Modeling and Control of Cyber-Physical Systems

Politecnico di Torino

Graziosi Luigi – *s331564*
Luppino Marco – *s333997*
Paglialunga Federico – *s328876*

Politecnico di Torino
A.A. 2023/2024

Summary

Project Part 2

Introduction

In our project, we focus on the **distributed control of a multi-agent magnetic levitation system**, a classic example of a cyber-physical system (CPS). The system consists of seven magnetic levitation units: one of these acts as the leader node $\boldsymbol{S_0}$, while the other six operate as follower nodes $\boldsymbol{S_i}$, $i = 1, 2, \dots, 6$. Each individual agent in the system, including both the leader and the followers, is described by identical state-space equations, obtained by linearizing the physical equations around a suitable equilibrium point. These equations are expressed as

$$\dot{x}_i = Ax_i + Bu_i \text{ and } y_i = Cx_i$$

where the matrices A, B, C are defined respectively as:

$$A = \begin{bmatrix} 0 & 1 \\ 880.87 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 \\ -9.9453 \end{bmatrix}, C = [708.27], D = [0]$$

Our task is to design a distributed control protocol that enables the multi-agent system to perform cooperative tracking tasks, asymptotically reducing the global disagreement error to zero. We use a communication network for the system described as a graph G augmented where the leader node is not considered. A way to represent graphs is through adjacency matrices, where each entry represents the weight of the edge (v_i, v_j), indicating that i receives information from j.

We designed two types of dynamic regulators since the state variables of the agents are not directly measurable due to $C \neq I$: one based on a **local observer** and the other on a **neighborhood observer**.

In the case of the **local observer**, each agent estimates the state using its local information and the output from other agents. This approach is computationally less complex but may be limited in terms of global performance, especially in the presence of noise or disturbances.

On the other hand, the **neighborhood observer** involves agents exchanging information with all their neighbors, creating a more densely connected network. This method can potentially enhance the system's overall performance but requires increased computational and communication complexity.

Regardless of the chosen approach, we need to conduct a numerical analysis to evaluate the effects of different leader node reference behaviors, such as constant signals, ramps, and sinusoidal signals, on the system. This involves modifying the dynamics of the leader by adjusting the eigenvalues of matrix A using the pole placement technique:

- For a **step** reference, we set one eigenvalue to zero and the remaining eigenvalues to have negative real parts.
- For a **ramp** reference, two coincident eigenvalues are needed with the remaining eigenvalues stable.
- For a **sinusoidal** reference, manipulation of the magnitude and phase of complex conjugate eigenvalues is necessary.

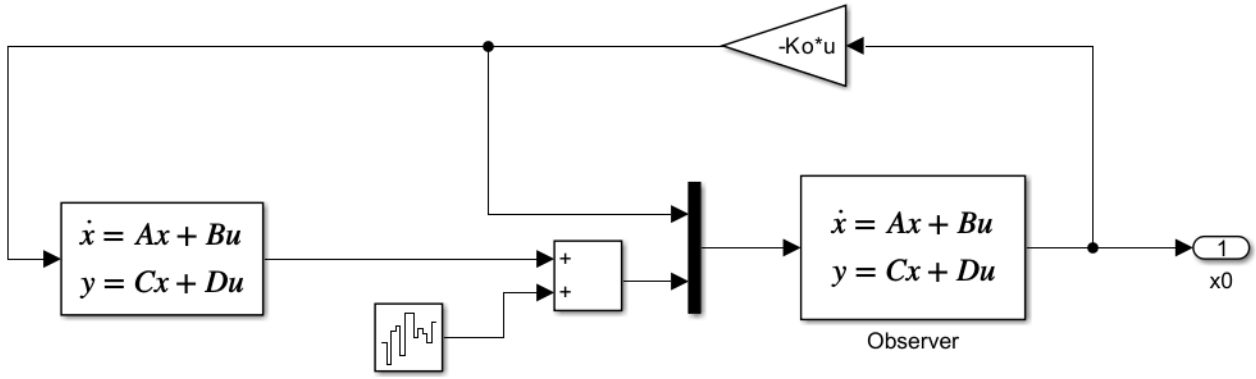
Case of Study I: Distributed Regulator based on Local Observers

In this case of study, we have 7 nodes, each implementing a local controller and a local observer. The follower nodes' structure is identical.

The local controller in the leader node S_0 is implemented by **feedback**, so that we have a State-Space system with given matrices A, B, C, D which gives the output. The output and the result of the feedback controller are given in input to the observer, which is implemented as a State-Space system which matrices are $A - L_1C$, $[B\ L_1]$, I_2, $\emptyset_{2,2}$, where L_1 is computed on `MATLAB` with the command `L1=place(A',C',eigsL1)'`, where the eigenvalues have $Re(\lambda_i) < 0, \forall i$.

The output of the observer is then multiplied by the matrix $-K_o$, which gives us the feedback input for our system and for the observer itself. The matrix K_0 is computed on MATLAB with the command `Ko=acker(A,B,eigs)`.
The choice of the eigenvalues we are going to assign is very important, as it defines the trajectory of the leader node and, consequently, of the follower nodes. In particular, we are going to analyse three kinds of reference trajectories:

- Step
- Ramp
- Sinusoidal



Pic. 1 – Leader system

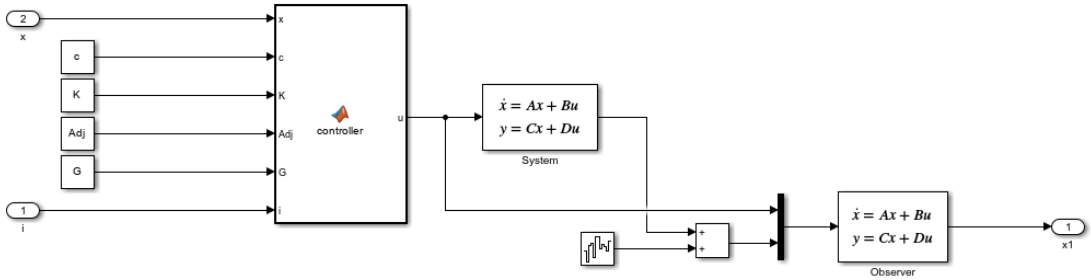
On the other hand, in addition to the controller given by K_0 (they have to be identical to the leader node otherwise they do not work), the follower nodes also implement a local controller given by the SVFB control protocol $u_i = cK\varepsilon_i$, where c is the coupling gain defined with respect to the constraint:

$$c > \frac{1}{2\min_{i \in \mathcal{N}} Re(\lambda_i)}$$

and $K = R^{-1}B^{\mathrm{T}}P$, according to the **Cooperative controller design Theorem**. The matrix P is the unique positive definite solution of the **algebraic Riccati equation** (**ARE**) computed on MATLAB by the command `P=are(A-B*K, B*R^-1*B', Q)`, where Q and R are diagonal positive definite matrices.
The controller block receives as inputs the number of the node i, the **adjacency matrix** Adj, the **coupling gain** c, the **pinning matrix** G and the **feedback state** x. It computes:

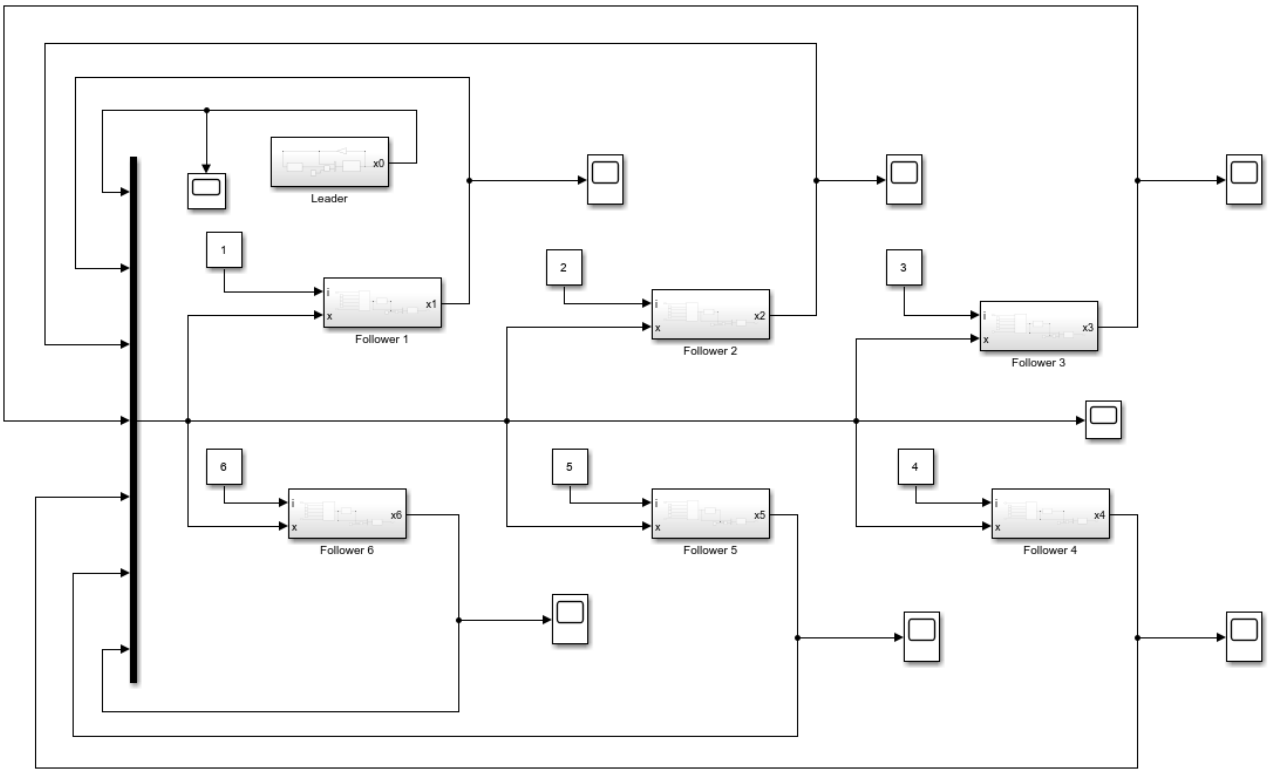
$$\varepsilon_i = \sum_{j=1}^{N} a_{ij}(x_j - x_i) + G_{ii}(x_0 - x_i)$$

where the first part of the equation is obtained on MATLAB by implementing a for loop on every node $j = 1, \dots, 6$. This controller is linked in input to the system controlled by the feedback $-K_0$.
The observer receives in input the output of the cooperative controller and output of the system. Note that, for graphical reasons, it has been preferred to implicitly implement feedback controller by inserting into the System State-Space block the matrices $A - BK_0$, B, C, D and into the Observer one the matrices $A - BK_0 - cFC$, $[B\ cF]$, I_2, $\emptyset_{2,2}$, where F is the local observer matrix obtained by computing the MATLAB command `F=place((A-B*Ko)',C',eigF)'`, $Re(eigF_i) < 0$.



Pic. 2 – Follower System

All the states $\hat{x}$ estimated by the observers of every node are combined by a multiplexer and feedback received in input by every follower node, because the controller will use it to compute ε_i (as explained before).

