiliary variable:

$$y\|c\|_2^2 - R^2 \;\; \Rightarrow \;\; y - 2x_n^\top c = -\|x_n\|_2^2. \tag{6}$$

Stacking all N equations together leads to the compact matrix form:

$$\underbrace{\begin{bmatrix} 1 & -2x_1^\top \\ 1 & -2x_2^\top \\ \vdots & \vdots \\ 1 & -2x_N^\top \end{bmatrix}}_{A} \begin{bmatrix} y \\ c \end{bmatrix} = \underbrace{\begin{bmatrix} -\|x_1\|_2^2 \\ -\|x_2\|_2^2 \\ \vdots \\ -\|x_N\|_2^2 \end{bmatrix}}_{b}. \tag{7}$$

Finally, the non linear scalar relation between y, c, and R is:

$$y = \|c\|_2^2 - R^2. \tag{8}$$

System (7) can be solved using Least Squares (LS), since in the presence of noisy data the linear system $Ax = b$ may not admit an exact solution. In this case, LS provides the sub-optimal approximation by minimizing the squared error,

$$\hat{x} = \arg\min_x \|Ax - b\|_2^2, \tag{9}$$

thus ensuring a consistent estimate of the circle parameters. This optimization problem can be solved in several ways, including through a closed-form solution (theoretical) or by using an iterative algorithm such as Gradient Descent (Algorithm). In this case, the solution is obtained with the `numpy.linalg.lstsq` function, which implements a stable variant of the closed-form expression $(A^\top A)^{-1} A^\top b$ by computing the Moore–Penrose pseudoinverse via SVD. The reason for using this function directly is that the details of how this optimization problem is solved are not relevant for the course, as it is considered an atomic operation.

In this way, the method detailed above was applied using the script "Task9_final.py". The procedure was straightforward, consisting of the computation of the matrices A and b, the solution of the LS problem through the `numpy.linalg.lstsq` function, and the construction of the image showing the solution obtained for the circle with the parameters (c_{ls}, R_{ls}).

Thus, by applying the method, the following parameters are obtained:

1. Center of circle: $c_{ls}^* = (-0.44638224, 1.50461084)$;
2. Radius of the circle: $R_{ls}^* = 0.59304295$.

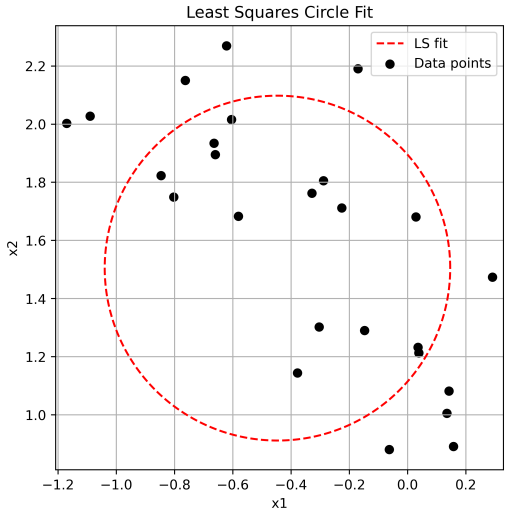


Figure 9: Least Squares circle fit obtained using `numpy.linalg.lstsq`, showing the fitted circle (c_{ls}, R_{ls}) and the original data points.

This result is satisfactory. However, since the noise present in the data was not taken into account, some divergence from the optimal solution will occur in this context. A better solution can be obtained through methods such as the one applied in Task 10, which takes into account the non-convex nature of the problem.

10 Task 10 [Numerical task]

In this task, the Levenberg–Marquardt (LM) algorithm will be implemented in order to fit a set of noisy data points $x_p \in \mathbb{R}^2$ to a circle. The optimization variables are the radius of the circle $R \in \mathbb{R}$ and the coordinates of the center $c \in \mathbb{R}^2$. Unlike the method used in Task 9, with this algorithm the non-convex problem is addressed directly, albeit with a twist. The basic principle of this method is the reduction to a least squares problem, where the functions can be linearized through a first-order Taylor expansion around the current iterate of the optimization variables. The initial estimate for the optimization variables is taken as the solution obtained in the previous task, $x(c_{ls}^*, R_{ls}^*)$.

