



**Politecnico
di Torino**

Politecnico di Torino

Modeling and Control of Cyber- Physical Systems

Project I

Sofia Longo 310183
Matteo Gravagnone 319634
Gabriele Martina 310789
Danilo Guglielmi 318083

Task 1

The purpose of this task is to solve the IST (Iterative Soft-Thresholding) algorithm for reconstructing sparse signals from incomplete and noisy measurements, solving a synthetic Lasso problem.

A random projection matrix $\mathbf{C} \in \mathbb{R}^{q,p}$ and a k -sparse signal $\mathbf{x} \in \mathbb{R}^p$ are generated.

\mathbf{x} is k -sparse meaning it has k non-zero values at random positions and with values ranging between $-b$ and $-a$ and between a and b , where a and b are two constants specified by the requirement.

Then, we simulate the behaviour of a system of q sensors that collect all the measurements of our objective state vector. For a realistic representation of the inaccuracy of the problem we corrupt the measurements with Gaussian noise (deviation = 10^{-2}).

The IST algorithm is then applied to reconstruct the original signal from the incomplete and noisy measurements.

The following code shows our implementation of the IST algorithm along with our implementation of the shrinkage/thresholding operator (below).

Algorithm 1 IST

- 1: Initialization: $x_0 = 0 \in \mathbb{R}^p$
- 2: **for all** $t = 1, \dots, T_{max}$ **do**
- 3: $x_{t+1} = \mathbb{S}_{\tau\Lambda} [x_t + \tau C^\top (y - Cx_t)]$
- 4: **end for**

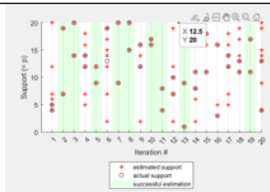
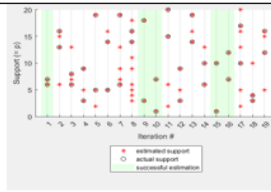
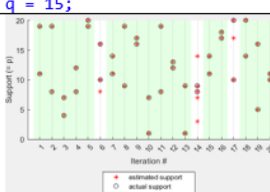

```
x_t = zeros(p, 1); % current estimation
while true
    grad = (C')*(y-C*x_t);
    x_t_next = shrinkage(x_t + tau*grad,
    tau*LAMBDA);

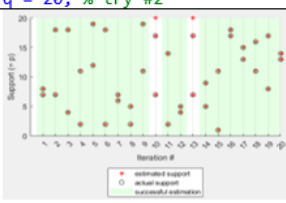
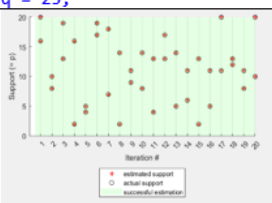
    count = count + 1;
    if norm(x_t_next - x_t, 2) < delta % Tmax
        break;
    end
    x_t = x_t_next;
end
```

A loop of iterations is executed until the change between two consecutive estimates of the signal x becomes lower than a tolerance threshold δ .

Stop criterion: $T = \text{first time instant s.t. } \|x_{T+1} - x_T\|_2 < \delta, \delta = 10^{-12}$.

We are going to check the algorithm's performance as we change some parameters and repeat over 20 iterations.

<p>lambda = 1/(100*tau);</p> <p>% These parameters will stay the same</p> <p>p = 20;</p> <p>iter = 20;</p> <p>tau = 0.012;</p>	<p>q = 10; % try #1</p>  <p>Success = 45%</p> <p>Iterations needed – Min / Max / Avg</p> <p>193 / 2833 / 1081.5</p>	<p>q = 10; % try #2</p>  <p>Success = 25%</p> <p>Iterations needed – Min / Max / Avg</p> <p>203 / 9141 / 1207.75</p>
	<p>q = 15;</p>  <p>Success = 85%</p> <p>Iterations needed – Min / Max / Avg</p> <p>107 / 641 / 222.3</p>	<p>q = 20; % try #1</p>  <p>Success = 100%</p> <p>Iterations needed – Min / Max / Avg</p> <p>101 / 213 / 141.6</p>

	$q = 20; \% \text{ try \#2}$  <p>Success = 90% Iterations needed – Min / Max / Avg 68 / 272 / 131.3</p>	$q = 25;$  <p>Success = 100% Iterations needed – Min / Max / Avg 76 / 198 / 115.15</p>
--	---	---

