Modeling and Control of Cyber-Physical Systems

Project I - Report

September 21, 2023

Carena Simone, s309521
Carmone Francesco Paolo, s308126
Mazzucco Ludovica, s315093
Pulinas Federico, s319881

Contents

List of Figures

1 Implementation of the Iterative Shrinkage-Thresholding Operator (IST)

When estimating the sensor state, some data might be corrupted by adversarial attack. In order for such errors to be correctable, it's necessary to check the sparsity of the attack vector by using the l_0-norm. Since the l_0-norm is non-convex and non-continuous, we must employ its closest convex approximation, which is the l_1-norm. Similarly as stated in the Compressed Sensing theory, relaxing the constraints on the l_1-norm, we formulate the problem as a LASSO:

$$\min_{x \in \mathbb{R}^p} \frac{1}{2} \|Cx - y\|_2^2 + \Lambda^T \|x\|_1 \tag{Eq. 1}$$

This kind of problem can be solved through the IST operator on a gradient step

$$x_{t+1} = \mathbb{S}_{\lambda\tau}\left[x_t + \tau A^T (y - Ax_t)\right] \tag{Eq. 2}$$

Where the IST operator is defined as follows

$$\mathbb{S}_\alpha(x) = \begin{cases} x - \alpha & \text{if } x > \alpha \\ x + \alpha & \text{if } x < -\alpha \\ 0 & \text{if } |x| \leq \alpha \end{cases} \tag{Eq. 3}$$

and shrinkage/thresholding represented by the $\pm\ \alpha$

1.1 Correct estimation of $\tilde{x}$ with respect to the sensors number q

In this task the number of sensors varies from 10 to 50 when q is small, the matrix is underdetermined, hence a less observable system with a smaller percentage of success. While q is increased the observability matrix becomes taller, admitting a greater percentage of success. By analizing the graph we can see that after a certain value of q, the percentage start to converge to 100%, stating the fact that the observability matrix reached a feasible size for the problem.

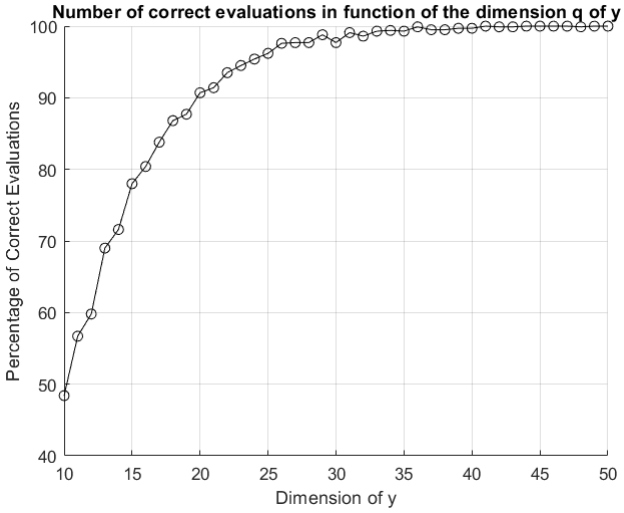


Figure 1: Percentage of correct evaluations in function of q

Regarding the **convergence time**, and considering the number of iterations related to the number of measurements:

