UNIVERSITÀ DI BOLOGNA

School of Engineering

Master Degree in Automation Engineering

Distributed Autonomous Systems

Distributed algorithms: classification and multi-robot coordination

Professors:
Giuseppe Notarstefano
Ivano Notarnicola

Students:
Mattia Guazzaloca
Daniele Paccusse

Academic year 2023/2024

Abstract

The work addresses two significant problems that can be solved enforcing distributed algorithms: distributed classification and multi-robot coordination. The first task involves developing a distributed algorithm for logistic regression classification, where the goal is to determine the parameters of a nonlinear separating function. We investigate the performance of this algorithm across different graph topologies, including path, cycle, star, and complete graphs, to highlight their impact on convergence rates and classification accuracy. The second task focuses on formulating an aggregative optimization problem to achieve coordinated movement of multiple robots, ensuring they maintain a tight formation while moving towards their respective targets. This problem is addressed using a distributed control algorithm tested also in ROS2. Numerical simulations validate the proposed methods, demonstrating effective classification with high accuracy and robust coordination of robot formations under various experimental setups. These results underscore the potential of advanced distributed algorithms in enhancing the efficiency and reliability.

Contents

Introduction

Motivations

The primary motivation behind this research is to show the efficiency of distributed algorithms in classification problems and multi-robot coordination.

In distributed systems, the ability to perform classification tasks without centralized control offers numerous benefits. In particular, they offer advantages in handling large-scale datasets efficiently by distributing computation across multiple nodes. Scalability is another key benefit, as distributed approaches can expand to accommodate growing dataset sizes, making them particularly well-suited for big data scenarios. The use of logistic regression in a distributed manner aims to leverage these advantages while maintaining high accuracy and efficiency in classification tasks.

For multi-robot systems, the challenge lies in enabling a group of robots to operate cohesively towards common objectives while also achieving individual goals. Aggregative optimization plays a crucial role in ensuring that robots can maintain formations, avoid collisions, and reach their respective targets.

Contributions

The work demonstrates how a classification problem can be effectively addressed without centralized control. An implementation of the Gradient Tracking algorithm is presented, showing the convergence of the algorithm in the tracking of the real gradient of the cost function.

By comparing the results of gradient tracking with a centralized method, the study provides insights into the accuracy of the distributed approach.

The work explores also a way to control multiple robots using a distributed algorithm. Unlike centralized methods where one controller makes all decisions, this approach spreads decision-making across all robots. By solving optimization problems and using a ROS2 package, the research shows how robots can stick together in formations and stay near their targets on their own. The algorithm has been tested in different situations.

The report unfolds as follows. In chapter 1, the classification problem is addressed, showing the results with both methods: centralised and distributed. In chapter 2, the

multi-robot coordination problem is solved, where in the section 2.3 it is shown how the robot can efficiently enter inside a corridor avoiding collisions, to reach the targets.

Chapter 1

Task 1: Distributed Classification via Logistic Regression

In this task, a classification problem is addressed by determining the parameters of a nonlinear separating function using a distributed algorithm that minimizes the Logistic Regression Function.

Initially, the distributed algorithm is implemented and tested by minimizing a simple quadratic cost function to ensure its functionality. This verification step employs Gradient Tracking.

Before applying Gradient Tracking to the main classification problem, a centralised method is used to classify the points.

1.1 Distributed Optimization

The quadratic cost function $\ell(z)$ is defined as:

$$\ell(z) = \sum_{i=1}^{N} \ell_i(z)$$

Where $\ell_i(z)$ and its gradient are defined as:

$$\ell_i(z) = \frac{1}{2} z_i^T Q_i z_i + R_i z_i \qquad i = 1, \ldots, n \tag{1.1}$$

$$\nabla l_i(z) = Q_i z_i + R_i \qquad i = 1, \ldots, n \tag{1.2}$$

where:

- $Q_i \in \mathbb{R}^{2\times 2}$ is the identity matrix
- $R_i \in \mathbb{R}^d$ is the i-th d-dimensional vector of zeros
- $z \in \mathbb{R}^d$ is the d-dimensional input vector.

- $\nabla \ell(z) \in \mathbb{R}^d$

The optimal solution z^* is found by setting the gradient equal to zero and is given by:

$$z^* = -\left(\sum_{i=1}^{n} R_i\right)\left(\sum_{i=1}^{n} Q_i\right)^{-1}$$

The cost at the optimal solution z^* is:

$$f(z^*) = \frac{1}{2}(z^*)^T\left(\sum_{i=1}^{n} Q_i\right)z^* + \left(\sum_{i=1}^{n} R_i\right)z^*$$

This is useful to make a comparison with the implemented Gradient Tracking to understand if it's working properly.

The Gradient Tracking is implemented as:

$$z_i^{k+1} = \sum_{j \in N_i} a_{ij} z_j^k - \alpha s_i^k \qquad z_i^0 \in \mathbb{R}^2$$

$$s_i^{k+1} = \sum_{j \in N_i} a_{ij} s_j^k + \nabla \ell_i\left(z_i^{k+1}\right) - \nabla \ell_i\left(z_i^k\right) \quad s_i^0 = \nabla \ell_i\left(z_i^0\right)$$

For every node, at each iteration, the gradients are summed up and the norm is evaluated to determine the gradient magnitude. If the gradient magnitude is less than $1e-6$, then the consensus is reached.

Different weighted graph patterns are tested. All the plots are in semi-logarithmic scale along x, to highlight the variations.

1.1.1 Path graph

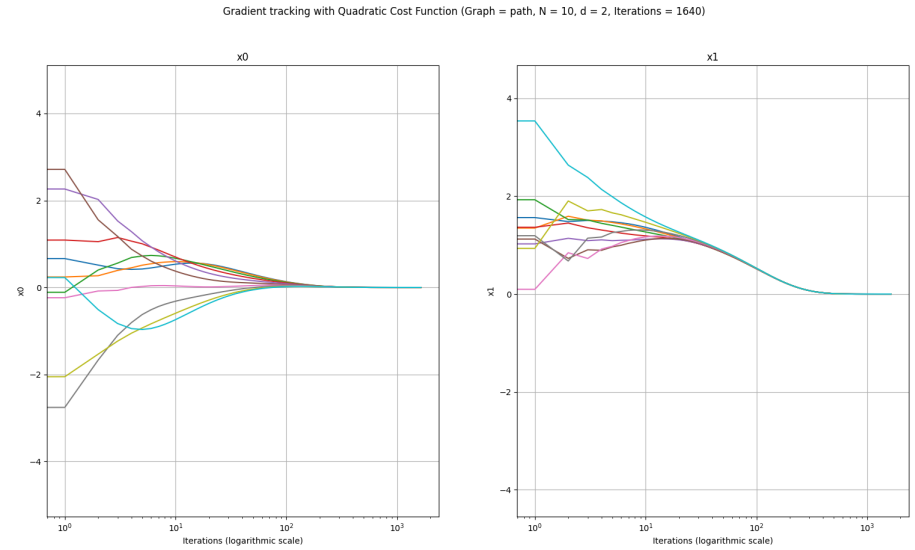

Figure 1.1: Evolution of the states for Path graph (semi-logarithmic scale)

