Modeling and Control of Cyber-Physical Systems

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

Politecnico di Torino
A.A. 2023/2024

Summary

Project Part 1

Introduction

In the first part of the project, we will explore some algorithms and apply the studied mathematical models in order to estimate the state of a system.
In particular, we will take into consideration the case of localization of some targets in a two-dimensional indoor area, which state is measured through sensors, some of which are under adversarial attack.
The following paragraphs will analyze five different situations and methods.

Task 1: Implementation of ISTA

The first task refers to the implementation of **IST Algorithm** to solve the **LASSO** problem:

$$\min_{x\in\mathbb{R}^p} \frac{1}{2}\|Cx-y\|_2^2+\lambda\|x\|_1$$

where λ is a scalar design parameter.

Goal

The main goal is to learn how the **Iterative Shrinkage Thresholding Algorithm** (**ISTA**) works and to understand how reliable it is. Moreover, we want to understand how the algorithm results change when the dimensions of the C matrix are modified, namely q (measurements of different sensor nodes) and p (different inputs).
The aim of this task is to implement ISTA using the shrinkage/thresholding operator. To do this, we change the parameters, such as τ and λ, and run it at least 20 times.

Implementation

Our code is composed of two different files. The first has the aim of *wrapper* of the operating function, which means that it only performs the statistical results, running several times the operating code like a *black box* and estimating the accuracy rate under different conditions.
The second file is the most important, because it contains the real implementation of the ISTA, computing it under given parameters. The core section of the code is composed of the while loop in which we apply a function called ***thresholding*** to update the state estimate. It receives as input the values $x+\tau * C^{\mathrm{T}}(y-Cx)$ and $\gamma=\{\gamma_i\}_n$ and, at each step, we update the state:

$$x_{k+1}=S_\gamma[x_k+\tau C^T]$$

Then we compute the square norm of the difference between the state at the step k and the state at the step $k+1$ in order to compare it with a tolerance threshold to define the ***ending condition***. The initial condition of the estimate is $x_0=\{0\}_n$.
The thresholding function is implemented in the file `thresholding.m`, it compares every element of the estimated state x with a given threshold γ with reference to the following rule:

