Universidad Zaragoza
1542

Trabajo Fin de Máster

Centroid estimation based in multi-agent dynamic consensus.

Estimación de centroide basado en consenso dinámico multiagente.

Autor

Pardina Quirós, Sergio

Directores

Aragüés Muñoz, María del Rosario

López Nicolás, Gonzalo

ESCUELA DE INGENIERÍA Y ARQUITECTURA

2024

Agradecimientos

A mis directores Rosario Aragüés y Gonzalo López, por introducirme a este tema de investigación, por su tutela, constante apoyo y orientación que han sido fundamentales para el éxito de este trabajo.

Al I3A por permitirme realizar este proyecto en un entorno de investigación de alto nivel y adquirir conocimientos prácticos y profesionales que sin duda marcarán mi futuro.

Este trabajo ha sido parcialmente financiado por el Programa de Becas y Ayudas del Instituto de Investigación en Ingeniería de Aragón (I3A).

Resumen

El uso de múltiples agentes para la resolución de un problema complejo puede significar una mejoría tanto en el tiempo de resolución del problema, la precisión de la misma o incluso permitir que tenga solución. Una posible forma de aplicar estos múltiples agentes es mediante algoritmos de consenso. Algoritmos donde diferentes agentes actúan de forma independiente en base a su información local pero son capaces de comunicarse entre sí. Esta comunicación les permite actualizar sus posiciones, estados o estimaciones en función de la discrepancia que haya con sus vecinos. Al final se termina llegando a un consenso compartido entre todos los agentes

En este trabajo buscamos estudiar diferentes algoritmos de consenso y aplicarlo a un problema de estimación. Se quiere estimar el centro de masas o centroide de un cuerpo deformable del cual no se conoce su geometría. El consenso se obtendrá mediante la detección de balizas que delimitan el cuerpo y son vistas por diferentes robots. Mediante el seguimiento individual de estas balizas por los robots, se tendrá entonces una estimación de dicho centro de masas o centroide a pesar de que ningún robot conocer más información que la información local de la baliza que está siguiendo y la información que le puedan proporcionar sus vecinos.

Summary

The use of multiple agents for the resolution of a complex problem can mean an improvement in both the time to solve the problem, the accuracy of it or even allow it to be solved. One possible way to apply these multiple agents is through consensus algorithms. Algorithms where different agents act independently based on their local information but are able to communicate with each other. This communication allows them to update their positions, states or estimates based on the discrepancy with their neighbors. Eventually they end up reaching a shared consensus among all the agents.

In this work we seek to study different consensus algorithms and apply them to an estimation problem. We want to estimate the center of mass or centroid of a deformable body whose geometry is not known. The consensus will be obtained by detecting beacons that delimit the body and are seen by different robots. Through the individual tracking of these beacons by the robots, we will then have an estimation of the center of mass or centroid even though no robot knows more than the local information of the beacon it is following and the information provided by its neighbors.

Table of contents

Chapter 1

Introduction

Since the introduction of the "Unimate" in the mid-20th century, the adoption of both mechanical and virtual systems that automatically operate in an autonomous way has steadily increased. These machines, commonly referred to as robots, have significantly boosted economic productivity and reduced the human effort required for repetitive tasks across industries like manufacturing, data processing, healthcare, and more.

In addition to reducing the need for human labor, they have also enabled the standardization of manufacturing processes and improved the safety of many high-risk activities like radioactive cleanup [1]. They have also enhanced human capabilities, such as the robotization of prosthetic limbs or the implementation of human-controlled robots for surgical interventions [11]. Moreover, the development of sophisticated robots has allowed humanity to access inhospitable environments such as the Moon or Mars at a fraction of the cost and risk that sending live humans would take.

However, robots are often treated as discrete units in their applications, limiting the potential for process improvements. By enabling communication between robots, their collective perspectives could be leveraged to enhance performance. This approach would not only enable a more refined space estimation but also improve task execution by allowing robots to work together toward a common goal. For example, consider the scenario of an oil spill in an oceanic region. It would be extremely time-consuming for a single robot or operator to survey the entire oil spill and localize its source, such as a sinking oil tanker. However, by deploying hundreds of drones, each covering a smaller area, the perimeter of the spill could be mapped much more quickly. Additionally, by combining their individual estimations and positions, the drones could provide a more accurate estimate of the oil tanker's location.

1.1 Motivation

Robotics is now commonplace across many fields such as search and rescue, environmental monitoring, and industrial automation. Multiple robots working in coordination can achieve objectives more efficiently and robustly than a single robot. Through the study of consensus algorithms we ask a group of robots to reach an agreement on shared goals, such as their position, trajectory, or those of a third party, like tracking an object. By taking advantage of the different measurements and estimations from the robots the variable to detect is more accurately followed despite individual differences and uncertainties.

As robotic systems grow more sophisticated, the need for reliable and scalable methods to coordinate their actions becomes critical. Consensus strategies tackle challenges related to distributed decision-making, where each robot only has local information but needs to align its actions with others to achieve a common objective. Furthermore, studying consensus in multi-robot systems opens new possibilities for decentralized control, reducing the dependency on centralized computation and communication, which can be a bottleneck in large-scale deployments and also represents a single point of failure.

Figure 1.1: An oil spill

Figure 1.2: A forest fire

Figure 1.3: Herding

Figure 1.4: From crisis management such as estimating the source of an oil spill or forest fire, to husbandry, like following a herd. Many different applications may benefit from multi-agent consensus.

Achieving consensus among multiple robots is helpful for the development of collaborative autonomy, where robots act independently but as cohesive teams. This enhances their collective intelligence and operational effectiveness. This research area not only contributes to the advancement of robotics but also has significant implications for future technologies in fields such as swarm intelligence, distributed computing, and network systems.

1.2 Goals

The goal of this report is studying a multi-agent system where several different robots track a deformable object.

By taking advantage of multi-agent consensus we expect to be able to estimate the center of mass or centroid of the deformable object. This is a task that would not be feasible to achieve by single uncommunicated robots in certain circumstances.

Besides making a proof of concept and verifying that using a multi robot consensus is effective, we also aim to define the range of validity under which the different parameters of the simulation must fall to achieve convergence. Additionally, we will also attempt assess its accuracy.

Specifically, we want to :

- Pose the previously unresearched problem of a deformable body that would benefit from dynamic consensus.
- Introduce the most commonly used dynamic consensus algorithms and reason which would be best for our problem.
- Differentiate between the distributed and the vector form of dynamic consensus algorithms and justify which one to choose.
- Verify that the discrete consensus results due to experimental simulations are valid.
- Analyze the conditions to achieve convergence in the dynamic consensus.
- Analytically compute an upper bound for the maximum tolerable error in the results.
- Study the influence of different parameters of the system in the convergence and final error of the estimated centroid.

In order to achieve this goals the main tool used for the experimental simulations will be the Python programming language. Using version 3.12.6 and Matlab for some prototyping. Most of the programming was done in the JupyterLab development environment. Specifically version 4.2.5

1.3 Structure of the report

The following report is structured as follows.

Firstly, in chapter 2 the problem to solve is posed and explained. Later on a definition of consensus is to be given and the difference between static and dynamic consensus is highlighted. Once both terms have been clearly explained a formal definition of the problem we are attempting to solve is written.

Some numerical simulations are studied in chapter 3 with the aim of illustrating which algorithm is the most appropriate for the problem at hand. We then discuss whether using its vector or distributed formulation is more useful.

Once said groundwork has been laid, we proceed with some mathematical work. We do this to calculate a mathematical set of conditions that connect the different variables of the problem with the possibility of convergence. Then the value of the step size chosen for the experimental simulations is shown and justified.

In chapter 5 we examine the deformable system scenario. We will provide an algebraic study to estimate its maximum possible error and perform its associated simulations for the experiments. We then also study some of the system parameters and how they affect the consensus.

Lastly we will discuss the conclusions gathered after the whole process in chapter 6. A small section proposing some possible future research avenues of interest is also included.

Chapter 2

Dynamic consensus - Problem definition

We aim at using several robots in order to estimate the centroid of a deformable object. The deformable object will be delimited by a set of points on its surrounding edge. We will call those objects beacons from now on. The different robots are capable of seeing one to all beacons, from which they will estimate a reference signal. This reference signal will be the point where each robot intends to converge to and is computed as the average position of the detected beacons. The agents are also capable of communication between each other to inform their neighbors about their current state. The state in our case will be defined as the bi-dimensional position in space.

2.1 Related work

Research has already been made about consensus in multiple agent systems. However in most of the studied cases each robot follows its own reference signal which is independent of the rest of robots or a body to track. Therefore the robots themselves behave almost independently of a global point or body to track.

2.2 Static consensus

Let N agents be in a $\mathbb{R}^2$ space. Each agent position is $\mathbf{x}_i$. And is provided with its own reference value $\mathbf{u}_i$ a constant value. The agents are able to communicate with each other be it directly or indirectly by communicating through a middle-man agent. The communication between robots may be represented with a strongly connected graph. The overall goal is to compute an estimation of the average value of the reference signals. That is $\mathbf{u}_{avg} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{u}_i$

Since the graph is assumed to be equally-weighted and bidirectional the agents will then asymptotically converge to the intended goal applying the following controller:

$$\dot{\mathbf{x}}_i(t) = -\sum_{j=1}^{N} a_{ij}(\mathbf{x}_i(t) - \mathbf{x}_j(t)). \tag{2.1}$$

To discretize the controller we may use the approximation of

$$\dot{\mathbf{x}}_i\left(t\right) \approx \frac{\mathbf{x}_i\left(k\right) - \mathbf{x}_i\left(k-1\right)}{\tau} = \dot{\mathbf{x}}_i\left(k\right), \tag{2.2}$$

for $t = k * \tau$. Which leads to the following discrete expression to update the states of the agents.

$$\mathbf{x}_i(k) = \mathbf{x}_i(k-1) - \tau a \sum_{j=1}^{N} a_{ij}(\mathbf{x}_i(k-1) - \mathbf{x}_j(k-1)). \tag{2.3}$$

Where a is a controller parameter that requires tuning. It is always of the form of $a = \tau * \mu$. Where μ is the simulation time step, and k might be any constant, variable or even function, depending on the algorithm and problem chosen. The variable a_{ij} is a binary variable to codify communication between robots i, j.

2.3 Dynamic consensus

In many cases the reference signal that each agent follows may not be constant with time, being instead a time dependent function $f(t)$, such that $\mathbf{u}_i\left(t\right) = \mathbf{f}_i\left(t\right)$.

In such case is necessary to include the signal variation inside the controller expression to allow the robots to take said signal variations into account.

The resulting controller in discrete time is therefore:

$$\dot{\mathbf{x}}_i(t) = \dot{\mathbf{u}}_i - a \sum_{j=1}^{N} a_{ij} * (\mathbf{x}_i(t - \Delta t) - \mathbf{x}_j(t - \Delta t)) \tag{2.4}$$

Which leaves the state update equation for the states (in our case position) of the i-th robots in a discrete time frame:

$$\mathbf{x}_i(k) = \mathbf{x}_i(k-1) + \mathbf{u}_i(k) - \mathbf{u}_i(k-1) - \tau a \sum_{j=1}^{N} a_{ij} * (\mathbf{x}_i(k-1) - \mathbf{x}_j(k-1)) \tag{2.5}$$

In both this case of dynamic consensus and the static consensus of section 2.2 the controller relies only on local information it may have access to, that is, the signal it views and the position of those other agents it can communicate with.

2.4 Formal definition of the problem

In this section we will give a formal overview of the problem we want to solve.

We aim to estimate the center of mass or centroid of a dynamically evolving deformable object.

The deformable object will be delimited by a series of beacons that limit the edges of the body. Since we would like the body to be deformable the different beacons will then be allowed to oscillate with their own direction. The different robots will sense a certain number of beacons, whose average position will be the reference signal they follow.

We will consider the true centroid of the object as the average position of all its delimiting beacons. The goal is then for the robots to asymptotically converge to the true centroid of the object thanks to the consensus. If that is not achievable then a reasonable accuracy estimation should be given to know the approximate position of the true centroid.

So let a set or robots ID'd as $i = 1, .., N$ and connected as a strongly connected, bidirectional unweighted graph. Strongly connected so all robots participate in the consensus. Bidirectional and unweighted as a desing decision for simplicity. The positions are given respectively by the vector $\mathbf{x_i}$. Inter-robot communications are codified by an adjacency matrix. In the scenario we study the links between robots are bidirectional and are all equally weighted. Therefore our matrix will be defined as :

$$a_{ij} = \begin{cases} 1 & \text{if robot } i \text{ and robot } j \text{ are linked} \\ 0 & \text{if robot } i \text{ and robot } j \text{ are not linked} \end{cases} \tag{2.6}$$

There is also a set of M beacons on the edges of a certain deformable body whose centroid we wish to track. The beacons are also identified by an index $m = 1, ..., M$. Their position is given by the vector $\mathbf{x}_m$, and it is known by any robot that views the beacon. The centroid of the deformable body is defined as the average position of all its beacons $\mathbf{x}_c = \frac{1}{M} \sum_{m=1}^{M} \mathbf{x}_m$. A small simplified drawing of the situation described is shown in Figure 2.1.

Each robot therefore knows the positions of those beacons it sees $\mathbf{x}_m^i$ and computes the reference signal it will follow. Its reference signal is obtained as the average position of the beacons it sees $\mathbf{u}_i = \frac{1}{M_b^i} \sum_{m=1}^{M} \mathbf{x}_m^i$. Where M_b^i is the number of beacons a robot is seeing.

The beacons will behave as harmonic oscillators and will oscillate to simulate the deformability of the object. As a design choice we decided that the robots converge towards the centroid, so that once they have converged, they all share the same state and only one of them is required to recover the centroid position. If we had not required the robots to head towards the centroid, its position would also be possible to estimate but it would require to compute the average of the position of all the robots. By making

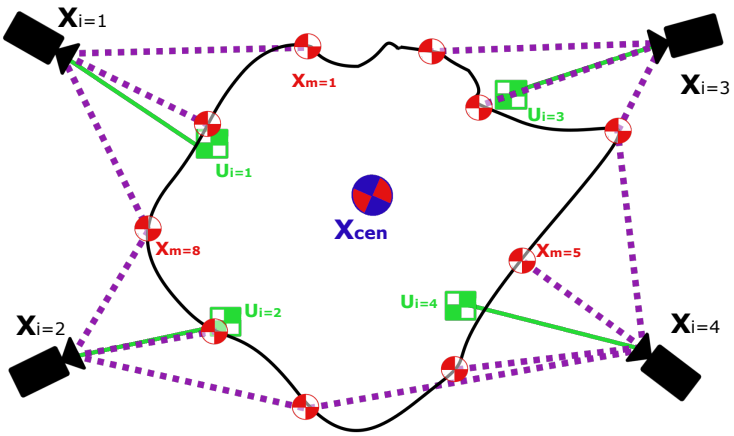
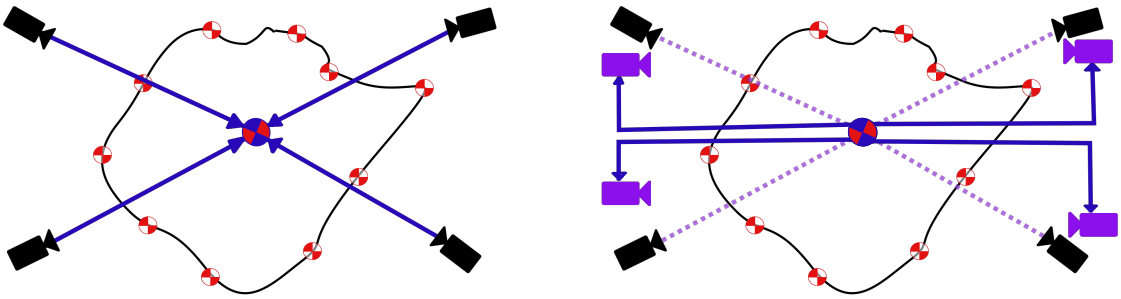


Figure 2.1: Simple drawing of the presented problem. A deformable object can be tracked through M beacons (red circular markers). Each of the N robots (black) may view several beacons (dotted). The goal is for the robots to reach a consensus regarding where the centroid of the body $\mathbf{u}_{avg}$ is (blue center marker) through their measured signals (square markers), which are the average positions of their respectively seen beacons.

each robot head for the centroid, the user only needs to establish communications with a single robot to know the coordinates of the centroid.

Another benefit of this choice: It also allows us to define formations for the robots to follow when tracking the centroid. This is achieved by simply adding a custom offset to each agent as it can be seen in Figure 2.2.



(a) Consensus towards centroid.

(b) Consensus with formation

Figure 2.2: Comparison between making a consensus converge towards the centroid or forcing it to follow a formation. In the second case the final formation of the agents (purple) is defined by an offset relative to the centroid.

In formation type consensus when the robot heads towards the centroid it also follows its own offset to ensure following its formation. This can only be assured as long as the beacons each robot views remains static, that is, a robot never loses track of its initially tracked beacons.

2.5 Perception matrix

Although not studied in this work, it may happen that there are not the same amount of agents as there are beacons to track. In such scenarios each robot may be tasked with sensing every beacon, only one beacon or any number in between. In order to better model this mathematically we may define the Perception Matrix.

$$p_m^i = \begin{cases} 1 \text{ beacon } m \text{ is viewed by robot } i \\ 0 \text{ beacon } m \text{ is not viewed by robot } i \end{cases} .$$

The resulting matrix has dimensions $N \times M$, where N is the number of robots and M is the total amount of beacons.

$$\begin{pmatrix} \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_N \end{pmatrix}_{N\times 1} = \mathbf{P}_{N\times M} \begin{pmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_M \end{pmatrix}_{M\times 1} \tag{2.7}$$

In this case $\mathbf{u}_i$ represents the reference signal read by the i-th robot and $\mathbf{x}_m$ is the state (position) of the m-th beacon.

Naturally, after the introduction of the perception matrix, the definition of $\mathbf{u_i}$ has to be redefined to take this new parameter into account. The expression that directly relates the sensed signal with the states of the beacons is

$$\mathbf{u}_i^{sum} = \sum_{m=1}^{M} p_m^i \mathbf{x}_m. \tag{2.8}$$

Where, $\mathbf{u}_i^{sum}$ is the signal measured by the robot i, and equals the sum of the positions $\mathbf{x}_m$ of all beacons seen by the i-th robot ($p_m^i \neq 0$). It is the consequence of doing the matrix multiplication of Equation 2.7.

But from the initial definition we are interested in the average signal, rather than the sum of signals, so a better computation would be to weight the sum in order to obtain the average value of the states (which we call average weighting):

$$\mathbf{u}_i = \frac{1}{\sum_{m=1}^{M} p_m^i} \mathbf{u}_i^{sum} = \frac{1}{\sum_{m=1}^{M} p_m^i} \sum_{m=1}^{M} p_m^i \mathbf{x}_m. \tag{2.9}$$

Which is the expression obtained for the average signal perceived by each robot after introducing the perception matrix.

Due to scope reasons, we elected to only focus on the case where $M = N$, each robot views only one beacon and every beacon is seen.

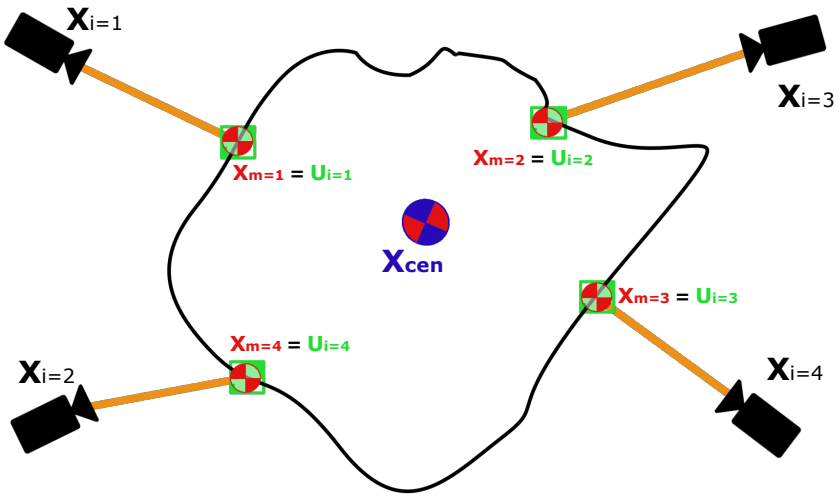


Figure 2.3: Simple drawing of the presented problem where the number of robots is equal to the number of beacons and each beacon is seen uniquely by a robot. A deformable object can be tracked through M beacons (red circular markers) whose positions coincide with the measured signals $\mathbf{u}_i$. Each of the N robots (black) views one beacon. The goal remains estimating the position of the centroid $\mathbf{x}_{cen}$.

We may study it without worrying of about losses of generality, since it is no more than the specific simpler case that fulfills that the perception matrix equals the identity matrix and no average weighting is necessary.

$$\mathbf{P}_{N\times N} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix}, \qquad \sum_{m=1}^{M} p_m^i = 1 \quad \forall i. \tag{2.10}$$

Chapter 3

Dynamic consensus algorithms

In this chapter we will introduce and analyze the different possible dynamic consensus algorithms studied for solving our particular problem. The strengths and weaknesses of each algorithm will be discussed and the outcomes of the simulations will also be shown in plots.

The comparison will be made by applying the different consensus algorithms to the same system. The system will consist in 10 agents following 10 beacons. Each beacon will move uniformly in the horizontal direction and will make a sudden jump in the y direction. The jump does not necessarily reflect the movement of an usual centroid in a real application but rather has been included to better view how the algorithm would deal with such a jump. Each beacon will also oscillate around its own position to model the deformations of the body.