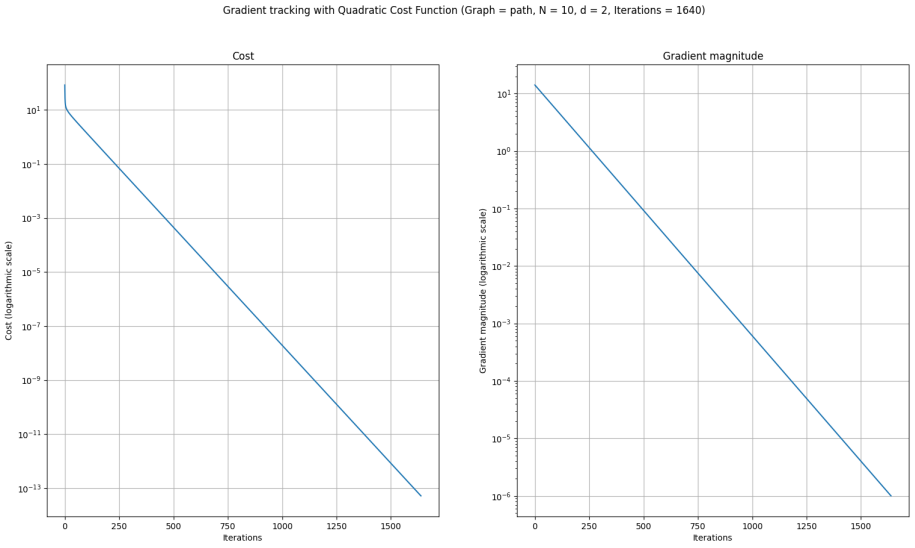

Figure 1.2: Cost and gradient magnitude Path graph

1.1.2 Cycle graph

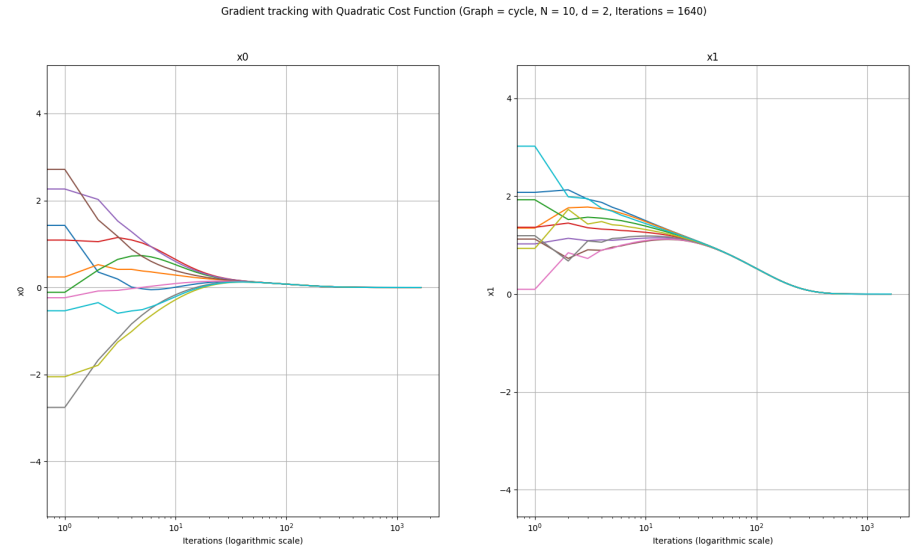

Figure 1.3: Evolution of the states Cycle graph

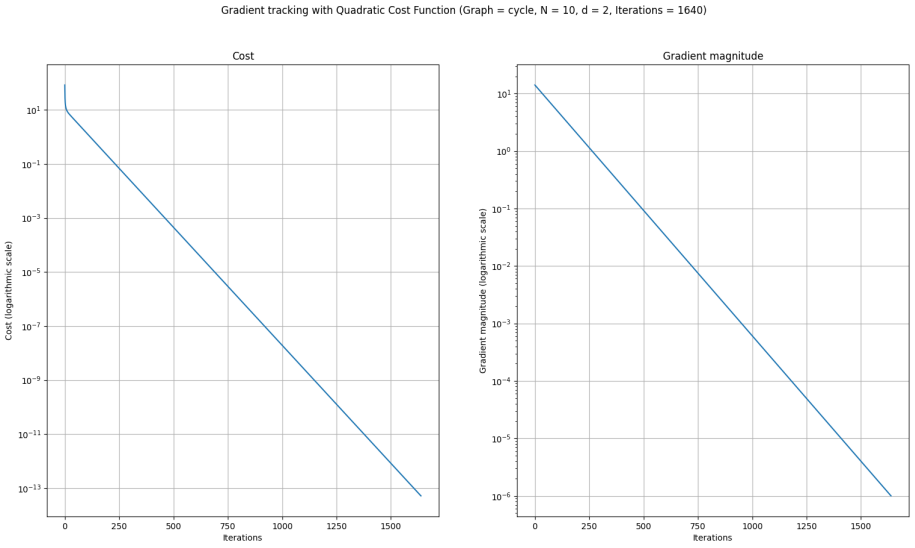

Figure 1.4: Cost and gradient magnitude Cycle graph

1.1.3 Star graph

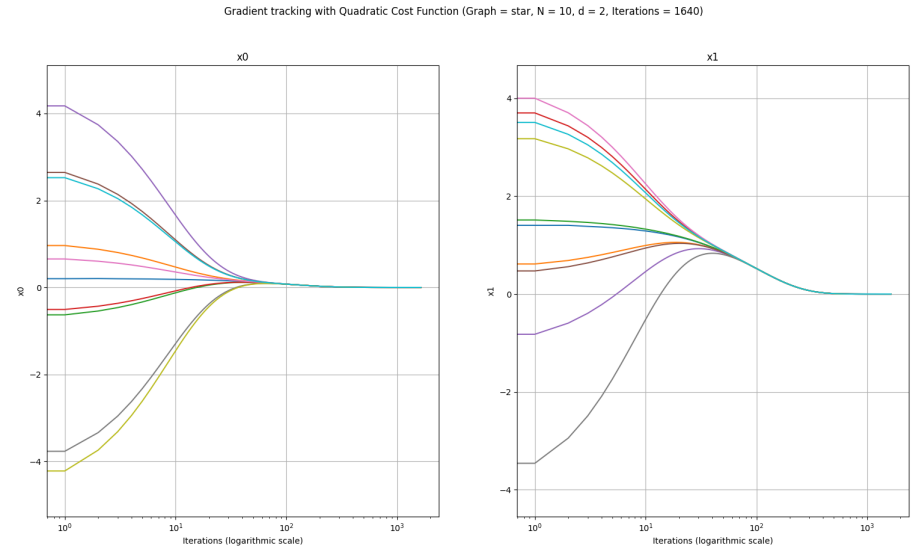

Figure 1.5: Evolution of the states Star graph

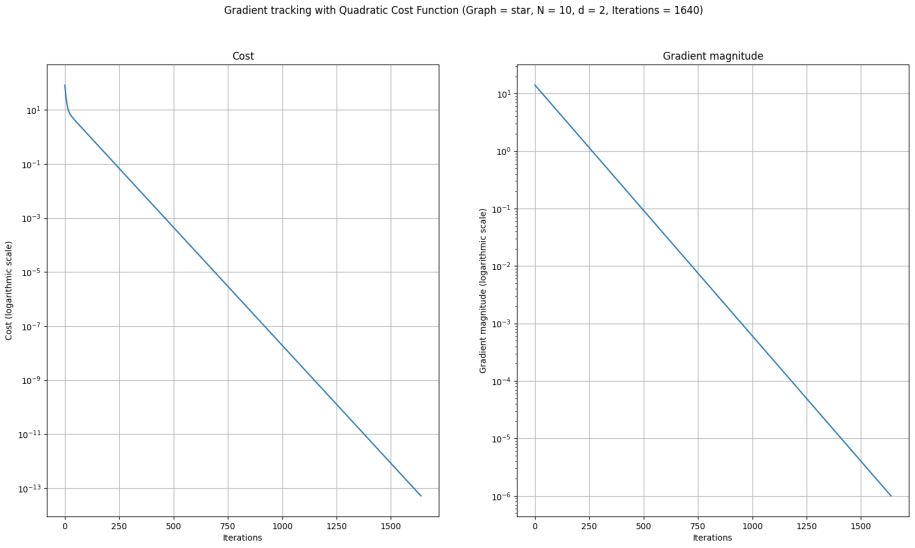

Figure 1.6: Cost and gradient magnitude Star graph

1.1.4 Complete graph

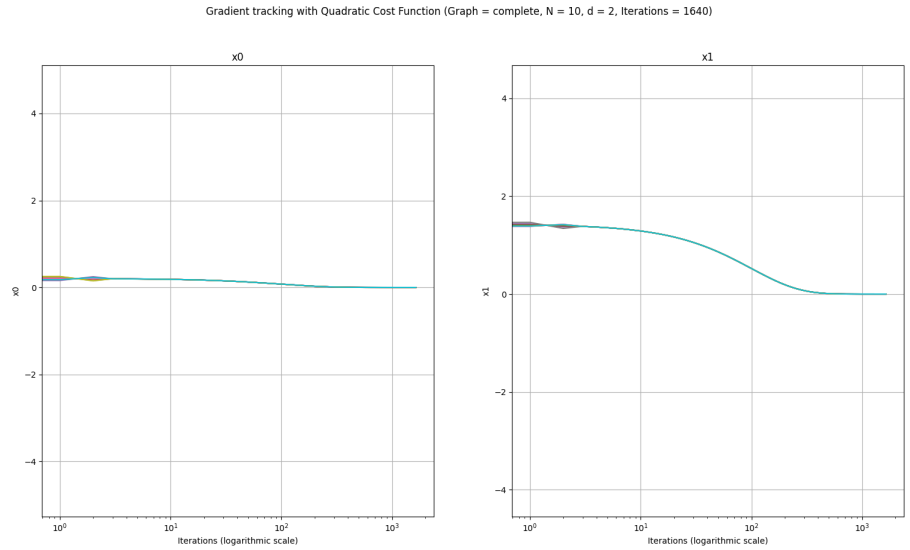

Figure 1.7: Evolution of the states Complete graph

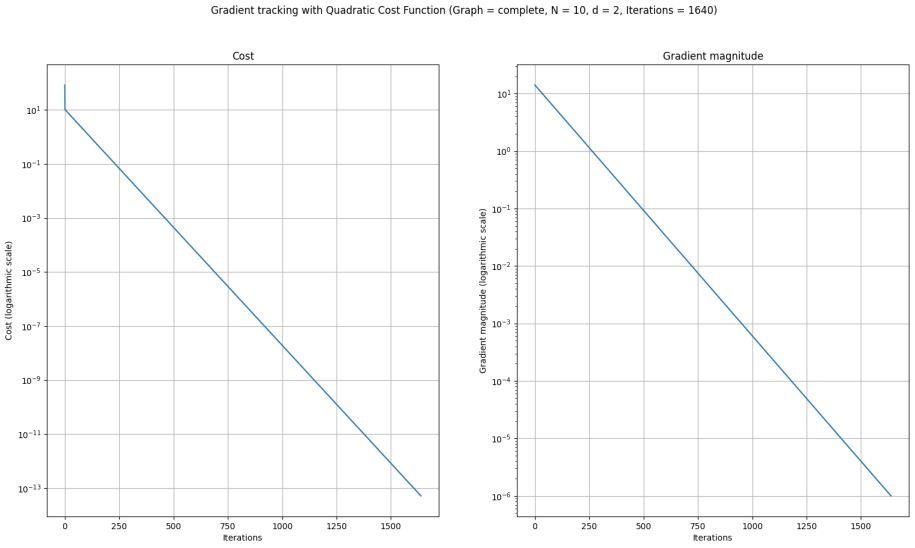

Figure 1.8: Cost and gradient magnitude Complete graph

1.1.5 Comments

In a complete graph, every node is directly connected to every other node. This structure exhibits the fastest convergence because information can be exchanged directly between all pairs of nodes in very few iterations. This allows for the most efficient averaging of information, as each node can immediately access updates from all other nodes.

In a cycle graph, each node is connected to exactly two other nodes, forming a closed loop. This structure converges slower than the complete graph but faster than the star and path graphs. Although each node can only communicate with its two neighbors in each iteration, the cyclic nature allows information to circulate throughout the network relatively efficiently.

A star graph has one central node connected to all other nodes, which are not connected to each other. The convergence rate of a star graph is slower than the cycle graph but faster than the path graph. The central node plays a crucial role by acting as a hub for information exchange.

In a path graph, each node (except the endpoints) is connected to exactly two other nodes, forming a linear chain. This structure shows the slowest convergence. Information must propagate sequentially from one end of the path to the other, resulting in a high number of iterations for updates to move throughout the entire network.

Higher connectivity typically results in faster convergence because information can be exchanged more freely and quickly between nodes.

1.2 Logistic regression classification: centralised and distributed

To classify the points using a centralised algorithm, a labelled dataset is needed. The labels are assigned using the equation of an ellipse as separating function:

$$dy^2 + by + cx^2 + ax = e^2$$

This has been rewritten as:

$$\left\{ x \in \mathbb{R}^2 \mid w^\top \varphi(\mathbf{x})) + b = 0 \right\},$$

where:

$$\omega = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}, \qquad b = -e^2, \qquad \mathbf{x} = \begin{bmatrix} x \\ y \\ x^2 \\ y^2 \end{bmatrix}$$

The label p is assigned to each point based on the following criteria:

$$\begin{cases} p = 1, & \text{if } w^\top \varphi(\mathcal{D}^m) + b \geq 0, \\ p = -1, & \text{if } w^\top \varphi(\mathcal{D}^m) + b < 0. \end{cases}$$

The total number of points is randomly assigned at runtime and it can be between 500 and 1000. Varying the parameters, different dataset patterns are obtained, such as an horizontal ellipse, a vertical one and a hyperbola. For each dataset, a centralised method and a distributed one has been applied to solve the classification problem. The centralised method simply updates the parameters as:

$$\theta^{(k)} = \theta^{(k-1)} - \alpha \sum_{j=1}^{M} \nabla l_j$$

Where M is the total number of points in the dataset.

The distributed method uses the gradient tracking algorithm described in section 1.1. The dataset has been split into N parts and the nodes are interconnected through a cycle graph.

Recalling that the logistic regression cost is defined as:

$$L = \sum_{j=1}^{\mathcal{M}} \log\left(1 + \exp\left(-p^j\left(w^\top \varphi\left(\mathbf{x}\right) + b\right)\right)\right)$$

The gradient of this function is:

$$\nabla L = \begin{pmatrix} \frac{\partial L}{\partial a} = \sum_{j=1}^{N} -p_j x_j \frac{\exp(-p_j \cdot sep\_fn(\mathbf{x}_j))}{1+\exp(-p_j \cdot sep\_fn(\mathbf{x}_j))} \\ \frac{\partial L}{\partial b} = \sum_{j=1}^{N} -p_j y_j \frac{\exp(-p_j \cdot sep\_fn(\mathbf{x}_j))}{1+\exp(-p_j \cdot sep\_fn(\mathbf{x}_j))} \\ \frac{\partial L}{\partial c} = \sum_{j=1}^{N} -p_j x_j^2 \frac{\exp(-p_j \cdot sep\_fn(\mathbf{x}_j))}{1+\exp(-p_j \cdot sep\_fn(\mathbf{x}_j))} \\ \frac{\partial L}{\partial d} = \sum_{j=1}^{N} -p_j y_j^2 \frac{\exp(-p_j \cdot sep\_fn(\mathbf{x}_j))}{1+\exp(-p_j \cdot sep\_fn(\mathbf{x}_j))} \\ \frac{\partial L}{\partial b} = \sum_{j=1}^{N} -p_j \frac{\exp(-p_j \cdot sep\_fn(\mathbf{x}_j))}{1+\exp(-p_j \cdot sep\_fn(\mathbf{x}_j))} \end{pmatrix}$$

Where:

$$sep\_fn(x) = \mathbf{w} \cdot \phi(\mathbf{x}) + \text{bias}$$

In the next sections results are shown.