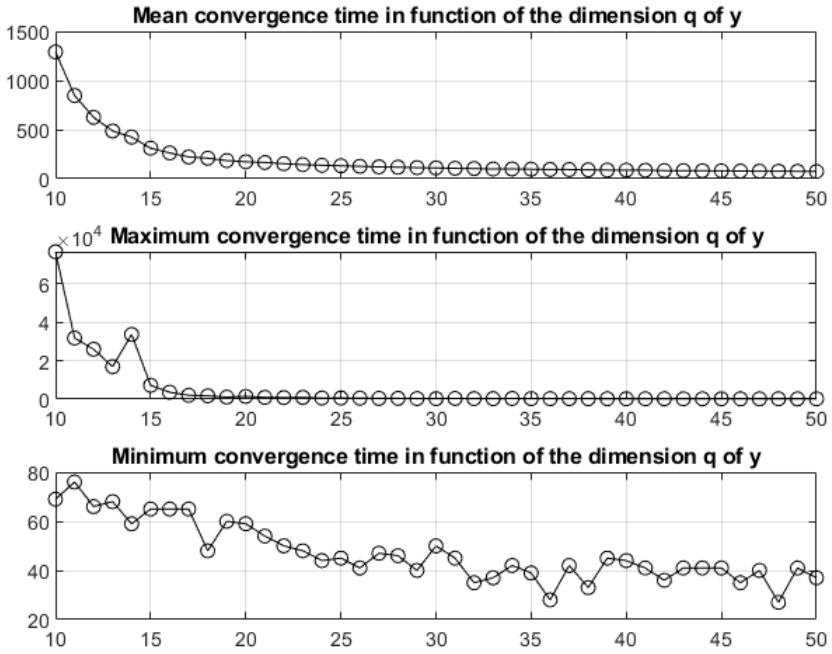


Figure 2: Number of iterations in function of q

1.2 Variation of τ

An interesting observation that can be done on the system is to check how the statistics vary according to the decrease of τ, keeping λ constant. It has to be clear that in this case what is changing is the gradient step length, while the IST term $\lambda\tau$ remains constant since we set $\lambda = \frac{1}{10\tau}$.

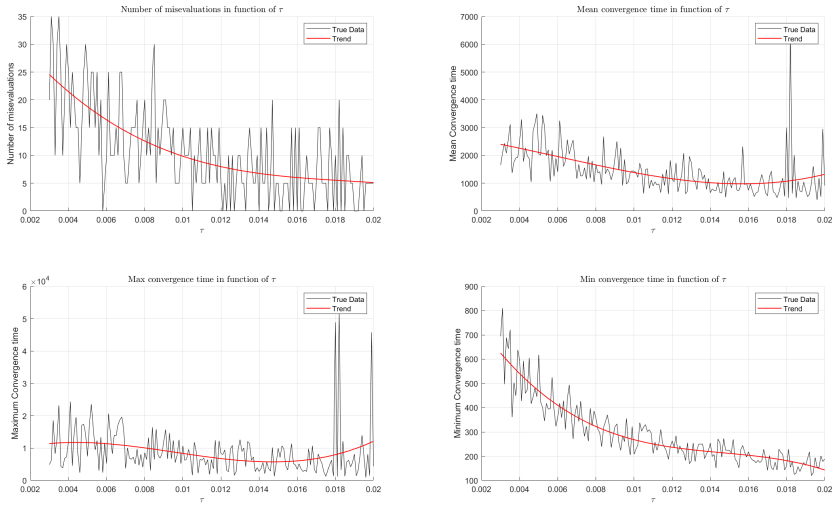


Figure 3: Number of miseavluations and iterations in function of τ

The plots above have been realized taking the maximum value of τ allowed to let the algorithm work ($||C||^{-2} - \epsilon$) and then letting it decrease. It does not have to reach too small values, if so

the algorithm will be stuck, in our case we chose 0.002 as lower bound.
The variation of τ influences how rapidly the IST moves toward smaller values of its argument, that is the function to be minimized : great values of τ mean great steps, resulting in smaller convergence times.

1.3 Variation of λ

Increasing its value and fixing τ, we obtain a variable value for the IST parameter of the $\lambda\tau$. As a consequence, the range of values affected by the threshold changes and the bigger it is, the larger the probability that values near zero are cancelled, leading to a lack of accuracy and misevaluations, although the convergence time decreases sharply.