In the minimization process, since the solution corresponding to the minimum of the local quadratic approximation (squared linearized residuals) may otherwise diverge to distant values, an ℓ_2 regularization term is introduced, as commonly done in the literature, in order to ensure that the iterative error does not become excessively large. It is important to note that the inclusion of this regularization term preserves the least squares nature of the cost function.

In order to apply the Levenberg-Marquardt (LM) algorithm, several preparatory steps are necessary. The first step involves defining the residual function f_p, where p denotes the index of the data point being linearized. Once this residual is specified, the cost function naturally follows as the minimization of the sum of squared residuals. The residual function is given by

$$f_p(c, R) = \|c - x_p\|_2 - R, \tag{10}$$

where $c \in \mathbb{R}^2$ represents the circle center, $R \in \mathbb{R}$ its radius, and x_p the pth data point. Using the residuals defined in (10), the corresponding cost function is expressed as

$$\min_{c \in \mathbb{R}^2,\, R \in \mathbb{R}} F(c, R) = \frac{1}{2} \sum_{p=1}^{P} f_p(c, R)^2. \tag{11}$$

To apply the LM algorithm, the residual function must be linearized around the current estimate. This is achieved through a first-order Taylor expansion, which requires computing the Jacobian of the residuals at each iteration. The linearized form of the residual function can be expressed as

$$f_p(c_k + \Delta c, R_k + \Delta R) \;\approx\; f_p(c_k, R_k) + J_p(c_k, R_k)\, \Delta x, \tag{12}$$

where

$$\Delta x = \begin{bmatrix} \Delta c \\ \Delta R \end{bmatrix}, \quad J_p(c_k, R_k) = \begin{bmatrix} \dfrac{c_{k,1} - x_{p,1}}{\|c_k - x_p\|_2} & \dfrac{c_{k,2} - x_{p,2}}{\|c_k - x_p\|_2} & -1 \end{bmatrix}.$$

Here, $J_p(c_k, R_k)$ denotes the Jacobian of the pth residual with respect to the parameters (c, R), evaluated at the current estimate (c_k, R_k).

To evaluate the stopping condition in the LM algorithm, the total Jacobian and its associated gradient norm must be computed. For a given iteration k, the total Jacobian J_k is obtained by stacking the individual Jacobians corresponding to each data point, as follows:

$$J_k = \begin{bmatrix} \frac{c_{k,1} - x_{1,1}}{\|c_k - x_1\|_2} & \frac{c_{k,2} - x_{1,2}}{\|c_k - x_1\|_2} & -1 \\ \frac{c_{k,1} - x_{2,1}}{\|c_k - x_2\|_2} & \frac{c_{k,2} - x_{2,2}}{\|c_k - x_2\|_2} & -1 \\ \vdots & \vdots & \vdots \\ \frac{c_{k,1} - x_{P,1}}{\|c_k - x_P\|_2} & \frac{c_{k,2} - x_{P,2}}{\|c_k - x_P\|_2} & -1 \end{bmatrix} \in \mathbb{R}^{P \times 3}. \tag{13}$$

Using this total Jacobian, the gradient of the cost function is computed as

$$\nabla F(c_k, R_k) = J_k^\top(c_k, R_k) f(c_k, R_k),$$

and the iteration is terminated once the gradient norm falls below the predefined ε:

$$\|\nabla F(c_k, R_k)\|_2 = \|J_k^\top(c_k, R_k) f(c_k, R_k)\|_2 < \varepsilon. \tag{14}$$