1.3 Classification with centralised method

1.3.1 Horizontal ellipse

Parameter	Value
Number of points	1000
Max iterations	1000
Stepsize	0.01
Classification error	0.1%

Table 1.1: Parameters of the Centralised Method

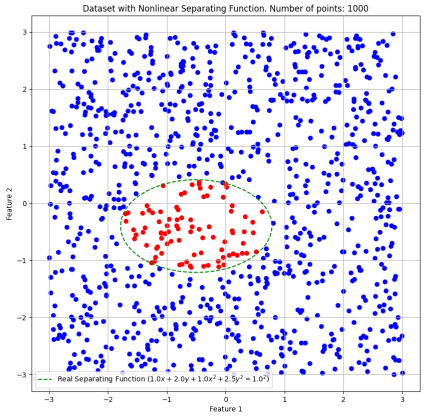

Figure 1.9: Horizontal ellipse dataset

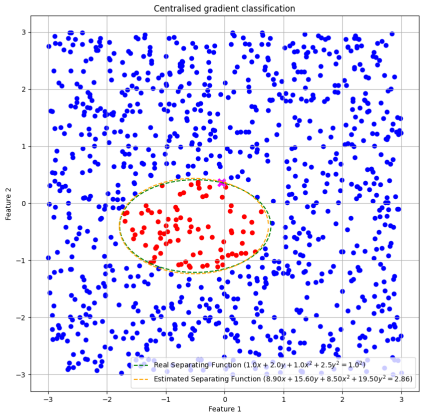

Figure 1.10: Result obtained

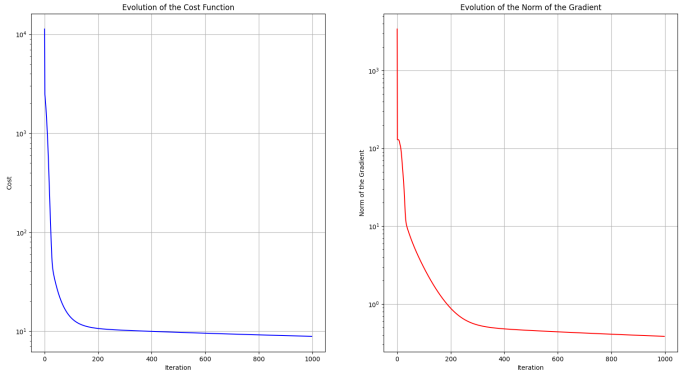

Figure 1.11: Cost and gradient magnitude

1.3.2 Vertical ellipse

Parameter	Value
Number of points	1000
Max iterations	1000
Stepsize	0.01
Classification error	0.2%

Table 1.2: Parameters of the Centralised Method

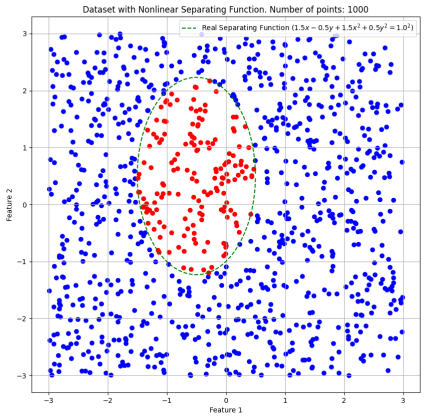

Figure 1.12: Vertical ellipse dataset

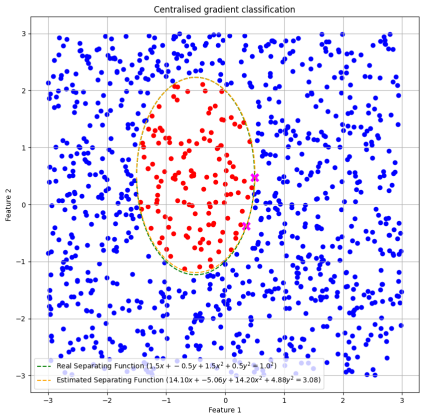

Figure 1.13: Result obtained

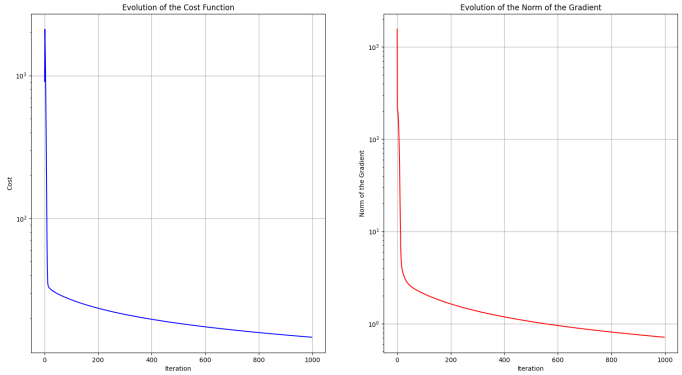

Figure 1.14: Cost and gradient magnitude (vertical ellipse)

1.3.3 Hyperbola

Parameter	Value
Number of points	1000
Max iterations	1000
Stepsize	0.01
Classification error	0.1%

Table 1.3: Parameters of the Centralised Method

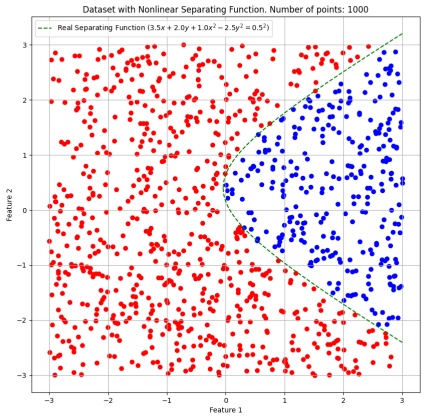

Figure 1.15: Hyperbola dataset

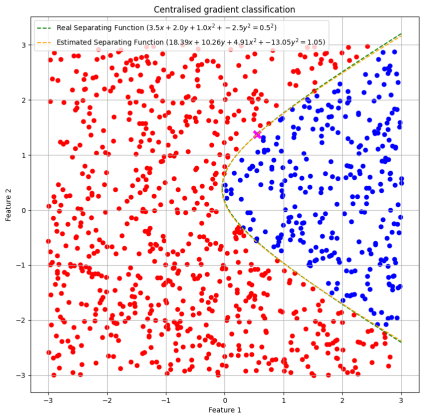

Figure 1.16: Result obtained

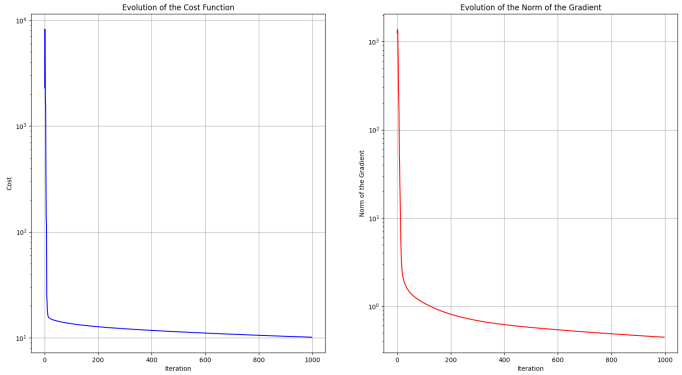

Figure 1.17: Cost and gradient magnitude (hyperbola)

1.4 Classification with Gradient Tracking

To show the convergence of the gradient tracking, for each node, the norm of the difference between the real gradient and its estimate s_i has been plotted.

1.4.1 Horizontal ellipse

Parameter	Value
Points	1000
Iterations	1500
Stepsize	0.01
Graph type	Cycle
Nodes	10
Error	0.4%