3.1 Basic dynamic consensus

The most basic formulation for achieving dynamic consensus in a system made up of N robots and N signals is the following (in vector notation):

$$\dot{\mathbf{x}}_i = -\mathbf{L}\mathbf{x}_i + \dot{\mathbf{u}}_i, \quad \mathbf{x}_i(0) = \mathbf{u}_i(0). \tag{3.1}$$

$\mathbf{x}_i$ is the position of a robot and $\mathbf{u}_i$ its associated signal.

We have committed a small abuse of notation by not specifying the continuous domain (t). For sake of clarity we define the continuous time expressions without (t), meanwhile, for discrete time expressions, we will specify the discrete time point (k), both to differentiate discrete and continuous and because it is necessary to follow along the equations.

From this expression and thanks to the approximation from Equation 2.2 it can be

obtained the first algorithm in discrete time for the dynamic consensus.

$$\mathbf{x}_i(k) = \mathbf{x}_i(k-1) + \tau \dot{\mathbf{x}}_i(k), k \geq 1, \tag{3.2}$$

$$\dot{\mathbf{x}}_i(k) = \mathbf{u}_i(k) - \mathbf{u}_i(k-1) - \tau \sum_{j=1}^{N} a_{ij} \left(\mathbf{x}_i(k-1) - \mathbf{x}_j(k-1)\right), \tag{3.3}$$

$$\mathbf{x}_i(0) = \mathbf{u}_i(0). \tag{3.4}$$

The shown system is written in its distributed loop representation. It could also be written in the form of a vector algorithm, which can be represented as

$$\mathbf{x}_i(k) = \mathbf{x}_i(k-1) - \tau \mathbf{L}\mathbf{x}_i(k-1) + \mathbf{u}_i(k) - \mathbf{u}_i(k-1), \tag{3.5}$$

$$\mathbf{u}_i(k) = \mathbf{u}_i(k) \tag{3.6}$$

For the sake of redundancy, we present the different compared algorithms only in its distributed representation. We will also briefly study the benefits and disadvantages of each representation and the choose the best one. This will be done in section 3.7.

The proposed system shown in the set of equations Equation 3.2, Equation 3.3 and Equation 3.4 is quite simple to understand. The update stage of the state (position) of the robots happens in Equation 3.2 where the controller action is added to the current position of the robots. The controller to apply is described by Equation 3.3. It is simply the sum of two terms. The first term $\mathbf{u}_i(k) - \mathbf{u}_i(k-1)$ appears from the Euler discretization of $\dot{\mathbf{u}}$ and is the change of the measured signal by the robots each step. The second part of the equation manages the consensus part of the controller. If two robots are able to communicate, then the difference between positions $\mathbf{x}_i$ and $\mathbf{x}_j$ will be added to the controller. This will cause those two robots to gradually converge to the same position. This is repeated for every possible pair of robots due to the summation and the communication is governed by the Laplacian matrix, from which a_{ij} comes from. The parameter a_{ij} may be only 1 or 0 depending on whether the two robots do or do not communicate (Equation 2.6).

Notice the Equation 3.4 that forces the initial states of the robots to be the same as the reference signals they measure. This allows for convergence towards the centroid goal $\mathbf{u}_{avg}(k) = \frac{1}{N}\sum_{j=1}^{N} \mathbf{u}_j(k)$. This final constraint may be relaxed to $\sum \mathbf{x}_i(0) = \sum \mathbf{u}_i(0)$.

The formulation while simple to implement and understand has an important drawback: Its initial constraint. If said condition is not fulfilled the robots reach a consensus outside the goal centroid. While all the robots will tend to converge to the same point in space that point will not necessarily be the average position of the beacons. The robots will converge to a point and then will perfectly track the motion of the centroid, but will suffer from an offset regarding it.

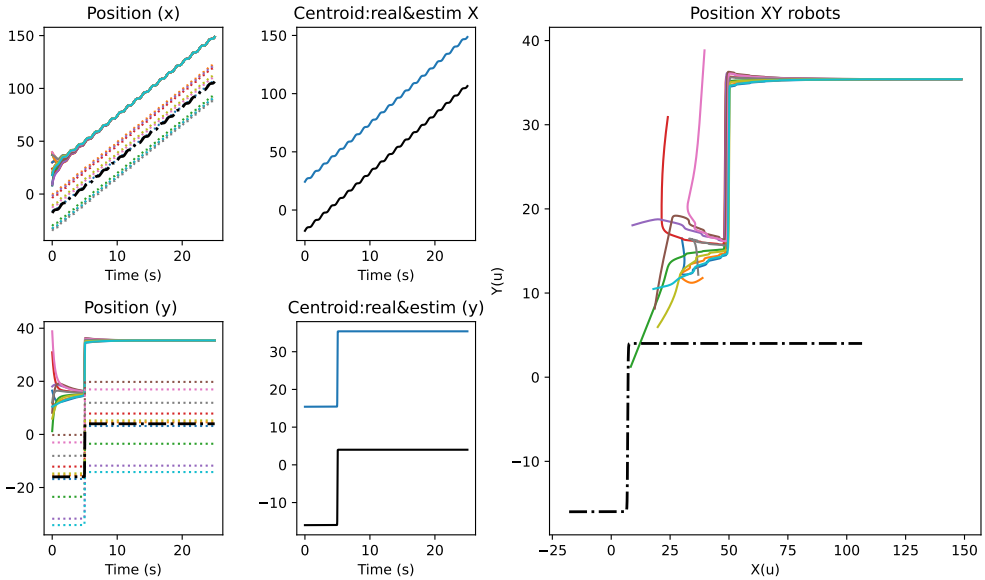


Figure 3.1: Simulation of a consensus of 10 Robots (solid) following 10 signals (dotted) using basic dynamic consensus. The left column shows the x (top) and y (bottom) coordinates of the robots (solid), the beacons (dotted) and the centroid (dashed) as time passes. The middle column shows the x (top) and y (bottom) coordinates of both the real centroid (black) and the estimated centroid (blue). The final bigger plot shows the xy position of the robots (solid) and compares it to the true centroid position (dashed), we'd like for them to coincide. When not fulfilling $\mathbf{x}_i(0) = \mathbf{u}_i(0)$, the robots converge towards an average signal and track it, but suffer an offset from the true average signal. We gently remind that the jump that happens is unlikely to happen in a real application, but we have modeled it nonetheless to view how well the algorithms react to unfavorable conditions.

In order to fulfill the initial condition all robots must have previous knowledge of the signal they perceive and have their initial state be the same as its value. So in this implementation not only is previous knowledge and positioning required for the consensus to be reached, the signals must be of the same type as the state of the robot, spatial coordinates, be them 1,2 or 3-dimensional.

While this flaw may not be critical depending on the task set to accomplish and the initial constraint could be circumvented with enough preliminary preparation, the algorithm is not ideal. We will introduce other algorithms later that attempt to fix those two weaknesses.

As it can be seen in the example simulated in Figure 3.1, not fulfilling the initial condition from Equation 3.4 causes the robots to converge with an offset. They all converge towards each other and track the movements of the signals with an offset. The y-coordinate and the small oscillations on the x-coordinates are correctly replicated, but it is done with an offset to the true centroid.

3.2 Dynamic consensus with drift towards signals

In the second implementation we introduce a new term in order to influence the robots to also move towards the signals they receive. With this addition we try to ensure that the consensus point where the robots converge is indeed the centroid of the deformable body we are trying to track.

$$\dot{\mathbf{x}}_i = -\mathbf{L}\mathbf{x}_i + \dot{\mathbf{u}}_i - \alpha\left(\mathbf{x}_i - \mathbf{u}_i\right), \tag{3.7}$$

The last term has been added to promote a drifting movement towards the signal. We may discretize the vector expression (Equation 2.2) in order to obtain the discrete distributed algorithm:

$$\mathbf{x}_i\left(k\right) = \mathbf{x}_i\left(k-1\right) + \tau\dot{\mathbf{x}}_i\left(k\right), k \geq 1, \tag{3.8}$$

$$\begin{aligned}\dot{\mathbf{x}}_i\left(k\right) = \mathbf{u}_i\left(k\right) - \mathbf{u}_i\left(k-1\right) - \tau\sum_{j=1}^{N} a_{ij}\left(\mathbf{x}_i\left(k-1\right) - \mathbf{x}_j\left(k-1\right)\right) + \\ + \tau\alpha\left(\mathbf{x}_i\left(k-1\right) - \mathbf{u}_i\left(k-1\right)\right),\end{aligned} \tag{3.9}$$

$$\mathbf{x}_i\left(0\right) = \mathbf{u}_i\left(0\right). \tag{3.10}$$

As it can be seen a new term has been added in Equation 3.9 that depends on the current state of a robot and the signal it measures. It essentially acts as adding a vector towards the direction of the measured signal so the movement of the robot has a component heading towards it. The influence of this term may be freely tuned as long as the studied conditions in chapter 4 are fulfilled.

By adding this term we let the robots converge individually to their respective measured signals and collectively together towards each other. Since it is assumed that all signals will be viewed by at least one robot, each robot will individually tend to the position of that beacon. As they also collectively converge to a consensus the final consensus point will be the average position of the beacons, the goal initially aimed to achieve.

Once the consensus has been reached, they will also perfectly track the movement of the centroid.

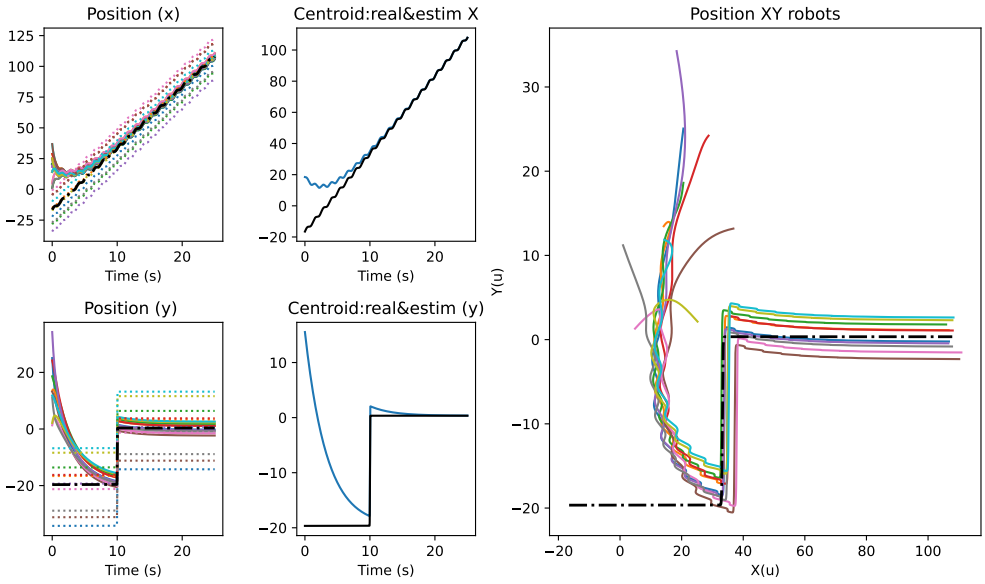


Figure 3.2: Simulation of a consensus of 10 Robots following 10 signals using basic dynamic consensus after adding a drift term. The condition $x_i(0) = u_i(0)$ remains unfulfilled. Again, the first column compares the x-y coordinates of agents, beacons and centroid. The middle column compares the x-y coordinates of the estimated and real centroids and the rightmost plot shows the evolution of the agents and the real centroid. This time although the robots converge towards the average signal and track its changes well, they still individually suffer an offset from the true average signal. The drift term causes the robots to slowly drift towards the true avg signal and compensate the offset, but after a while it stabilizes around a still existing but smaller offset.

The remaining drawback that still has not been fixed is the initial constraint, although the average centroid is tracked correctly, the robots do not fully converge their positions towards it, as it can be seen in Figure 3.2. If we had decided not to require that, the problem would be solved. Since we want the robots to converge to the average centroid, the algorithm is still not adequate.

This may be solved by adding a second equation to the system, leaving a new algorithm that will be further studied on section 3.4.

3.3 Dynamic consensus for directed graphs

On *Tutorial on Dynamic Average Consensus*[9] a different algorithm is proposed.

$$\dot{\mathbf{q}}_i = \alpha\beta \sum_{j=1}^{N} a_{ij} \left(\mathbf{x}_i - \mathbf{x}_j\right), \tag{3.11}$$

$$\dot{\mathbf{x}}_i = -\alpha \left(\mathbf{x}_i - \mathbf{u}_i\right) - \beta \sum_{j=1}^{N} a_{ij} \left(\mathbf{x}_i - \mathbf{x}_j\right) - \mathbf{q}_i + \dot{\mathbf{u}}_i, \tag{3.12}$$

$$\mathbf{x}_i \left(t_0\right),\ \mathbf{q}_i \left(t_0\right) \in \mathbb{R} \quad s.t \quad \sum_{i=1}^{N} \mathbf{q}_i \left(t_0\right) = 0. \tag{3.13}$$

This achieves convergence for any set of values of the tunable parameter α. One of the biggest arguments in favor of this implementation is the optionality of knowledge of the Laplacian. In this method, agents are not required to know their respective columns of the Laplacian. It does not require exchange with both in and out neighbors.

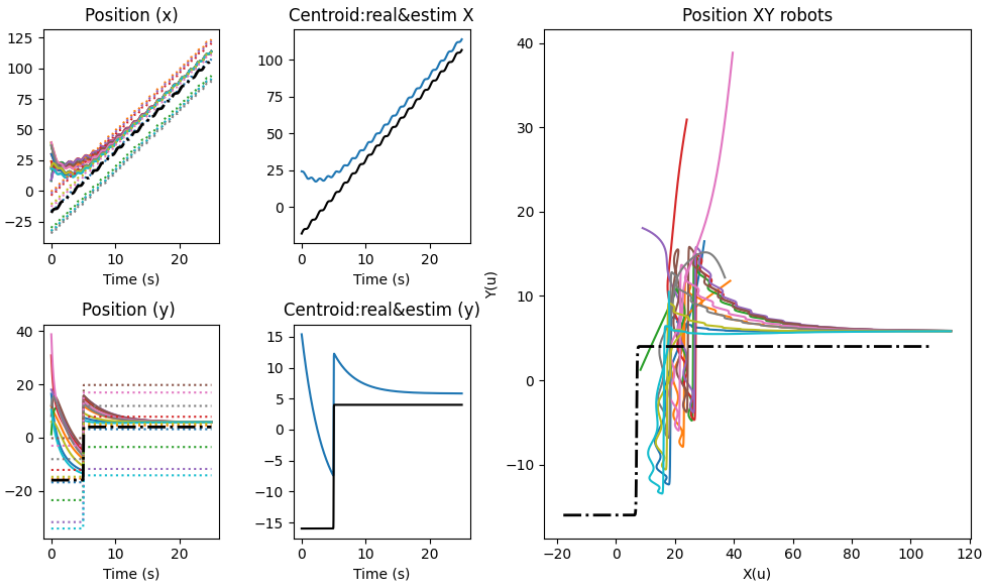


Figure 3.3: Simulation of a consensus of 10 Robots following 10 signals using the proposed dynamic consensus while not fulfilling $\sum \mathbf{q}_i(0) = 0$. It can be seen that although the robots converge towards the average signal and track its changes well, they suffer an offset from the true average signal, smaller than in other cases, but an offset remains.

The resulting implementation in discrete time is as follows:

$$\mathbf{q}_i(k) = \mathbf{q}_i(k-1) + \tau\alpha\beta \sum_{j=1}^{N} a_{ij}\left(\mathbf{x}_i(k-1) - \mathbf{x}_j(k-1)\right), \tag{3.14}$$

$$\mathbf{x}_i(k) = \mathbf{x}_i(k-1) + \tau\dot{\mathbf{x}}_i(k), \tag{3.15}$$

$$\begin{aligned}\dot{\mathbf{x}}_i = \frac{\mathbf{u}_i(k) - \mathbf{u}_i(k-1)}{\tau} - \alpha\left(\mathbf{x}_i(k-1) - \mathbf{u}_i\right)(k-1) - \\ - \beta \sum_{j=1}^{N}\left(\mathbf{x}_i(k-1) - \mathbf{x}_j(k-1)\right) - \tau\mathbf{q}_i(k),\end{aligned} \tag{3.16}$$

$$\begin{aligned}\mathbf{x}_i(k) = \mathbf{x}_i(k-1) + \Delta\mathbf{u}_i(k-1) - \tau\alpha\left(\mathbf{x}_i(k-1) - \mathbf{u}_i(k-1)\right) - \\ - \tau\beta \sum_{j=1}^{N}\left(\mathbf{x}_i(k-1) - \mathbf{x}_j(k-1)\right) - \tau\mathbf{q}_i(k).\end{aligned} \tag{3.17}$$

After running its corresponding simulation we obtain the Figure 3.3 which represents the evolution of a set of robots that like in the case of Figure 3.1 converge. Unlike the first case, the final offset around which the robots stabilize is smaller, and therefore the estimated average signal is closer to the true one. However there is still an offset, regardless of how small it is, which depending on our goal could be extremely problematic.

3.4 Dynamic consensus independent of initial conditions

In this section we present and briefly comment another consensus algorithm that does not require the initial states of the robots to fulfill a set of initial constraints. In vector form it is:

$$\dot{\mathbf{q}}_i = -\mathbf{L}_I\mathbf{q}, \tag{3.18}$$

$$\dot{\mathbf{x}}_i = \alpha\left(\mathbf{x}_i - \mathbf{u}_i\right) - \mathbf{L}_p\mathbf{x}_i + \mathbf{L}_I^T\mathbf{q}_i + \dot{\mathbf{u}}_i, \tag{3.19}$$

$$\mathbf{q}_i(t=0),\ \mathbf{x}_i(t=0) \in \mathbb{R}.$$

Notice that two different Laplacian matrices $\mathbf{L}_I, \mathbf{L}_p$ are used. This will in turn let the agents have an extra degree of freedom to follow a better tracking. Once the

discretization has been applied we get the resulting algorithm for dynamic consensus :

$$\mathbf{x}_i(k) = \mathbf{x}_i(k-1) + \tau \dot{\mathbf{x}}_i(k), \tag{3.20}$$

$$\mathbf{q}_i(k) = \mathbf{q}_i(k-1) + \sum_{j=1}^{N} b_{ij}\left(\mathbf{x}_i(k-1) - \mathbf{x}_j(k-1)\right), \tag{3.21}$$

$$\dot{\mathbf{x}}_i = \frac{\Delta \mathbf{u}_i(k-1)}{\tau} + \alpha\left(\mathbf{x}_i(k-1) - \mathbf{u}_i(k-1)\right) + \tag{3.22}$$
$$+ \sum_{j=1}^{N} a_{ij}\left(\mathbf{x}_i(k-1) - \mathbf{x}_j(k-1)\right) + \sum_{j=1}^{N} b_{ij}\left(\mathbf{q}_i(k) - \mathbf{q}_j(k)\right),$$
$$\mathbf{q}_i(t=0),\ \mathbf{x}_i(t=0) \in \mathcal{R}.$$

Where $\Delta \mathbf{u}_i(k) = \mathbf{u}_i(k) - \mathbf{u}_i(k-1)$ and b_{ij} is equivalent to the a_{ij} variable for the Laplacian of $\mathbf{q}$. This more complicated algorithm allows convergence without requiring the initial position of the robots to fulfill any conditions other than being real values, which makes it more versatile than the other implementations shown before.

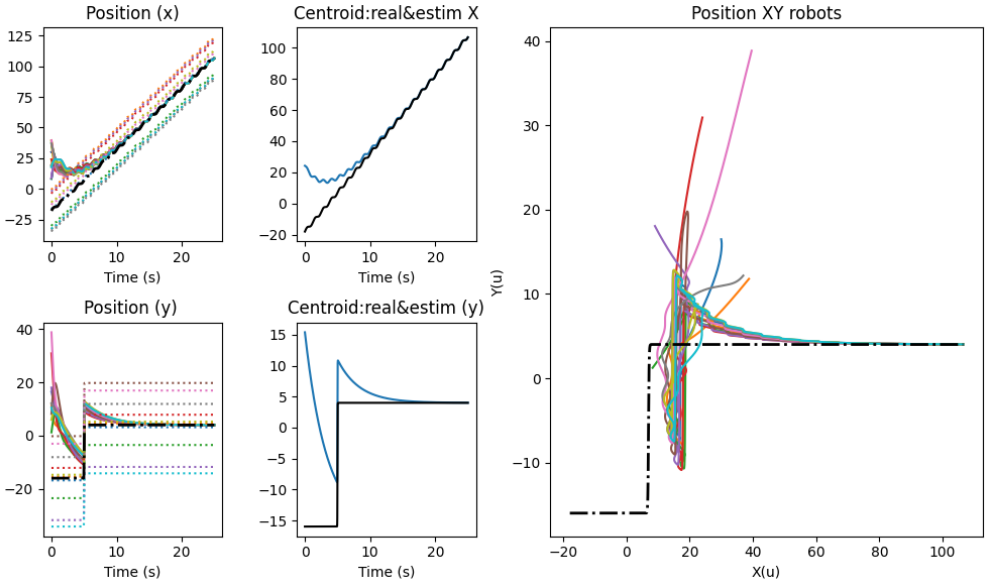


Figure 3.4: Simulation of a consensus of 10 Robots following 10 signals using the proposed dynamic consensus. No initial condition is required. As it can be seen the robots converge among each other and converge towards the true average signal they follow, yielding the best results so far.

As shown in Figure 3.4, the results obtained through this algorithm are the best ones so far, despite being the most complex one. For this reason we elected to use this algorithm whenever a simulation is conducted and is not otherwise specified, this will mostly happen whenever the simulation refers to a deformable body.