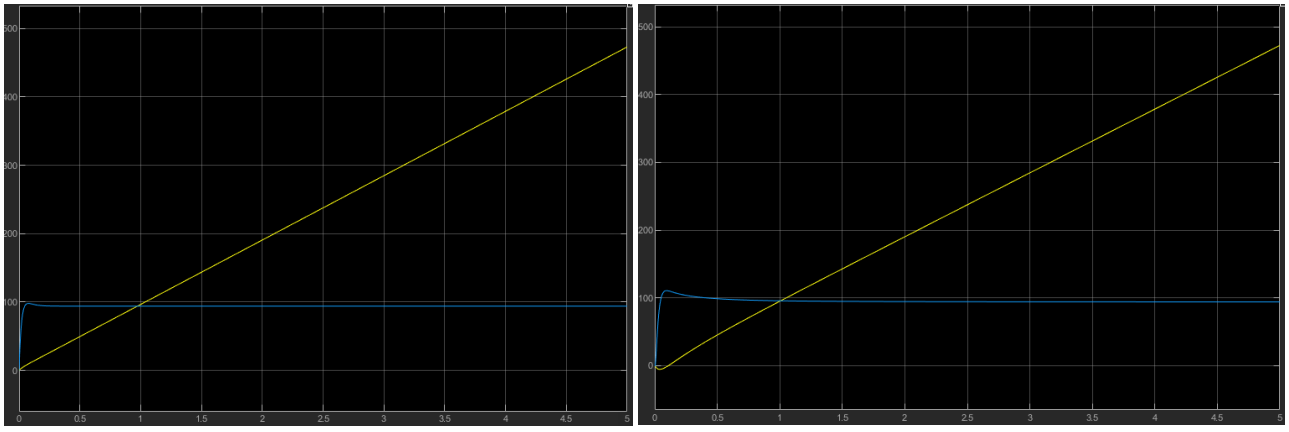


Pic. 3 – The entire system

It has been added a **White Gaussian Noise** to every follower output.

Results

The follower nodes correctly track the Leader, independently from the noise added to the system output, which only causes more time of convergence.



Pic. 4 - Leader and follower state evolution

We have noted that if we add too much noise on the leader, the state diverges from the reference input.

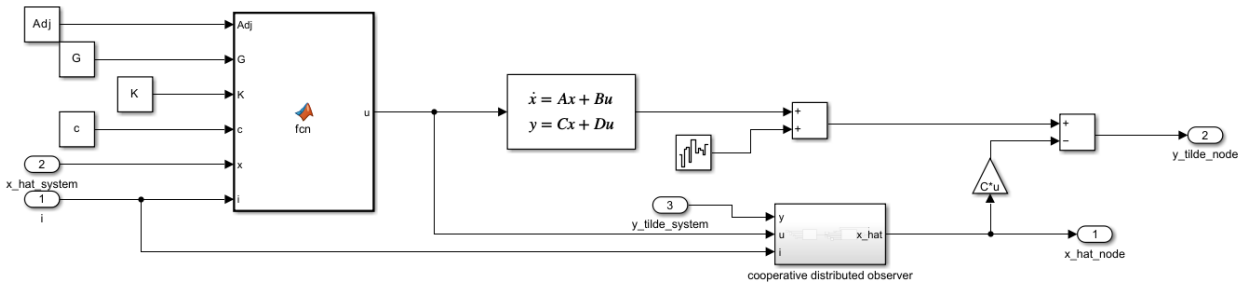
Case of Study II: Distributed Regulator based on a distributed Neighbourhood Observer structure

In the second case, a neighbourhood observer and a local controller is implemented on each follower with the primary goal of estimating the state of the system through the assessment of the neighbourhood.

As in the previous case, the local controller in the leader node is implemented by feedback, placing negative eigenvalues to K_0, the only difference is that in this case we also compute the $\tilde{y}$, that is the difference between the real output of the agent and the estimation provided by the local observer.

The inputs of the agent provided by the neighbourhood observer are $\hat{x}$ and $\tilde{y}$ that represent the state of all the agent present in the system. For each follower, the input goes through the control block and through the observer.

The follower nodes implement a local controller given by the SVFB control protocol $u_i = cK\hat{\varepsilon}_i$. The $\hat{\varepsilon}_i$ is the local neighbourhood state estimation error, computed using $\hat{x}$ instead x .



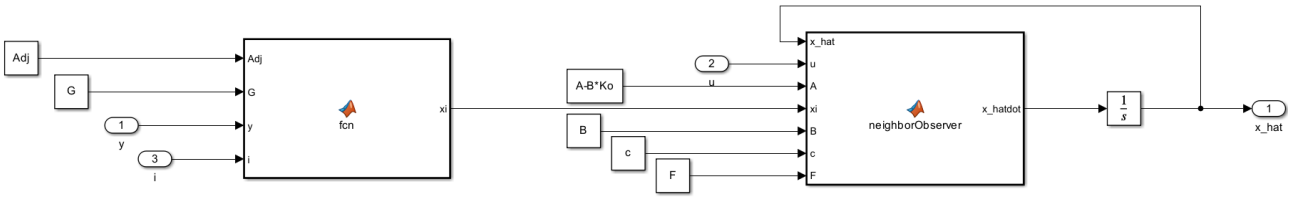
Pic. 5 - Structure of follower

$\hat{x}$ and $\tilde{y}$ are provided by the **cooperative distributed observe**r, having these dynamics:

$$\dot{\hat{x}}_i = A\hat{x}_i + Bu_i - cF\xi_i$$

where the term ξ_i is the ***local neighbourhood output estimation*** **error** given by:

$$\xi_i = \sum_{j=1}^{N} a_{ij}(\tilde{y}_j - \tilde{y}_i) + G_{ii}(\tilde{y_0} - \tilde{y}_i)$$