Table 1.4: Parameters

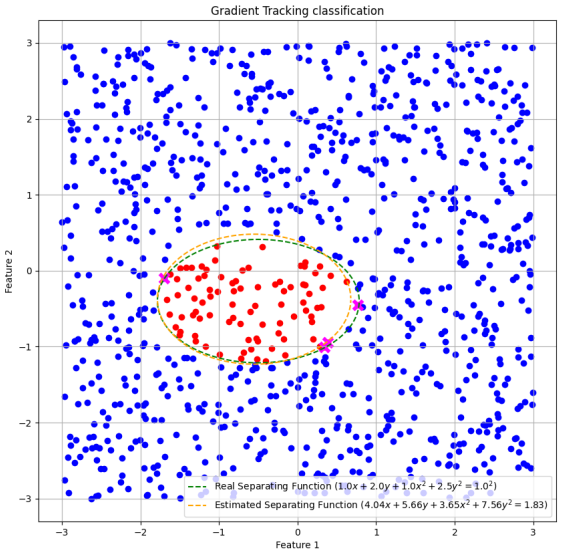

Figure 1.18: Results obtained

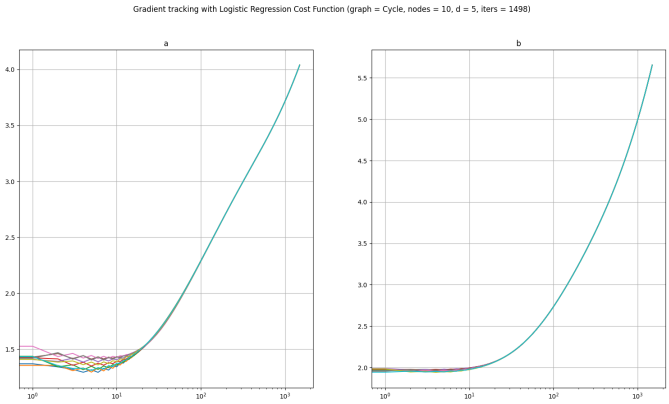

Figure 1.19: Evolution of parameters a and b

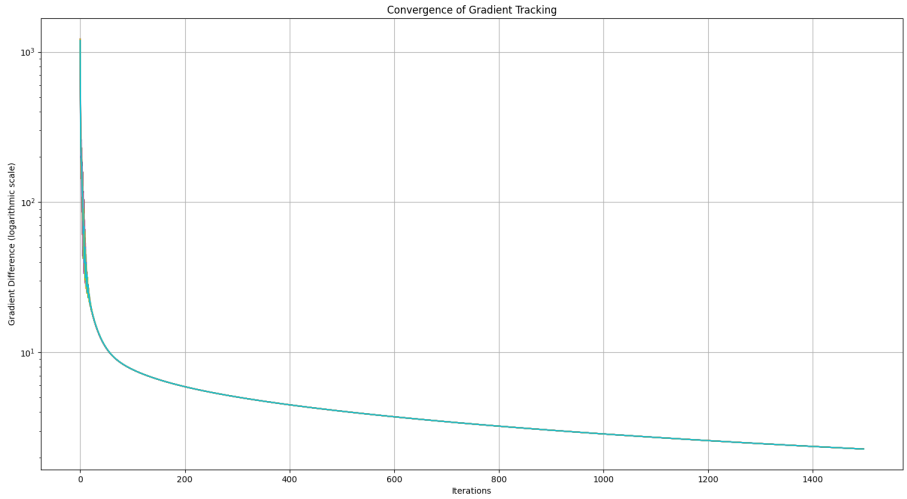

Figure 1.20: Convergence of Gradient tracking

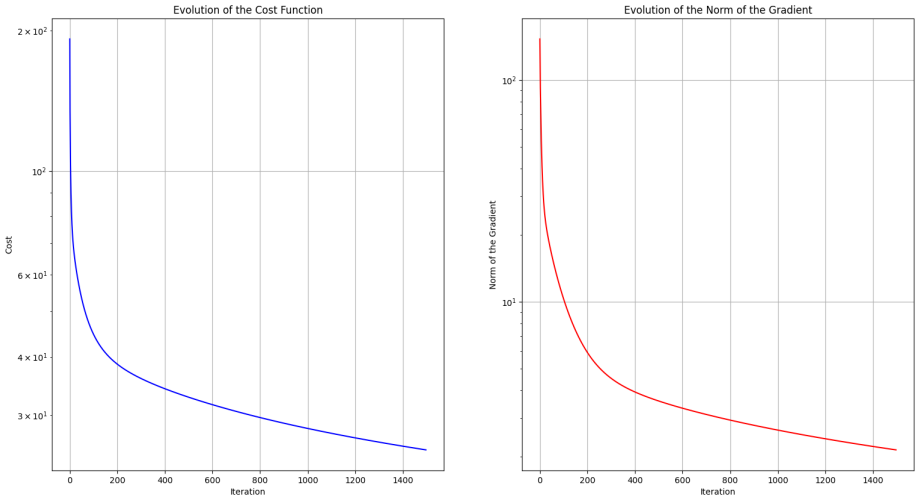

Figure 1.21: Cost and gradient magnitude with Gradient Tracking

1.4.2 Vertical ellipse

Parameter	Value
Points	1000
Iterations	3500
Stepsize	0.001
Graph type	Cycle
Nodes	10
Error	0.4%

Table 1.5: Parameters

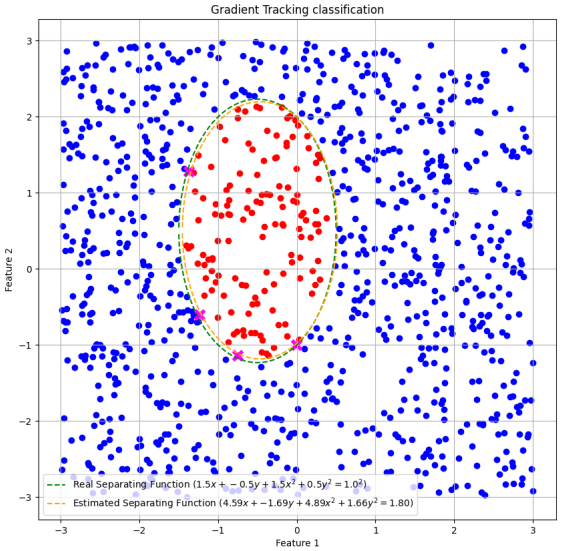

Figure 1.22: Results obtained

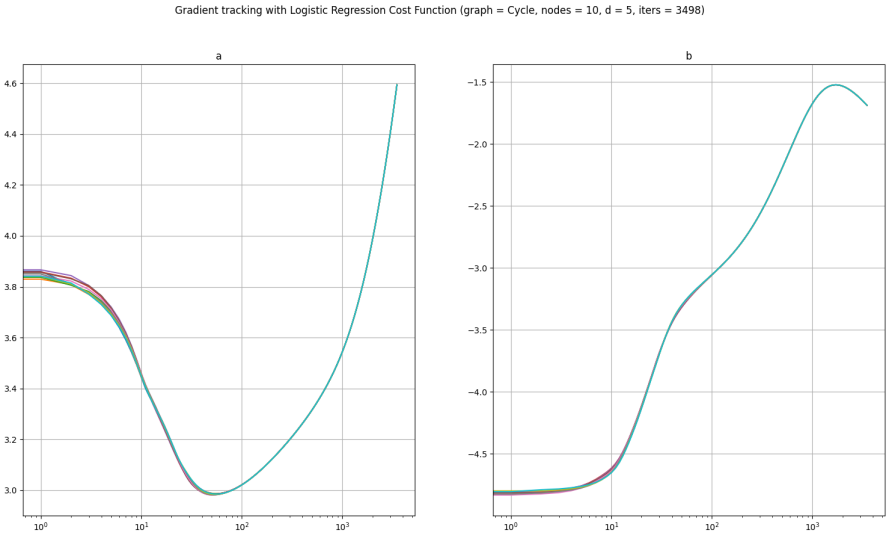

Figure 1.23: Evolution of parameters a and b (vertical ellipse)

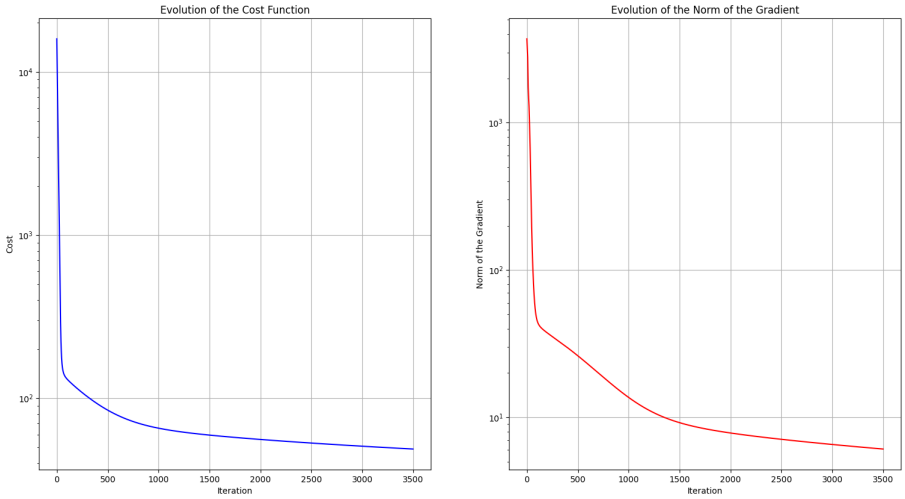

Figure 1.24: Cost and gradient magnitude for Gradient Tracking (vertical ellipse)

1.4.3 Hyperbola

Parameter	**Value**
Points	1000
Iterations	3500
Stepsize	0.005
Graph type	Cycle
Nodes	10
Error	0.1%

Figure 1.25: Parameters

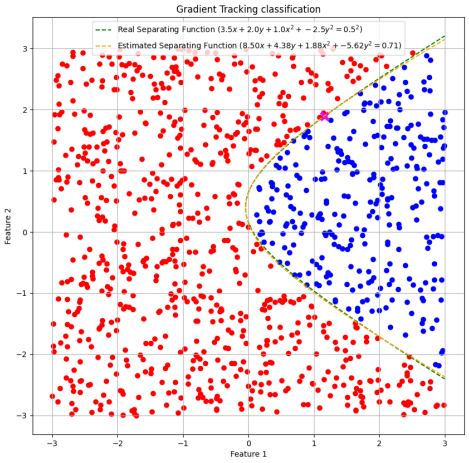

Figure 1.26: Result obtained

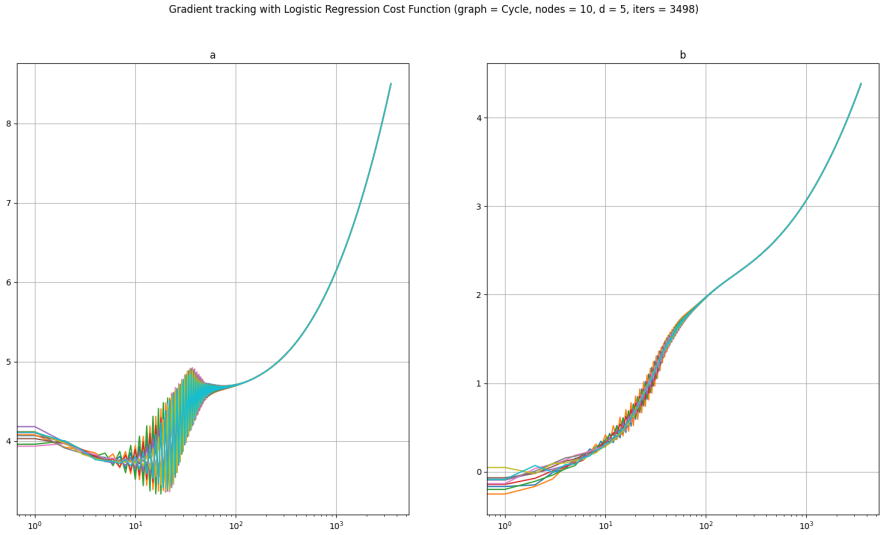

Figure 1.27: Evolution of a and b parameters (hyperbola)

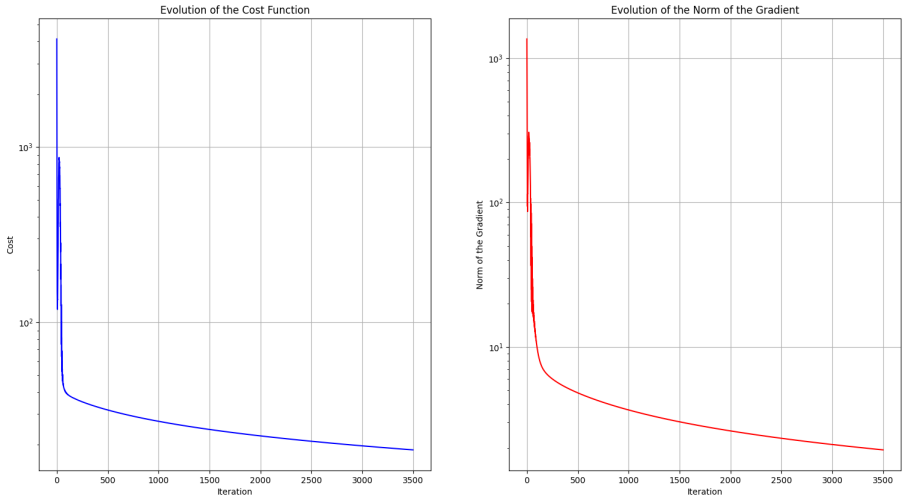

Figure 1.28: Cost and gradient magnitude (hyperbola)

Chapter 2

Task 2: Aggregative Optimization for Multi-Robot Systems

The aim of this task is to develop a distributed control algorithm that enables the robots to maintain a tight formation while simultaneously staying close to their respective targets.

To obtain this, an aggregative optimization problem is formulated, at first in a simple python script, then in a ROS2 package.