In short, this algorithm is more complex than those shown before and is less intuitive to explain and understand, but its results are significantly more robust due to not

needing to fulfill any initial constraints. It also depends on two different Laplacian matrices that indirectly reflect the communication network of the robots unlike the algorithm of section 3.3, which only requires a single Laplacian matrix. That, a priori, is a fundamental difference between both methods and a clear weakness of this implementation. However this is not of great concern. Using for both variables $\mathbf{L}_I$ and $\mathbf{L}_p$ the same Laplacian matrix still allows the system to converge, but at a slower pace. In fact, the simulation shown in Figure 3.4 illustrates a system where both Laplacian matrices are the same ($a_{ij} = b_{ij}$). As can be seen, convergence is reached without any issues.

3.5 Selecting the best algorithm

All the presented algorithms have their own strengths and weaknesses. We have decided to use the last one of them (section 3.4). It provides the best results and no offset appears at any moment, regardless of the values of the variables at $t = 0$. Although this algorithm involves two different Laplacian matrices in its formulation, it can also be used with a single system matrix, which alleviates the main disadvantage that was noted in relation to the algorithm discussed in section 3.3. The latter also needed to initialize $\sum_{i=1}^{N} \mathbf{q}_i(t_0) = 0$. Which is still a weakness compared to our chosen algorithm.

The first algorithm introduced is not suitable since it also requires a t_0 initialization of the variables. Without this initialization the tracking suffers an offset that we would like to avoid.

The second algorithm, that in , demonstrates great tracking. Unlike the first and third algorithms it introduces no offset but has a weakness that our chosen algorithm overcomes. The robots of the second algorithm do not converge to the centroid, they converge around it. Their average position coincides with the centroid. This is a behavior, that while useful in some cases is not what we are looking for.

In short, we have chosen the last algorithm for the experiments. This choice is because it provides the best results, with no offset at any time, regardless of the initial values. Additionally, it avoids the need for a t_0 initialization of variables, unlike the rest algorithms, and ensures accurate tracking without any of the drawbacks of the other methods.

3.6 Alternative formulation z

Introduced by Kia et al.[9], another way of formulating the different consensus algorithms is through the use of the intermediate variable

$$\mathbf{z}_i(k) = \mathbf{x}_i(k) - \mathbf{u}_i(k). \tag{3.23}$$

The main advantage of this method is that, by using the $\mathbf{z}$ variable as claimed by the original authors, the individual robots stop working with their state $\mathbf{x}_i$ and work exclusively with $\mathbf{z}$. This allows them to avoid the need for knowledge of the future (in our case real-time) signal $\mathbf{u}_i(k)$ during their computation of their own working variable $\mathbf{z}_i$ during the consensus. Which is useful in scenarios such as those where the reference is sampled from another online algorithm.

However, this does not solve the issue of requiring knowledge of the future signal (one step ahead) if one aims to determine the robot's state. If we try to compute the state of the robot $\mathbf{x}_i$ we will still require knowing the value of the signal $\mathbf{u}_i(k)$. Even if we have managed to compute $\mathbf{z}_i(k)$ without requiring it. One needs to know $\mathbf{u}_i(k)$ to add it to the previously computed $\mathbf{z}_i(k)$ to know the state of a robot.

This representation may be useful for the robots because they no longer need to keep track of their states, instead working solely with the parameter $\mathbf{z}$. For us, as users and experiment designers, it provides no particular advantages. For that reason we elected to choose the base formulation when implementing our code, since it saves us from having to recalculate $\mathbf{x}$ from $\mathbf{z}$ and the final results remain the same.

Because we do not use this formulation we decided not to include the formulation of the algorithms with this new auxiliary variable. Instead, they are included as part of Appendix A for those interested

3.7 Vector or individual formulation of the algorithm

Several algorithms were introduced in the earlier sections of this chapter. All of these can be written and implemented in two different forms. The first one being the vector representation of the algorithm, the second one, the distributed individual form of the algorithms.

The individual distributed algorithm is more appropriate when applying these algorithms to a real multi-robot system. Each robot i would work with its own copy

of the algorithm focusing only on itself and that information relevant to it. This is a realistic implementation of the situation at hand and leverages the distributed computing power of having several robots working simultaneously. It also does not require all information to be shared among robots or any other entities, saving memory. Our simulations and experiments have been done in a single computer since the fundamental objective of this work is to study both the algorithms and the problem to be solved. Estimating the centroid of a deformable object, analyzing and ideally assessing some indicator variables, such as the rate of convergence or the expected uncertainty. Implementing the algorithms into real robots is not the goal of the experiments at this stage. Conducting all the experiments in a single computer allows great savings in both implementation time and simplicity, however it comes with some trade-offs.

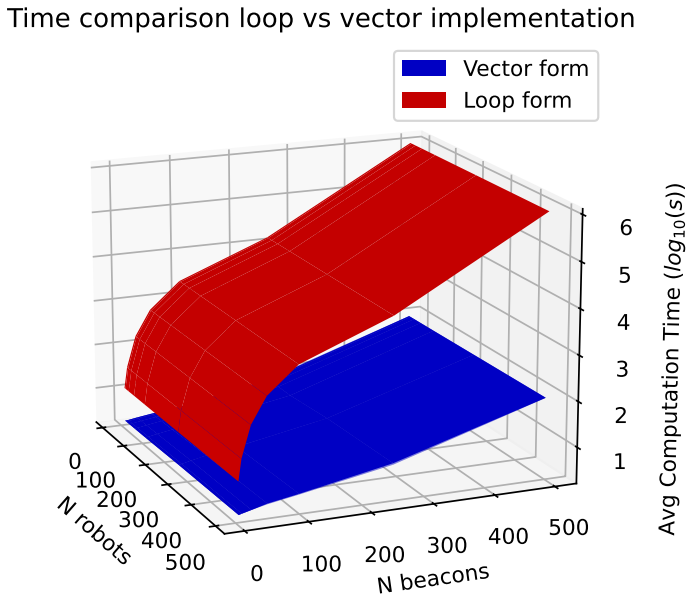


Figure 3.5: Comparison of the computation times as a function of the number of robots and beacons to track for the vector algorithm (blue) and the distributed loop algorithm (red). The distributed algorithm requires far greater amounts of time for its computations.

The distributed algorithms require more computational power and therefore simulation time than the vector expressions. The computer must iterate over every virtual robot and simulate as if it were said robot, so the natural parallelization of the problem by having different robots plays no role here. This in turns means that the loop-filled distributed algorithm takes more computation time than the vector

representation, especially in our chosen programming language. Python mathematical libraries tend to be optimized to work better and faster with matrix operations rather than the somewhat slow default loop.

The time improvements are quite notable, especially as the amount of robots and beacons increases. We may visualize the time increase and its dependency with the number of robots of the system by running some simple experiments and computing the time cost. We repeat this several times with both algorithms in order to create a map representing the evolution. We illustrate it in Figure 3.5. In both cases, the simulations use the robust algorithm to avoid comparing different algorithms and avoid any issues that may arise there. The initial conditions, such as the state of the robots $\mathbf{x}_i$, their $\mathbf{q}_i$ values and the position of the beacons $\mathbf{x}_m$ from which each robot computes its reference signal $\mathbf{u}_i$ are initialized randomly.

In the case of the vector algorithm we plot the average times computed from 30 different instances of the experiment and their $\pm\sigma$ values. For the distributed algorithm, no statistical average is shown. Each simulation took considerably longer than the vector case, meaning that repeating each simulation 30 times was unfeasible. Only a single simulation instance is shown.

Another advantage of using vector notation for this particular study is the compactness of the formulation, which allows us as experiment designers to more easily access, analyze and study the variables of the system with time. Some of these variables are the robot states $\mathbf{x}_i$, the signals values $\mathbf{u}_i$ or even the true centroid of the object $\mathbf{x}_{cen}$, which we may easily calculate due to our complete control and knowledge of the simulation parameters and values.

Chapter 4

Convergence analysis and discretization constraints

It is of great importance that our chosen system works and therefore converges. The number of agents to be used, time step of the simulation and the connectivity between agents and between agents and beacons are all important factors that play a role in the feasibility of accomplishing the task at hand.

If the set of variables that comprise our proposed solution do not have adequate values, consensus will not be reached, which in turn means that no centroid is estimated, instead the agents states and its estimations may diverge.

To better study this problem, we will mathematically analyze the valid ranges for the simulation parameters that ensure convergence.

4.1 Ensuring convergence for the dynamic consensus algorithm

For the sake of length in this section we will mathematically analyze and provide bounds for the tunable parameters for only the drift algorithm (section 3.5).

We specifically choose that one because it is the one that provides the best balance between complexity and interest. It is not as simple as the basic dynamic consensus but its analysis is not as complex and elaborate as that of the robust algorithm. In contrast with the rest of algorithms shown, this one has not been studied in the bibliography previously. We give a mathematical estimation of the conditions whose fulfillment is required to ensure convergence for the algorithm.

If the reader were interested in the error estimation and analytical analysis of the remaining algorithms, the expected error and convergence rate are all studied in the bibliography.

Let us then begin with the analysis of the drift algorithm:
Given that the communication graph of the agents is bidirectional, unweighted and strongly connected, its associated connectivity matrix has some interesting properties which we may leverage in order to estimate a range of values for the different parameters to ensure its convergence. The eigenvalues of the system in section 3.5 are:

$$\lambda(\mathbf{M}) = (1 - a\alpha) - a\lambda_i(\mathbf{L}). \tag{4.1}$$

The $\mathbf{L}$ indicates the Laplacian matrix related to the graph. Due to the properties of Laplacian matrices it is fulfilled that its eigenvalues follow:

$$0 = \lambda_i \leq \lambda_2 \leq \lambda_3 \leq ... \leq \lambda_n \leq ... \leq \lambda_{max}, \tag{4.2}$$

$$0 \geq -a\lambda_2 \geq -a\lambda_3 \geq ... \geq -a\lambda_n \geq ... \geq -a\lambda_{max}, \tag{4.3}$$

$$1 \geq 1 - a\alpha - a\lambda_2 \geq 1 - a\alpha - a\lambda_3 \geq ... \geq 1 - a\alpha - a\lambda_n \geq 1 - a\alpha - a\lambda_{max}. \tag{4.4}$$

Where a is a tunable parameter in the simulation that meets $a \geq 0$.

In order to advance with the required conditions we may glean some different results from the same expressions.

$$\begin{cases} 1 - a\alpha \leq 1 \implies -a\alpha \leq 0 \implies a\alpha > 0 \\ 1 - a\alpha \geq 1 - a\alpha - a\lambda_{max} \implies a\lambda_{max} \geq 0 \implies a \geq 0 \because \lambda_{max} \geq 0 \\ 1 - a\alpha - a\lambda_{max} \geq -1 \implies 2 \geq a(\alpha + \lambda_{max}) \\ 1 - a\alpha - a\lambda_i \leq 1 \implies a(\alpha + \lambda_i) \geq 0 \\ 1 - a\alpha - a\lambda_i \geq -1 \implies a(\alpha + \lambda_i) \leq 2 \end{cases} \tag{4.5}$$

Therefore we arrive at

$$0 \leq a(\alpha + \lambda_i) \leq a(\alpha + \lambda_{max}) \leq 2. \tag{4.6}$$

We then compute that $\frac{2}{a} - \lambda_i \geq \alpha$ which is valid as long as $a \neq 0 \implies a > 0$. Since $a > 0$ and in undirected graphs $\lambda_i \geq 0$ then it follows that $\alpha \in (-\lambda_i, \frac{2}{a} - \lambda_i)$. The α represents the attraction between the robot and its measured reference signal, so having a negative value would mean that it is repelled, the opposite of our goal. So $\alpha \in (0, \frac{2}{a} - \lambda_i)$. From that expression we may also compute a range for a, taking the extreme value of $\alpha = 0$, then $\frac{2}{a} - \lambda_i > 0 \implies a < \frac{2}{\lambda_i}$.

Considering Equation 4.2 we may compute the range of a by computing its limits through the biggest (λ_{max}) and smallest (λ_i) non-zero eigenvalues.

$$a \in \left(\frac{2}{\lambda_{max}}, \frac{2}{\lambda_2}\right). \tag{4.7}$$

The property of $a \geq 0$ originates from its original definition in the paper *Convergence speed of dynamic consensus with delay compensation*[3]. Where instead of a the tunable parameters are defined as β and τ. In our case we elected to rewrite them as $a = \beta * \tau$. The valid range of both the original parameters indicate that they have both to be greater than 0.

Concretely $\tau \in \left(0, \frac{1}{\beta\mathcal{N}}\right)$ as τ represent the time step size of the simulations and β fulfills $\beta > 0$. $\mathcal{N}$ is the number of robots in the system.

We may therefore compute the valid range of values of a.

$$a = \tau\beta \implies \beta = \frac{a}{\tau}, \tag{4.8}$$

$$0 \leq \tau \leq \frac{1}{\beta\mathcal{N}} \implies 0 \leq \tau \leq \frac{1}{\frac{a}{\tau}\mathcal{N}}, \tag{4.9}$$

$$\tau \leq \frac{\tau}{a\mathcal{N}} \implies a\mathcal{N} \leq 1. \tag{4.10}$$

So $a\mathcal{N} \leq 1$. We also obtained before that $a < \frac{2}{\lambda_2}$. Therefore both conditions are required to be fulfilled for whichever value of a is used.

$$a \leq \frac{1}{\mathcal{N}} \wedge \frac{2}{\lambda_{max}} \leq a \leq \frac{2}{\lambda_2}. \tag{4.11}$$

These conditions hold for any kind of graph shape and movement of the centroid. They only depend on the nature of the connectivity graph and Laplacian.

We have derived a set of conditions that relate the tunable parameters of the simulation with the expected convergent behavior. In short, in the case of the algorithm shown in section 3.5, we may ensure its convergence and reaching a consensus as long as the following conditions happen:

$$\begin{aligned}
&G = (V, E), \quad E \subseteq V \times V, \\
&\forall u, v \in V, \exists \text{ a path from } u \to v \text{ and } v \to u, \\
&w(e) = 1 \,\text{for all}\, e \in E, \\
&0 \leq \alpha \leq \frac{2}{a} - \lambda_i, \\
&a \leq \frac{1}{\mathcal{N}}, \\
&\frac{2}{\lambda_{max}} \leq a \leq \frac{2}{\lambda_2}, \\
&a = \tau\beta, \quad \tau > 0, \quad \beta > 0,
\end{aligned}$$

with the first three conditions just meaning that the communication graph of the agents is bidirectional, strongly connected and unweighted, with an associated connectivity matrix M.

Therefore this validity range for the parameters will remain for any centroid we study and any set of robots, independent of how densely they communicate as long as the resulting connectivity graph is strongly connected, bidirectional and unweighted.

All simulations that are shown in this work and fulfill its associated algorithm convergence conditions, which may be different depending on the algorithm.

4.2 Discretization as a source of error

The dynamic consensus algorithms studied in this work and the bibliography are most of the time defined in a continuous time domain. While this allows the mathematical analysis to be easier to study it also means that they technically cannot be represented in computer experiments. Implementing any continuous experiment in a computer is impossible because computers they are inherently discrete machines.

However the experiments can be implemented. This is achieved by rewriting the dynamic consensus expressions in a discrete form, allowing the computer to work with those expressions, enabling us to run experiments.

The definition of derivative works discretely

$$\frac{df}{dt} = \lim_{\tau \to 0} \frac{f\left(t+\tau\right) - f\left(t\right)}{\tau}. \tag{4.12}$$

Evidently our τ will not infinitely tend to zero, but will end up with a finite value. From which an error will appear due to using the same expression.

That is, our discretization will follow the expression:

$$\frac{df}{dt} = \frac{f\left(k*\tau+\tau\right) - f\left(k*\tau\right)}{\tau}. \tag{4.13}$$

Also known as the Euler discretization method.

Where k is the current simulation step and τ the time interval of each step. It follows that $t = k*\tau$. We would like to quantify the error that this representation has.

In order to do so let us define the Local Truncation Error (LTE) and the Global Accumulated Error (GAE). The LTE is defined as the quantity between the true value (that is, the value we would obtain if we were using continuous time) and the value obtained via the discretization.

The GAE is the sum of all the LTE occurred in every former step of the simulation up until the current one.

The LTE value may be estimated from the Taylor series around the point of a function [8]:

$$y(t_0+\tau) = y(t_0) + \tau y'(t_0) + \frac{1}{2}\tau^2 y''(t_0) + \mathcal{O}\left(\tau^3\right). \tag{4.14}$$

The LTE then is proportional to τ^2. The GAE is the sum of all the former LTE and is the difference between the true value and the estimated value using Euler $GAE = y_{true} - y_k = \sum^k \frac{1}{2}\tau^2 y''(t_0) + \mathcal{O}(\tau^3)$.
Since the number of steps computed depend on the time step $k = \frac{t_i - t_0}{\tau}$ the sum is $\left(\frac{1}{2}\tau^2 y''(t_0) + \mathcal{O}(\tau^3)\right)\frac{t_i - t_0}{\tau}$. Which leaves us with a negligible error of $\mathcal{O}(\tau^2)$ and the main source of error in Euler integration is

$$\frac{1}{2}\tau y''(t_0)(t_i - t_0) \propto \tau(t_i - t_0). \tag{4.15}$$

The error associated with the Euler discretization method is linearly dependent on both the step size and the duration of the simulation. This error is larger compared to more refined methods, such as midpoint discretization or the Runge-Kutta family of integration algorithms, though these methods are typically more costly to implement.

It is crucial to choose a sufficiently small time step τ for the simulation to avoid errors that could invalidate the results.

Which τ is adequately small for our experiments? In our case, we chose the time step to be $\tau = 10^{-4}$ after applying two different accuracy tests. Firstly we used the algorithm introduced by Douglas Wilhelm [14] in his course *Numerical methods* to estimate an optimal time step balancing precision and efficiency. To verify the result of the test, we also compared the analytical and computational results using the τ given by the first test. We did so for a simplified problem to be able to confirm that the given time step with the first step returns results adequately accurate compared with the analytically expected values.

We do not fully include these algorithms in the body of this thesis, as the main goal of this work is to study the dynamic consensus of a deformable body, not estimating the introduced Euler error. Although the tests indicated that much larger time steps that finally chosen we opted for $\tau = 10^{-4}$ as a more cautious choice to ensure robustness and reliability throughout the simulations. The simulation times for this τ value were also not particularly long, which helped make that decision. For those interested in further details of each of the algorithms or wishing to verify these results, a full explanation and implementation of the tests used are included in the appendix Appendix B.

Chapter 5

Deformable system. Analysis and experiments

In this section we derive an analytical expression that provides us with a maximum possible error between the estimated centroid by the robots and the true estimated centroid. This expression defines the upper limit of the disagreement for worst case scenarios. After that we will proceed with the experiments and simulations made for the robust algorithm.

5.1 Bounds for the maximum expected error

It is assumed that in this specific case each beacon is viewed by one robot, so no beacons are left unattended.

Departing from the expression computed in *Tutorial on Dynamic Average Consensus*[9] we may build upon said computation to provide maximal error for our specific scenario.

Let

$$|e^i(i)| \leq \kappa e^{-\lambda(t-t_0)} \left\| \begin{bmatrix} \mathbf{w}_{2:N}(t_0) \\ \overline{\mathbf{e}}(t_0) \end{bmatrix} \right\| + \frac{\kappa \|\mathbf{B}\|}{\lambda} \sup_{t_0 \leq \tau \leq t} \left\| \left(\mathbf{I}_N - \frac{1}{N} \mathbf{1}_N \mathbf{1}_N^T \right) \dot{\mathbf{u}}(\tau) \right\|. \quad (5.1)$$

be the error boundary for the robust algorithm of a non-deformable dynamic system. In our case we want to study a scenario whose signals are not perfect. We have elected to model this as an oscillation around the true signal, therefore $\mathbf{u}_{def} = \mathbf{u} + \Sigma$, where Σ_i is the oscillation of the i-th beacon and is defined as $\Sigma_i = A \cos\alpha_i \sin(t + \phi_i)$. The first term is $\cos\alpha_i$ or $\sin\alpha_i$ depending on the coordinate of the deformation. α_i and ϕ_i are parameters specific to each beacon, that rule the direction around which the beacon oscillate and its initial phase. This ensures that the beacons do not move uniformly.

From the chosen modeling of the beacons and its corresponding signals it follows that

$$\frac{\kappa \left\| \mathbf{B} \right\|}{\lambda} \sup_{t_0 \leq \tau \leq t} \left\| \left(\mathbf{I}_N - \frac{1}{N} \mathbf{1}_N \mathbf{1}_N^T \right) \dot{\mathbf{u}} \left(\tau \right) \right\| \rightarrow$$
$$\rightarrow \frac{\kappa \left\| \mathbf{B} \right\|}{\lambda} \sup_{t_0 \leq \tau \leq t} \left\| \left(\mathbf{I}_N - \frac{1}{N} \mathbf{1}_N \mathbf{1}_N^T \right) \dot{\mathbf{u}} \left(\tau \right) + \left(\mathbf{I}_N - \frac{1}{N} \mathbf{1}_N \mathbf{1}_N^T \right) \dot{\mathbf{\Sigma}} \left(\tau \right) \right\|. \tag{5.2}$$

Due to space we will rename the right side of the second expression as $\frac{\kappa \left\| \mathbf{B} \right\|}{\lambda} \sup_{t_0 \leq \tau \leq t} \left\| A \right\|$.