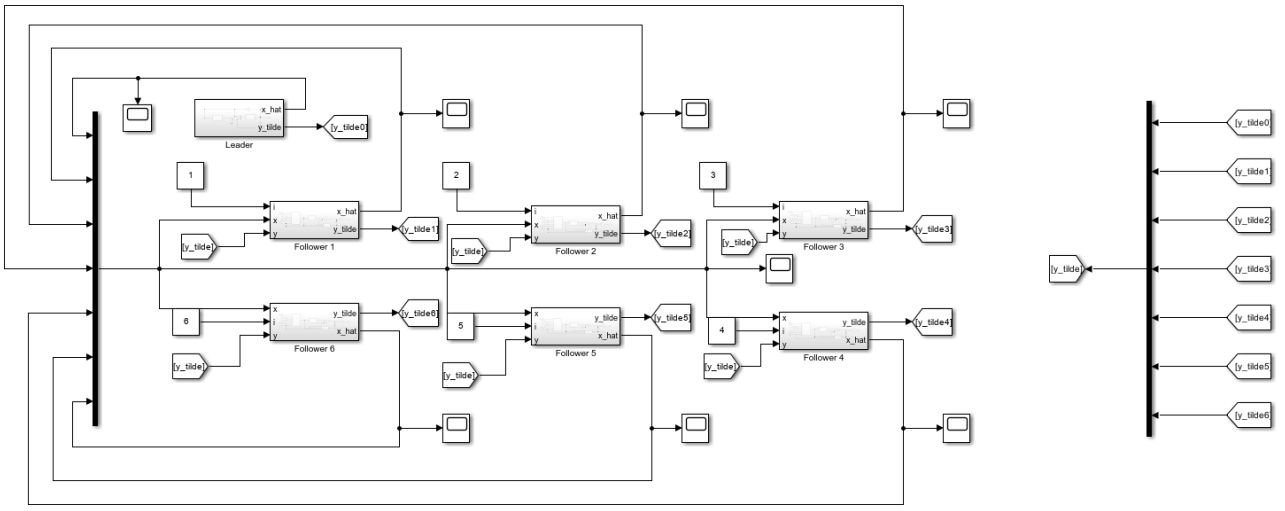


The **local closed-dynamics** is given by:

$$\dot{x}_i = Ax_i + cBK\left(\sum_{j=1}^{N} a_{ij}(\hat{x}_j - \hat{x}_i) + g_i(\hat{x_0} - \hat{x}_i)\right)$$

$$\dot{\hat{x}}_i = A\hat{x}_i + Bu_i - cF\left(\sum_{i=1}^{N} a_{ij}(\tilde{y}_j - \tilde{y}_i) + g_i(\tilde{y}_j - \tilde{y}_i)\right)$$

The following photos is the entire system built on Simulink.

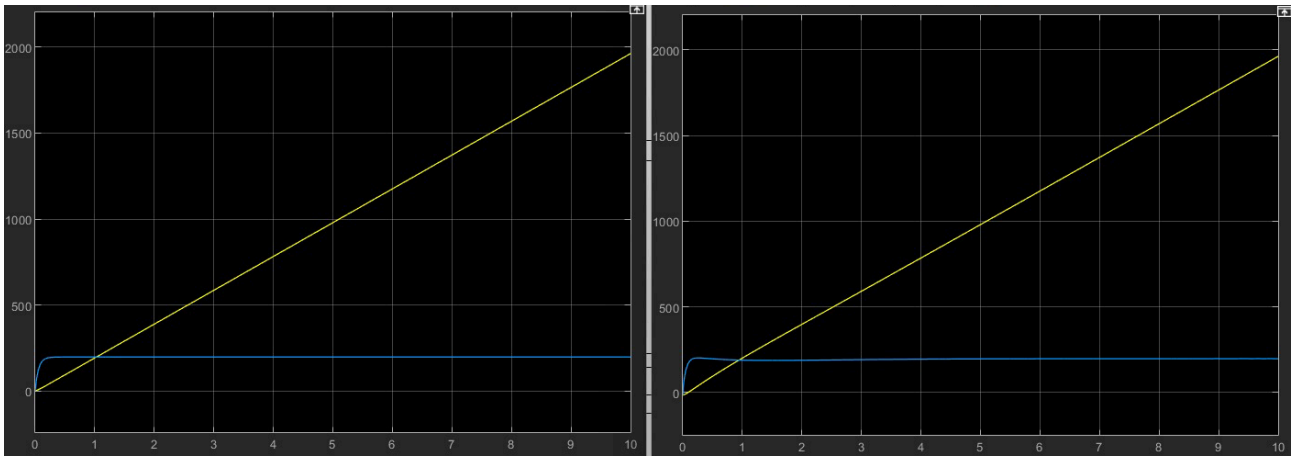


Pic. 6 – The entire system

It has been added a **White Gaussian Noise** to every follower output.

Results

After some simulations and analysis all the followers of the system converge to the state of the leader. Also adding white noise on the output of the follower node, consequently on the data exchanged by the follower, the system converges to the consensus.

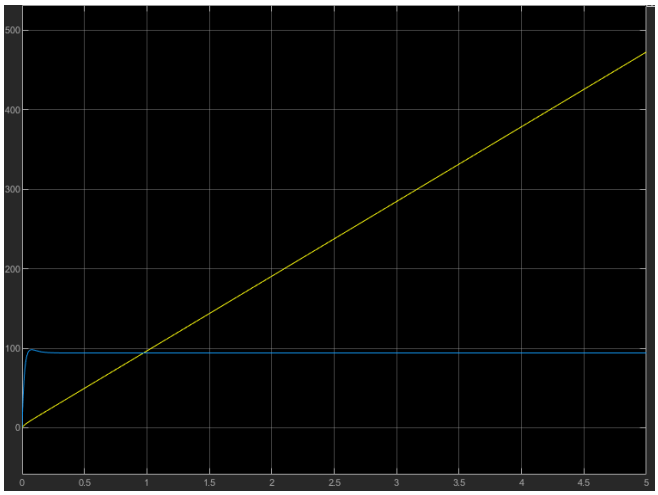


Pic. 7 - Leader and follower state evolution

We have noted that if we add too much noise on the leader, the state diverges from the reference input.

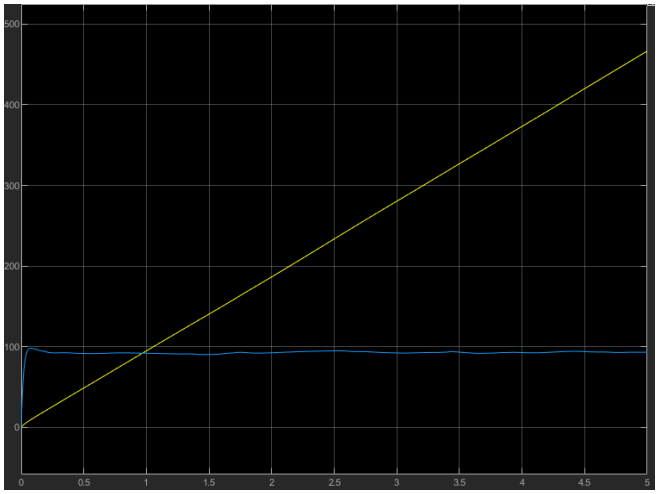
Comparison

By adding noise also to the leader node output, we can notice that the behaviour is not correct anymore, because the trajectory diverges after some time (it does not follow the defined state evolution).
As follows we can notice how also a white noise with a little variance can create unwanted oscillations of velocity in the case of a ramp.