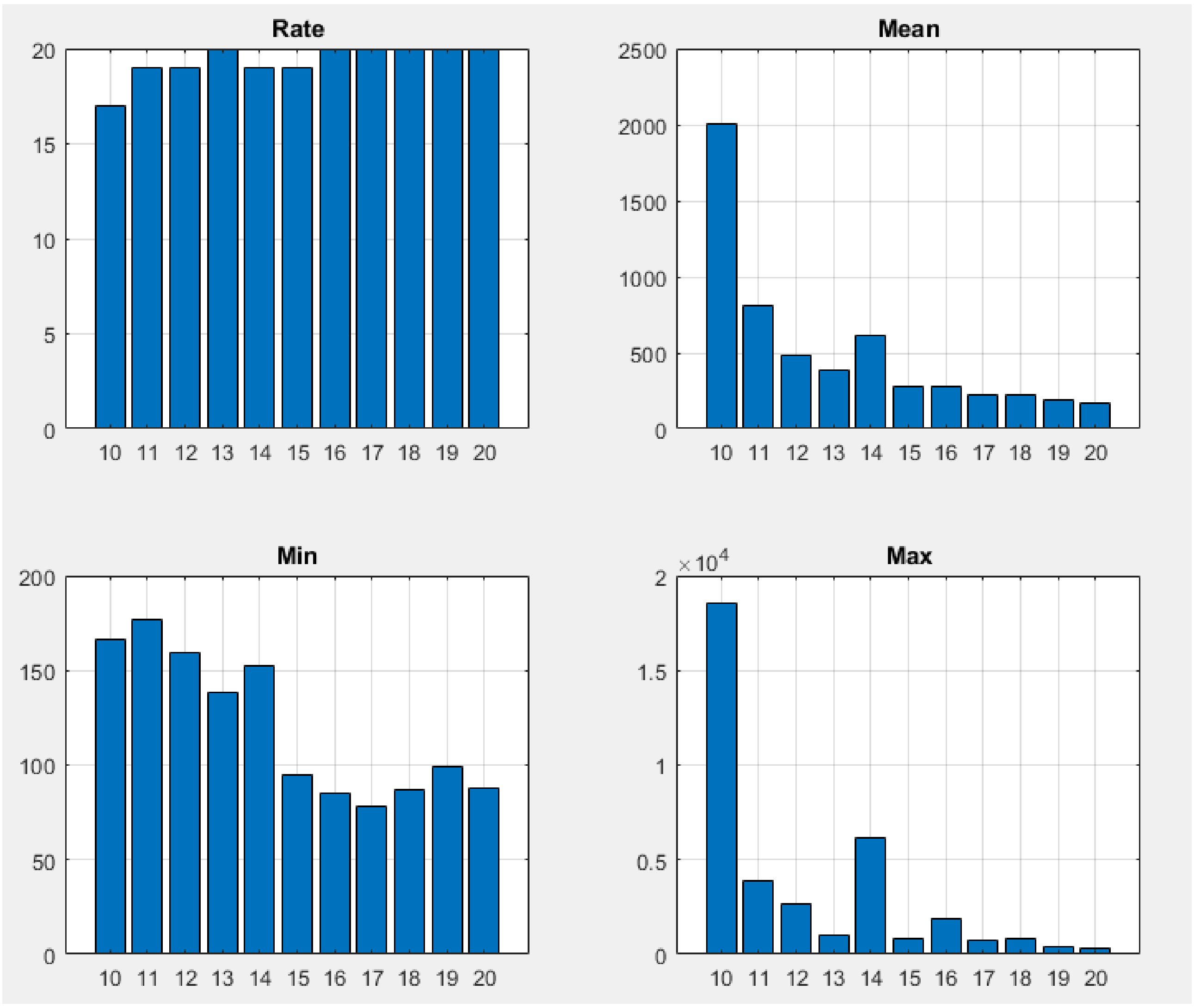
$$S_{\gamma_i}(x_i)=\begin{cases} x_i-\gamma_i & if\ \ x_i>\gamma_i \\ x_i+y_i & if\ x_i<-\gamma_i \\ 0 & if\ |x_i|\le\gamma_i \end{cases}$$

At the end, we apply a pruning process to the estimated results to set all values in the estimate that are below a chosen tolerance to 0. This tolerance is selected based on observations of the estimates.

Result

After 20 iterations, we can analyze the obtained results. When q is lower than p, we achieve approximately 85% accuracy. By increasing the value of q until it equals p, we can correctly estimate all 20 supports. The maximum number of iterations to meet the exit condition is 7938, the minimum is 134, and the average is 1367. Moreover, we notice that lower values of τ increase the number of iterations needed to reach the exit condition.
Conversely, when keeping tau constant and varying λ, we observe that lower values of λ require more iterations, while higher values of λ require fewer iterations.

Plot



Pic. A - Bar graphics with respect to the value of q (x-axis)

Task 2: Secure estimation under sparse sensor attacks

In the `Task 2`, we need to apply ISTA given the measurements $y = C\tilde{x} + a + \eta$ to estimate the position of the targets with h-sparse sensors under attacks.

Goal

The central purpose is to understand and demonstrate that the ISTA works and is highly accurate even in the presence of sensor attacks when these attacks are h-sparse and $h << q$.

Our task is divided into two cases: the first where the attacks are **unaware**, and the second where the attacks are **aware**. We say that the measurements are affected by attacks when they are given by $y = Cx + a$, where a represents the sensors' attacks.

The attacks a_i are unaware when they do not consider the measurements of the sensors. They simply add random uniformly distributed values. In the case of aware attacks, instead, a is an attack of the form $y = C(x + w)$, where a is a multiple of the real measurements (y_{true}), i.e., $a = Cw$.

Implementation

This task comprehends three different files, the first is a wrapper which aims to gather some statistical information about the estimate accuracy in the two cases of "unaware" or "aware" attacks by running the two cases 20 times.