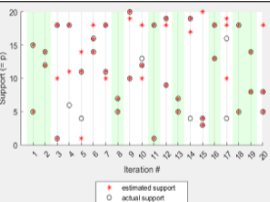
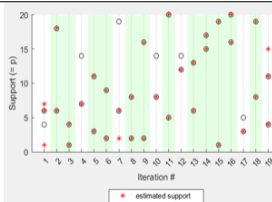
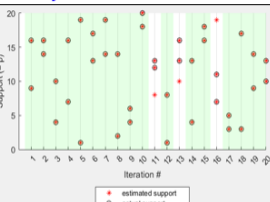
From the initial data ($p=20$, $q=10$) our success rate hovers around 40/50%.

The measurements show that by increasing q up to obtaining a square matrix ($q=20$, $p=20$), the success rate gradually increases until it reaches an average of 95%. Additionally, the number of iterations needed to reach a final estimate generally decreases as the number of sensors increases along with the number of rows of matrix C .

It is noteworthy that the maximum number of iterations needed to compute the estimate goes from reaching over 10.000 in the worst case, to remaining below 300 when $q = 20$. The same applies to the minimum and mean of number of iterations.

In the case of a TALL matrix, where $q > p$ and the system is overdetermined, the success rate is almost always above 95% and can even reach 100%. In this case, the values related to the number of iterations do not vary drastically compared to the case of the square matrix.

Increasing the number of rows of the projection matrix leads to a higher success rate in reconstructing sparse signals from incomplete and noisy measurements, as the experiments show. Additionally, the total number of required iterations decreases with increasing matrix size, for both square and overdetermined matrices.

$\lambda = 1/(10 \cdot \tau);$ % These parameters will stay the same $p = 20;$ $\text{iter} = 20;$ $\tau = 0.012;$	$q = 10; \% \text{ try \#1}$  <p>Success = 40% Iterations needed – Min / Max / Avg 127 / 581 / 273.1</p>	$q = 15;$  <p>Success = 60% Iterations needed – Min / Max / Avg 77 / 369 / 170.65</p>
	$q = 20;$  <p>Success = 85% Iterations needed – Min / Max / Avg 82 / 258 / 154.9</p>	

Increasing the regularization parameter λ , proportional to the parameter τ , leads to a decrease in the success rate of the IST algorithm for reconstructing sparse signals. For instance, when $q = p$, the success rate decreases from 90% to 85%.

By increasing λ the predicted capacity of the model starts worsening causing underfitting of the model to the data, this means that the shrinking thresholding operator doesn't "cut" enough and doesn't recognize which contributions are effectively useful for the prediction.

'tau' is called "learning rate" [1] and it affects the convergence time, in particular, it influences to what extend newly acquired information overrides old information. The larger tau is the more we take into account the newly computed gradient in the evolution of our estimate. However, a too high learning rate will make the estimate jump over minima, but a too low learning rate will either take too long to converge or get stuck in an undesirable local minimum.

This trend is reflected in our experiments.

[1] https://en.wikipedia.org/wiki/Learning_rate

Task 2

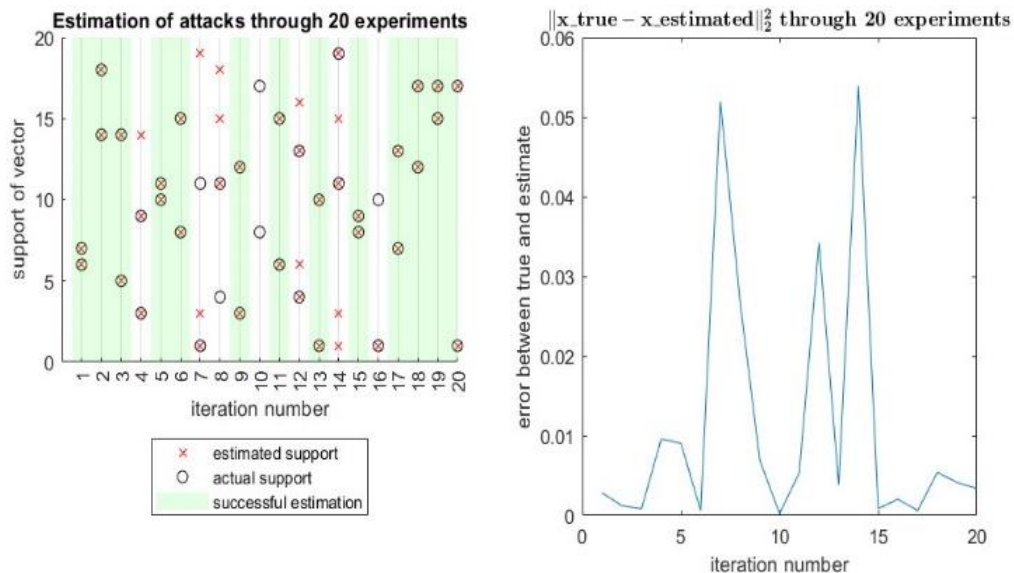
Starting from Task 1, the IST algorithm is reformulated to reconstruct a non-sparse signal and either an "aware" or an "unaware" sparse attack.