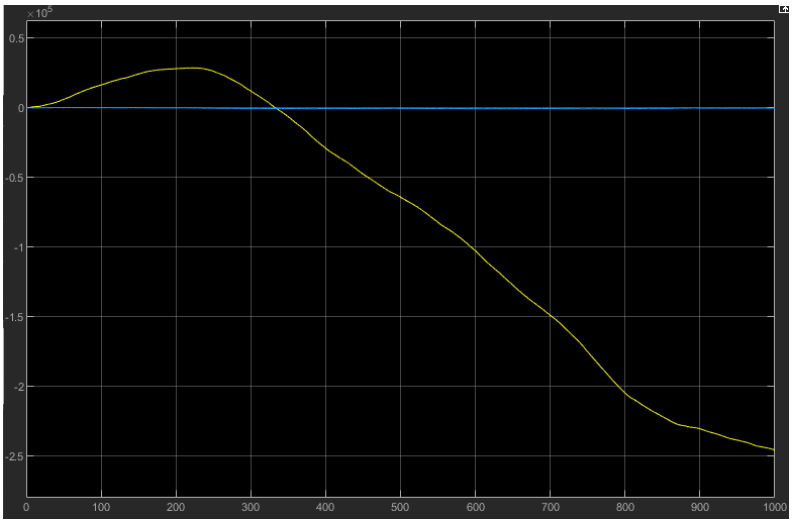


Pic. 8a - State variables without noise on leader

Pic. 8b - State variables with little noise on leader



By increasing the variance on leader noise, we will observe instability:

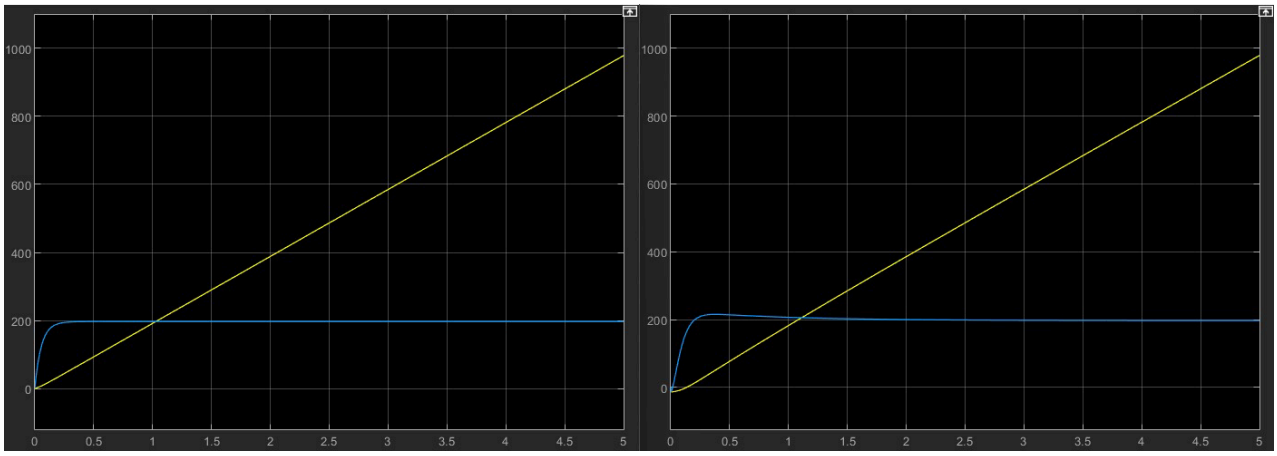


Pic. 8c - State variables with much noise on leader

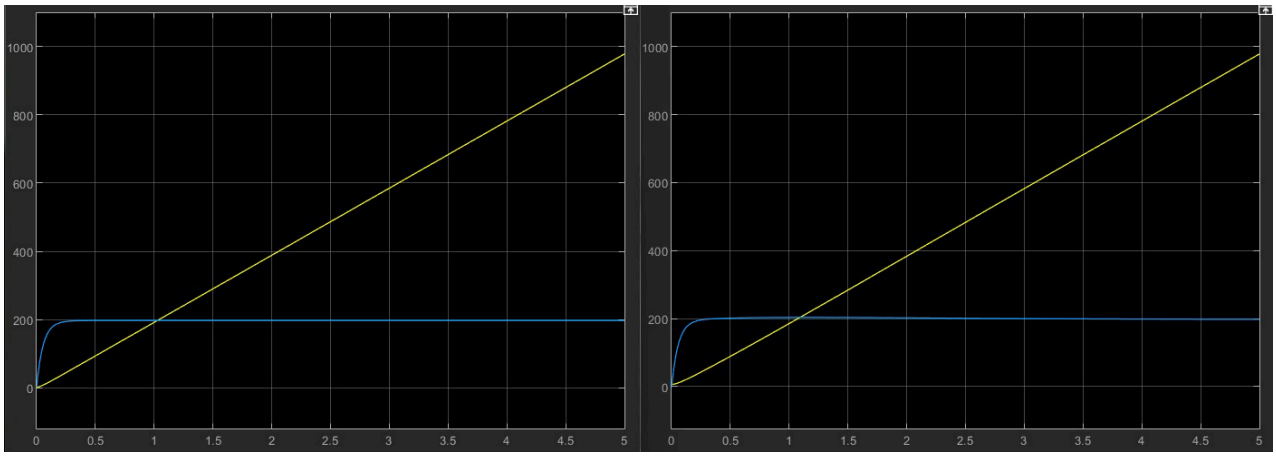
In our discussion we will consider that the leader node is not affected by any noise (the noise block has variance equal to 0). The addition of noise on the followers' outputs causes more instability and a higher time of convergence.