In particular, the "unaware" case is implemented into the file `Task2_unaware.m` and consists in h sparse attacks with random values uniformly distributed in the set $[-2, -1] \cup [1, 2]$. This implementation resumes the preceding one by adding to the output the attacks; specifically, we define $G = [C \quad I_q]$ and $\tilde{z} = [\tilde{x} \quad a]^{\mathrm{T}}$. The resulting measurements are so given by $y = G\tilde{z} + \eta$.

The "aware" case, instead, is implemented into the file `Task2_aware.m` and requires that the attacker would know the measurements of the sensors. So given q nodes, $h \ll n$ sensors are modified by adding to the real value of the measurements under attack its half:

$$y_{i-new} = y_{i-old} + \frac{1}{2} y_{i-old} = \frac{3}{2} * (C\tilde{x} + \eta)$$

In both cases, the implementation of ISTA is the same as the `Task 1`, with initial condition of the estimate equal to $z_0 = \{0\}_{n+q}$. The estimate in this task includes both the values of the state $\tilde{x}$ and the support of the attacks a.

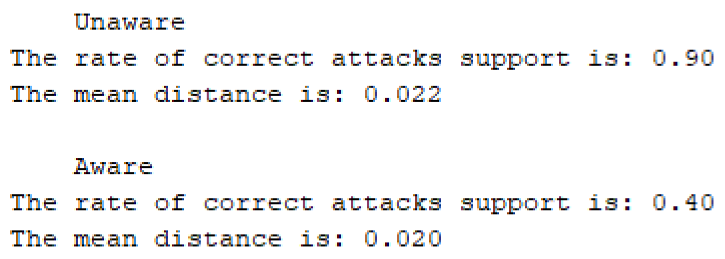
Result

After 20 iterations, we have noticed that there are differences between the "aware" case and the "unaware" one.

The rate of correct estimation of the attacks in the case of unaware attacks is about 90%, whereas the rate in the case of aware attacks is about 40%. This difference is due to the structure of the attacks. In the case of an aware attack, the offender knows the structure of the system and it is able to modify the value of the sensor output.

On the other hand, when the attack is unaware, the offender only adds a random value to the sensor output, which, in our discussion, is normally distributed; therefore, in this case it is easier to identify the sensor under attack.

Additionally, the square norm distance, defined as $\|\tilde{x} - \hat{x}\|_2^2$, which is the distance between the true value and the estimation, follows this trend. The mean distance is 0.025 in the case of an aware attack, and 0.001 in the case of an unaware attack. The state estimate is acceptably close in both situations.

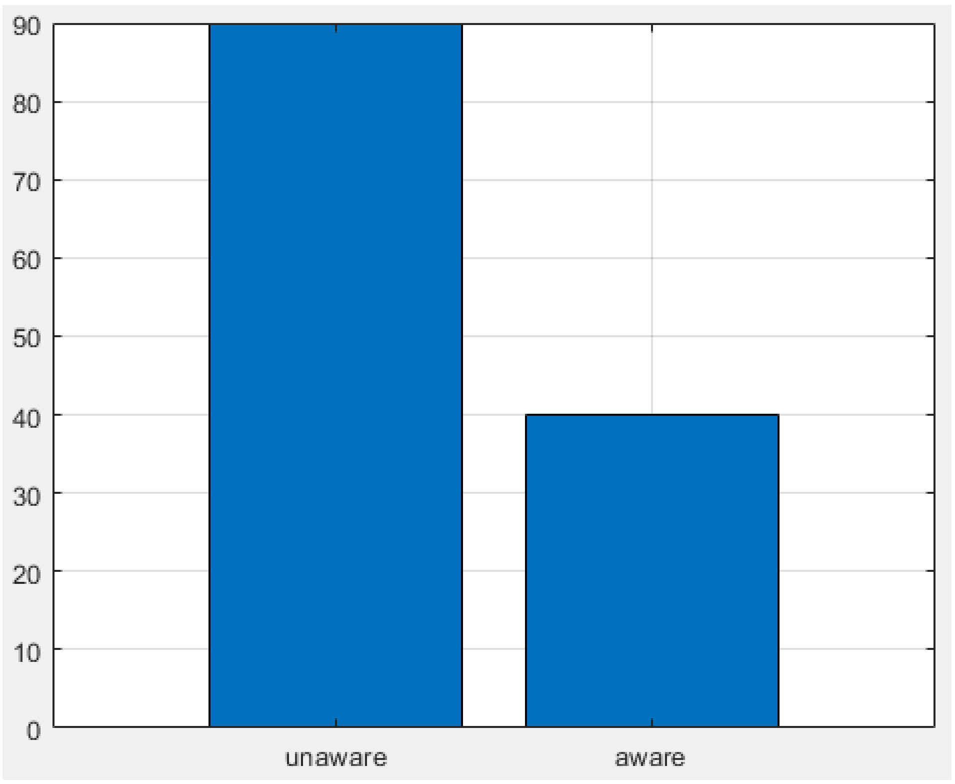


```
    Unaware
The rate of correct attacks support is: 0.90
The mean distance is: 0.022

    Aware
The rate of correct attacks support is: 0.40
The mean distance is: 0.020
```