2.1 Problem Set-up

The aggregative optimization problem is formulated as:

$$\min_{z\in\mathbb{R}^2} \sum_{i=1}^{N} \ell_i\left(z_i, \sigma(z)\right) = \sum_{i=1}^{N} \gamma \cdot \|z_i - target_i\|^2 + \|z_i - \sigma(z)\|^2,$$
$$\sigma(z) \triangleq \frac{\sum_{i=1}^{N} z_i}{N}.$$

The partial derivative of the cost $\ell_i(z_i, \sigma(z))$ with respect to z_i is given by:

$$\nabla_{z_i} \ell_i(z_i, \sigma(z)) = 2 \cdot \gamma \cdot (z_i - \text{target}_i) + 2 \cdot (z_i - \sigma(z))$$

Similarly, the partial derivative of the cost $\ell_i(z_i, \sigma(z))$ with respect to $\sigma(z)$ is:

$$\nabla_{\sigma(z)} \ell_i(z_i, \sigma(z)) = 2 \cdot (z_i - \sigma(z))$$

The overall gradient is:

$$\nabla \ell_i(z_i, \sigma(z)) = \begin{bmatrix} \nabla_{z_i} \ell_i(z_i, \sigma(z)) \\ \nabla_{\sigma_i} \ell_i(z_i, \sigma(z)) \end{bmatrix} = \begin{bmatrix} 2 \cdot \gamma \cdot (z_i - target_i) + 2 \cdot (z_i - \sigma(z)) \\ 2 \cdot (z_i - \sigma(z)) \end{bmatrix}$$

Where:

- z_i is a vector that contains x and y position of a robot (node) i
- $target_i$ is a vector that defines the x and y coordinate of the target of robot i
- $\sigma(z)$ is the barycenter (aggregative variable)
- γ is balancing factor that adjusts the importance of the two competing objectives in the optimization problem.

The algorithm implemented to solve this optimization problem is the aggregative tracking distributed optimization algorithm:

$$
\begin{aligned}
z_i^{k+1} &= z_i^k - \alpha \left(\nabla_1 \ell_i \left(z_i^k, s_i^k \right) + \nabla \phi_i \left(z_i^k \right) v_i^k \right) && z_i^0 \in \mathbb{R}^{n_i} \\
s_i^{k+1} &= \sum_{j \in \mathcal{N}_i} a_{ij} s_j^k + \phi_i \left(z_i^{k+1} \right) - \phi_i \left(z_i^k \right) && s_i^0 = \phi_i \left(z_i^0 \right) \\
v_i^{k+1} &= \sum_{j \in \mathcal{N}_i} a_{ij} v_j^k + \nabla_2 \ell_i \left(z_i^{k+1}, s_i^{k+1} \right) - \nabla_2 \ell_i \left(z_i^k, s_i^k \right) \quad v_i^0 = \nabla_2 \ell_i \left(z_i^0, s_i^0 \right)
\end{aligned}
$$

As in section 1.1, the gradient magnitude is evaluated during the execution. To show the convergence of the algorithm, the norm of the difference between the quantities that must be tracked (the barycenter and $\sum_i^N \nabla_2 \ell_i(z_i, \sigma)$)) and their estimations of each node s_i and v_i are plotted.

Several experiments were conducted to demonstrate the implementation's behavior. The first one involved setting targets close to the robot, whereas in the second experiment, targets were placed further away. These scenarios were simulated across various γ values to illustrate their impact on the simulation results.