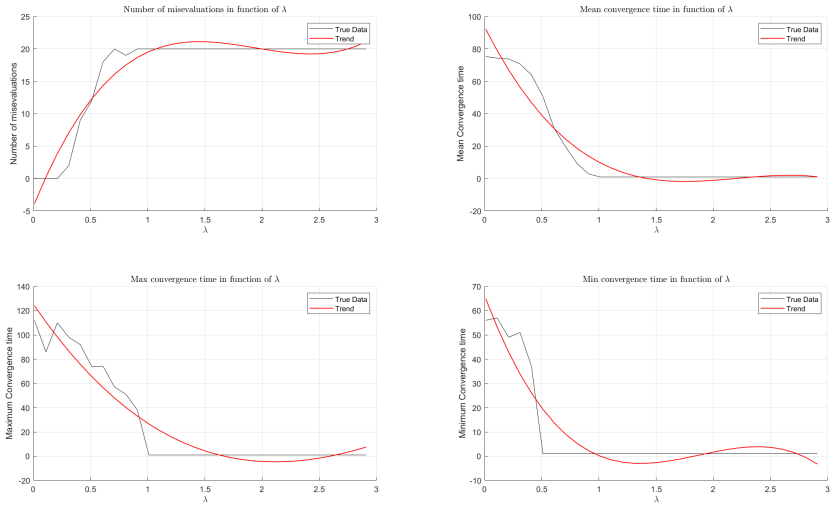


Figure 4: Number of miseavluations and iterations in function of λ

2 Secure estimation under sparse sensor attacks

Since sensor nodes are physically distribuited, it's plausible to assume that just a few of them can be attacked at the same time, hence the name sparse sensor attacks. We now deal with *static LTI systems* described as follows

$$\begin{cases} \tilde{x}(k+1) & = \tilde{x}(k) = \tilde{x} \\ y(k) & = C\tilde{x} + \eta + a \end{cases} \qquad \text{(Eq. 4)}$$

where a is the attack vector, and η are the measurement noises. The system state $\tilde{x}$ depends uniquely on its starting condition, since the input matrix B is null, and the dynamics matrix A is equal to the identity matrix I, which means that the system has no dynamics or behavior of its own. The output measurements y depends on the output matrix C and the state $\tilde{x}$, but not on the input u, since the feedthrough matrix D is null.
Our goal is to estimate the support of the attack vector (i.e. the indices) and the state $\tilde{x}$. By running the IST operator (Eq. 2) with the following parameters

$$\varepsilon = 10^{-8},\ \tau = \|C\|_2^{-2} - \varepsilon,\ A = \begin{bmatrix} C & I \end{bmatrix},\ \tau\Lambda = \begin{bmatrix} 0 & \dots & 0 & 0.001 & \dots & 0.001 \end{bmatrix}^T.$$

we're able to obtain the estimated vectors x_{found} and a_{found}. We compare the support of the latter with the support of the real attack vector a, eventually increasing the miss counter: a miss indicates a wrongly estimated support of the attack vector. Finally, we consider accuracy as the similarity between x_{found} and x; the lower the norm, the higher the quality of the estimation: when the norm is null, the state estimate and the true state are identical. In order to draw more accurate conclusions, we run the same experiment $T = 1000$ times, and we assume only $h = 2$ sensors are under attack. In the following experiments, the measurement data assumes values $\in [-2, -1] \cup [1, 2]$, and we analyze the results computing the system with and without noise, more precisely, the noise-free case assumes $\eta = 0$, while the case with noise assumes $\eta \sim \mathcal{N}(0, \sigma^2)$. We expect the presence of noise to disturb the estimation, leading to an increase number of misses, i.e. the number of times the attack vector wasn't correctly estimated.

2.1 Unaware attacks

In the unaware attack scenario, we assume that the components a_i are uniformely distribuited in $[-2, -1] \cup [1, 2]$, just like the measurement data.
Regarding the noise, both cases yield comparable results. The majority of estimations exhibit a norm close to zero, indicating an almost exact approximation of the state, and the number of misses is respectively 23.2% for the noise-free scenario and 26.3% when the noise is present.