Thanks to the triangle inequality $\left\| \mathbf{x} + \mathbf{y} \right\| \leq \left\| \mathbf{x} \right\| + \left\| \mathbf{y} \right\|$ it follows that

$$\sup_{t_0 \leq \tau \leq t} \left\| A \right\| \leq \sup_{t_0 \leq \tau \leq t} \left(\left\| \left(\mathbf{I}_N - \frac{1}{N} \mathbf{1}_N \mathbf{1}_N^T \right) \dot{\mathbf{u}} \left(\tau \right) \right\| + \left\| \left(\mathbf{I}_N - \frac{1}{N} \mathbf{1}_N \mathbf{1}_N^T \right) \dot{\mathbf{\Sigma}} \left(\tau \right) \right\| \right). \tag{5.3}$$

From one of the basic properties of the *supremum* $\sup \left(f + g \right) \leq \sup \left(f \right) + \sup \left(g \right)$, from which we obtain that

$$\sup_{t_0 \leq \tau \leq t} \left(\left\| \left(\mathbf{I}_N - \frac{1}{N} \mathbf{1}_N \mathbf{1}_N^T \right) \dot{\mathbf{u}} \left(\tau \right) \right\| + \left\| \left(\mathbf{I}_N - \frac{1}{N} \mathbf{1}_N \mathbf{1}_N^T \right) \dot{\mathbf{\Sigma}} \left(\tau \right) \right\| \right) \leq \tag{5.4}$$
$$\leq \sup_{t_0 \leq \tau \leq t} \left(\left\| \left(\mathbf{I}_N - \frac{1}{N} \mathbf{1}_N \mathbf{1}_N^T \right) \dot{\mathbf{u}} \left(\tau \right) \right\| \right) + \sup_{t_0 \leq \tau \leq t} \left(\left\| \left(\mathbf{I}_N - \frac{1}{N} \mathbf{1}_N \mathbf{1}_N^T \right) \dot{\mathbf{\Sigma}} \left(\tau \right) \right\| \right). \tag{5.5}$$

By substitution and the initial expression from Equation 5.1 we may compute a boundary expression as a function of the modeled beacons deformations with regard to the original undeformed problem.

$$\left| e^i_{Def} \left(t \right) \right| \leq \left| e^i \left(t \right) \right| + \frac{\kappa \left\| \mathbf{B} \right\|}{\lambda} \sup_{t_0 \leq \tau \leq t} \left(\left\| \left(\mathbf{I}_N - \frac{1}{N} \mathbf{1}_N \mathbf{1}_N^T \right) \dot{\mathbf{\Sigma}} \left(\tau \right) \right\| \right). \tag{5.6}$$

Due to the way the matrix is defined, the product inside the norm is

$$\left\| \left(\mathbf{I} - \frac{1}{N} \mathbf{1}_N \mathbf{1}_N^T \right) \dot{\mathbf{\Sigma}} \left(\tau \right) \right\| = \left\| \begin{pmatrix} b & c & \dots & c \\ c & b & \dots & c \\ \vdots & \vdots & \ddots & \vdots \\ c & c & \dots & b \end{pmatrix} \begin{pmatrix} \dot{\Sigma}_1 \\ \dot{\Sigma}_2 \\ \vdots \\ \dot{\Sigma}_N \end{pmatrix} \right\| = \left\| \begin{pmatrix} b\dot{\Sigma}_1 + c\dot{\Sigma}_2 + \dots + c\dot{\Sigma}_N \\ \vdots \\ c\dot{\Sigma}_1 + c\dot{\Sigma}_2 + \dots + b\dot{\Sigma}_N \end{pmatrix} \right\|. \tag{5.7}$$

Where $b = 1 - \frac{1}{N}$, $c = -\frac{1}{N}$ and each component $\dot{\Sigma}_i$ is the perturbance component that the signal u_i suffers and causes the deformation. Notice that we have written $\dot{\Sigma}$ rather than $\dot{\Sigma} \left(\tau \right)$. To simplify the notation we drop the $\left(\tau \right)$.

We may simplify further the expression at the cost of relaxing again the inequality through the triangle inequality.

$$\left\| \begin{pmatrix} b\dot{\Sigma}_1 + c\dot{\Sigma}_2 + \cdots + c\dot{\Sigma}_N \\ \vdots \\ c\dot{\Sigma}_1 + c\dot{\Sigma}_2 + \cdots + b\dot{\Sigma}_N \end{pmatrix} \right\| =$$

$$= \sqrt{\left(b\dot{\Sigma}_1 + c\dot{\Sigma}_2 + \cdots + c\dot{\Sigma}_N\right)^2 + \cdots + \left(c\dot{\Sigma}_1 + c\dot{\Sigma}_2 + \cdots + b\dot{\Sigma}_N\right)^2} \leq$$

$$\leq \left\| b\dot{\Sigma}_1 + c\dot{\Sigma}_2 + \cdots + c\dot{\Sigma}_N \right\| + \left\| c\dot{\Sigma}_1 + b\dot{\Sigma}_2 + \cdots + c\dot{\Sigma}_N \right\| + \cdots + \left\| c\dot{\Sigma}_1 + c\dot{\Sigma}_2 + \cdots + b\dot{\Sigma}_N \right\|.$$

Since $\|x_1 + \cdots + x_N\| \leq |x_1 + x_2 + \cdots + x_N|$ and $|x_1 + \cdots + x_N| \leq |x_1| + |x_2| + \cdots + |x_N|$ we may apply those conditions to the result, yielding:

$$\left\| b\dot{\Sigma}_1 + c\dot{\Sigma}_2 + \cdots + c\dot{\Sigma}_N \right\| + \left\| c\dot{\Sigma}_1 + b\dot{\Sigma}_2 + \cdots + c\dot{\Sigma}_N \right\| + \cdots + \left\| c\dot{\Sigma}_1 + c\dot{\Sigma}_2 + \cdots + b\dot{\Sigma}_N \right\| \leq$$

$$|b\dot{\Sigma}_1 + c\dot{\Sigma}_2 + \cdots + c\dot{\Sigma}_N| + |c\dot{\Sigma}_1 + |b|\dot{\Sigma}_2 + \cdots + c\dot{\Sigma}_N| + \cdots + |c\dot{\Sigma}_1 + c\dot{\Sigma}_2 + \cdots + b\dot{\Sigma}_N| \leq$$

$$b|\dot{\Sigma}_1| + |c||\dot{\Sigma}_2| + \cdots + |c||\dot{\Sigma}_N| + \cdots + |c||\dot{\Sigma}_1| + |c||\dot{\Sigma}_2| + \cdots + |b||\dot{\Sigma}_N|. \tag{5.8}$$

Note, that since c takes its absolute value it will now have a value of $c = \frac{1}{N}$.

By simply computing the sum of Equation 5.8 we obtain

$$\begin{aligned} &|\dot{\Sigma}_1|\left(b + (N-1)\,c\right) + |\dot{\Sigma}_2|\left(b + (N-1)\,c\right) + \ldots |\dot{\Sigma}_N|\left(b + (N-1)\,c\right) = \\ &= |\dot{\Sigma}_1|\left(\frac{N-1}{N} + (N-1)\,\frac{1}{N}\right) + \\ &\qquad + |\dot{\Sigma}_2|\left(\frac{N-1}{N} + (N-1)\,\frac{1}{N}\right) + \\ &\qquad + \cdots + \\ &\qquad + |\dot{\Sigma}_N|\left(\frac{N-1}{N} + (N-1)\,\frac{1}{N}\right) = \\ &= \left(2\frac{N-1}{N}\right)\left(|\dot{\Sigma}_1| + \cdots + |\dot{\Sigma}_N|\right) \end{aligned} \tag{5.9}$$

Trough applying some properties of the norm-2 and the triangle inequality we have reached then this expression from the original norm

$$\left\| \left(\mathbf{I} - \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^T\right)\dot{\mathbf{\Sigma}}\left(\tau\right) \right\| \rightarrow \left(2\frac{N-1}{N}\right)\left(|\dot{\Sigma}_1| + \cdots + |\dot{\Sigma}_N|\right), \tag{5.10}$$

the right part of it being a more conservative value of the right side one.

We may use said result to estimate a more conservative maximum error by replacing it in Equation 5.6:

$$|e^i_{Def}| \leq |e^i(t)| + \frac{\kappa\,\|\mathbf{B}\|}{\lambda} \sup_{t_0 \leq \tau \leq t} \left(\left(\frac{2\,(N-1)}{N}\right)\left(|\dot{\Sigma}_1| + ... + |\dot{\Sigma}_N|\right)\right). \tag{5.11}$$

Whose main benefit comes from not needing to do any matrix operations at the cost of increased cautiousness.

In the specific case of our study, we elected to model the deformation of our modeled as oscillations $\Sigma_i = A\cos(\alpha_i)\sin(t+\phi_i)$. Its derivative being $\dot{\Sigma}_i = A\cos(\alpha_i)\cos(t+\phi_i) \leq A$. And due to the image of the cosine functions being $Im(\cos) \in [-1,1]$ we may estimate the worst possible case for the sum of the oscillation components of the signals $|\dot{\Sigma}_1| + ... + |\dot{\Sigma}_N| \leq AN$.

By replacing this term in the supremum of Equation 5.11 we get

$$\left(\left(\frac{2(N-1)}{N}\right)\left(|\dot{\Sigma}_1| + ... + |\dot{\Sigma}_N|\right)\right) \leq \left(\frac{2*(N-1)}{N}AN\right) = 2A(N-1).$$

Which leaves us with a estimation of the deformable error that, known the associated non-deformable error $|e^i(t)|$ requires no *supremum* operator and grows linearly with the amount of agents N.

$$|e^i_{Def}| \leq |e^i(t)| + \frac{\kappa\|\mathbf{B}\|}{\lambda}2A(N-1). \tag{5.12}$$

For our default experiment $A = 1$.

5.1.1 Alternative bounds for the maximum error

We propose an alternative way of estimating the maximum error bound for the deformable body.

In this case we will apply the following norm-2 property: $\|x\|_2 \leq \sqrt{N}\|x\|_\infty$.

Applying the condition to the Equation 5.7 we obtain:

$$\left\|\begin{pmatrix} b\dot{\Sigma}_1 + c\dot{\Sigma}_2 + \cdots + c\dot{\Sigma}_N \\ \vdots \\ c\dot{\Sigma}_1 + c\dot{\Sigma}_2 + \cdots + b\dot{\Sigma}_N \end{pmatrix}\right\|_2 \leq \sqrt{N}\left\|\begin{pmatrix} b\dot{\Sigma}_1 + c\dot{\Sigma}_2 + \cdots + c\dot{\Sigma}_N \\ \vdots \\ c\dot{\Sigma}_1 + c\dot{\Sigma}_2 + \cdots + b\dot{\Sigma}_N \end{pmatrix}\right\|_\infty. \tag{5.13}$$

Where, as a reminder, the infinity norm is defined as $\|x\|_\infty = \max(|x_1|, ..., |x_N|)$.

It follows then that the right side of the Equation 5.13 may be rewritten as:

$$\sqrt{N}\left\|\begin{pmatrix} b\dot{\Sigma}_1 + c\dot{\Sigma}_2 + \cdots + c\dot{\Sigma}_N \\ \vdots \\ c\dot{\Sigma}_1 + c\dot{\Sigma}_2 + \cdots + b\dot{\Sigma}_N \end{pmatrix}\right\|_\infty = \sqrt{N}\sup_{0\leq i\leq N-1}\left|b\dot{\Sigma}_i + c\sum_{j\neq i}\dot{\Sigma}_j\right|. \tag{5.14}$$

Since we are working with the original values of $b = \frac{N-1}{N}$ and $c = -\frac{1}{N}$ it follows:

$$\sqrt{N}\sup_{0\leq i\leq N-1}\left|\frac{N-1}{N}\dot{\Sigma}_i + -\frac{1}{N}\sum_{j\neq i}\dot{\Sigma}_j\right| \tag{5.15}$$

and we may use this expression for the bound of the maximum error, akin to Equation 5.11. Which leaves us with:

$$|e^i_{Def}| \leq |e^i(t)| + \frac{\kappa\|\mathbf{B}\|}{\lambda}\sup_{t_0\leq\tau\leq t}\left(\sup_{0\leq i\leq N-1}\left(\left|\frac{\sqrt{N}(N-1)}{N}\dot{\Sigma}_i - \frac{\sqrt{N}}{N}\sum_{j\neq i}\dot{\Sigma}_j\right|\right)\right). \tag{5.16}$$

As it can be seen the boundary expression is definitely more cumbersome since it requires a double *supremum* search, or in other words, searching the supremum of two different variables, the time τ and the $i-th$ element of the deformation vectors.

However as we will show next, this method provides a less conservative bound than that of Equation 5.11 under certain conditions.

5.1.2 Comparison between boundary expressions

We will briefly study when each of the expression for the maximum error perfmors better (those in Equation 5.11 and Equation 5.16).

We may do so since we can rewrite Equation 5.11 so that it the same form as that of Equation 5.16.

$$\sup_{t_0\leq\tau\leq t}\left(\sup_{0\leq i\leq N-1}\left(\frac{2\left(N-1\right)}{N}|\dot{\Sigma}_i|+\frac{2\left(N-1\right)}{N}\sum_{j\neq i}\left|\dot{\Sigma}_j\right|\right)\right). \tag{5.17}$$

As it is evident, the *supremum* operator associated to i is variable because the contents of its parenthesis amount to the sum of all elements of the vector multiplied by $\frac{2(N-1)}{N}$, but in writing it like so it allows us to better reflect the structure of Equation 5.16 and define a comparison.

Besides the *supremum*, the rest of the expressions are equal, so we may simply compare the contents of each *supremum* to view which expression leaves us with a smaller error. Let $\dot{\mathbf{\Sigma}}$ be a vector of deformation function which we apply to both expressions. Since the vector and its components $\dot{\Sigma}_i$ are the same in both cases, we may compare them by just focusing on those elements dependent on the number of robots N.

From that we may then define the functions

$$f(x)=\frac{2\left(x-1\right)}{x}+\frac{2\left(x-1\right)}{x},\quad g(x)=\text{abs}\left(\frac{\sqrt{x}\left(x-1\right)}{x}-\frac{\sqrt{x}}{x}\right).$$

When $f(x)>g(x)$, the error associated to $f(x)$ and therefore Equation 5.11 provides a bigger maximum error estimation and vice versa. We may compare both methods and compute the best ranges of N for each method by subtracting both functions.

When $f(x)>g(x)$ and therefore $f(x)-g(x)>0$, $f(x)$ estimates a greater maximum error, which in turn means that $g(x)$ provides a less conservative estimate and its associated expression (Equation 5.16) would be preferable for that number of robots. The equation $h(x)=f(x)-g(x)$ may be easily solved graphically.

In Figure 5.1 both expressions have been compared. From the presented logic, the most appropriate expression for small amounts of robots -up to 18- is the one shown

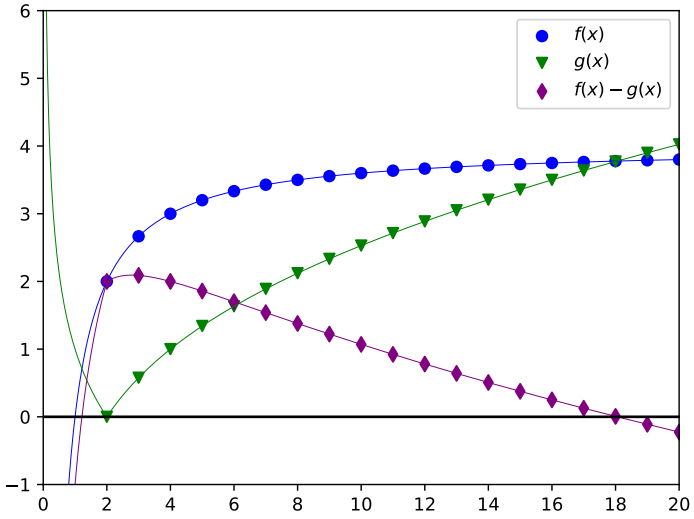


Figure 5.1: Each expression as a function of the amount of robots and its comparison by means of subtraction. Shown in markers. For up to $N = 18$ robots, $f(x)$ provides a more cautious error estimation. For greater amounts $f(x) - g(x)$ crosses 0 so $g(x)$ provides more conservative estimations. The continuous equivalent of the expressions is also shown to better view their tendencies.

in Equation 5.16. As N increases, the expression of Equation 5.11 becomes better performing until it outperforms the alternative for $N > 18$.

It worth mentioning that both Equation 5.11 and Equation 5.16 provide a maximum value for the disagreement between the centroid of the deformable object estimated by each robot and the true position of the centroid. In reality the final error encountered in simulations will usually be far smaller than that provided by the expression. However, they do give an upper limit for it as a worst case scenario.

5.1.3 Estimated bounds and real error

Firstly we will compare how well-performing the expression Equation 5.12 is. In doing so we will conduct a simulation over time and compare the actual disagreement between the estimated centroid of each robot and the true centroid. We will then compare it with the maximum allowed error given by the boundary expression. Additionally, we will analyze the contribution of each term of the final sum in Equation 5.11 is.

We chose to only focus on the expression from Equation 5.12 because it is significantly simpler to implement.

From Figure 5.2 one may see that the first addend of the boundary expression, which we will call T_1 starts being the biggest source of it but decays exponentially. In the figure is shown as the blue area. The second term T_2 is the green area. Its

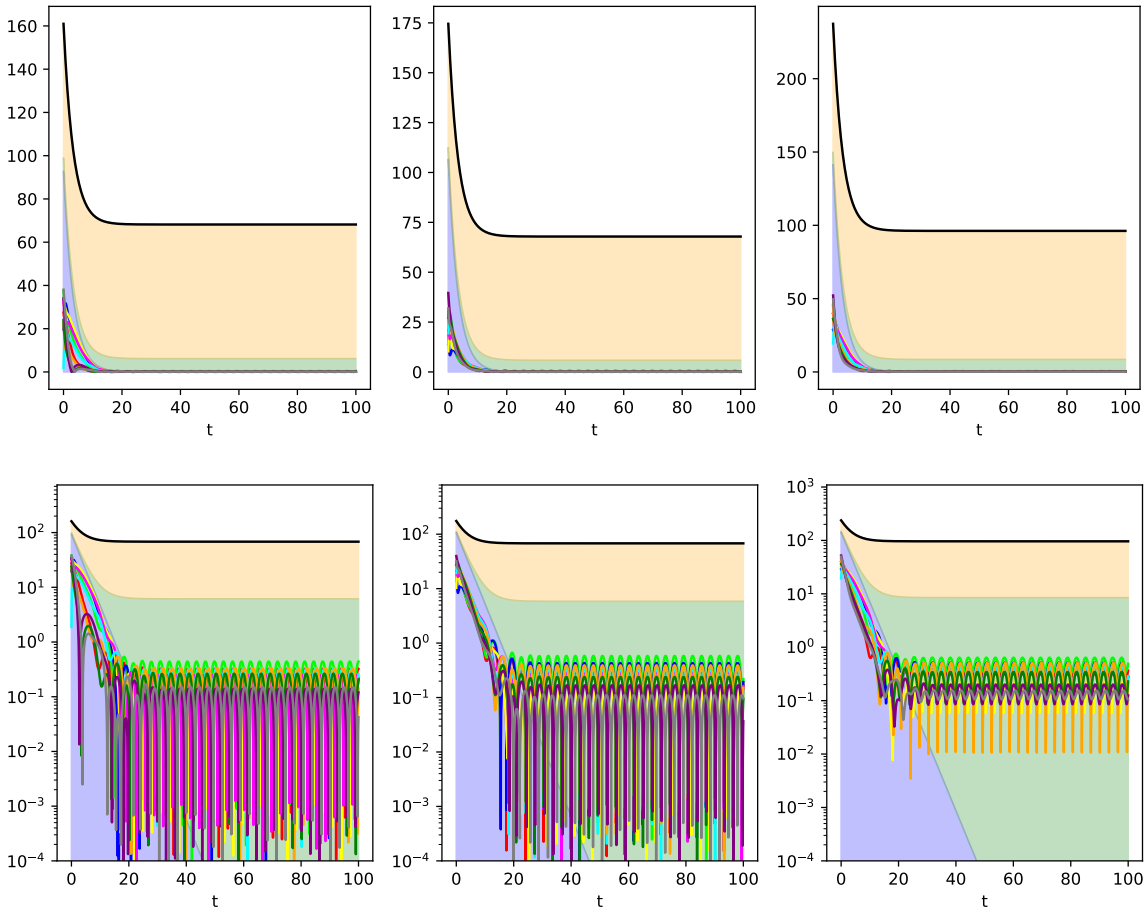


Figure 5.2: Disagreement of each robot (solid lines) and the bounded maximum disagreement (colored areas). The different terms of the bounded disagreement are uncoupled to view their individual impact. Over time the biggest influence on the bound ends up being from the new term derived for deformations. Each plot shows the error in x, y and $||\mathbf{x}||$ coordinates. In the top row the disagreement is displayed in linear scale, which allows to easily gauge how much of the error each source is. In the bottom row the scale is logarithmic. This allows a better comparison of the signals in regard to the upper bound.

value does not increase over time, except for the replacement of the decaying first term. Surprisingly this behavior also occurs in the y-coordinate despite the sudden jump coded in the movement. Finally, the last term T_3 which we derived for the deformation is completely constant the whole time. Not surprising, $\frac{\kappa||B||}{\lambda} 2A\,(N-1)$ does not depend on time.

We also compare in Figure 5.3 the final bound provided by the Equation 5.12 and contrast it with the actual final disagreement. We simulate the system until $t = 500$, which we consider as an approximation of $t \to \infty$. From there, we calculate the final disagreement and the boundary error using the formula we derived from Equation 5.12.

We study the effect of the number of robots $N \in (2, 25)$ and plot the behavior.