We said there is a lower bound for choosing the value of the coupling gain c, but we had not spoken of the upper bounds.

By increasing the coupling gain the follower systems have a smaller transient, i.e., it converges more rapidly. The only upper bound would be that imposed by the velocity of the leader node, because follower nodes velocity saturates when they try to converge faster than the leader state evolution.

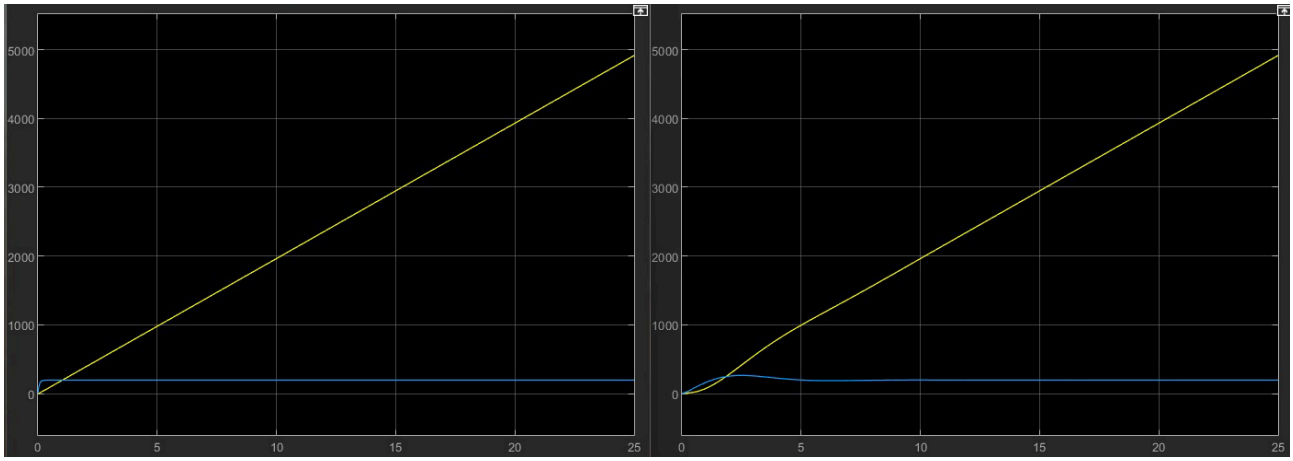


Pic. 9a – c = 2.2724

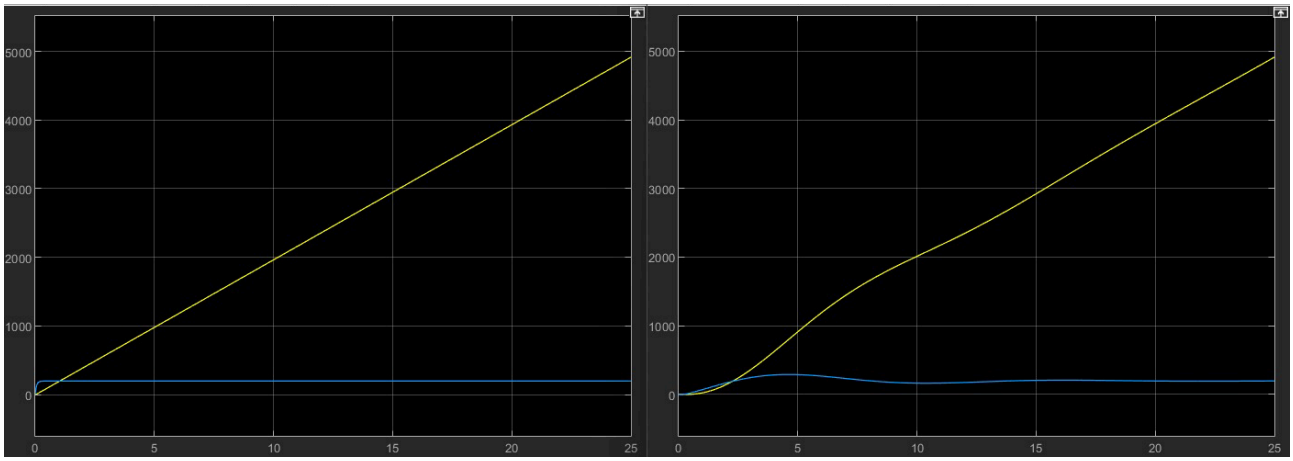


Pic. 9b – c = 227.24

By increasing the value of R we notice that the time of convergence of the state variables increases too. Actually, we can see that the velocity of the followers speeds up slowly.