Pic. B - Comparison of rates between unaware and aware attacks on 20 runs

Plot



Pic C – Rate of correct support estimation

Task 3: Target localization under sparse sensor attacks

In the `Task 3` we need to estimate the position of three targets positioned in a room of 100 cells and 25 sensors. To localize the targets and identify the sensors under attack, implement ISTA to solve the following weighted Lasso of the **RSS-Fingerprinting** method:

$$\min_{x\in\mathbb{R}^p, a\in\mathbb{R}^q} \left\| (D \quad I) \binom{x}{a} - y \right\|_2^2 + \lambda_1 \|x\|_1 + \lambda_2 \|a\|_1$$

The RSS-fingerprinting is a localization technique that employs the received signal strength (RSS) to determine the position of a target within an indoor area.

Goal

The objective is to understand and implement the RSS-fingerprinting technique by applying the ISTA. RSS-fingerprinting relies on a grid where sensors are placed to correlate the received signal strength with different positions. It consists of two phases: the ***training phase***, where each sensor measures and stores the RSS in order to create a dictionary D; and the ***runtime phase***, during which each target broadcasts a signal and sensors measure the RSS.
In our case, the training phase has already been executed and we are provided with D, with which we must implement the ISTA algorithm and determine which cells of the grid are occupied by a target and which sensors are under attack.

Implementation

This task is implemented in the file `Task3.m`. It applies ISTA like in the previous tasks but with different data. In this case, the output measurements are obtained by multiplying the state with a dictionary D, given as known by loading the file `localization.mat`. The output measurements are also given in this task.

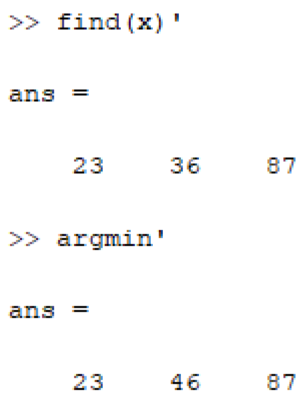
Like previous tasks, we define a matrix $G = [D \quad I_q]$ and then we normalize it; the estimate is a column vector $z \in \mathbb{R}^{p+q}$ initially equal to a vector of zeros: $z_0 = \{0\}_{p+q}$. The vector Γ comprehends p values equal to $10\lambda\tau$ and q values equal to $20\lambda\tau$.
Other methods can be applied to this problem. We focused on the ***k-NN* algorithm**, assuming that the sensors are *attack-free*. It is characterized by k chained loops (three in our case, because we assume that the number of targets is known) from 1 to p.
Each iteration computes the square-norm distance between the measurements y and the sum of the k values in the dictionary. This is compared to the actual minimum value, if it is lower, we update it and save its arguments. At the end we obtain an estimate of the support of the state (the position of the targets) given by:

$$[x_1, \ldots, x_k] = \arg\min \left\| \sum \left(D(:,i) \right) - y \right\|_2^2$$

If we want to consider the presence of attacks and estimate their support, we must know the number of sensors attacked and add other chained loops, as many as the number of attacks.