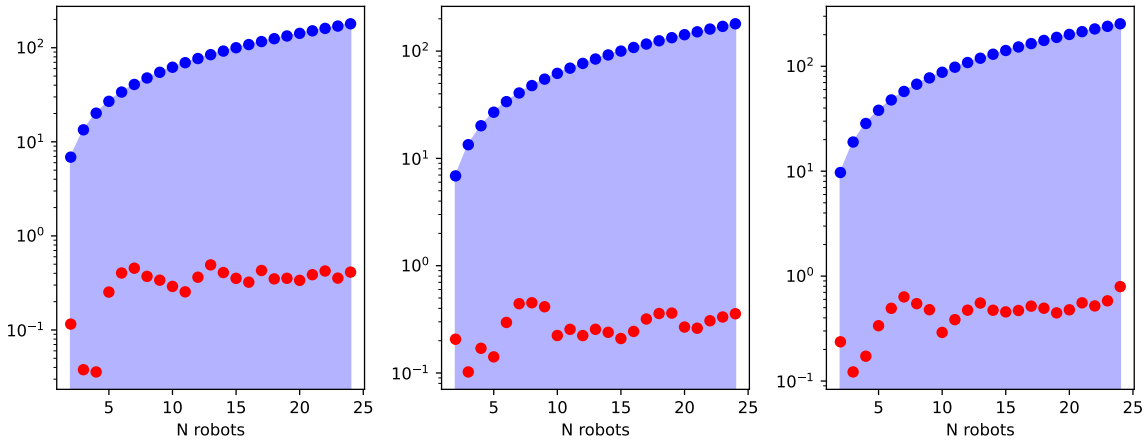


Figure 5.3: Final disagreement for $t = 500s$ (red markers) and its associated upper bound (blue area) for different amounts of robots. From left to right e_x, e_y and $||e||$ The upper bound error is always an overestimation and grows even larger as N increases.

As shown Figure 5.3 the final disagreement is always smaller than the estimation provided by the bound expression. In some cases the difference is of several orders of magnitude. The computed expression is then cautious at the cost of poorer accuracy. It also grows as N grows.

It is important to note, that the shown results are not technically fully correct. The boundaries shown are only the component $\frac{\kappa||B||}{\lambda}2A\left(N-1\right)$ of the whole Equation 5.12. We chose not to include the other term referring to the error from the undeformed problem. We decided to just plot the deform component, since in this case is enough to guarantee that the full result will be an upper bound.

We proceed to explain why. From equation Equation 5.12 the upper bound is:

$$|e^i_{Def}(t)| \leq |e^i(t)| + \frac{\kappa||B||}{\lambda}2A(N-1). \tag{5.18}$$

From the original bound equation for the undeformed problem Equation 5.1 we get

$$|e^i_{Def}(t)| \leq \kappa e^{-\lambda(t-t_0)} \left\| \begin{bmatrix} \mathbf{w}_{2:N}\left(t_0\right) \\ \overline{\mathbf{e}}\left(t_0\right) \end{bmatrix} \right\| + \tag{5.19}$$

$$+ \frac{\kappa\left\|\mathbf{B}\right\|}{\lambda} \sup_{t_0 \leq \tau \leq t} \left\| \left(\mathbf{I}_N - \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^T \right) \dot{\mathbf{u}}\left(\tau\right) \right\| + \frac{\kappa||\mathbf{B}||}{\lambda}2A(N-1). \tag{5.20}$$

Which is a sum of three terms T_1, T_2, T_3. By definition, the norm of any matrix or vector $||\mathbb{R}^{n\times m}|| \geq 0$. The first term is a norm, exponentially damped with time so $T_1 \geq 0$. The second term T_2 is the norm of a matrix and the fraction $\frac{\kappa||B||}{\lambda} > 0$ by definition of κ, λ [9]. The *supremum* of a positive function is greater or equal to zero. So $T_2 \geq 0$ too.

Since neither the number of robots N nor the amplitude of oscillations can be negative, the third term T_3 will also be positive.

It follows then that $T_1 \leq T_1 + T_2 \leq T_1 + T_2 + T_3$ and also $T_1 \leq T_1 + T_2 \leq T_1 + T_2 + T_3$ As long as $T_3 \geq e_{real}$, we can be confident that the rest of the expression will maintain an upper bound. Therefore, we can focus solely on T_3 to qualitatively observe how the upper bound behaves as N grows, keeping in mind that the other two terms will further increase the upper bound. Including the additional terms would not provide significant new insights, aside from increasing the value of the blue area, and computing the *supremum* would unnecessarily complicate the implementation.

5.1.4 Conclusions

In this section of the chapter we have introduced two different expressions to estimate the maximum upper bound of the error between the centroid of the deformable body and the estimated centroid by the robots.

While this expressions differ in complexity, both have their place, one performs better for low amounts of robots (Equation 5.16), while other is more efficient as the number of pairs increases (Equation 5.12).

We also studied how closely the upper bound predicted by one of the formulas (Equation 5.12) aligns with the actual error observed in experiments. Our results showed that the upper bound tends to be quite large, indicating that reducing the gap between the upper bound and the true error could be a promising direction for future research.

Furthermore, our experimental observations confirm that the upper bound increases as the number of robot-beacon pairs in the system grows, while the true final error does not exhibit any strong trends.

5.2 Deformable body - Experiments

In this section we present the results obtained from simulating the original posed problem. A deformable body that follows a certain trajectory, and its centroid is estimated by a group of agents, each viewing a single beacon. We will firstly simulate a baseline scenario and viewing which results appear and which conclusions and hypothesis we may develop from those.

The parameters of the simulation are shown in Table 5.1. As it can be seen the algorithm to be used will be that of section 3.4 and the connectivity between robots will be a circle graph, that is, each robot communicates exclusively with its two neighbors. We gently remind that α is a tunable parameter of the algorithm that controls the attraction between robot and signal. In this case we have written m_x and m_y referring to translational movement of the deformable body in its respective coordinates. It may be surprising the m_y component since it models a sudden jump by means of the sigmoid function. As explained at the beginning of chapter 3 this type of behavior is not really expected in most tracking applications, since the tracked goal does not usually have almost instantaneous spatial jumps, however we decided to include it anyway to simulate and study if the algorithm works still even when facing difficult odds, like losing for a moment the tracked objective or a third party having moved the body.

The last column corresponds to the modeling of the deformation effect that each beacon is subjected to.

Those parameters indicated as $\sim \mathcal{R}$ are those whose value is randomly initialized, just like with the initial states of the agents $\mathbf{x}_i$, the initial positions of the beacons $\mathbf{x}_m$ and the initial values of the variable $\mathbf{q}_i$. As a reminder, we work the case where there are as many robots as beacons and each robot views a beacon, which means $\mathbf{x}_m = \mathbf{u}_i$.

However, it is important to mention that the initial agent states $\mathbf{x}_i$, the beacon positions $\mathbf{u}_i$ and $\mathbf{q}_i$ remain equal for every experiment. That is, if we for example

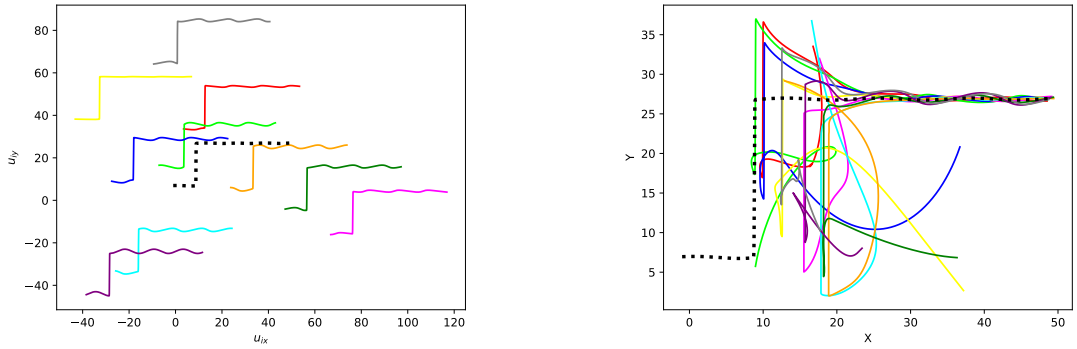
Tabla 5.1: Default values used for the experiment. Unless otherwise specified, all experiments have been done following this characteristics.

Algorithm	*section* 3.4	t_{max}	25s	Deform	$A_i \cos\theta_i \sin(f_i t + \varphi_i)$ $A_i \sin\theta_i \sin(f_i t + \varphi_i)$
N_{robots}	10	α	0.3	Amplitude	$A_i = 1$
$N_{beacons}$	10	m_x	$2t$	Direction	$\theta_i \sim \mathcal{R}$
Connectivity	Circle graph	m_y	$\frac{20}{1+e^{-100*(t-5)}}$	Frequency	$f_i = 1$
τ	10^{-4}	L_p	$L_p = L_i$	Phase	$\varphi_i \sim \mathcal{R}$

want to compare the influence of the amplitude on the convergence, those variables are generated randomly and kept for all the amplitude values we aim to study.

5.3 Baseline experiment

In this section we show the experimental results for the baseline simulation parameters.



(a) Positions of the beacons (solid) and centroid position (dotted).

(b) Positions of the robots (solid) and centroid position (dotted).

Figure 5.4: Movement of the beacons and the robots as time passes. The robots reach a consensus and converge around the average position of the beacons, the centroid.

The experimental results are satisfactory, as shown in Figure 5.4. The robots converge around the average position of the beacons despite each beacon being affected by independent deformations. Even so it can be seen that the convergence is not absolute, the robots oscillate around the convergence point. This is shown in Figure 5.5.

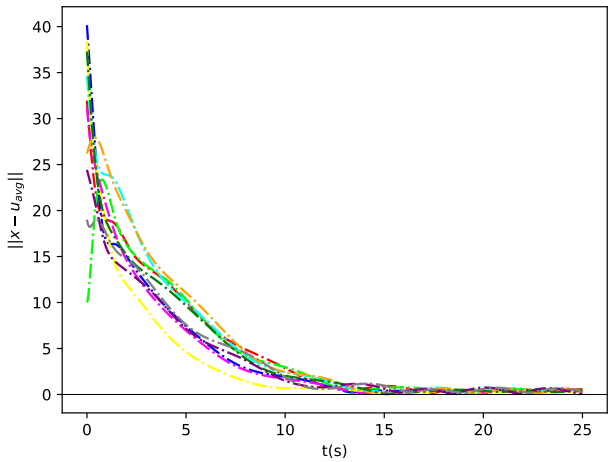


Figure 5.5: Disagreement between the position of each robot and the average position of the beacons. Ideally as t increases it should converge towards zero. The deformation of the beacons causes the disagreement to oscillate around zero.

This is undoubtedly caused by the oscillations of the signals. Each robot moves by taking into account two things, the position of the rest of the robots, which makes them cluster together and it also follows it associated beacon. Since the beacons oscillate and the part of the movement of the robots is about replicating the movement of the beacons they also oscillate. Since the phase at which each beacon, and in turn each robot, are different, the deformations do not cancel it each other out, leaving us with a new source of error.

We may see that oscillating error by graphing the disagreement between the position of each robot and the position of the centroid. Since we are working 2-dimensionally, we plot the euclidean distance for the disagreement.

The disagreement falls towards zero, but at no point it remains stable on zero. It oscillates due to the effect of the deformation function on each of the beacons. This oscillation of the disagreement is bounded by the amplitude of the deformation oscillations. In this case $A = 1$, so the maximum error after convergence of an individual robot and the centroid is no larger than $e_{max} = A = 1$.

5.3.1 Different oscillations

We may study the problem when the oscillations are characterized by different expressions. In our case we elected to keep the deformations periodic and bounded. We also chose some functions that are not differentiable in their whole domain to study whether that has an effect on the robustness of the algorithm and its convergence.

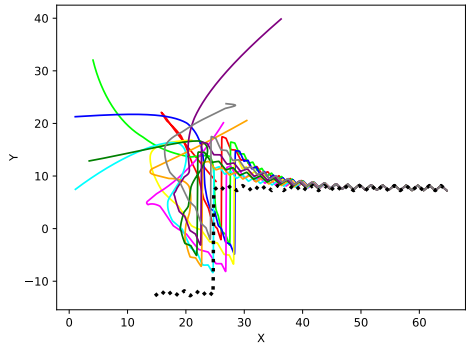
We will show then the effects of replacing the sinusoidal deformation with:

- **A triangular wave**: Shown in Figure 5.6
- **A sawtooth wave**: Shown in Figure 5.7
- **A square wave**: Shown in Figure 5.8
- **An extremely oscillating wave**: Shown in Figure 5.9 In this case, rather than trying to simulate a somewhat realistic movement for the beacons to follow, we attempt to create an unrealistic signal to study how robust the algorithm is.

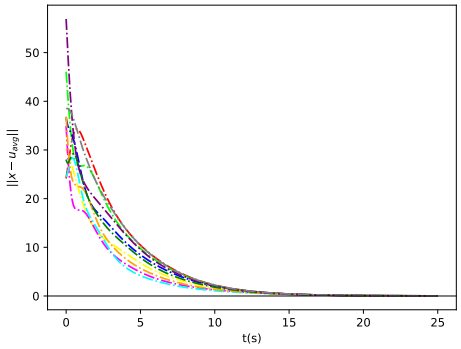
  It works as follows. The deformation of each signal is either $A = 5$ or $-A = -5$, the initial value being decided randomly, then on each iteration step (so, every 10^{-4} seconds of simulation time) the sign of the deformation of each beacon flips. We do so for both the x-coordinate and the y-coordinate.

  In this case, as shown in Figure 5.9, despite all beacons starting with the same phase an error end ups appearing due to the extreme oscillation. After computing it the offset with regard to the true centroid is $e \approx 7.07$.

From the fact that the value of the oscillation has been defined as $A = 5$, its euclidean distance of the disagreement would be computed as $e = \sqrt{A_x^2 + A_y^2} = \sqrt{50} \approx 7.07$. In this extreme cases it appears that the agents converge to a consensus whose estimated centroid has an error equal to the euclidean distance of the different coordinate components.

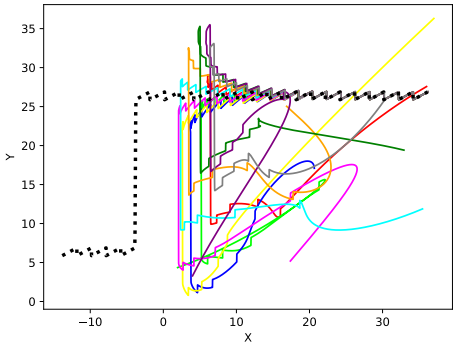


(a) Positions of the robots (solid) and centroid position (dotted).

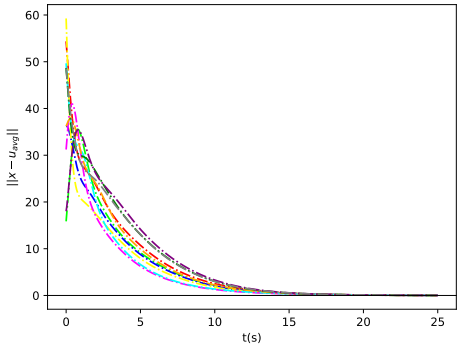


(b) Disagreement between the centroid and each robot position.

Figure 5.6: Results from the triangle wave. Despite it being not being differentiable in its whole domain, the centroid tracking is achieved. No residual error happens due to the deformation because all the triangle signals begin with the same phase.

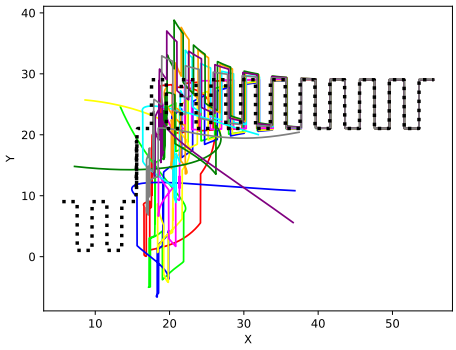


(a) Positions of the robots (solid) and centroid position (dotted).

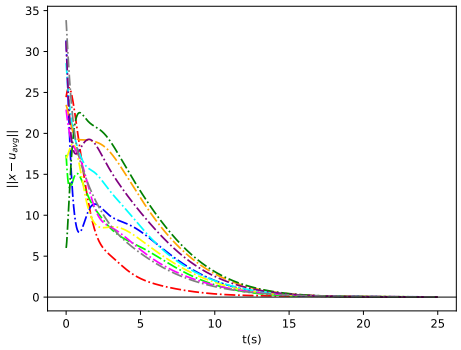


(b) Disagreement between the centroid and each robot position.

Figure 5.7: Results from the sawtooth wave. Despite it being not being differentiable in its whole domain, the centroid tracking is achieved. No residual error happens due to the deformation because all the sawtooth signals begin with the same phase.

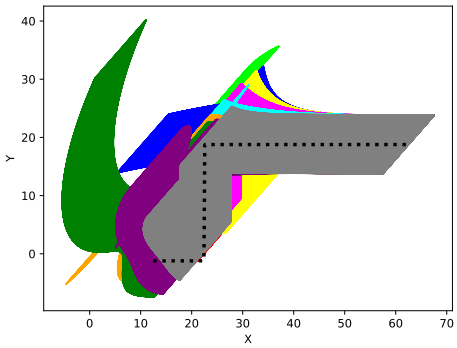


(a) Positions of the robots (solid) and centroid position (dotted).

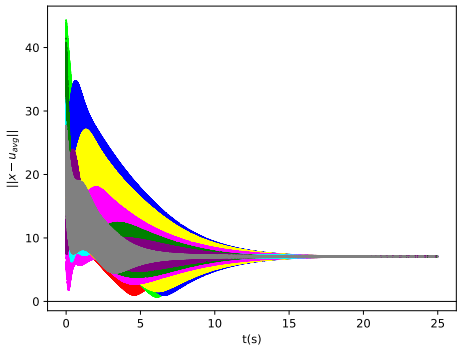


(b) Disagreement between the centroid and each robot position.

Figure 5.8: Results from the square wave. Despite it being not being differentiable in its whole domain, the centroid tracking is achieved. No residual error happens due to the deformation because all the square signals begin with the same phase.



(a) Positions of the robots (solid) and centroid position (dotted).



(b) Disagreement between the centroid and each robot position.

Figure 5.9: Results from the flipping wave. This time the perfect tracking is not achieved. Unlike the other scenarios an error appears.

5.4 Parameters of the simulation

In this section we study the influence of some of the different parameters defined in the simulation and what effects they have in system.

The main aspects we will study in each case beside reviewing the simulation results after changing the value of the parameter will be the average disagreement in each of the cases and its convergence rate. We have defined the convergence rate as the time it takes for the global disagreement of the robots to be a fraction of the initial disagreement. Both agents and beacons are initialized at N random positions.

Therefore a $CR_{0.1}$ will be the time in seconds that it takes for the robots to converge to a state where the average disagreement is 0.1 times the initial disagreement.

5.4.1 Number of agents and beacons

Firstly we will study how the complexity of the system affects the consensus. We will do so by changing the amount of agents and beacons that take part in the experiment. They will however, remain with a ratio of 1 to 1, each beacon is viewed by one robot.

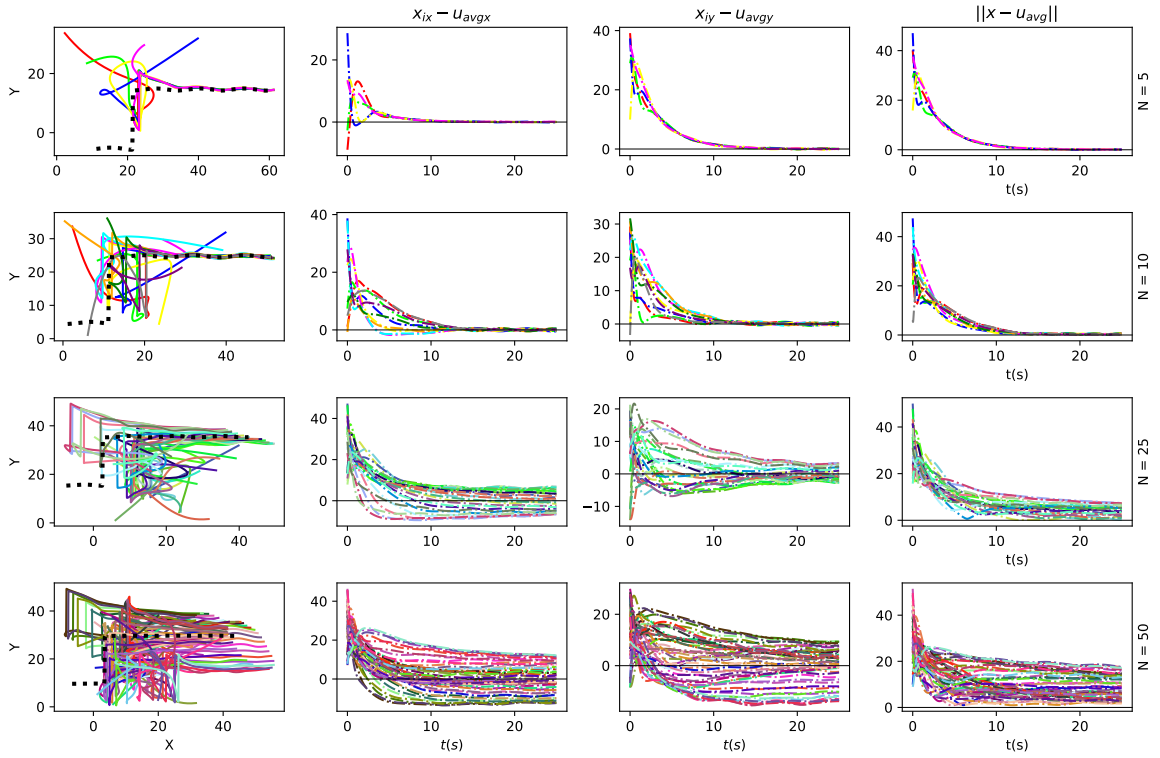


Figure 5.10: Results obtained when comparing the same system with 5, 10, 15, 25 robots and beacons. The convergence time increases as the complexity of the system grows.