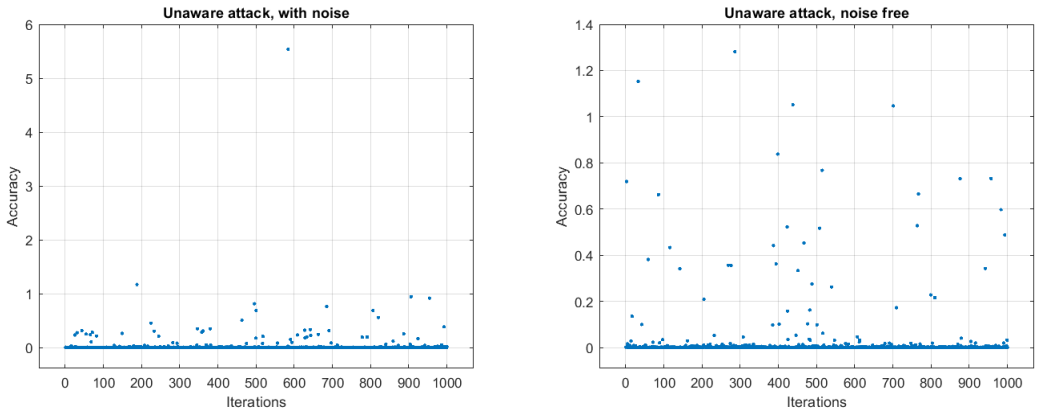


Figure 5: Accuracy$^{(0)}$ in function of the number of iterations, unaware attack

2.2 Aware attacks

In this experiment the attacker is able to access the sensors measurements, and perturbs them of a quantity equal to $a_i = \frac{1}{2}y_i$. The parameters are identical to the previous case.
Considering the numerical resemblance between the perturbation induced by the attack vector and the sensor data, we expected an higher misprediction rate compared to the previous scenario, reaching values sightly above 30%. In fact, if any component of the predicted attack vector is less than or equal to 1.3, its altered value would remain within our permissible range (since $1.3(1 + \frac{1}{2}) < 2$). Consequently, the algorithm would encounter difficulty in identifying the actual corrupted value.
The number of times the attack vector wasn't correctly estimated, is respectively 31.3% for the noise-free scenario and 34.8% when the noise is present.

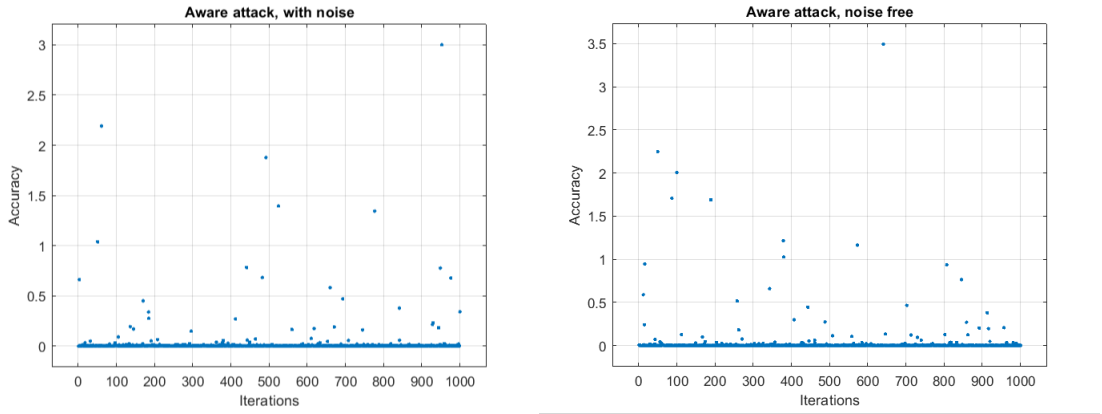


Figure 6: Accuracy[0] in function of the number of iterations, aware attack.