Result

The results obtained from the IST Algorithm indicate that it performs excellently. Indeed, as we can observe, the outcomes are correct and in line with the expected solution. Our algorithm converges and is able to accurately estimate which grid cells are occupied by the target and which sensors are under attack. In our problem we obtained that the support of the state is $\{23, 36, 87\}$ while the support of the sensors under attack is $\{12, 16\}$.
On the other hand, evaluating the results obtained from the k-*NN* algorithm, we can notice that it is not as precise, as one of the state estimates is incorrect. This occurs because k-*NN* considers the state of the sensors as if they were attack-free. In our problem, the support of the state estimated is $\{23, 46, 87\}$.
An additional problem with k-*NN* is that, if we wanted to estimate the attacks, the computational complexity would become very high, as it is proportional to the number of variables to be estimated (3 for the states and 2 for the attacks), leading us to perform 10^{10} iterations, which is a significantly large number. At the end, k-*NN* algorithm requires to know the number of targets and sensors under attack.



```
>> find(x)'

ans =

    23    36    87

>> argmin'

ans =

    23    46    87
```

Pic. D – Comparison with the support estimated by applying ISTA (first) and the support estimated with k-NN (second)

Task 4: Sparse Observer

In the `Task 4` we start from the preceding problem in the case of moving targets, according to the following dynamics:

$$x(k+1) = Ax(k)$$

where A and the output are given. In our problem, we deal with three targets moving left at each step.

Goal

The objective is to implement the **RSS-fingerprinting** technique in case of moving targets with the sparse observer. The **Sparse Observer** is an algorithm used to estimate the state of a dynamic system using only a limited set of measurements.
It represents an estimation technique that leverages the sparse structure of the system to obtain accurate state estimates even with a limited number of observations.
We want to **correctly estimate the state and the sensors under adversarial attack**. We also want that once the state has converged it follows the correct evolution of the targets, which means that the state estimation moves to the left at each step. In our implementation we will consider three cases: u*naware* attacks, *aware* attacks on fixed sensors, a*ware* attacks on *changing* sensors.

Implementation

This task is implemented in the file `Task4.m`. We have obtained the estimate of state and the estimated of sensor under attack thanks to the implementation of **Sparse Observer**.
Like previous tasks, we define a matrix $G = [D \quad I_q]$ and then we normalize it; the estimate is a column vector $z \in \mathbb{R}^{p+q}$ initially equal to a vector of zeros: $z_0 = \{0\}_{p+q}$
The sparse observer uses the ISTA algorithm as implemented in `Task 3` to estimate $\hat{z}^+$ at each iteration starting from the previous **prediction** of the state and of the attacks' support. We also save the local result at each iteration in the Z_matrix (code variable) in order to visualize the change of estimation during time. In last, we update the prediction of $\hat{z}$ taking the estimate of the current state ($\hat{z}^+$) and multiplying with matrix A. Our implementation also considers an aware state, which randomly creates the support of true state and sensors under attacks. Then, we also generate the measurements' matrix Y. Then, we add to real measurements the aware attacks, following the same rule of other tasks:

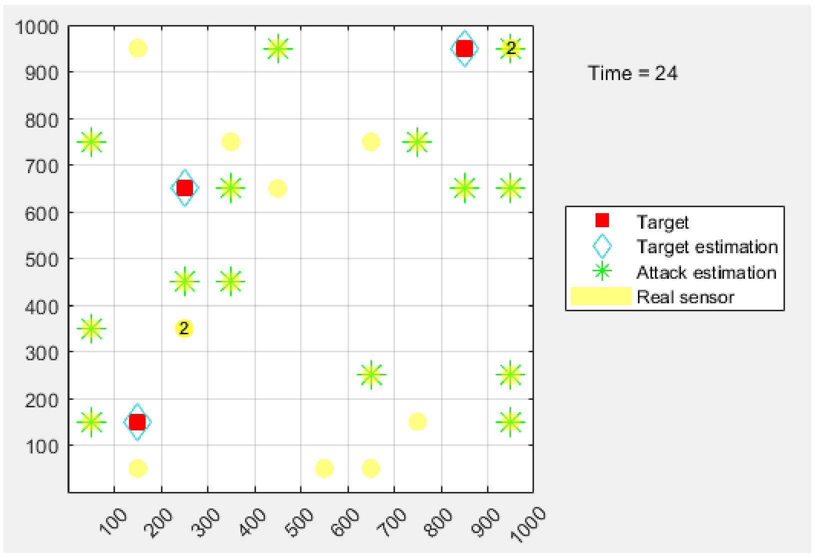
$$y_{i-new} = y_{i-old} + \frac{1}{2} y_{i-old} = \frac{3}{2} * (C\tilde{x} + \eta)$$

The matrix is generated in n_{iter} steps by applying the preceding formula. In the change sensor mode, the support of the true attacks' changes after $n_{iter}/2$ iterations.
Notice that in the aware case the estimate stops to the preceding targets' positions for graphical reasons.
A graphical visualization of the sensors' estimation at each time k is available thanks to the changes made to the function `plot_field.m` and to the function we created `plot_sensor.m`.

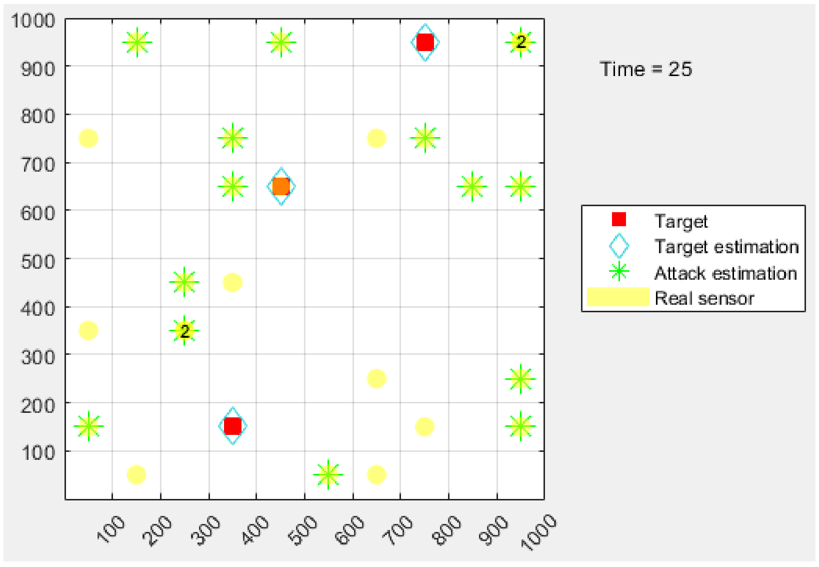
Result

In the *unaware* case, we can observe that the algorithm converges at the 24th iteration, as we can also see in the plot. The target state estimates accurately match the actual target states, and the estimates for the attacked sensors are precise.
In the case of *aware* attacks on two sensors, we see that the algorithm converges at a variable step (depending on the randomly generated matrix A) and correctly estimates both the state of the targets and the attacked sensors. Even when the sensors under aware attack change, our sparse observer accurately estimates both the system state and the attacked sensors.
When significantly increasing the number of attacked sensors, the system state is still estimated correctly, but the exact number of attacked sensors is not. Since the most important aspect in this kind of problem (i.e., real situations) is the state estimation, the implementation appears to be *resilient* to the aware attacks we conducted on the sensors.