A cycle of 20 experiments is performed to evaluate the success rate and the L2 norm of the reconstruction error.

In the following pictures, the two different types of attacks are represented: on the left, there is the estimation of attacks during 20 experiments; on the right, the error between the true value of the signal and the estimate one.

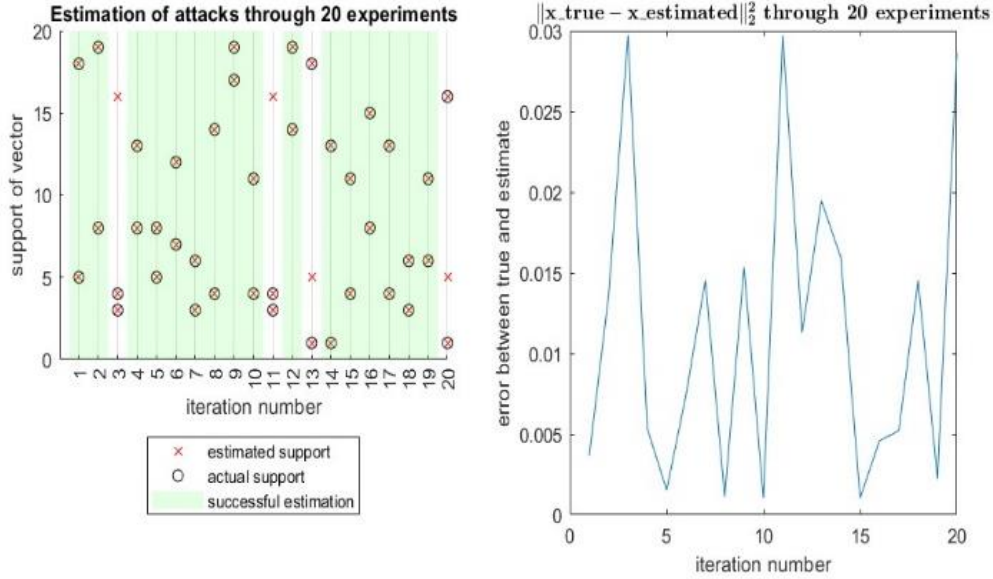
Aware Attack

Success Rate: 65% (range 46%~68%)



Unaware Attack

Success Rate: 80% (range 65%~95%)



The reason for the difference in success rates between the two types of attacks can be attributed to the fact that an unaware attack is uniformly distributed in a certain domain, while the aware attack depends on our sensors' measurements.

We have noticed that the accuracy of the estimate of the attack is strictly linked to the proportion between the magnitude of the attack and the magnitude of that sensor measurement. In particular, the greater the magnitude of the attack is with the measurement, the more "detectable" the attack appears to be.

Given that, in the context of unaware attacks, the extent of the attack ranges in absolute value between [1, 2], therefore its contribute to the algorithm is always significant, which can identify it and estimate it in most cases. Here, our medium success rate hovers around 80%.

On the other hand, when simulating the aware attack, the perturbation is dependent on the measurement itself; in particular, it corrupts the measurement by a half. Therefore, when our measurement is too small the attack becomes too low to be noticed at all. This makes it harder for the IST algorithm to separate the attack from the original signal and reconstruct the original signal accuracy. Here, instead, our medium success rate is approximately 65%: significantly lower.