By increasing the number of robots and beacons of the system we are also increasing the complexity of the whole. It is reasonable that the convergence time increases just like it happens in our simulations.

The average disagreement of the robots paints a similar picture as shown in Figure 5.11. Notice how the oscillating minimum error remains, due to the oscillating deformation of the beacons and the disagreement falls slower the greater the amount of entities in our system. The convergence rate grows at different speeds. We show it in the convergence rate graph in Figure 5.12.

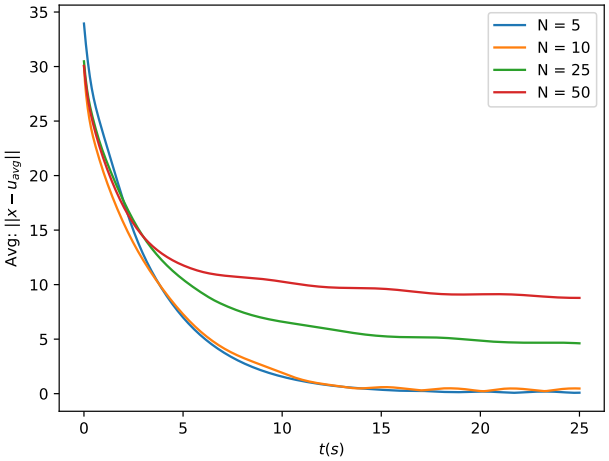


Figure 5.11: Average disagreement between robots and centroid. The more robots the more complex the system and the slower the disagreement shrinks.

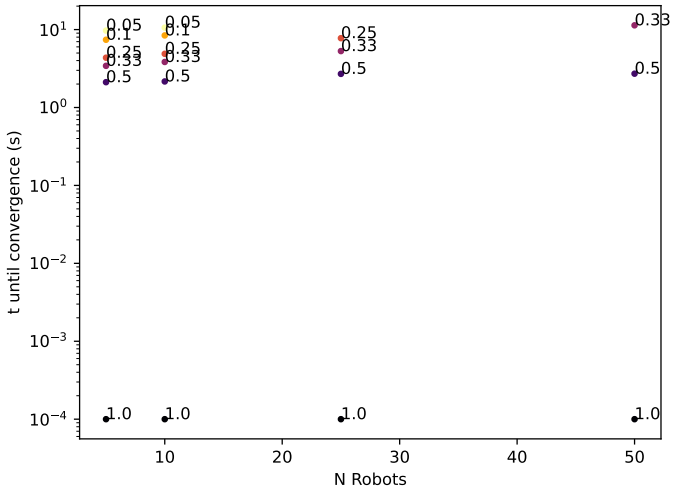


Figure 5.12: Time of simulation until a CR_x is reached. The difference CR grow at different speeds. The lines uniting those points of same have slightly different slopes

As the number of robots increases, so does the convergence time. This is especially noticeable for lower CR values. Those who imply the smallest disagreement between robots and the true centroid of the deformable body grow quite quickly, in fact, for most the cases of 25 and 50 robots, that convergence is not reached within a simulation time of $t = 25$ seconds.

5.4.2 Amplitudes

We will study how impactful the amplitudes of the oscillations from which we have modeled the deformations are. We will change the magnitude of the amplitude and we will also study a case where the amplitude of each beacon is random.

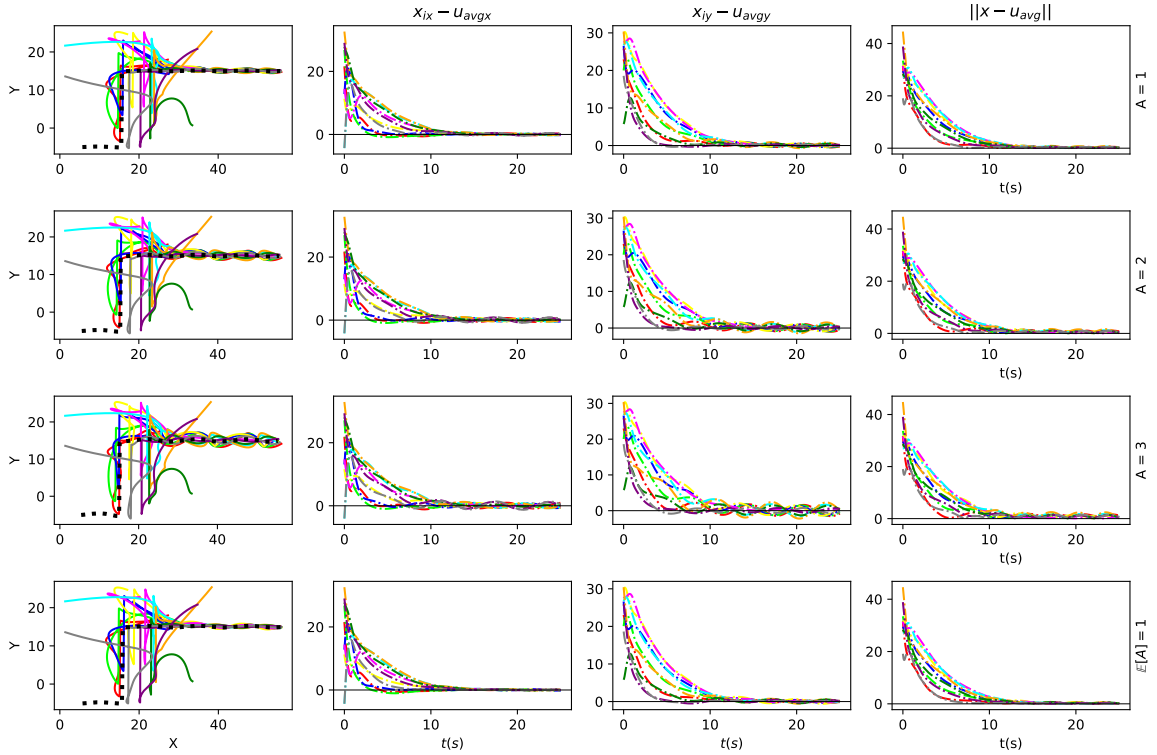


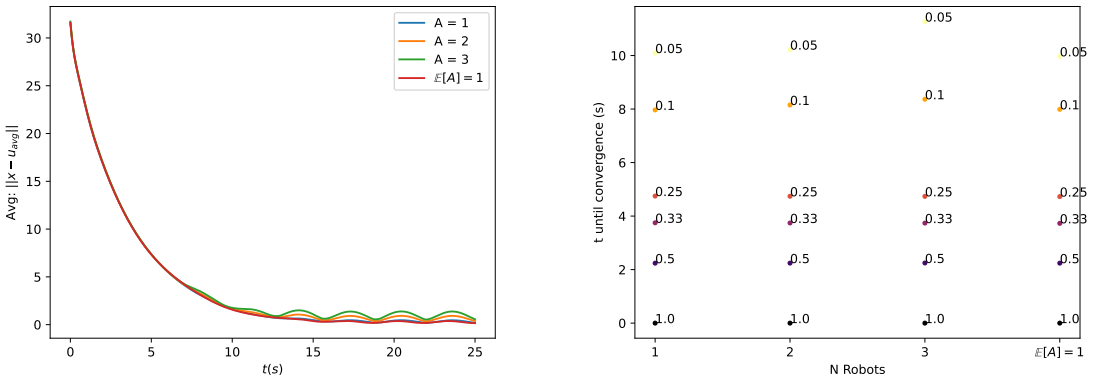
Figure 5.13: Results obtained when changing the amplitude magnitude. First row is default amplitude and next two are after multiplying it by a factor of 2 and 3. In the final row the amplitudes where each generated randomly from a uniform distribution of expected value 1. Besides some small difference in the disagreement, no changes in the convergence rate.

We see in this scenario that the only effect the amplitude of the oscillations has on the experiments happens on the disagreement. As it appears, the greater the amplitude of the oscillations the higher the disagreement ends up being. We may better appreciate it from the average disagreement.

As shown in Figure 5.14, the convergence rate does not seem to be correlated with the amplitude of the oscillations.

The disagreement between the estimated and true centroids is influenced by the amplitude, especially after the robots have converged, otherwise the main error source comes from the initial position of the robots. Once the robots have converged, the maximum disagreement increases as the amplitude grows.

This somewhat supports the bound for the error we estimated earlier in section 5.1.



(a) Disagreement between the estimated centroid and the real one for different-sized amplitudes.

(b) Studied influence for different oscillation amplitudes. No particular difference in CR for the amplitudes is found.

Figure 5.14: Studied influence for different oscillation amplitudes.

We observe a maximum error that depended linearly on $2A\,(N-1)$, similar to the results from our experiments. Do keep in mind that we are not showing the maximum error, which is the actual value Equation 5.12 provides, but it is useful to compare to observe that it indeed grows linearly as the amplitude increases just like the upper bound for the error. Since the disagreement grows linearly with the amplitude, we would expect the upper bound for the error to grow linearly too (or even faster), otherwise, there would be a point where the real error exceeds the upper bound, which is not consistent with the definition of upper bound.

5.4.3 Connectivity

Let us now study what is the effect of changing the connectivity between robots. We expect the convergence rate to fall as communication density increases. In other words, the more connected and communicated the robots are between each other, the sooner we expect them to converge and estimate the centroid because they can exchange information with more agents at every moment.

For the connectivity, we increase the neighbor reach. Starting with the initial base circle graph, we gradually increase the number of neighbors each robot can communicate with, until every robot is able to communicate with every other robot. We also do the extreme scenario where robots are not allowed to communicate with each other.

From Figure 5.15, initially increasing the number of neighbors accelerates the convergence speed, having a noticeable effect when increasing from $0 \rightarrow 2$ neighbors and from $2 \rightarrow 4$ neighbors. However, as shown in Figure 5.16, further increases in the

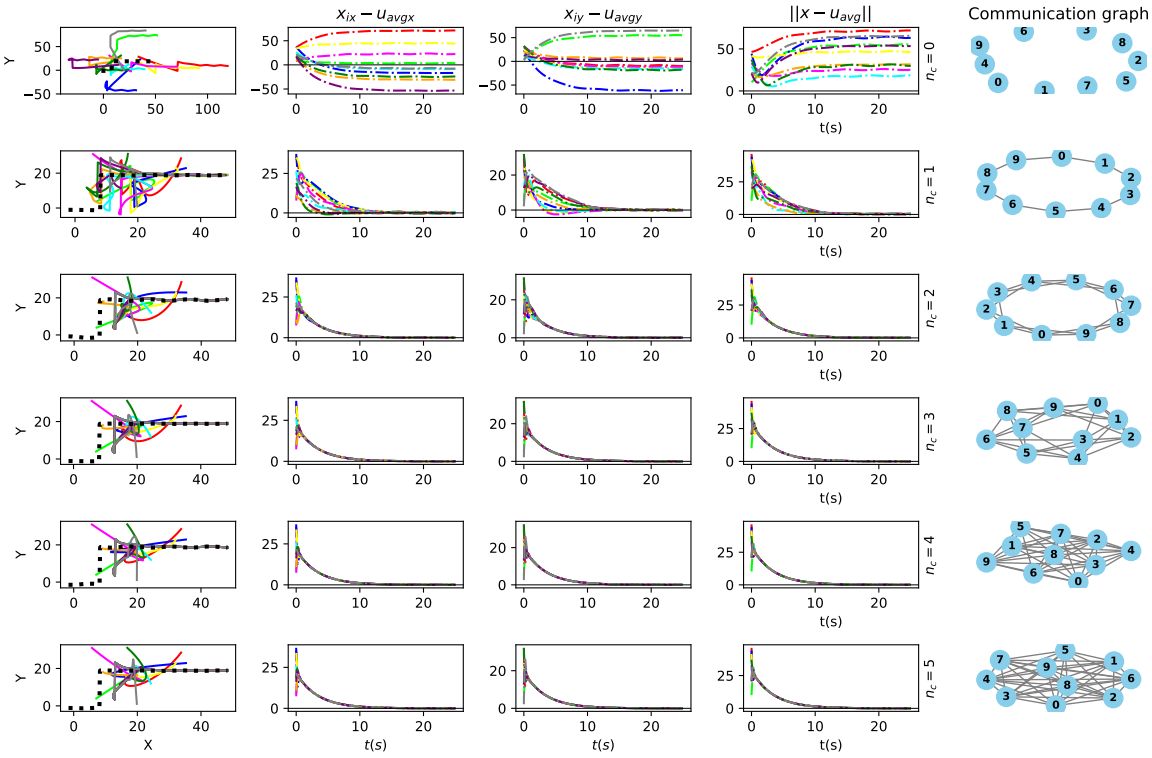
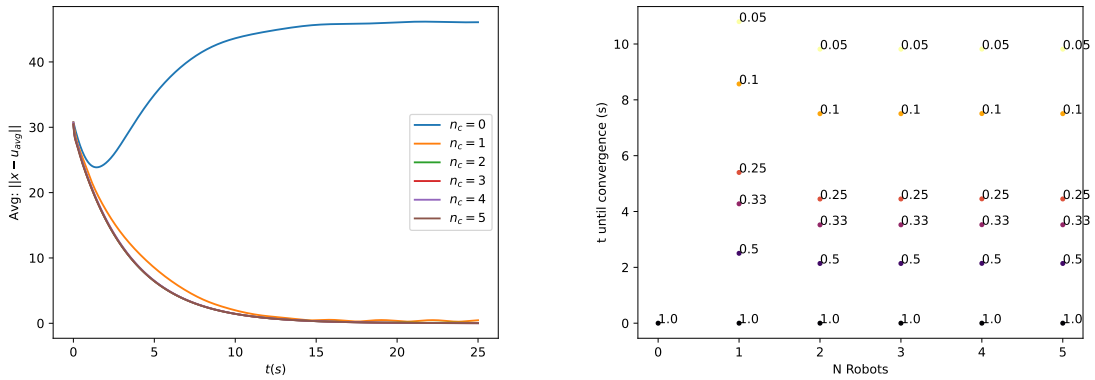


Figure 5.15: At first sight it appears that higher connectivity causes a faster convergence. However in reality, there is no noticeable difference between allowing each robot to have 4 or more neighbors. The connectivity graph in each case is shown as the last column.

number of the connected neighbors does not provide any improvements.



(a) Disagreement between the estimated centroid and the real one for different amounts of neighbors.

(b) Convergence rate for different amounts of neighbors. For the first cases it lowers, no further effects.

Figure 5.16: Studied influence for different connectivity degrees.

Unsurprisingly, in the case of 0 neighbors, the error does not improve over time. In this particular scenario no communication is allowed, there is no consensus to be

reached.

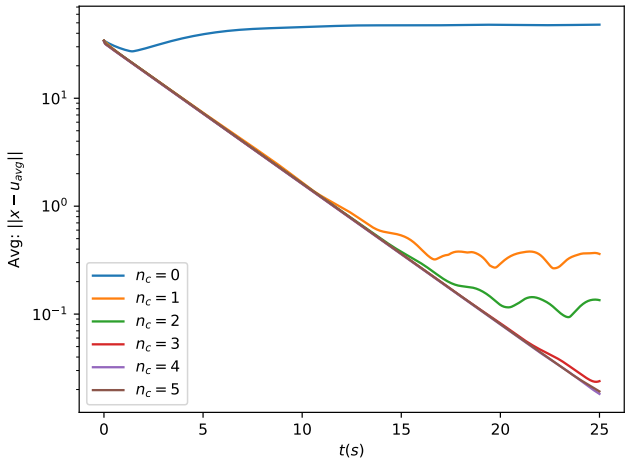


Figure 5.17: Centroid estimation error for differently connected graphs. The more interconnected, the better the estimation is.

An improvement in the centroid estimation can be seen as the number of neighbors increases. It happens after convergence is achieved and the more interconnected the lower the error is. To better appreciate this, is necessary to use logarithmic scale on the y axis (Figure 5.17).

Unexpectedly, the convergence speed only changes when moving from 1-distance neighbors (those adjacent in the graph) to 2-distance neighbors (adjacent and those adjacent to them). After that increases the intercommunication seems not to have an effect in the convergence speed like Figure 5.16b shows.

5.4.4 Frequencies

We analyze in this subsection changing the frequencies.

Similar to the amplitudes case, the frequencies will be multiplied by a factor of 2,3 and randomly set from a uniform distribution of expected value 1. The baseline case is also included to facilitate comparison

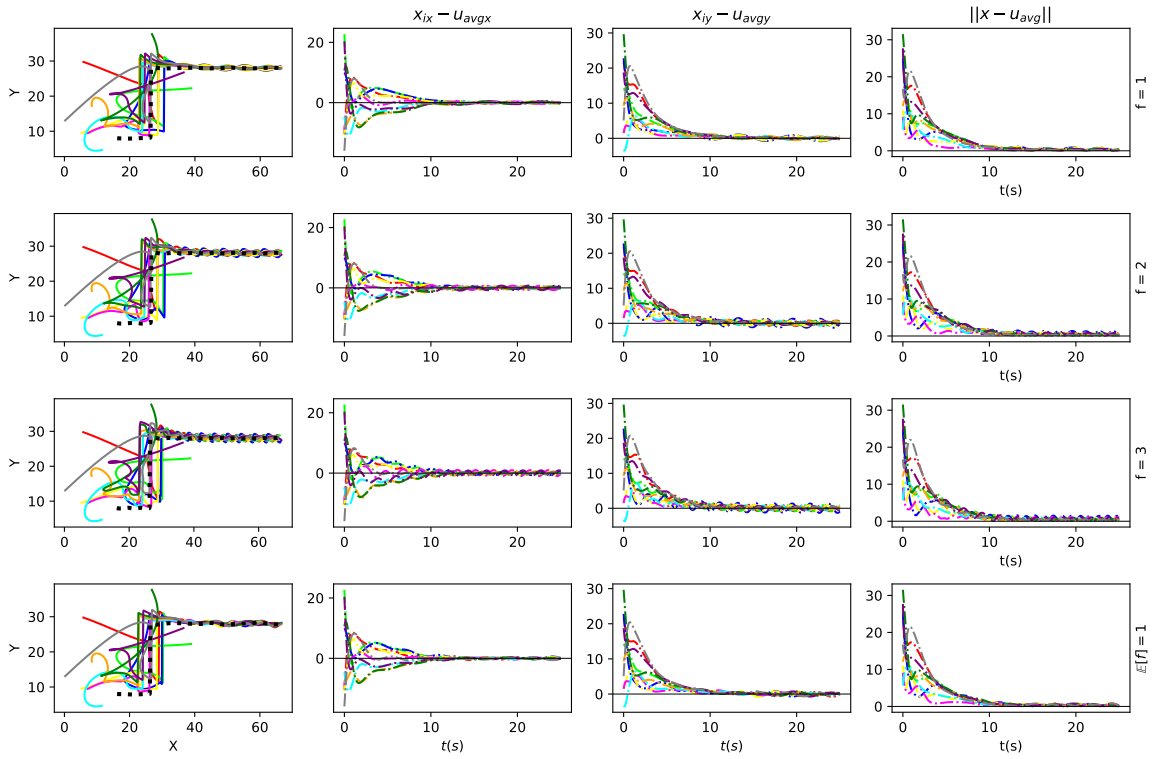
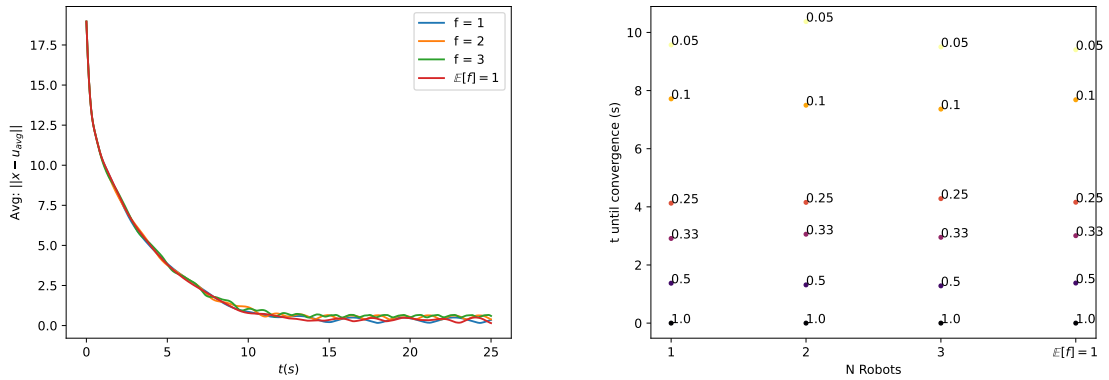


Figure 5.18: Each row represents an experiment with a certain frequency scale. No particularly significant effects can be seen.

Like in the amplitude case, the frequency parameter does not play an important role in the convergence rate (Figure 5.19b). Unlike the amplitude, there is no influence either on the magnitude of the disagreement between estimated and true centroid (Figure 5.19a). Frequencies only affect the speed of the oscillations in the error, but no jumps in quality happen.



(a) Average disagreement for different deformation frequencies. The frequency is not a source of error by itself.

(b) The convergence time is not affected by the frequency at which the beacons oscillate. Regardless of magnitude or randomness.

Figure 5.19: Studied influence for different frequencies.

5.4.5 Initial positions

Now we study the initial placement of the robots. They will still be randomly initialized, but their position will be scaled by a factor of 1, 2, 5 and 10.