Results with $\gamma = 1$

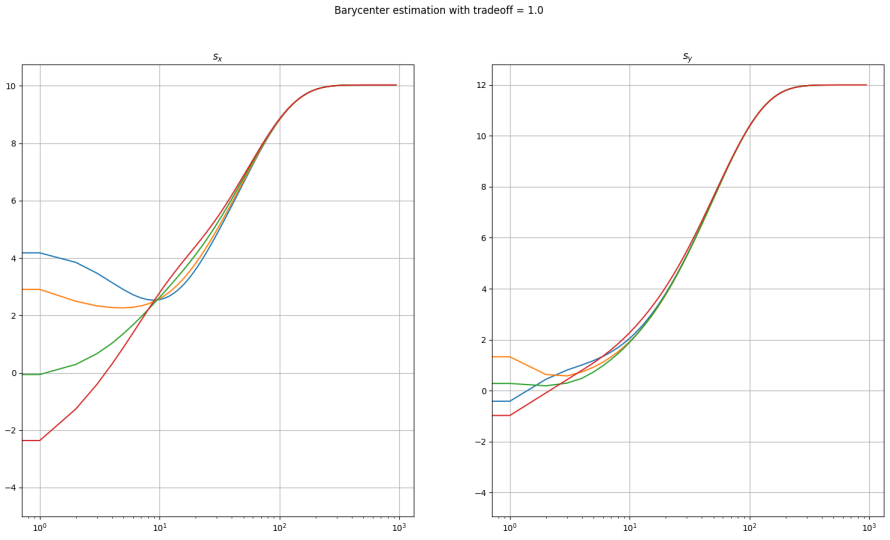

Figure 2.1: Evolution of barycenter estimation with $\gamma = 1$

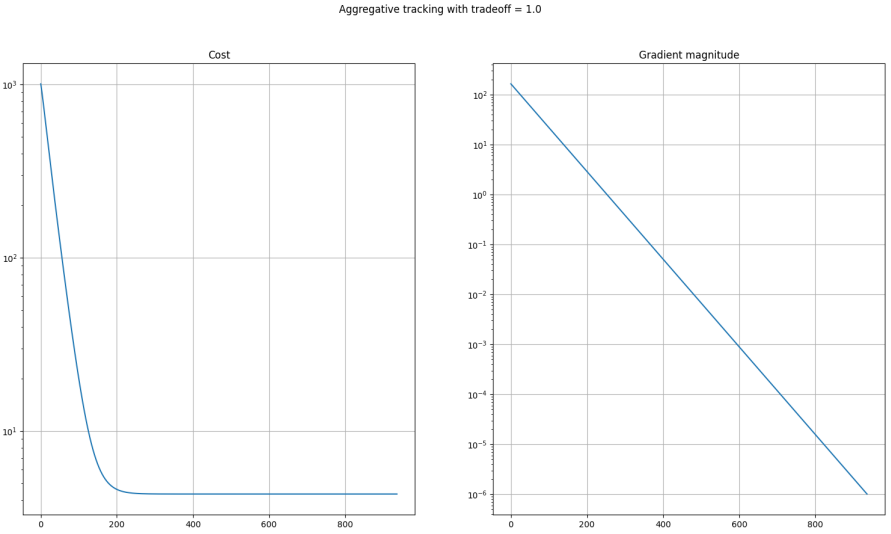

Figure 2.2: Cost and gradient magnitude with $\gamma = 1$

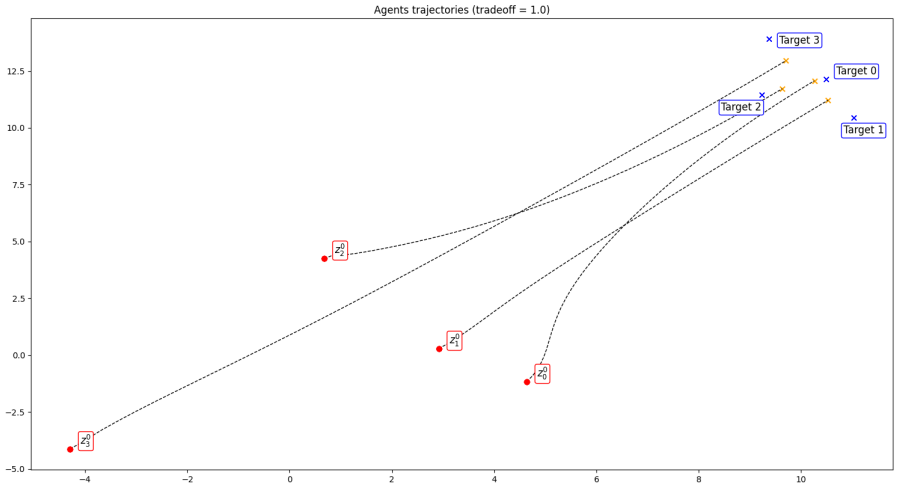

Figure 2.3: Behaviour of robots with $\gamma = 1$

Results with $\gamma = 0.1$

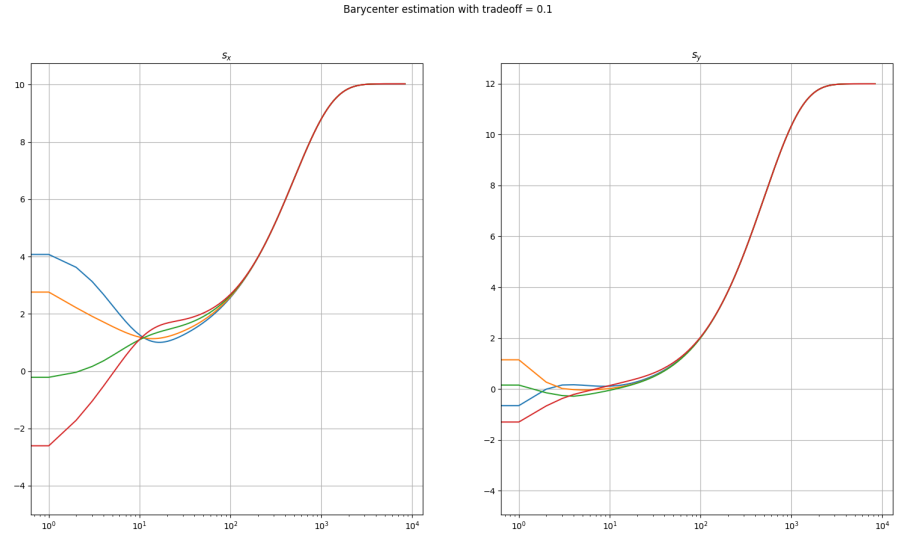

Figure 2.4: Barycenter estimation with $\gamma = 0.1$

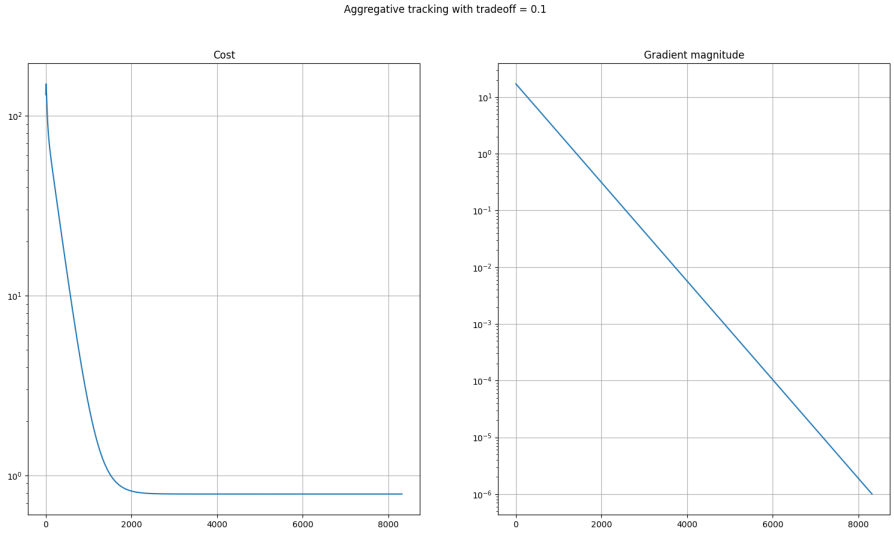

Figure 2.5: Cost and gradient magnitude with $\gamma = 0.1$

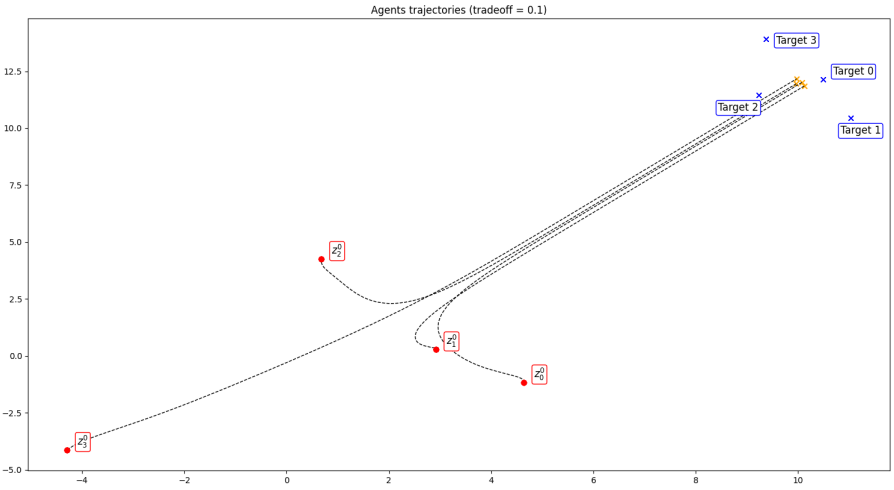

Figure 2.6: Behaviour of robots with $\gamma = 0.1$

Results with $\gamma = 10$

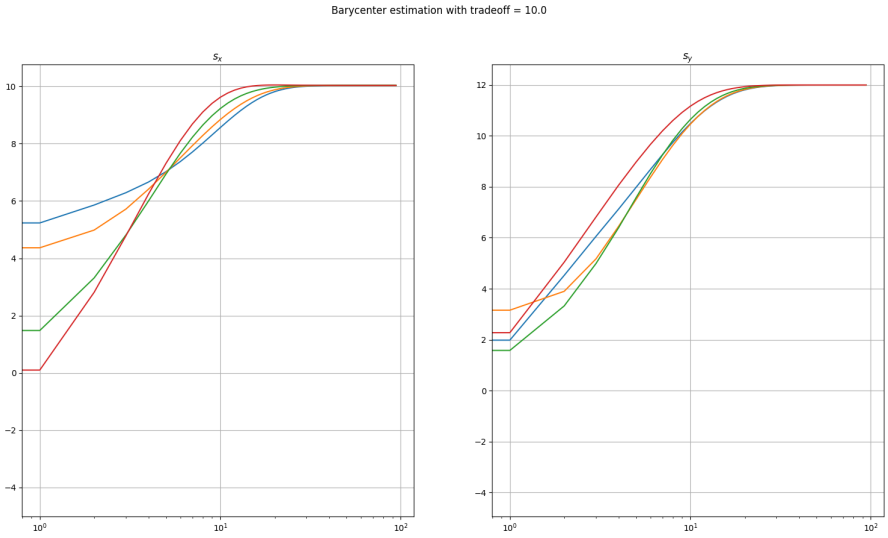

Figure 2.7: Barycenter estimation with $\gamma = 10$

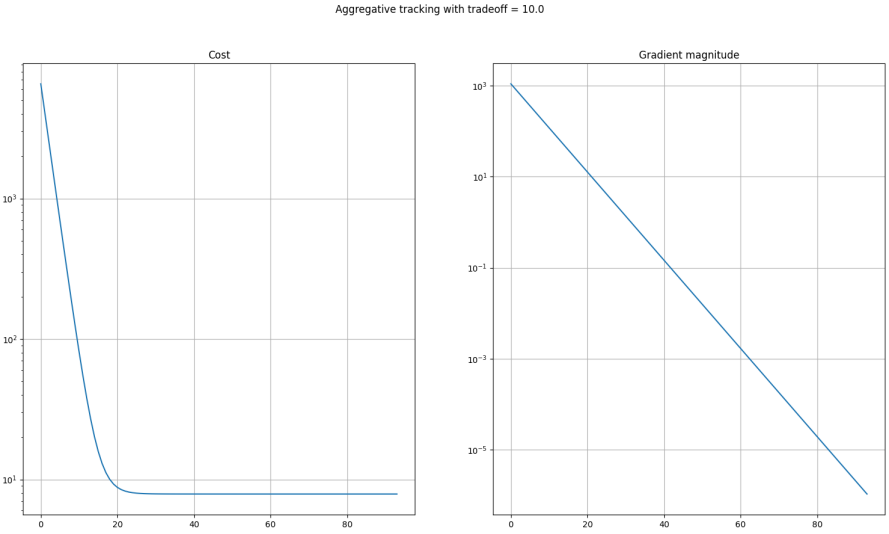

Figure 2.8: Cost and gradient magnitude with $\gamma = 10$

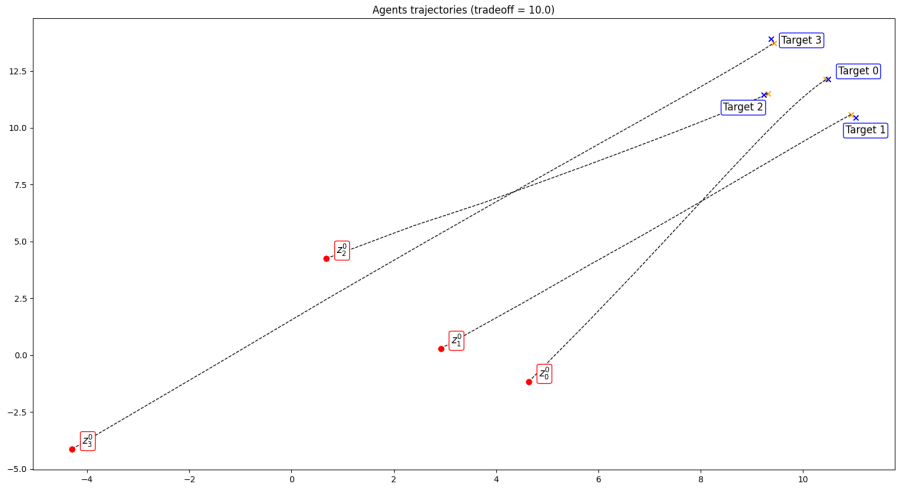

Figure 2.9: Behaviour of robots with $\gamma = 10$

Some experiments were performed also with a smaller distance between robots and targets, obtaining the same behaviour (for brevity only the final plots are reported).

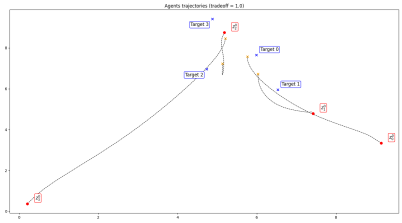

Figure 2.10: Robots near targets, $\gamma = 1$

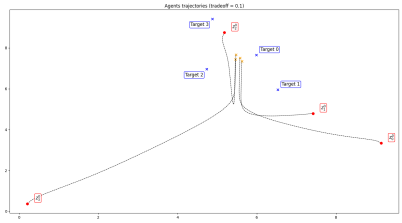

Figure 2.11: Lower distance, $\gamma = 0.1$

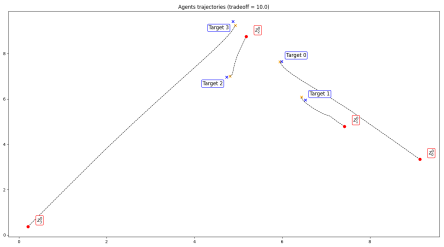

Figure 2.12: Robots near targets, $\gamma = 10$

2.2 ROS2 implementation

Two ROS2 packages has been created into a workspace. The code has been organized as:

```
ros
└── src
    ├── message
    │   └── msg
    │       ├── NodeData.msg
    │       └── PlotterData.msg
    └── surveillance
        ├── launch
        │   ├── corridor.launch.py
        │   └── surveillance.launch.py
        └── surveillance
            ├── aggregative_tracking.py
            ├── costs_fn.py
            ├── functions.py
            ├── phi.py
            ├── plotter.py
            └── warden.py
```

The package *message* is used to define proper message structures used to implement communication between robots, while in the *surveillance* packages two ROS2 nodes are coded: *warden* is the node that implements the aggregative tracking algorithm as in section 2.1, while *plotter* is an auxiliary node created to plot the overall behaviour of the system.

The launch file *surveillance.launch.py* is used to create the nodes with proper parameters. The γ and the initial distance between robots and targets can be passed as an argument while launching the launch file. Experiments shows that the behaviour is the same as described in section 2.1.