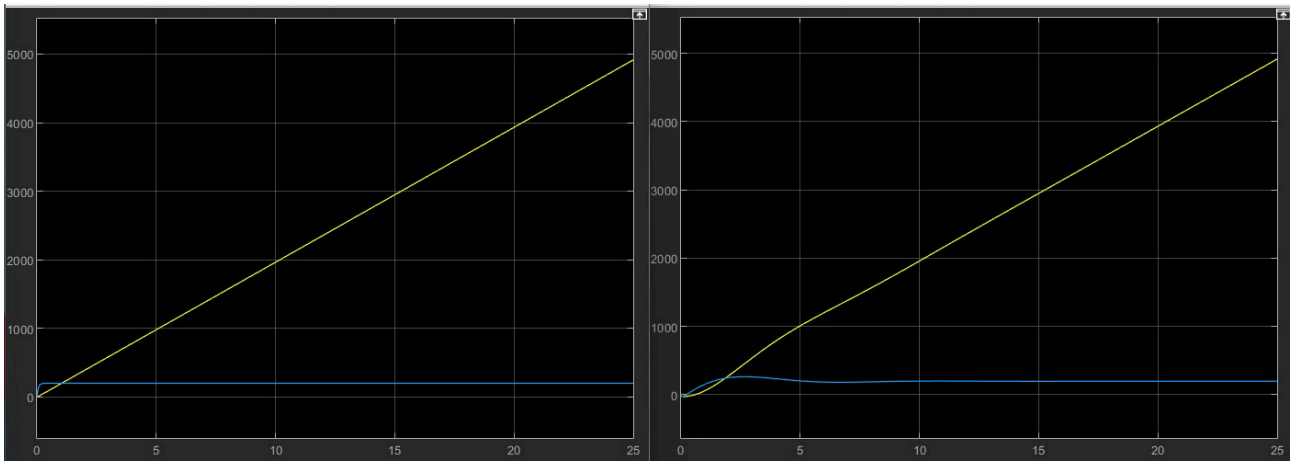


Pic. 10a – R = 1

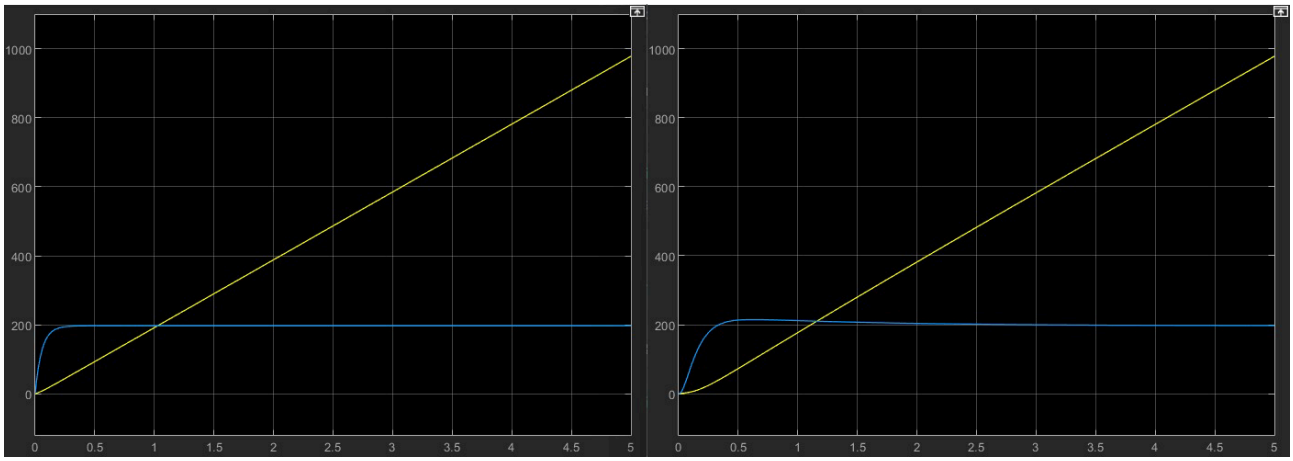


Pic. 10b – R = 10

Similar considerations can be reached by decreasing the value of matrix Q. In fact, if the values inside the matrix Q are higher, the followers tracking will converge faster.



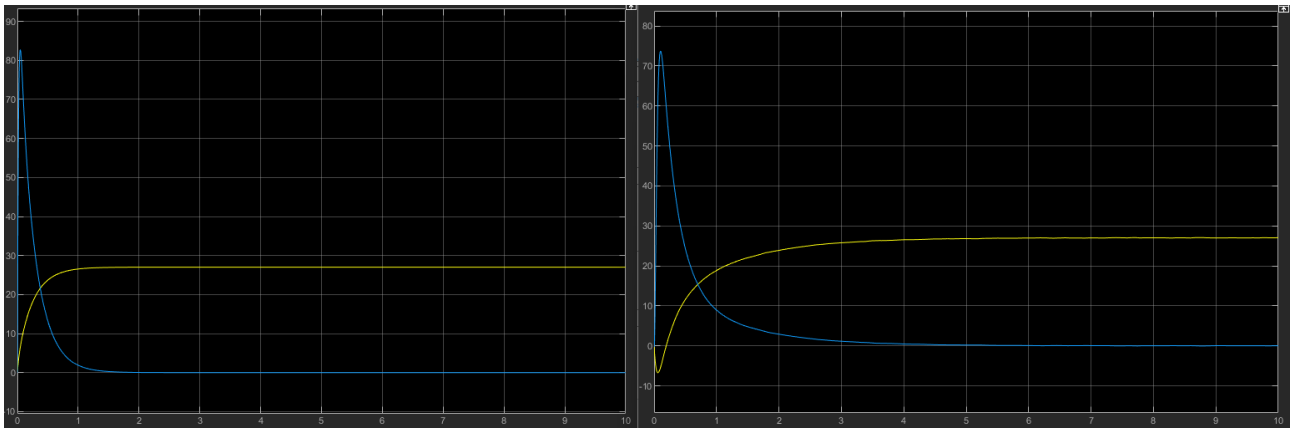
Pic. 11a – Q=eye(2)



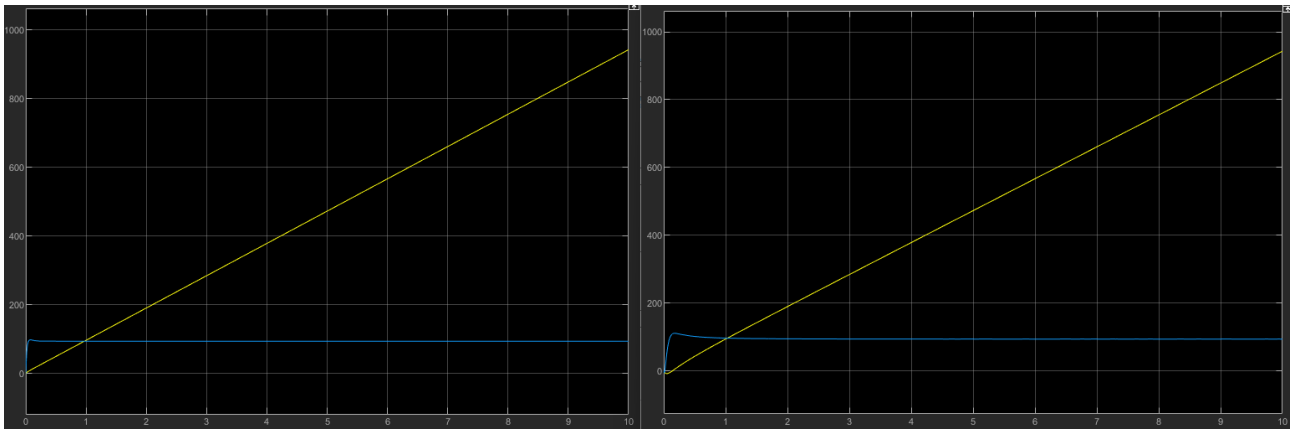
Pic. 11a – Q = 200*eye(2)