At each iteration, the parameter increment Δx is computed by solving the following linear system (implemented in code using `np.linalg.lstsq`):

$$\min_{x \in \mathbb{R}^3} \left\| \begin{bmatrix} J_k \\ \sqrt{\lambda_k}\, I \end{bmatrix} x \; - \; \begin{bmatrix} J_k x_k - f(x_k) \\ \sqrt{\lambda_k}\, x_k \end{bmatrix} \right\|_2^2. \tag{15}$$

Once Δx is obtained, the parameters are updated through a simple additive rule:

$$\begin{bmatrix} c_{k+1} \\ R_{k+1} \end{bmatrix} = \begin{bmatrix} c_k \\ R_k \end{bmatrix} + \Delta x, \tag{16}$$

where the damping parameter λ_k is adapted at each iteration depending on whether the cost function decreases.

With this setup, the method was implemented in code. The first phase consisted of applying the algorithm from Task 9 in order to obtain the initial estimate. Then, the total Jacobian was computed (evaluating its norm), together with the residual function at each data point, and the matrices A and b were assembled and evaluated. After that, these matrices from the linear system were passed to the `numpy.linalg` solver, and the iterate was updated.

To better understand the evolution of the Jacobian, namely the convergence of the method, a plot of the iteration number k versus the norm of the Jacobian was generated, the output is shown in Figure 10. In addition, a comparison was made between the circle obtained with LS and LM, which is shown in Figure 11. Applying the method yields the following parameters:

1. Center of the circle: $c_{lm}^* = (-0.8792,\ 1.104)$;
2. Radius of the circle: $R_{lm}^* = 0.9086$.

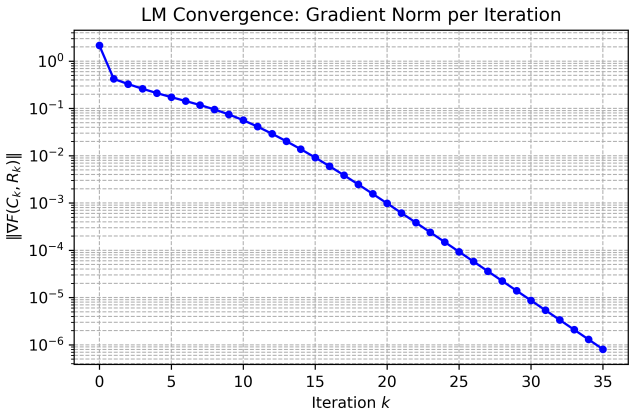


Figure 10: Evolution of the Jacobian norm with respect to the iteration number k.

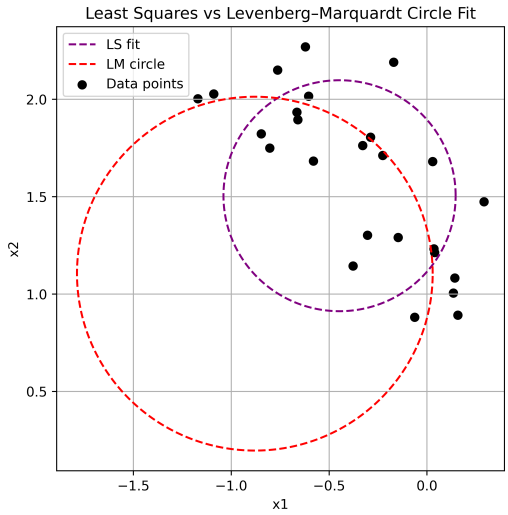


Figure 11: Comparison between the circle obtained with LS and LM.

By analyzing Figure 10, it can be verified that the algorithm converged, yielding a stable solution that met our objective. However, when examining Figure 11, a significant difference can be observed between the circles generated by the LS and LM methods. This discrepancy may be due to the Levenberg-Marquardt algorithm bypassing a local minimum, suggesting that a more stringent regularization regime could produce a result more consistent with the LS method. Alternatively, this outcome may stem from the intrinsic properties of the dataset, as different point configurations (for example, the data set of circle 1) showed greater similarity between the solutions produced by both methods.