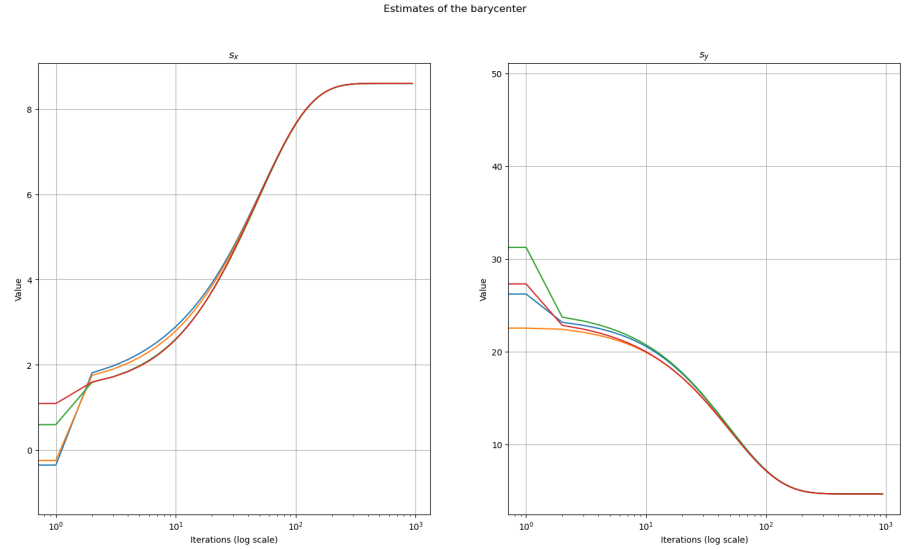

Figure 2.13: Barycenter estimation in ROS2 with $\gamma = 1$

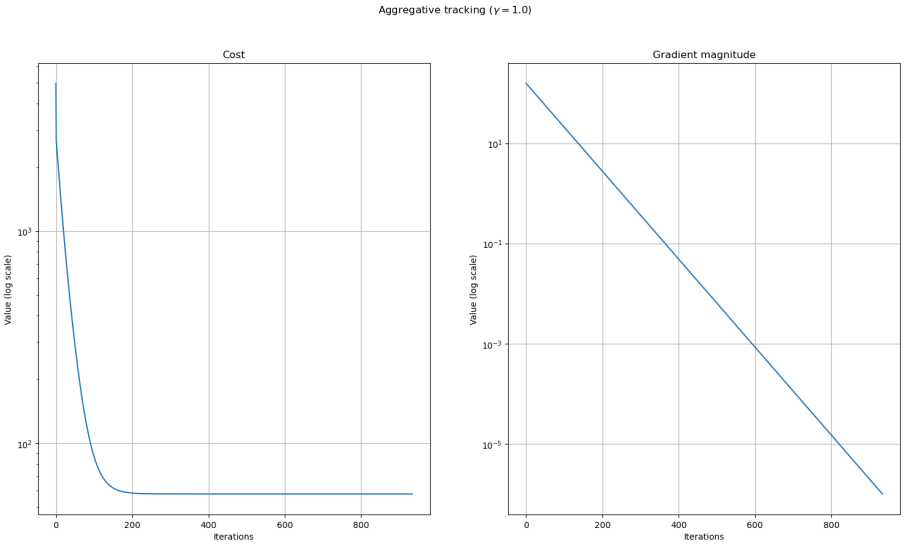

Figure 2.14: Cost and gradient magnitude in ROS2

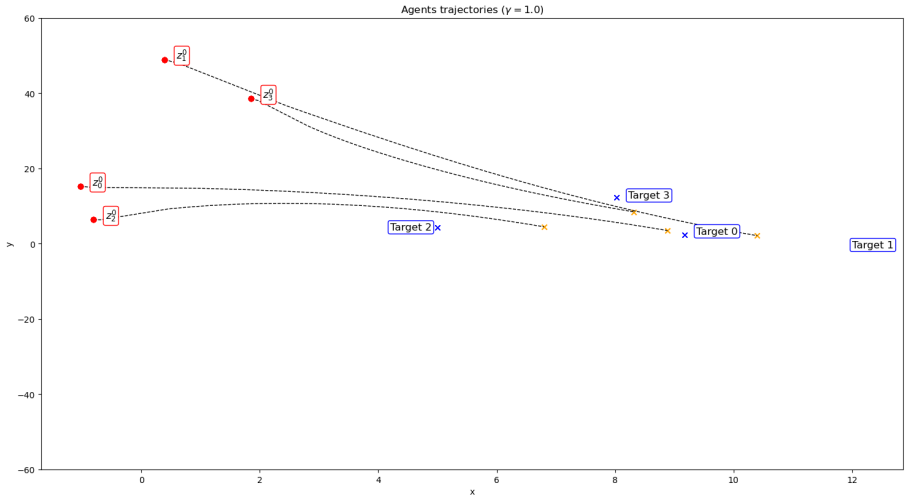

Figure 2.15: Agents behaviour in ROS2

An animated simulation is also present, showing the nodes that move toward the targets.

2.3 Moving inside a corridor

To make the robot able to move inside a corridor, a feasible region has been designed using a particular function. The robots are constrained to be in the feasible region. The feasible region is described by two local inequality constraint that each node must always satisfy:

$$\begin{aligned} g_1(z_i) \leq 0 \quad \forall i = 1, \ldots, n \\ g_2(z_i) \leq 0 \quad \forall i = 1, \ldots, n \end{aligned}$$

Where:

$$\begin{aligned} g_1(x_i, y_i) &= -1e^{-5} \cdot x_i^4 - 2 + y_i \leq 0, \quad i = 1, \ldots, n \\ g_2(x_i, y_i) &= -1e^{-5} \cdot x_i^4 - 2 - y_i \leq 0, \quad i = 1, \ldots, n \end{aligned}$$

Those inequality constraints are embedded in the cost function $\ell(z, \sigma(z))$ by using a barrier function. The overall cost becomes:

$$\min_{z \in \mathbb{R}^2} \sum_{i=1}^{N} \gamma \cdot \|z_i - target_i\|^2 + \|z_i - \sigma(z)\|^2 - \log(-g_1(z_i)) - \log(-g_2(z_i))$$

The gradient $\nabla \ell(z, \sigma(z))$ is given by:

$$\nabla \ell(z, \sigma(z)) = \sum_{i=1}^{N} \nabla \ell_i(z_i, \sigma(z))$$

Where:

$$\nabla \ell_i(z, \sigma(z)) = \begin{bmatrix} \nabla_z \ell_i(z, \sigma(z)) \\ \nabla_\sigma \ell_i(z, \sigma(z)) \end{bmatrix} \in \mathbb{R}^4$$

And:

$$\nabla_z \ell_i(z, \sigma(z)) = \begin{bmatrix} 2 \cdot \gamma \cdot (z_{i1} - target_{i1}) + 2 \cdot (z_{i1} - \sigma_1) \\ 2 \cdot \gamma \cdot (z_2 - target_2) + 2 \cdot (z_2 - \sigma_2) \end{bmatrix} + \begin{bmatrix} \frac{-1 \times 10^{-5} \times 4 \times z_{i1}^3}{g_1} \\ \frac{-1 \times 10^{-5} \times 4 \times z_{i1}^3}{g_2} \end{bmatrix} + \begin{bmatrix} \frac{-1 \times 10^{-5} \times 4 \times z_2^3}{g_1} \\ \frac{-1 \times 10^{-5} \times 4 \times z_2^3}{g_2} \end{bmatrix}$$

$$\nabla_\sigma \ell_i(z, \sigma(z)) = \begin{bmatrix} 2 \cdot (z_{i1} - \sigma_1) \\ 2 \cdot (z_2 - \sigma_2) \end{bmatrix}$$

Where:

- z_{i1} and z_{i2} are the x and y position coordinates of the robot.
- σ_{i1} and σ_{i2} are the x and y coordinates of the estimation of the barycenter.
- $target_{i1}$ and $target_{i2}$ are the x and y components of the target position for robot i.

The following plots show the results obtained.

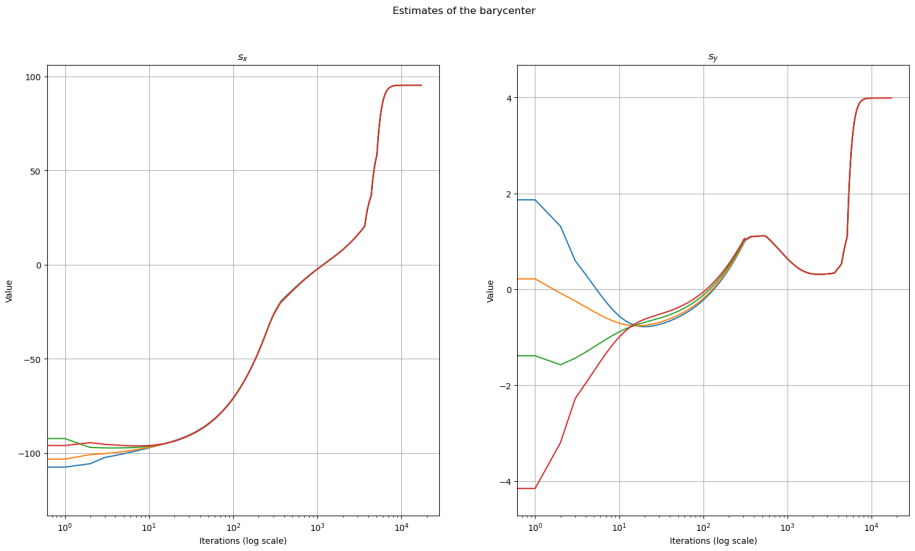

Figure 2.16: Barycenter estimation in ROS2 with corridor (case 1)

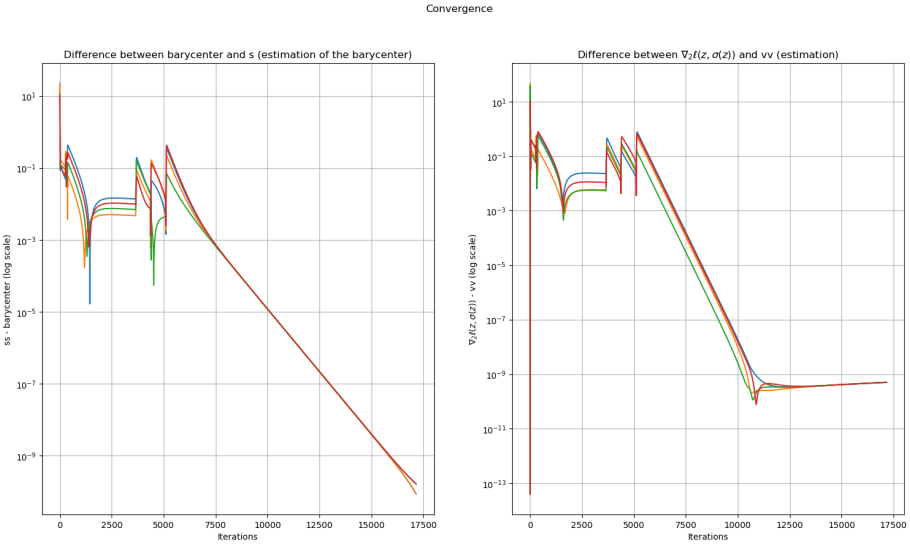

Figure 2.17: Showing convergence of the algorithm (case 1)

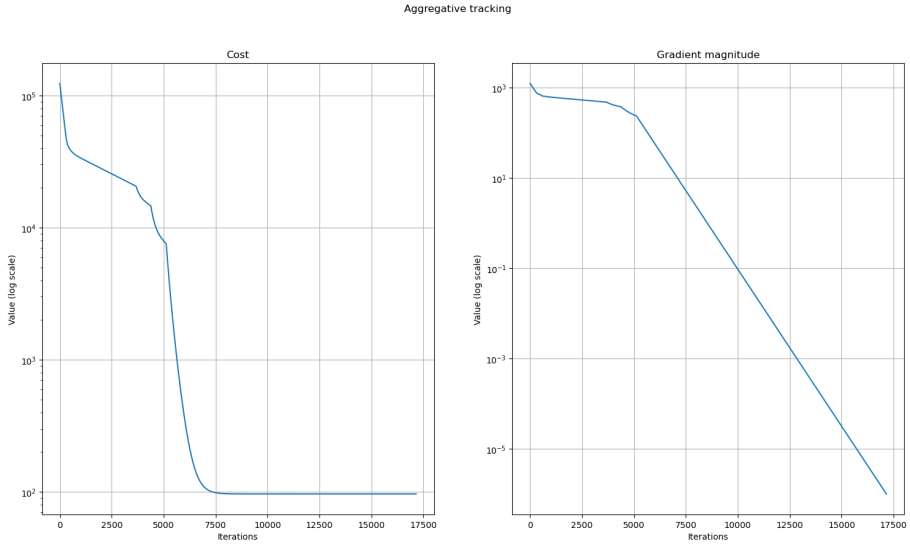

Figure 2.18: Cost and gradient magnitude ROS2 (case 1)