Plot of three kinds of references

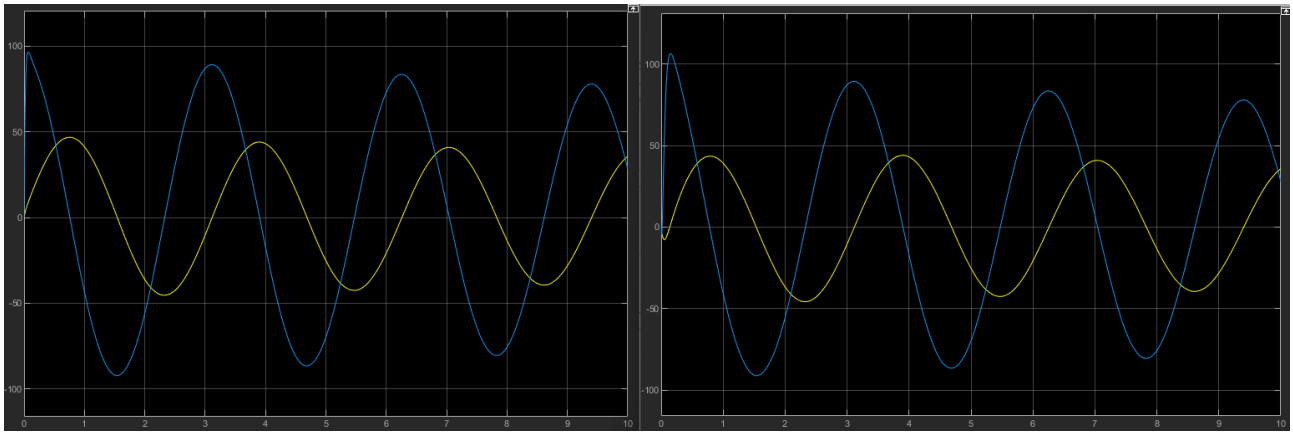
The plots of the two simulations are quite similar. As follows there will be the plots of the first case of study.



Pic. 12 – Step



Pic. 13 – Ramp



Pic. 14 – Sinusoidal signal

Final considerations

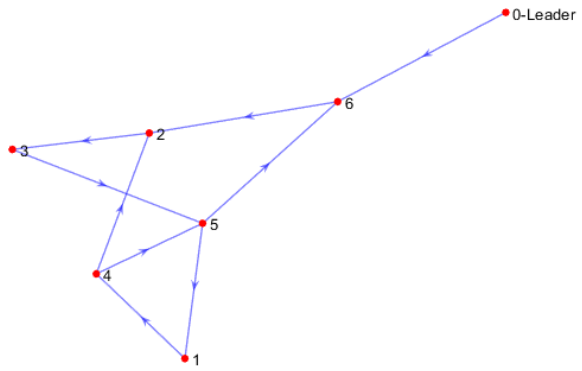
The project investigated the distributed control of a multi-agent magnetic levitation system, a key example of cyber-physical systems (CPS). By employing local and distributed neighborhood observers, it was demonstrated that follower nodes can effectively track the state of the leader node, even under noisy conditions.

However, managing noise, particularly when introduced into the leader node, presents a significant challenge as it can destabilize the system and cause undesirable oscillations. Additionally, optimizing the coupling gain c revealed the need to balance response speed and system stability; excessive increases in gain can lead to the saturation of the follower nodes' speeds. Variations in control parameters, such as the R and Q matrices, further emphasized the necessity for precise calibration to maintain optimal convergence times.

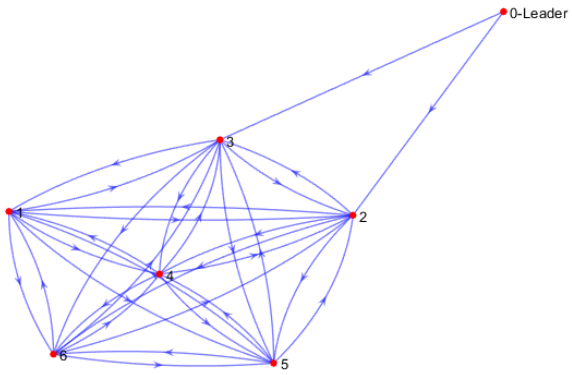
As far as concerning the communication networks, we found three possible configurations:

- **A)** The weights of the arcs are high, how we theoretically expected, the follower nodes tracking is slower
- **B)** In this configuration, we have all the follower nodes connected each other with the same weight, while the leader is connected only to nodes 2 and 3
- **C)** This configuration is a middle view between the preceding ones.

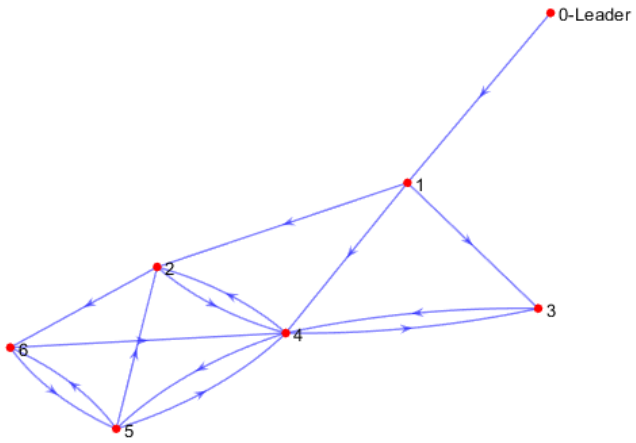
We wrote the function `graph_plot.m` to plot the full digraphs.



Pic. 15 – Configuration A



Pic. 16 – Configuration B



Pic. 16 – Configuration C