[0]Accuracy is defined as $\|x(t) - \hat{x}(t)\|_2^2$, where $x(t)$ is the real state at time t, and $\hat{x}(t)$ is the estimated state at time t. A smaller squared distance indicates a better estimation as it reflects a closer proximity to the real value.

3 Target Localization with Real Data

The target localization problem consists in estimating the position of one or more targets on a given indoor environment. Such problem can be intuitively solved by using a Nearest Neighbor approach, or by setting a LASSO problem. The former becomes NP-hard when trying to estimate more than one target, hence, since the Received Signal Strength (RSS) is additive, it's suggested to relax the problem and to reformulate it as the latter. During the *training phase*, a matrix dictionary D is built to store the RSS fingerprints of a target moving through all the cells as its columns, while each row houses the sensor measurements for each cell. In the *runtime phase*, each sensor makes an estimate of the target's location and sends its own measurement to the *fusion center*. The data is then processed by comparing the measurement vector y with the columns of the dictionary D, with the closest match being selected. This procedure of retrieving the target position is accomplished by solving a LASSO problem, as seen in Task 1. In order to achieve accurate results it is necessary to normalize the data contained in the dictionary D, as shown in Eq. 5

$$\begin{pmatrix} 0 \\ 0 \\ 0.2604 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \Rightarrow \begin{pmatrix} 1.3497 \\ 0.8801 \\ 0 \\ 0 \\ 0.7710 \\ 0.5104 \\ 9.0544 \end{pmatrix} \tag{Eq. 5}$$

The right vector contains the results obtained without normalizing the dictionary, whereas the left vector corresponds to the data obtained using the normalized signature map. It is important to note that the correct result is actually the one on the left, since the target is located in the seventh cell.

3.1 Target Localization under Sensors Attacks

By using the RSS fingeprints it is also possible to estimate the position of the target under sensors attacks. We considered a single aware attack on the i-th sensor, where the attacker perturbs the sensor by a quantity equal to $a_i = \frac{1}{5}y_i$, each time tampering a different one, in order to evaluate the effect produced by each one of them. The obtained results obtained are Eq. 6

$$\begin{pmatrix} 0 \\ 1.3526 \\ 0 \\ 0 \\ 0 \\ 0 \\ 8.5006 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 8.7072 \end{pmatrix} \begin{pmatrix} 0 \\ 1.1158 \\ 0 \\ 0 \\ 0 \\ 0 \\ 5.5698 \end{pmatrix} \begin{pmatrix} 0 \\ 1.3886 \\ 0 \\ 0 \\ 0 \\ 0 \\ 3.6118 \end{pmatrix} \begin{pmatrix} 0 \\ 1.8790 \\ 0 \\ 0.2914 \\ 0 \\ 0 \\ 8.7293 \end{pmatrix} \begin{pmatrix} 0 \\ 1.2479 \\ 0 \\ 0 \\ 0 \\ 0 \\ 8.7374 \end{pmatrix} \tag{Eq. 6}$$

and show that the target is estimated to be in the seventh position in the grid, independently from which sensor was attacked. This is evinced by noticing that the highest value in the vector is always in the seventh position.

4 Sparse Observer

Until now, we dealt with *static* problems described in the Eq. 4. Let us now consider the scenario in which the A matrix differs from the identity matrix, meaning that the target reaches a different position at each time instant. In order to solve the dynamic version of the localization problem there are two possibilities: either following a *static* approach or a *dynamic* one. The first aims at recovering the state of the wireless sensor network by waiting T time instants and only after that computing $x(0)$ through the (pseudo)inversion of the observability matrix $\mathcal{O}_T$, with the assumption that the rank$(\mathcal{O}_T) = n$, where n is the dimension of the state vector. Although this method requires minimal computational effort, it is not efficient since we have to wait a certain time interval before being able to estimate $x(0)$ and then compute recursively the position of the target at any k. Conversely, adopting a dynamic approach enables us to obtain an estimate of the vector right from the start, at the expense of a lower initial accuracy since the initial state is a random guess, and higher computational costs caused from utilizing an online algorithm. In particular the sparse observer algorithm merges *a middle step* of the IST:

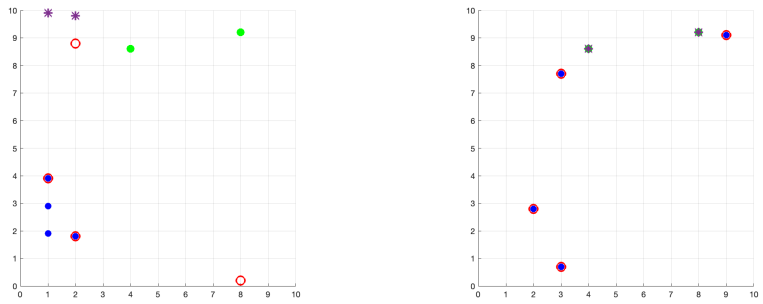
$$\hat{z}(k+\frac{1}{2}) = \mathbb{S}_{\lambda\tau}\left[\hat{z}(k) + \tau G^T(y(k) - G\hat{z}(k))\right] \qquad \text{(Eq. 7)}$$

with another based on the knowledge of the system equations:

$$\begin{cases} \hat{x}(k+1) & = A\hat{x}(k+\frac{1}{2}) \\ \hat{a}(k+1) & = \hat{a}(k+\frac{1}{2}) \end{cases} \qquad \text{(Eq. 8)}$$

4.1 Unaware time-invariant attack

In this first case, we considered each attack with a value of $a_i = 30$. The initial scenario and the one after T time steps are reported below.



(a) First iterations (b) Setting after T time steps

Figure 7: Blue dots are the estimated postions, red circles the real ones, green dots are the sensors under attack and stars are the estimated attacks.

As it concerns the performance of the aforementioned algorithm, it has been observed that a time interval of $T = 50$ covers its transient state quite comprehensively, after which the results become stable and accurate. By selecting $T = 100$, we were able to better capture the different phases of the procedure. As shown in Figure 8, it is evident that at a certain point, the predictions stabilize. At the end of the simulation time, all estimations were correct, as demonstrated in Figure 7, where the predictions and real data markers overlap.

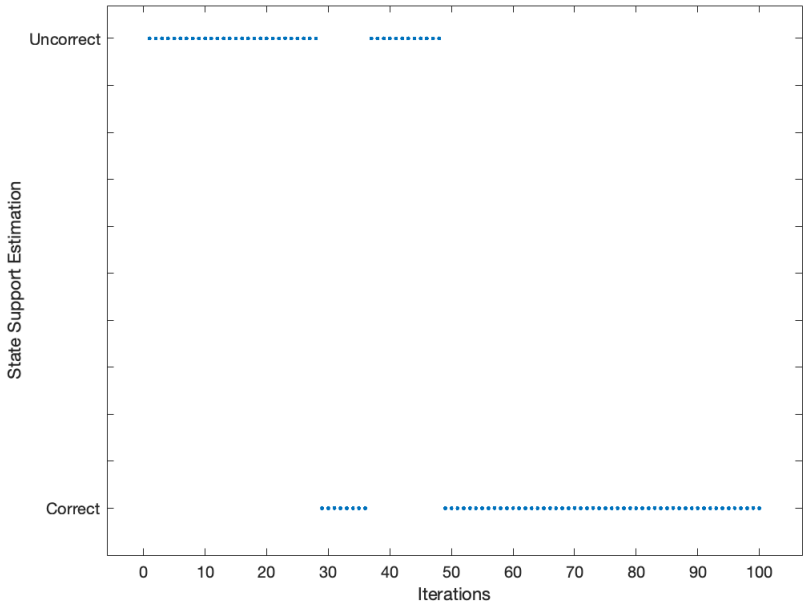


Figure 8: Change in prediction correctness

Indices in the support of the initial x and of a are chosen from the uniform discrete distribution. Along with the simulations performed, it has been noticed that selecting particular (not random) indices the transient interval of the algorithm needed to reach all correct estimations fell rapidly to a few steps as shown here:

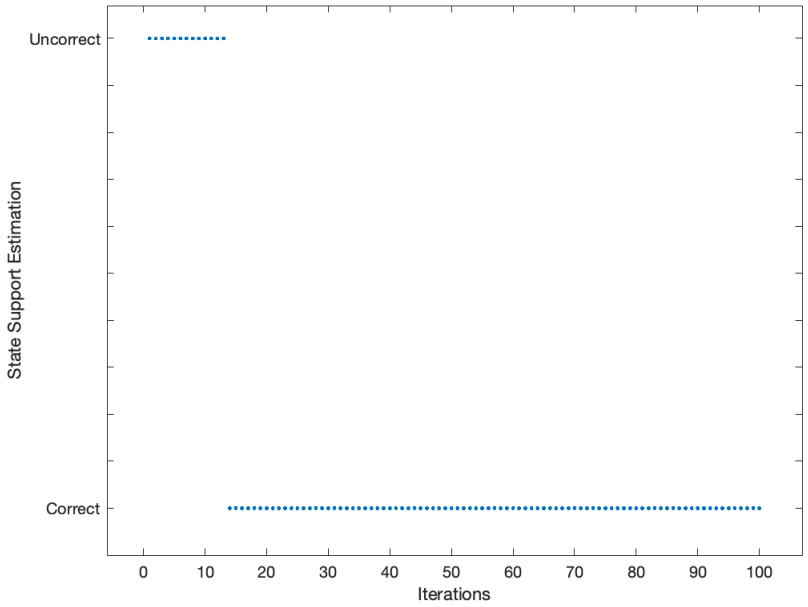
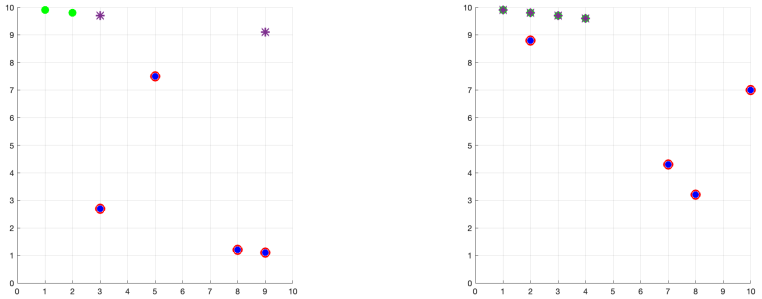


Figure 9: The support estimation in function of iterations number

In this case, we set the support of x as follows $\mathrm{S}_x = \{10,\ 20,\ 30,\ 40\}$

4.2 Aware time-varying attack

Now let us consider the aware-attack case the attacker perturbs the measurement sensor with a quantity equal to $a_i = \frac{1}{2}y_i$. The test has been performed using the same parameters of the previous case, but, while the position of the targets continues to be well estimated, the algorithm fails estimating the correct sensors attacked. This problem can be simply solved changing the entries in the vector Λ which manages the shrinking strength and to what extent the vector $\begin{bmatrix} x & a \end{bmatrix}^T$ has to be sparsified. Thus, setting the attack vector components to large values such as in their hundreds, allowed us to estimate the correct attacks even earlier than in the unaware attack case. Results are presented below.



(a) Using $\Lambda = \begin{bmatrix} 10 & 10 & ... & 20 & 20 \end{bmatrix}$ (b) Using $\Lambda = \begin{bmatrix} 10 & 10 & ... & 200 & 200 \end{bmatrix}$

Figure 10: Difference in performance of the algorithm with respect to Λ.