11 Task 11 [Numerical task]

A circle with center = $(1.2214, 5.0494)$ and radius = 4.0836 was obtained using Federated Learning. For comparison, we also plotted a circle based on the LS method based on all points from the four datasets combined. From Figure 12, we can see that these two circles almost overlap, which is a very good sign. This confirms that the ALM-BCD algorithm was correctly implemented and that the distributed approach reproduces the optimal solution with high precision, without requiring data sharing between agents.

Furthermore, the graph in Figure 13 and Table 2 show a rapid reduction in constraint violation, as it went from around 0.059 to less than 0.01 in just four ALM iterations. This demonstrated the effectiveness of the consensus mechanism. We can also observe from Table 2 that the number of BCD iterations decreases dramatically throughout the process, indicating that the system converges progressively more efficiently as agents approach consensus.

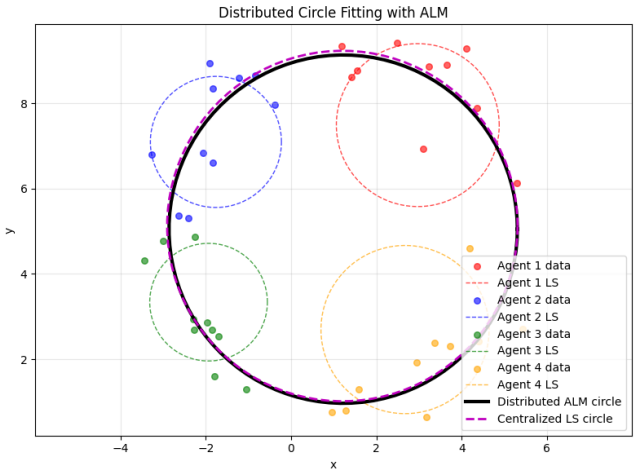


Figure 12: The four local datasets and the corresponding four local circles fitted by the four agents on their own, plus the circle fitted by Federated Learning and the one fitted by LS.

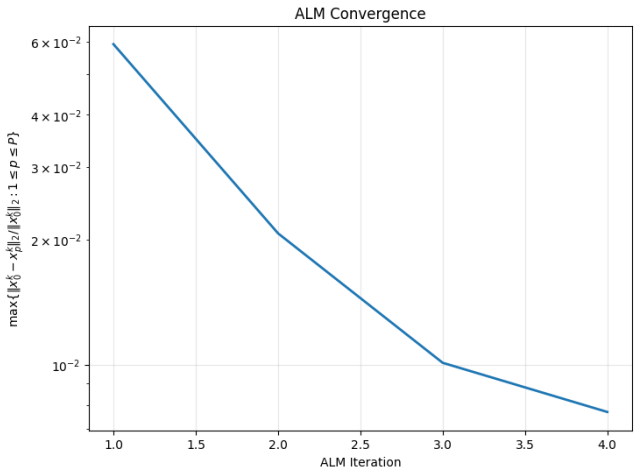


Figure 13: Maximum constraint violation across ALM iterations, for the case c = 100

Agent	# Points	(x_c, y_c)	R
1	10	(2.97, 7.49)	1.91
2	10	(-1.76, 7.09)	1.54
3	10	(-1.93, 3.34)	1.38
4	10	(2.67, 2.70)	1.97

Table 1: Least-squares circle fitting performed locally by each agent.

ALM Iteration	BCD Iterations	$\frac{\|x_0^k - x_p^k\|_2}{\|x_0^k\|_2}$	(x_c, y_c)	R
1	170	0.0592		
2	3	0.0207		
3	3	0.0101		
4	2	0.0077	(1.2214, 5.0494)	4.0836

Table 2: ALM–BCD convergence summary and constraint violation per iteration.