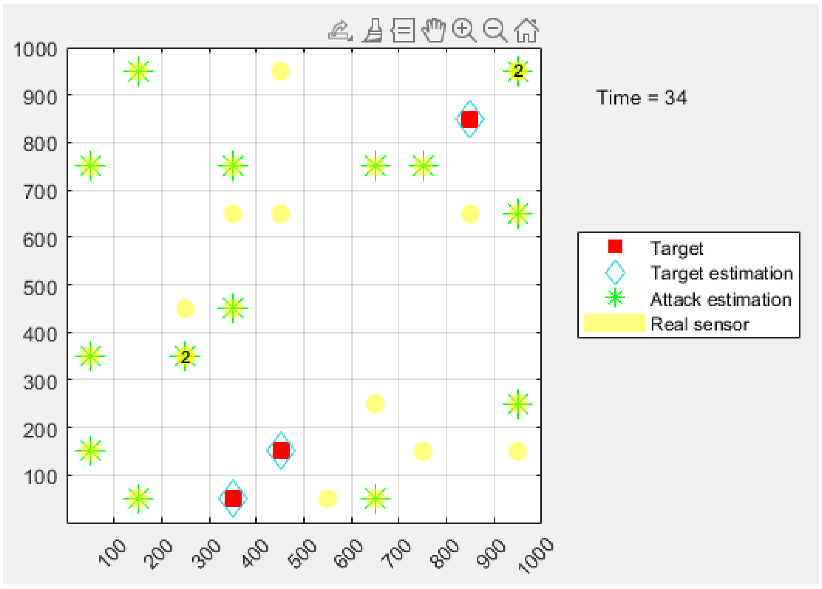
Plot



Pic. E – Unaware simulation



Pic. F – Aware simulation



Pic. G – Aware simulation + sensors' attacks changing

Task 5: Distributed targets localization under sparse attacks

In this task, we revisit the target localization problem from $Task$ 3 and address it in a distributed manner using the **Distributed Iterative Shrinkage-Thresholding Algorithm** (**DISTA**). The DISTA is used when Cyber-Physical Systems (CPS) are not composed of multiple agents and a fusion center, but the information is distributed among the various components of the CPS. The advantages of using distributed processing include:

- **Privacy:** Each device can keep some information private, ensuring data confidentiality.
- **Short Range Communication:** Devices only need to communicate with nearby nodes, reducing communication range and potentially lowering energy consumption.
- **Resilience:** The failure of a single device would not prevent the overall processing, enhancing the system's robustness and reliability.

Goal

The main objective of this task is to verify whether the application of DISTA can achieve the consensus condition. In the given dynamical system $x(k+1) = Qx(k)$, where Q is the matrix representing the connectivity graph of the sensors, Q solves the ***Consensus problem*** if there exists α such that

$$\lim_{k\to\infty} x(k) = \alpha\mathbf{1}$$

where alpha represents the average consensus problem: $\alpha = \frac{1}{q}\sum_{i=1}^{q} x^i(0)$ and $\mathbf{1} = \begin{pmatrix}1\\1\\\vdots\\1\end{pmatrix}$.

Implementation

This task is implemented in the file `Task5.m`. It exploits the same ISTA method of the preceding tasks applied to the distributed localization. At each iteration, we calculate the local mean as $\sum_{i=1}^{q} Q_{i,j} z^{(j)}(k)$ with a for-loop, where Q is given, while z is the estimate of the state and of the sensors' attacks, initialized at $z_0 = \{0\}_{p+q}$. After that, we can recall the `thresholding` function, which receives as parameters:

1. $\sum_{i=1}^{q} Q_{i,j} z^{(j)}(k) + \tau G_i^{\mathrm{T}}(y_i - G_i z^{(i)}(k))$
2. $\Gamma = \tau\lambda = 4 * 10^{-7} * (10, \ldots, 10, 0.1, \ldots, 0.1)$

Finally, we update the terminating condition by summing the squared norm-2 of the difference between old and new z estimations. At the end of each loop the stopping variable is checked, if it respected the algorithm ends, otherwise the loop starts again. Before each loop starts, the terminating variable is set to 0.

After estimating the state and the attacks, the data are pruned and cleaned in order to eliminate unwanted values.