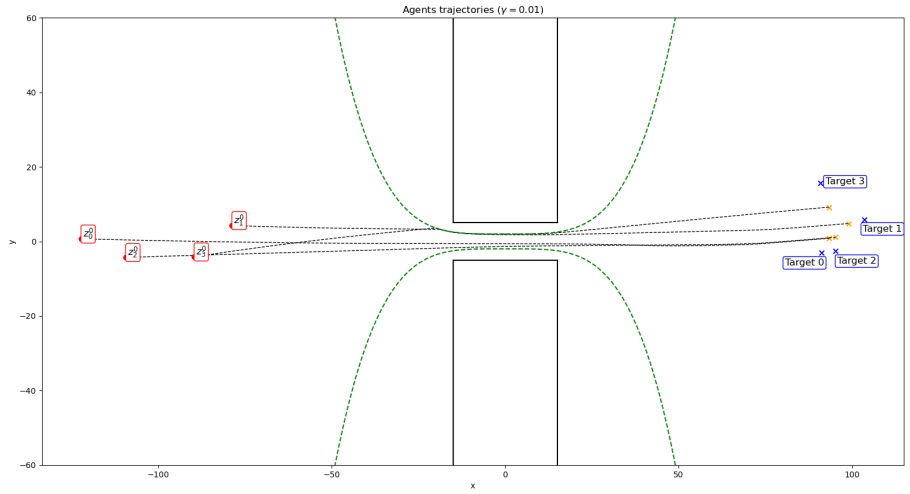

Figure 2.19: Agents trajectories ROS2 with corridor (case 1)

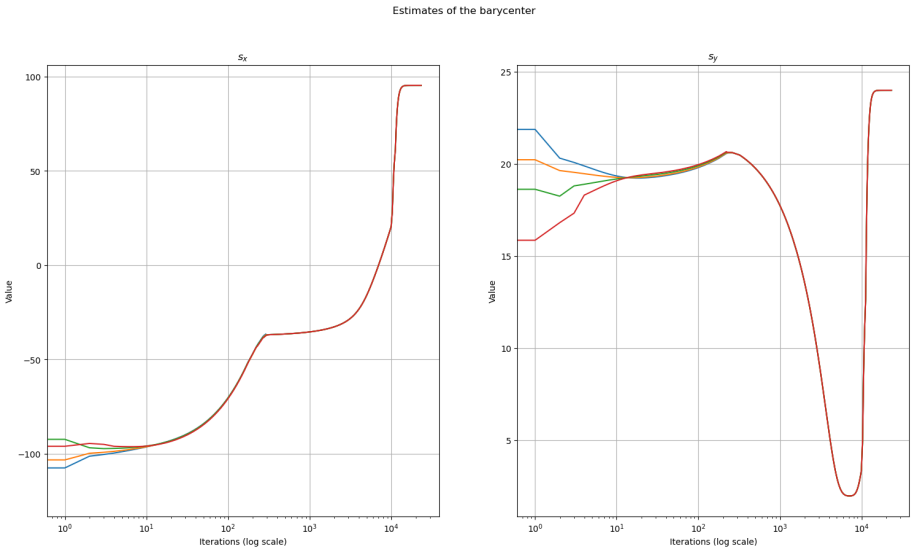

Figure 2.20: Barycenter estimation (case 2)

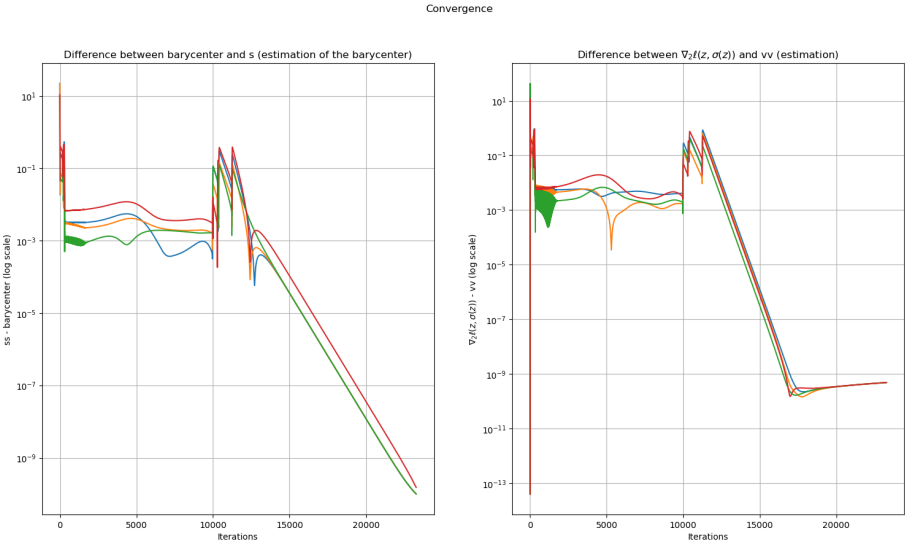

Figure 2.21: Showing convergence of the algorithm (case 2)

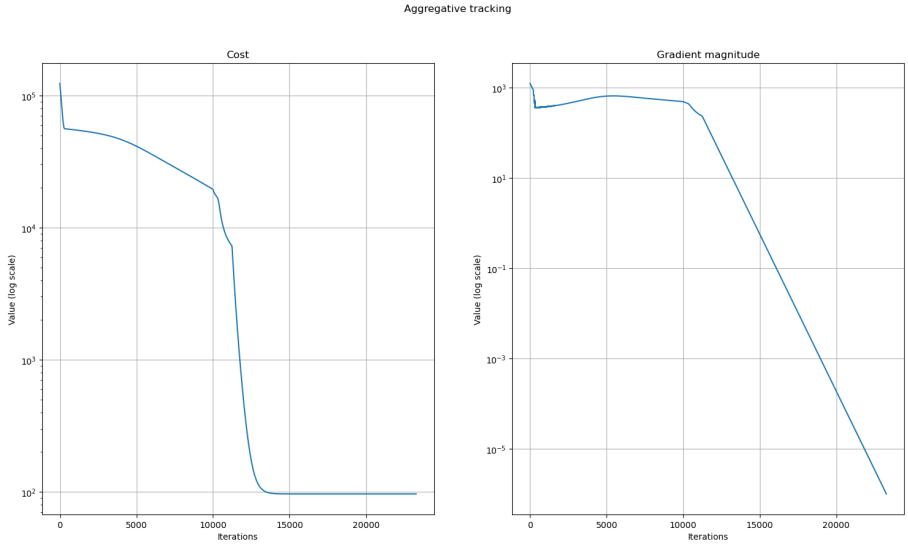

Figure 2.22: Cost and gradient magnitude (case 2)

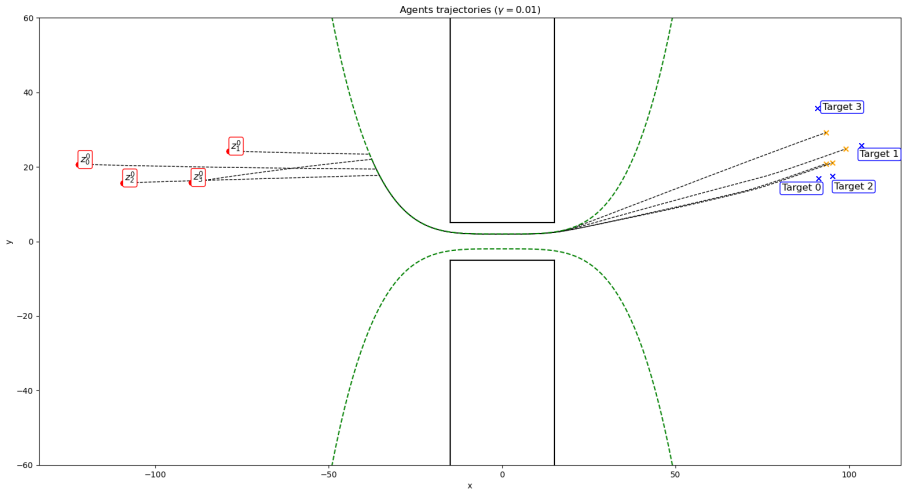

Figure 2.23: Agents trajectories ROS2 with corridor (case 2)

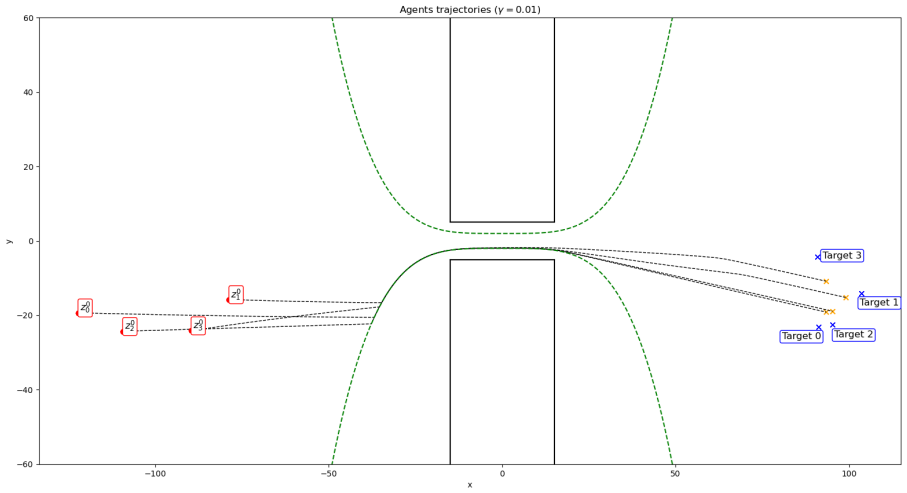

Figure 2.24: Agents trajectories ROS2 with corridor (case 3)

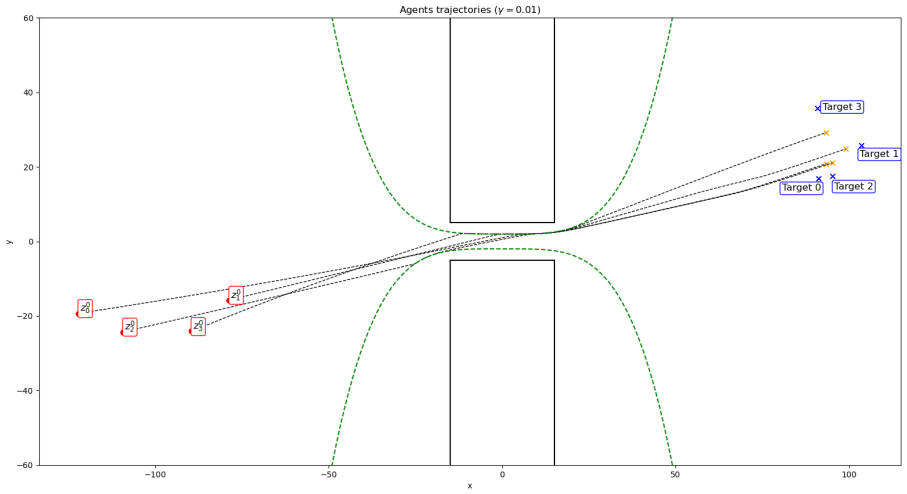

Figure 2.25: Agents trajectories ROS2 with corridor (case 4)

Conclusions

The work demonstrates how distributed algorithms can be efficiently implemented to solve complex problems.

It shows how the Gradient Tracking algorithm can successfully be implemented to solve a classification problem, comparing it with a traditional centralized algorithms.

Moreover, the development of a distributed control algorithm for multi-robot systems shows how a complex problem can be solved without necessarily having a centralised planning. Addressing challenges such as formation maintenance and proximity to targets, the work offers a practical solution with broad applicability. The use of ROS2 for implementation and testing ensures the results can be applied also in real-world scenarios.

Overall, this report showcases the feasibility and benefits of distributed algorithms in both classification tasks and multi-robot coordination.