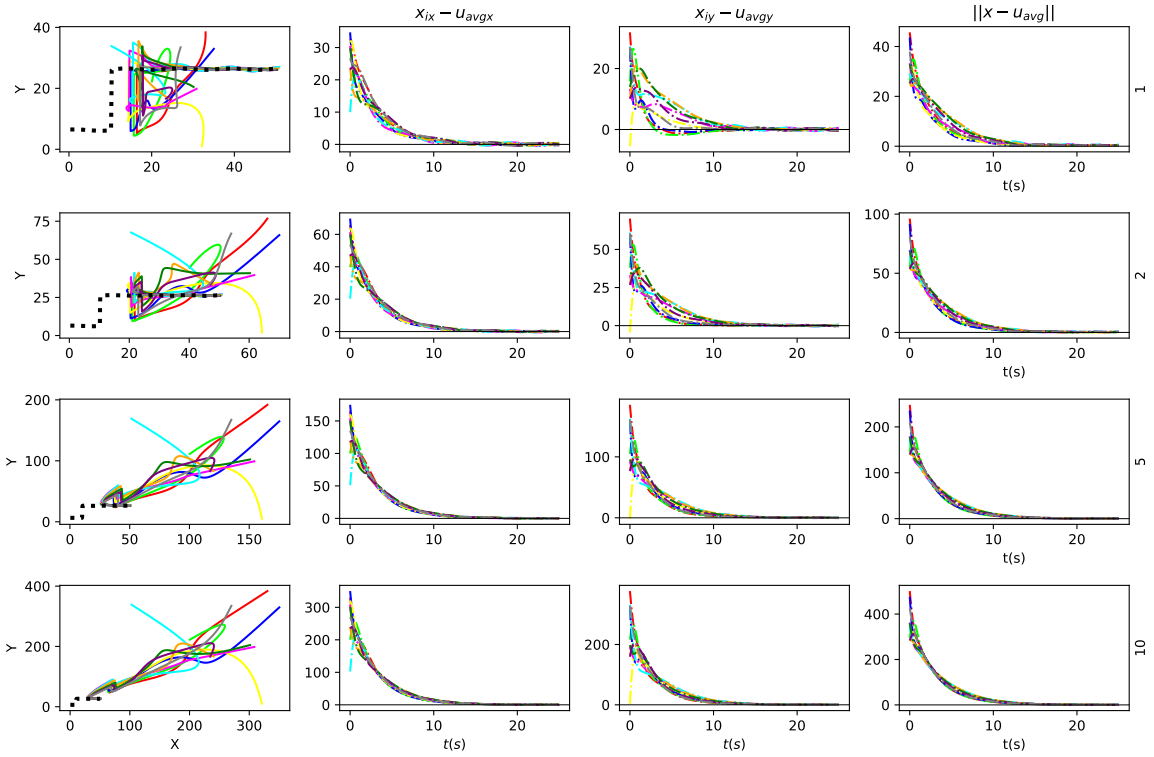
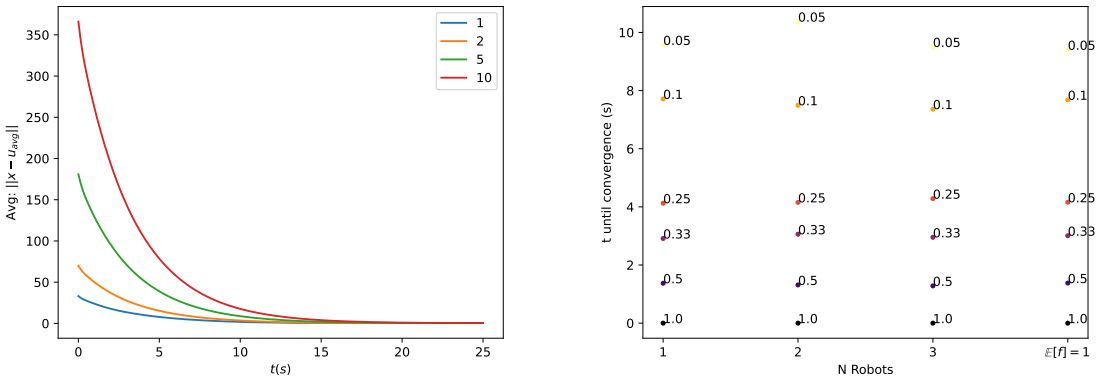


Figure 5.20: The initial position of the robots does have great influence on how the disagreement evolves. The further away the robots start the longer it takes them to reach the same disagreement as robots that started closer together.

We would expect that the closer together the agents start to begin with, the faster

they will converge. The convergence towards the centroid remains to be seen. Let us see the centroid estimation and its convergence rates in Figure 5.21.



(a) Disagreement from the initial position. The initial bigger errors need more time to reach the same disagreement as close robots.

(b) The evolution of the CR is not influenced by the initial position of the robots. Convergence is achieved in every case.

Figure 5.21: Studied influence for different initial positions.

Unsurprisingly, the farther away the robots begin from each other and the beacons, the longer it takes them to converge towards a zero value disagreement. However, it is remarkable, that since we have studied the convergence rate relative to its initial error, it does not change regardless of the initial error. It takes the same time to go from $CR_{0.5} = E_0 * 0.5$ to $CR_{0.05} = E_0 * 0.05$, regardless of the value of the initial error E_0. This does not hold for absolute disagreements too. It takes longer for the robots that started further away (scale = 10) to reach a disagreement of 1 that it costs the robots that started closer.

Relative convergence is constant regardless of the scale of the initial position of the robots, the absolute convergence, however is slower the further away the robots start.

5.4.6 Trajectory

Finally, we study if the trajectory. So far in every experiment the deformable body has followed the same trajectory: it moved right and at some point in time a jump happened in the y-coordinate.

In this section we study what happens when the trajectory is different. We will study a circular trajectory, a spiral trajectory, an exponential one and finally a periodical jump trajectory. In this last trajectory, the centroid jumps to a random $y \in [-1, 1]$ coordinate after every second. It moves linearly in the horizontal direction.

As can be seen convergence is reached for all studied trajectories. With no significant differences.

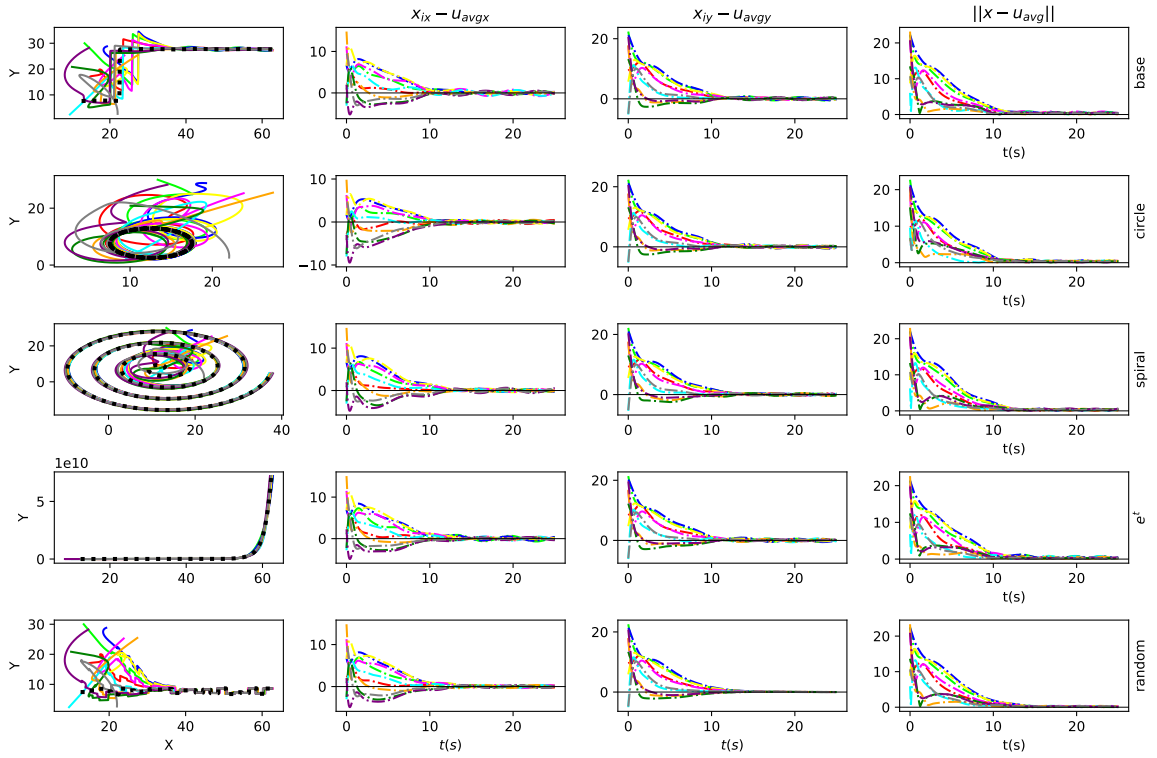
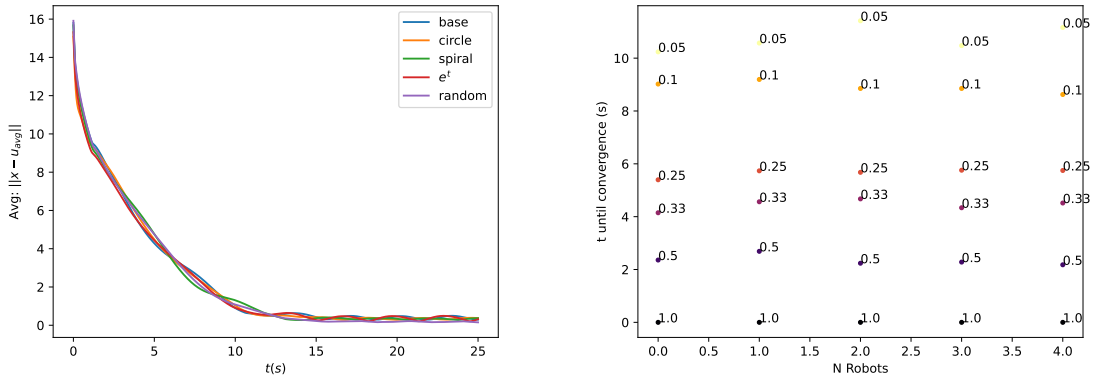


Figure 5.22: Spatial evolution, disagreement in coordinates x and y and total disagreement for each of the trajectories studied. The baseline one, a circular trajectory, a spiral trajectory, an exponential trajectory and a periodical jump trajectory.



(a) Disagreement for each of trajectories.

(b) CR for each of the trajectories

Figure 5.23: Studied influence for different initial positions.

Figure 5.23 shows that the disagreement between the estimated centroid and the true one fall similarly for all cases, with the only difference being the oscillations of the deformed beacons. It appears then, that the trajectory followed by the centroid does not play a crucial role on the convergence and tracking of said centroid.

The convergence rate remains constant for the different trajectories. This supports the conclusion.

However, this does not paint the whole picture. In some cases, where the speed of the centroid increases rapidly enough, the robots lose the capacity to track it. This can be seen after extending the simulation time of the exponential trajectory and plotting the disagreement between the estimated centroid and the true centroid.

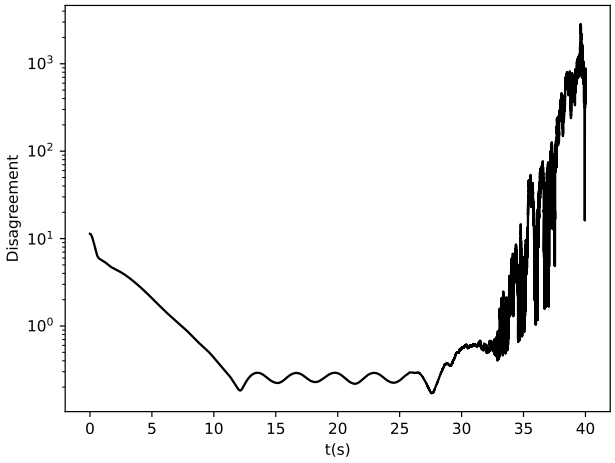


Figure 5.24: Disagreement between the estimated and true centroid for an exponential trajectory for a longer period of time.

As shown in Figure 5.24 the centroid is tracked successfully until $t \approx 27$ seconds, after that, it moves too fast for the robots to keep up and the error grows.

In this particular scenario of exponential tracking, the speed of the centroid is $\frac{de^t}{dt} = e^t$ and the tracking loss happens after $t = 27$s. The speed at which the centroid should move is $v = e^{27} \approx 5 * 10^{11}$. That is physically impossible since its several orders of magnitude over the speed of light. However that does not make the result any less significant for real life applications. In real-world implementations the robots will not operate instantaneously like in our experiments and are constrained by physical limitations, such as their maximum movement speed or their communication speed. So it is expected that this phenomenon will appear even in less extreme trajectories and should be taken into account.

5.5 Experimental conclusions

In the second half of this chapter, we present simulations of the posed problem.

Firstly, we define the baseline parameters of the simulation and investigate how varying certain parameters affects the behavior of the system.

Changing the type of oscillations that define the deformation has little effect, except

in the most extreme cases where the deformation becomes unrealistic. Otherwise, switching between different oscillation types (triangular, sawtooth, or square waves) results in minimal changes, with convergence and tracking still occurring as expected.

Other parameters, however, had a more significant impact on the system's behavior. Increasing the number of robots and beacons slowed down the convergence, while changing the frequency of the oscillations had little effect on the outcome. On the other hand, increasing the amplitude of the deformation noticeably impacted the final error between the estimated centroid and the true centroid. Similarly, better connectivity and data sharing among the robots reduced the final error, though increased connectivity led to diminishing returns on convergence speed.

As expected, starting with robots placed further from each other and the true centroid resulted in a longer time until a certain level of disagreement was reached. Surprisingly, however, the rate of convergence remained consistent relative to their initial error.

Finally, the trajectory exhibited an interesting behavior. For rapidly accelerating centroids, there were cases where the robots were not fast enough to track the centroid, causing the error to increase after initially reaching convergence.

Chapter 6

Conclusions and future work

6.1 Conclusions

In this work, we studied and compared different algorithms for estimating the centroid of a deformable object. Through careful analysis, we identified the most robust consensus algorithm as the best choice. Although it is more complex, this algorithm proved to be the most reliable for achieving consensus among robots while maintaining performance under dynamic conditions. As detailed in section 3.5, this algorithm was selected as the default for all subsequent simulations in this work.

We also compared the time cost of implementing the consensus algorithm in two approaches: a distributed method and a vector-based approach. Each method has its strengths and weaknesses. The distributed approach is more realistic and closer to how the algorithm would be implemented in a real-world application, while the vector-based approach is more abstract but simpler to implement for research purposes. The vector-based approach proved to be more effective, offering faster runtime, easier design, and simpler implementation for experiments.

We chose $\tau = 0.0001$ although larger values would have worked to (for more information consult Appendix B). While larger time steps would still maintain accuracy within an acceptable threshold, we opted for the smaller τ to minimize risk. Importantly, despite the small τ, the computation time was not excessively long. For more details, the full analysis can be found in the appendix.

Finally, we studied a deformable object with the goal of using consensus to estimate its centroid. We considered the simplest scenario of a deformable object with beacons delimiting its shape, where the deformation consists of oscillations in a random direction.

Building on the error estimation for the undeformed problem presented in Equation 5.1, we derived another mathematical expression to constrain the maximum error of the centroid estimation for a deformable body whose deformation consist in

oscillations. Two different methods were proposed for computing this estimation, each providing more precise error boundaries depending on the number of robots involved. Although both methods offered conservative error limits, they successfully established useful constraints.

We also explored the influence of various system variables on the convergence and final accuracy of the centroid estimation. These variables included the number of robot-beacon pairs, the amplitude of the oscillating deformation, and the trajectory followed by the deformable body. Some parameters had a significant impact on either the estimation error or convergence time, while others had minimal influence.

6.2 Future work

One potential future work direction involves modeling the deformable object as a set of interconnected bodies, represented by springs. This approach could have practical applications in fields such as biological physics, where proteins are often modeled in a similar way. By adopting this modeling technique, we could capture more complex time evolution in the system. Specifically, since the position of the centroid might be difficult-or even impossible-to determine at all times, the focus would shift to estimating the likely shape of the deformable body. This would involve reaching a consensus between two key variables: the position of the centroid and a tolerance radius. The tolerance radius would ideally be the minimum radius whose circumscribed circumference fully encloses the deformable body.

Another promising avenue of research could involve a deeper exploration of the perception matrix defined in this thesis. Investigating how changes in the perception matrix could affect dynamic consensus would be an interesting line of study.

Additionally, it may be worthwhile to consider scenarios where robots lose track of beacons. In such cases, the perception matrix would change dynamically as the system evolves over time, which would introduce new challenges and potential insights for the model.

Appendix A

Algorithms alternative formulation: z

In this Appendix we write the algorithms from chapter 3 in the alternative formulation initially proposed by Kia et al.[9] that does not require knowing the future signal $\mathbf{u}_i$ to update the state of the robots. We will apply it to all 4 algorithms studied.

The essential idea behind this proposal is the definition of an auxiliary variable:

$$\mathbf{z}_i(k) = \mathbf{x}_i(k) - \mathbf{u}_i(k). \tag{A.1}$$

With this change of variable the algorithms may be rewritten so that its application does not rely on the signal $\mathbf{u}_i(k)$ viewed by each robot $\mathbf{x}_i(k)$. We just write the discrete form of the algorithms since as we will explain later, this notation is not adequate for our case.

Basic Dynamic Consensus Applying the change of variable to the system defined in Equation 3.2 we obtain:

$$\mathbf{z}_i(k) = \mathbf{z}_i(k-1) + \mathbf{u}_i(k-1) + \dot{\mathbf{z}}_i(k) + \dot{\mathbf{u}}_i(k) - \mathbf{u}_i(k),$$
$$\dot{\mathbf{z}}_i(k) + \dot{\mathbf{u}}_i(k) = \mathbf{u}_i(k) - \mathbf{u}_i(k-1) - \tau \sum_{j=1}^{N} a_{ij} \left(\mathbf{z}_i(k-1) + \mathbf{u}_i(k-1) - \mathbf{z}_j(k-1) - \mathbf{u}_j(k-1)\right),$$
$$\mathbf{z}_i(k) = \mathbf{z}_i(k-1) - \tau \sum_{j=1}^{N} \left(\mathbf{z}_i(k-1) + \mathbf{u}_i(k-1) - \mathbf{z}_j(k-1) - \mathbf{u}_j(k-1)\right),$$
$$\mathbf{z}_i(t=0) = \mathbf{0}.$$

Dynamic Consensus with drift towards signals Again, we use the auxiliary variable in order to get:

$$
\begin{aligned}
&\mathbf{z}_i(k) = \mathbf{z}_i(k-1) + \mathbf{u}_i(k-1) + \dot{\mathbf{z}}_i(k) + \dot{\mathbf{u}}_i(k) - \mathbf{u}_i(k),\\
&\dot{\mathbf{z}}_i(k) + \dot{\mathbf{u}}_i(k) = \mathbf{u}_i(k) - \mathbf{u}_i(k-1) -\\
&\qquad - \tau \sum_{j=1}^{N} a_{ij} \left(\mathbf{z}_i(k-1) + \mathbf{u}_i(k-1) - \mathbf{z}_j(k-1) - \mathbf{u}_j(k-1)\right) + \tau\alpha \mathbf{z}_i(k-1),\\
&\mathbf{z}_i(k) = \mathbf{z}_i(k-1) - \tau \sum_{j=1}^{N} \left(\mathbf{z}_i(k-1) + \mathbf{u}_i(k-1) - \mathbf{z}_j(k-1) - \mathbf{u}_j(k-1)\right) + \tau\alpha \mathbf{z}_i(k-1),\\
&\mathbf{z}_i(t=0) = \mathbf{0}.
\end{aligned}
$$

Dynamic consensus for directed graphs By virtue of the auxiliary variable we may rewrite the original expressions into:

$$
\begin{aligned}
\dot{\mathbf{q}}_i(k) &= \beta\alpha \sum_{j=0}^{N} a_{ij} \left(\mathbf{z}_i(k) + \mathbf{u}_i(k) - \mathbf{z}_j(k) - \mathbf{u}_j(k)\right),\\
\mathbf{q}_i(k) &= \mathbf{q}_i(k-1) + \tau \left(\beta\alpha \sum_{j=0}^{N} a_{ij} \left(\mathbf{z}_i(k) + \mathbf{u}_i(k) - \mathbf{z}_j(k) - \mathbf{u}_j(k)\right)\right),\\
\dot{\mathbf{z}}_i(k) + \dot{\mathbf{u}}_i(k) &= -\alpha\left(\mathbf{z}_i(k-1)\right) -\\
&\quad - \beta \sum_{j=0}^{N} a_{ij} \left(\mathbf{z}_i(k-1) + \mathbf{u}_i(k-1) - \mathbf{z}_j(k-1) - \mathbf{u}_j(k-1)\right) - \mathbf{q}_i(k) +\\
&\quad + \frac{\mathbf{u}_i(k) - \mathbf{u}_i(k-1)}{\tau},\\
\mathbf{z}_i(k) + \mathbf{u}_i(k) &= \mathbf{z}_i(k-1) + \mathbf{u}_i(k-1) +\\
&\quad + \tau \left(-\alpha\left(\mathbf{z}_i(k-1)\right) - \beta \sum_{j=0}^{N} a_{ij} \left(\mathbf{z}_i(k-1) + \mathbf{u}_i(k-1) - \mathbf{z}_j(k-1) - \mathbf{u}_j(k-1)\right)\right) -\\
&\quad - \tau \left(\mathbf{q}_i(k) + \frac{\mathbf{u}_i(k) - \mathbf{u}_i(k-1)}{\tau}\right).
\end{aligned}
$$

Dynamic Consensus independent of initial conditions And finally for the robust algorithm:

$$
\begin{aligned}
\mathbf{z}_i(k) + \mathbf{u}_i(k) &= \mathbf{z}_i(k-1) + \mathbf{u}_i(k-1) + \tau\left(\dot{\mathbf{z}}_i(k) + \dot{\mathbf{u}}_i(k)\right), \\
\dot{\mathbf{z}}_i(k) + \dot{\mathbf{u}}_i(k) &= \frac{\Delta\mathbf{u}(k)}{\tau} + \alpha\mathbf{z}_i(k) + \left(\sum_{j=1}^{N} a_{ij}\left(\mathbf{z}_i(k) + \mathbf{u}_i(k) - \mathbf{z}_i(k) - \mathbf{u}_i(k)\right)\right) + \\
&\quad + \left(\sum_{j=0}^{N} b_{ij}\left(\mathbf{q}_i(k) - \mathbf{q}_j(k)\right)\right), \\
\mathbf{q}_i(k) &= \mathbf{q}_i(k-1) + \alpha\tau\sum_{j=1}^{N} b_{ij}\left(\mathbf{z}_i(k) + \mathbf{u}_i(k) - \mathbf{z}_j(k) - \mathbf{u}_j(k)\right) \\
\mathbf{z}_i(k) &= -\mathbf{u}_i(k) + \mathbf{z}_i(k-1) + \mathbf{u}_i(k-1) + \Delta\mathbf{u}(k) + \tau\alpha\mathbf{z}_i(k) + \\
&\quad + \tau\left(\sum_{j=1}^{N} a_{ij}\left(\mathbf{z}_i(k) + \mathbf{u}_i(k) - \mathbf{z}_i(k) - \mathbf{u}_i(k)\right)\right) + \tau\left(\sum_{j=0}^{N} b_{ij}\left(\mathbf{q}_i(k) - \mathbf{q}_j(k)\right)\right).
\end{aligned}
$$

Appendix B

Criteria for the selection of an adequate τ

In this appendix we study which size of the τ step would be small enough to guarantee a certain level of accuracy. We will consider then that accuracy is reached with the following criterion: if the result from the discrete simulation after a certain t_n is inside the interval $y_k \in (y_t - \varepsilon, y_t + \varepsilon)$, we consider the result and τ to be adequate. The y_k is the value of a discrete computation of the function in step k and y_t is the value of the analytical form of the function in continuous domain.