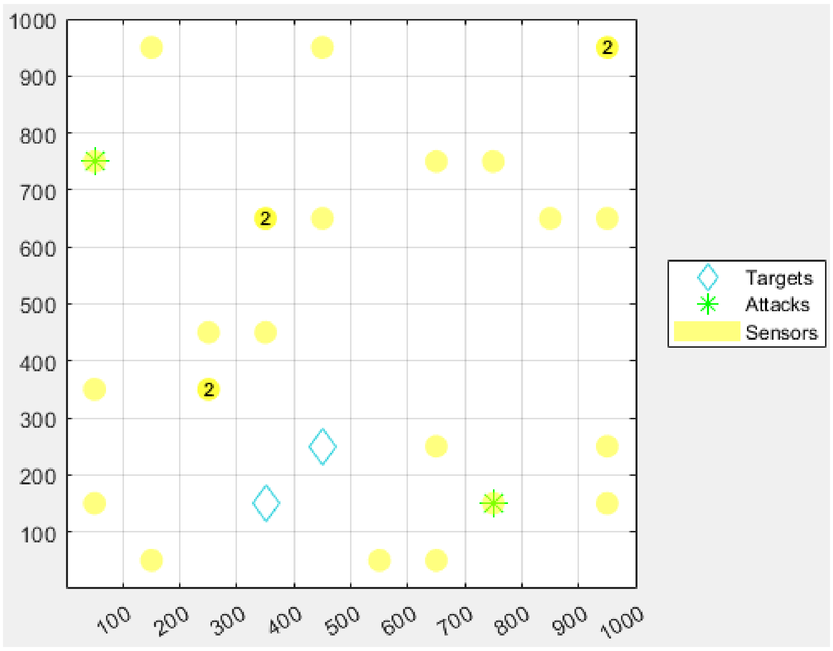
Result

In this chapter, we have presented the results of our study on the application of the distributed ISTA (DISTA) method for target localization. We are assuming that the network is perfectly synchronized, and the nodes collaborate properly.

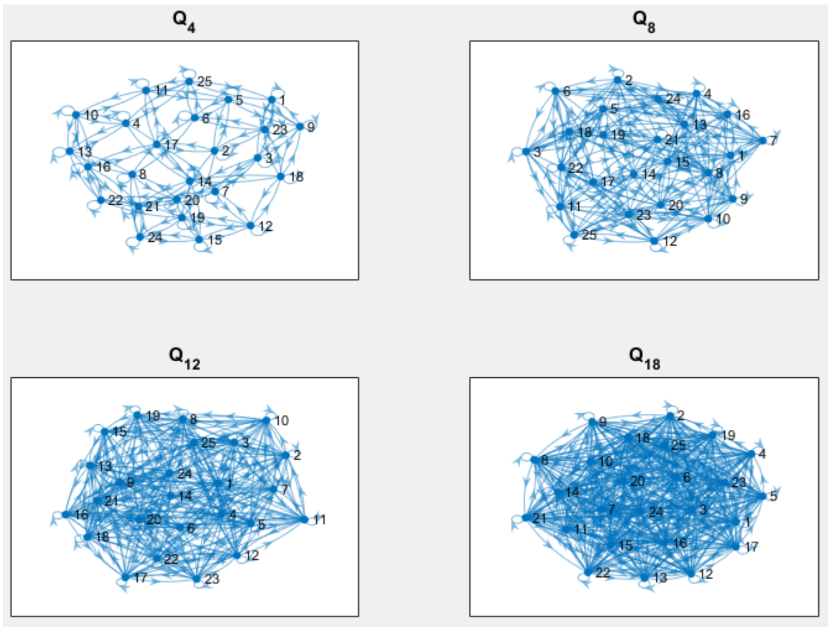
Our findings confirm that DISTA successfully achieves consensus among the distributed nodes. The iterative nature of the algorithm ensures that all nodes converge to a common solution, proving the robustness and reliability of the distributed approach. The final estimation produced by DISTA is accurate.

In this case, the eigenvalues of Q agree with the **Perron-Frobenius Theorem** and Q is doubly stochastic, so, the problem converges to a consensus that is the global mean.

Plot



Pic. H – Graphical representation of targets and attacks estimates



Pic. I – Graphs examples representations