5 Distribuited Estimation

Distributed estimation entails the absence of a *fusion center* that can process all sensor measurements and provide a unique state estimation. Instead, individual sensors must work collaboratively to achieve a consensus on the system state. In other words, all the sensors have to compute a similar estimate by influencing one another. This requires sharing information, but sharing raw measurement data such as y_i (i-th sensor measurement) or C_i (how the i-th sensor manipulates the state to achieve the output) is not feasible due to privacy concerns and limited storage resources on the sensor nodes. As an alternative, each sensor computes its own local estimate $x^{(i)} \in \mathbb{R}^n$ of the system state $\tilde{x}$ and shares that with the other sensors. The goal of this task is to estimate the state and the support of the attack vector.
Estimating the attack vector support produces similar results on all topologies, with the third performing sightly better than the others.

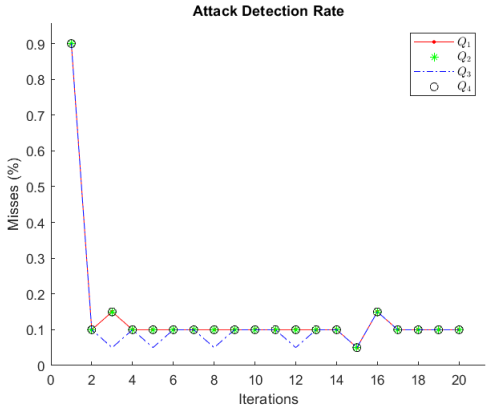


Figure 11: Number of misses while estimating the attack vector support, considering all topologies

Estimating the system's state
the third topology performed the worst out of all fours, while the rest behaved in a similar way.

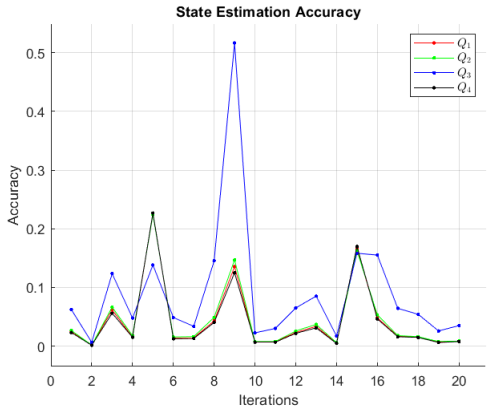


Figure 12: State estimation accuracy$^{(0)}$ in function of the number of iterations

The provided topologies are capable of solving the consensus problem, as they all satisfy the sufficient condition for resolving it. Infact, they are described by are doubly stochastic matrices, whose leading eigenvalues are unique and equal to one. The topology that yields the fastest convergence is the fourth one, described by the matrix Q_4, since its squared essential spectral radius is the smallest out of four. The speed of convergence has a decreasing exponential mode – the lower the base, the fastest the convergence to zero, as shown in figure 13.

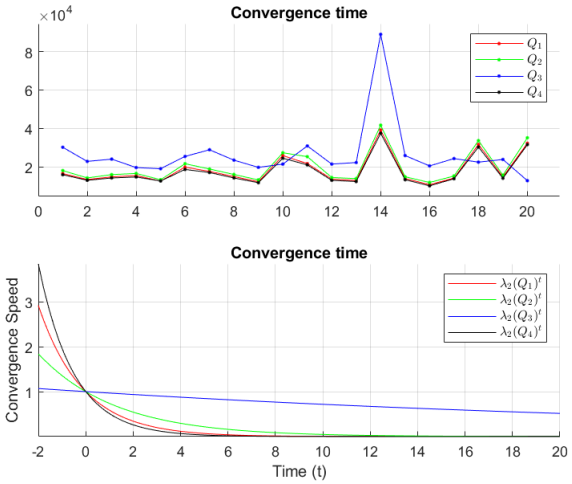


Figure 13: Convergence time in relation to as seen experimentally (up) and theoretically (down)