We therefore need to estimate a correct amount of steps to take during the simulation (and therefore indirectly define the step size) and check whether or not its *GAE* allows it to fulfill the imposed condition.

For the sake of length and ease of understanding, we will work with a toy problem of 3 1-dimensional robots fully interconnected and the algorithm of basic dynamic consensus. For the first part we will use the algorithm shown by Douglas Wilhem [14] so as to appraise the minimum required number of steps to do the simulation for different time horizons. Its main benefit is that of not requiring to provide an estimate of the error given a certain admissible uncertainty threshold without the need for the true value of the solution.

After that we will compare it the performance of different τ values with the final error with the analytical solution of the equivalent system of ordinary differential equations, to check that the results given by the initial algorithm are valid. This will allow us to claim with a certain degree of certainty that the chosen τ is good enough for the rest of simulations on this work.

B.1 Estimating the error without true reference

We will proceed with the algorithm described in the *Numerical Methods* course taught by Douglas Wilhelm [14]. The algorithm is shown provided in Algorithm 1

Algorithm 1 Estimate valid τ of Euler discretization

function SOLVE(n)
  $\tau \leftarrow \frac{(t-t_0)}{n}$
  $\tau_2 \leftarrow \tau/2$
  Integrate y for τ, y_n
  Integrate y for τ_2, y_{2n}
  $e \leftarrow 2 * |y_n - y_{2n}|$
  Return e
end function
Require: $0 < n \in \mathbb{N}$
  ε
Ensure: $e(t - t_0) \leq \varepsilon * (t - t_0)$
  n
  $e \leftarrow$ SOLVE(n)
  while $e > \varepsilon\,(t - t_0)$ **do**
    $n \leftarrow 2 * n$
    $e \leftarrow$ SOLVE(n)
  end while
  $\tau_f \leftarrow \frac{(t-t_0)}{2n}$

After implementing such algorithm we checked that some values for both the time horizon of the simulation $t - t_0 = 1s$ and $n = 500$ to check if its corresponding $\tau = \frac{t-t_0}{2n}$ would provide results inside the tolerance threshold. After doing the simulations we reached that for as long as the expected tolerance $\varepsilon = 0.01$, taking $n = 500$ is enough to fulfill the algorithm condition and assume that the τ is good enough.

Although we believe that this tolerance would be functional and work well for our problem, we elected to instead use $\epsilon = 0.001$. For which the sweet spot is around $n = 1500$ for time horizons of up to $20s$. As long as computation does not prove to be at odds with accuracy we elected to chose as our default $\tau = 0.0001$. This would be equivalent to having $n = 10.000$ for the shortest time horizon of $t = 1$, and even more steps as the time horizon increases.

This method is an estimation made without really being able to compare the final result to the true value. We will proceed in the next section to use the τ selected with this criteria and compare the simulation results with those of a toy problem that may be solved analytically.

If the chosen τ from the test of 1 provided good results (within a tolerance deemed sufficient) in the analytical test too we could make the case that this algorithm and its resulting τ may be used for the rest of simulations where we do not have an analytical solution to compare to.

B.2 Comparing the analytical Solution for our problem with the chosen τ

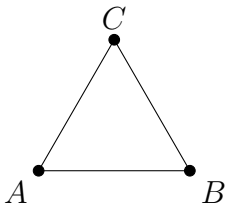
From section 3.1 it can be easily seen that the basic dynamic consensus is a system of differential ordinary equations in the form of:

$$\mathbf{x}' = \mathbf{A}\mathbf{x} + \mathbf{f}(t). \tag{B.1}$$

Where $\mathbf{A} = -\mathbf{L}$ and $\mathbf{f}(t)$ is the vector of signals that each robot follows as time evolves.

Let us consider the specific case of 3 agents all connected among each other and whose reference signals are sine waves offseted by a certain amount. We choose this particular case of the problem due to its simplicity, but it may be generalized to more complex systems with a greater amount of robots who follow different signals each.

Let then this particular graph and its corresponding Laplacian matrix $\mathbf{L}$ be:



Laplacian Matrix: $\begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}$

We get the following system of ODEs:

$$\begin{pmatrix} \mathbf{x}'_\mathbf{A} \\ \mathbf{x}'_\mathbf{B} \\ \mathbf{x}'_\mathbf{C} \end{pmatrix} = \begin{pmatrix} -2 & 1 & 1 \\ 1 & -2 & 1 \\ 1 & 1 & -2 \end{pmatrix} \begin{pmatrix} \mathbf{x}_\mathbf{A} \\ \mathbf{x}_\mathbf{B} \\ \mathbf{x}_\mathbf{C} \end{pmatrix} + \begin{pmatrix} \cos t \\ \cos t \\ \cos t \end{pmatrix}. \tag{B.2}$$

Where the vector of cosines comes from the fact that $\mathbf{f}(t) = \dot{\mathbf{u}}$ and we have defined the vector of signals the robots follow as:

$$\mathbf{u} = \begin{pmatrix} O_0 + \sin t \\ O_1 + \sin t \\ O_2 + \sin t \end{pmatrix}. \tag{B.3}$$

In order to solve the homogeneous system we compute the eigenvectors of the matrix and its corresponding eigenvalues.

$$\mathbf{v_1} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \lambda_1 = 0, \qquad \mathbf{v_2} = \begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix}, \lambda_2 = -3, \qquad \mathbf{v_3} = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}, \lambda_3 = -3. \tag{B.4}$$

Which leaves us with the homogeneous system's fundamental matrix:

$$\phi = \begin{pmatrix} 1 & -e^{-3t} & -e^{-3t} \\ 1 & e^{-3t} & 0 \\ 1 & 0 & e^{-3t} \end{pmatrix} \begin{pmatrix} \cos t \\ \cos t \\ \cos t \end{pmatrix} + \mathbf{C_o}. \tag{B.5}$$

Therefore the general family of solutions to the homogeneous system of equations is $\mathbf{x}_{sol} = \phi\mathbf{C}$, whose terms are

$$x_1 = c_1 - c_2e^{-3t} - c_3e^{-3t}, \tag{B.6}$$

$$x_2 = c_1 + c_2e^{-3t}, \tag{B.7}$$

$$x_3 = c_1 + c_3e^{-3t}. \tag{B.8}$$

We want to solve the nonhomogeneous system to take into account the signals each robot follows. This in turn means that each c_i is not a constant but a function of t.

We may compute the solution by simply solving the following expression:

$$\phi\frac{d\mathbf{c}}{dt} = \mathbf{f}(t) \implies \frac{d\mathbf{c}}{dt} = \phi^{-1}\mathbf{f}(t) \implies \mathbf{c} = \int \phi^{-1}\mathbf{f}(t)dt + Co. \tag{B.9}$$

After computing the inverse matrix of ϕ we obtain the integral

$$\mathbf{c} = \int \frac{1}{3}\begin{pmatrix} 1 & 1 & 1 \\ -e^{3t} & 2e^{3t} & -e^{3t} \\ -e^{3t} & -e^{3t} & 2e^{3t} \end{pmatrix}\begin{pmatrix} \cos t \\ \cos t \\ \cos t \end{pmatrix} dt + \begin{pmatrix} c_o \\ c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} \sin t + c_o \\ c_1 \\ c_2 \end{pmatrix} \tag{B.10}$$

Therefore, once $\mathbf{c}$ is known, the solution of the system may be computed by multiplying $\mathbf{x} = \phi\mathbf{c}$. Which leaves us with the solution:

$$\mathbf{x} = \begin{pmatrix} c_o + \sin t - c_1e^{-3t} - c_2e^{-3t} \\ c_o + \sin t + c_1e^{-3t} \\ c_o + \sin t + c_2e^{-3t} \end{pmatrix}. \tag{B.11}$$

Since we are working with the basic dynamic consensus, it is required for the initial states of the robots to match the initial value of their respective signals.

We get

$$\mathbf{u}(t=0) = \begin{pmatrix} \sin 0 + O_0 \\ \sin 0 + O_1 \\ \sin 0 + O_2 \end{pmatrix} = \begin{pmatrix} O_0 \\ O_1 \\ O_2 \end{pmatrix} = \mathbf{x}(t=0). \tag{B.12}$$

By imposing the initial condition $\mathbf{x}(t = 0)$ we may compute the values of the c_i constants. Result from which we finally obtain the particular solution of the problem for our specific scenario.

$$\begin{aligned} x_1 =& \frac{O_0 + O_1 + O_2}{3} + \sin t - \frac{2O_1 - O_0 - O_2}{3}e^{-3t} - \frac{2O_2 - O_0 - O_1}{3}e^{-3t}, \\ x_2 =& \frac{O_0 + O_1 + O_2}{3} + \sin t + \frac{2O_1 - O_0 - O_2}{3}e^{-3t}, \\ x_3 =& \frac{O_0 + O_1 + O_2}{3} + \sin t + \frac{2O_2 - O_0 - O_1}{3}e^{-3t}. \end{aligned} \tag{B.13}$$

We have therefore computed the analytical solution for this specific instance of the problem of dynamic consensus in its analytical form. The behavior of said solution as a function of time is provided in Figure B.1

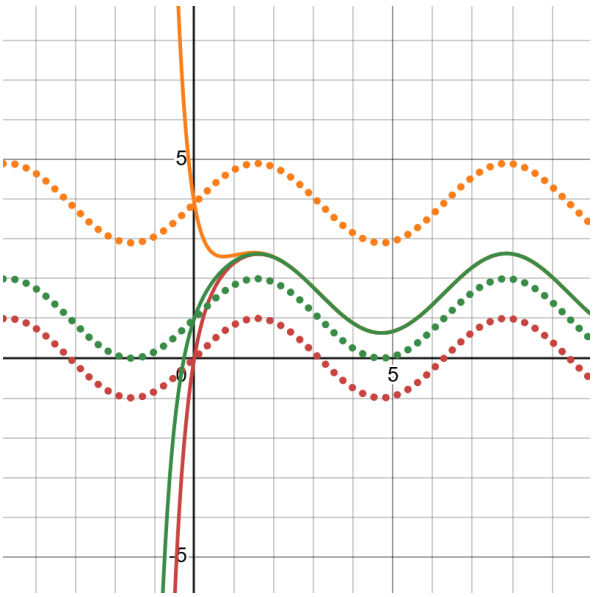


Figure B.1: Solution of the solved problem, the robots (solid) quickly converge together to the average signal despite each following its own signal (dotted).

Given that we have computed the analytical solution of the toy problem, we may use its values as a benchmark for evaluating whether a τ is small enough to guarantee us that the results after simulating will not be degraded by significant error introduced while using the Euler discretization.

This test is quite simple and will be done twofold, we will solve this same dynamic consensus problem computationally with different values for τ and t. Firstly we will do the relaxed test, where we will relax the permitted error by taking into account the time horizon. We define the admissible error $|\mathbf{x}^i(t)_{anal} - \mathbf{x}^i(t)_{comp}| < \varepsilon * t$. Where $\epsilon = 0.01$

Then we will strictly compare if the discrepancy between the analytical solution and the computational solution is within the tolerance threshold such that $|\mathbf{x}^i(t)_{anal} - \mathbf{x}^i(t)_{comp}| < \varepsilon$.

If the first condition were to be fulfilled it would be safe to assume that the chosen τ works well for that time horizon t. We will however focus on the second, more strict condition in order to check whether a τ is sufficiently safe for our task. We are therefore leveraging more accuracy in exchange for slower computation times.

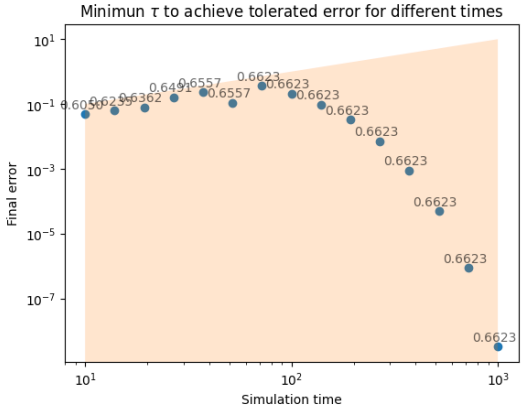


Figure B.2: Minimum τ obtained for the first criterion, where the accuracy is relative to the simulation time. Any smaller τ values will yield equally or more accurate simulations

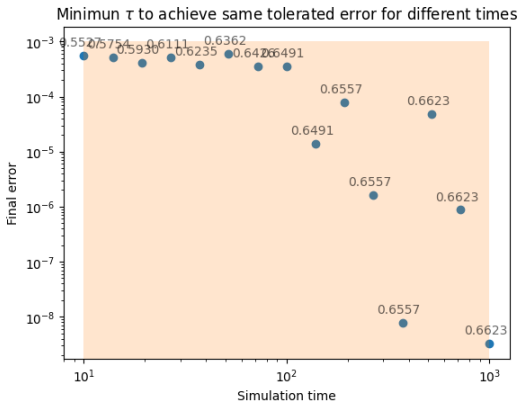


Figure B.3: Minimum τ obtained for the second, stricter criterion, where the goal accuracy is absolute. Any smaller τ values will yield equally or more accurate simulations

Surprisingly, for similar values of τ the error (defined as the norm of difference between the position of the centroid computationally estimated and the position of the analytically computed centroid) falls as the simulation time increases. As shown in Equation 4.15 we would expect it to grow linearly as the simulation time increases, which does not happen.

There is nonetheless an explanation for this behavior. As claimed in *Propagation of Errors in Euler Methods* by Akanbi M.A [2]. When an initial value problem is inherently stable the local errors may damp out as the function variable (in our case t) increases. For studying the inherent stability of the problem we may use the same method explained in the *Math 563 Lecture Notes of Duke University*[15].

By considering our problem as a base problem and a perturbation $\mathbf{v} = \mathbf{y}(t) + \varepsilon\mathbf{d}$, where in our case $\mathbf{v} = \dot{\mathbf{x}}$, $\mathbf{y}(t) = \mathbf{A}\mathbf{x}$ and $\varepsilon\mathbf{d} = \mathbf{f}(t)$ we may study the stability through the Jacobian matrix $\mathbf{J}$ of the base system. If none of the eigenvalues have a positive real part, the problem is stable.

Since $\mathbf{A} = -\mathbf{L}$, all its eigenvalues are $\lambda_i \leq 0$. So the condition is fulfilled and the problem is stable. Because the stability condition is fulfilled, the claim about the local errors may apply, providing us an explanation why the error unexpectedly damped rather than grow.

In both cases we fulfill our requirements with τ values around $\tau \approx 0.5$. Erring on the side of caution and because its corresponding simulation times are not particularly time-consuming, we will then do most simulations with $\tau = 0.0001$ unless otherwise specified.

Bibliography

[1] J. Abouaf. "Trial by fire: teleoperated robot targets Chernobyl". In: *IEEE Computer Graphics and Applications* 18.4 (1998), pp. 10–14. DOI: 10.1109/38.689654.

[2] M.A Akanbi. "Propagation of Errors in Euler Methods". English. In: *Scholars Research Library* 2.2 (2010), pp. 457–469. ISSN: 0975-508X. URL: https://www.scholarsresearchlibrary.com/articles/propagation-of-errors-in-euler-methods.pdf (visited on 11/12/2024).

[3] Rosario Aragues et al. "Convergence speed of dynamic consensus with delay compensation". en. In: *Neurocomputing* 570 (Feb. 2024), p. 127130. ISSN: 09252312. DOI: 10.1016/j.neucom.2023.127130. URL: https://linkinghub.elsevier.com/retrieve/pii/S0925231223012535 (visited on 10/02/2024).

[4] Li Chi-Kwong. "Norms and Norm inequalities". English. College of William and Mary, Department of Mathematics, Aug. 2008. URL: https://www.math.wm.edu/~ckli/ima/note-3.pdf (visited on 10/18/2024).

[5] Paul Dawkins. *Differential Equations - Repeated Eigenvalues*. Nov. 2022. URL: https://tutorial.math.lamar.edu/Classes/DE/RepeatedEigenvalues.aspx (visited on 10/07/2024).

[6] Randy A. Freeman, Peng Yang, and Kevin M. Lynch. "Stability and Convergence Properties of Dynamic Average Consensus Estimators". In: *Proceedings of the 45th IEEE Conference on Decision and Control*. ISSN: 0191-2216. Dec. 2006, pp. 338–343. DOI: 10.1109/CDC.2006.377078. URL: https://ieeexplore.ieee.org/document/4177692/?arnumber=4177692&tag=1 (visited on 11/21/2024).

[7] Grant B. Gustafson. *Differential Equations and Linear Algebra*. English. Vol. I and II. A Course for Science and Engineering. Salt Lake City, Utah: University of Utah, Department of Mathematics, 1999. ISBN: 979870549112, 9798711123651. URL: https://www.math.utah.edu/~gustafso/debook/deBookGG.pdf (visited on 10/10/2024).

[8] Robert Israel. "Error Analysis of the Euler Method". English. Website. University of British Columbia, Jan. 2002. URL: https://personal.math.ubc.ca/~israel/m215/euler2/euler2.html (visited on 10/02/2024).

[9] Solmaz S. Kia et al. "Tutorial on Dynamic Average Consensus: The Problem, Its Applications, and the Algorithms". In: *IEEE Control Systems Magazine* 39.3 (June 2019), pp. 40–72. ISSN: 1941-000X. DOI: 10.1109/MCS.2019.2900783. URL: https://ieeexplore.ieee.org/document/8716798/?arnumber=8716798 (visited on 10/02/2024).

[10] Jiri Lebl. "Notes on Diffy Qs: Differential Equations for Engineers". en. In: *2024-05-15* 6.7 (May 2024), p. 466. ISSN: 978-1706230236. URL: https://www.jirka.org/diffyqs/diffyqs.pdf.

[11] Michael E. Moran. "The da Vinci Robot". In: *Journal of Endourology* 20.12 (2006). PMID: 17206888, pp. 986–990. DOI: 10.1089/end.2006.20.986. eprint: https://doi.org/10.1089/end.2006.20.986. URL: https://doi.org/10.1089/end.2006.20.986.

[12] Emilio Sánchez Sánchez-Ortiga and Vicente Ferrando Martín. *Ecuaciones diferenciales en Física*. es. Universitat Politècnica de València: Escuela técnica superior de ingeniería del diseño, 2023. ISBN: 978-84-09-51855-5. URL: https://riunet.upv.es/bitstream/handle/10251/194853/EcuacionesDiferencialesenF%C3%ADsica.pdf?sequence=1.

[13] William F. Trench. *3.1: Euler's Method*. en. Jan. 2020. URL: https://math.libretexts.org/Courses/Monroe_Community_College/MTH_225_Differential_Equations/03%3A_Numerical_Methods/3.01%3A_Euler's_Method (visited on 10/30/2024).

[14] Douglas Wilhem. "Error analysis of Euler's, Heun's and the 4th-order Runge-Kutta method". English. Lecture Notes, ECE 204 Numerical Methods. University of Waterloo, Department of Electrical & Computer Engineering, Mar. 2021. URL: https://ece.uwaterloo.ca/~dwharder/nm/Lecture_materials/pdfs/7.1.1.1.4%20Error%20analysis%20of%20these%20methods.pdf (visited on 10/02/2024).

[15] Jeffrey Wong. "Math 563 Lecture Notes ODE Initial value problems (Part I)". English. Lecture Notes. Duke University, 2020. URL: https://services.math.duke.edu/~jtwong/math563-2020/lectures/Lec6-IVPs.pdf (visited on 11/21/2024).