Indeed, when we generate x such that it lies in a domain larger by at least an order of magnitude, we clearly notice that the trend reverses: aware attacks become more easily detectable whereas unaware attacks noticeability decreases.

The L2 norm of the reconstruction error was also higher in the case of the "aware attack" with a maximum of 0.055; instead, in the case of the "unaware attacks", the maximum value of the difference between the true value of the signal and the estimated one is 0.03. These results indicate that the IST algorithm is less effective when dealing with an aware attack compared to an unaware attack.

Task 3

The aim of this experiment is to use the RSS-fingerprint method for indoor localization to determine the location of the only attack and that of the targets.

Starting from the Lasso problem of task 2, we have perturbed the measurements each time on a different sensor and the perturbation is always set at $1/5 \cdot y_i$.

The situation can be described as follows:

-> $q = 6$ sensors, $p = 7$ cells

-> a matrix D ($q \times p$) that represents the fingerprint map (sensor measurements stored in each cell)

-> the measurement vector y .

When the stop condition is satisfied, the algorithm ends and shows results: w_t is a vector of $p+q$ elements that shows, in the first p positions, which cells contain the target sensors and, in the last q positions, which sensor is attacked. Below, the tables show which sensor was perturbed in each experiment and the targets' locations estimated by the algorithm. The color is increasingly green as the estimate become larger and more certain.

The tables bring to the eye the fact that the number of identified targets is not always the same. Also, even though we know for sure that the sensor under attack is only one, the algorithm seems to bring out more than one attack per experiment.

However, by observing the estimate of the targets, we notice that the highest absolute value is always associated with the seventh cell. This suggests that the other non-zero values obtained are very likely false positives. This is probably due to the lack of data (6 sensors for 7 cells are surely insufficient) or to some parameters that require tuning.

The same behavior occurs for the estimate of the attack. In this case, there are multiple non-zero values, but only one of them stands out among the others and it is always in the correct attacked position.

The lack of information, the lack of data, and the choice of non-optimal parameters not only result in false positives but often lead to completely erroneous measurements and in some cases, such as #3 and #4, the sensor under attack is not identified at all.

FIRST EXPERIMENT

Attacked sensor	Estimated attacks on sensor number	Estimated attacks	Target measurment
1	1	-5.2231	0
	2	0	1.3526
	3	0.5315	0
	4	0	0
	5	0	0
	6	0	0
			8.5006

SECOND EXPERIMENT

Attacked sensor	Estimated attacks on sensor number	Estimated attacks	Target measurment
	1	0	0
2	2	-3.6393	0
	3	1.1628	0
	4	0	0
	5	-0.5836	0
	6	0	0
			8.5006

THIRD EXPERIMENT

Attacked sensor	Estimated attacks on sensor number	Estimated attacks	Target measurment
	1	-0.0144	0
	2	0.4116	1.1158
3	3	0	0
	4	2.1701	0
	5	0	0
	6	0	0
			5.56986

FOURTH EXPERIMENT

Attacked sensor	Estimated attacks on sensor number	Estimated attacks	Target measurment
	1	-0.1	0
	2	0.4057	1.3526
	3	4.6856	0
4	4	0	0
	5	0	0
	6	0	0
			8.5006

FIFTH EXPERIMENT

Attacked sensor	Estimated attacks on sensor number	Estimated attacks	Target measurment
	1	0	0
	2	0	1.8790
	3	0.2338	0
	4	0	0.2914
5	5	-4.5564	0
	6	0	0
			8.7293

SIXTH EXPERIMENT

Attacked sensor	Estimated attacks on sensor number	Estimated attacks	Target measurment
	1	0	0
	2	0.1152	1.2479
	3	0.5397	0
	4	0	0
	5	0	0
6	6	-4.8199	0
			8.7374

Task 4

Task 4 is focused on a localization problem in which four targets are known to be moving in a square room partitioned in a-hundred cells. Twenty-five sensors are deployed in random spots within the room and they had been trained separately, resulting in a dictionary D of RSS fingerprints considered uncorrupted.

Two random sensors are targeted each time the experiment is performed. The type of attack can be chosen at the beginning of the Matlab file [line 39] simply specifying the variable *attack_type* to be either "AWARE" or "UNAWARE".

- Unaware attacks perturb the sensors' measurements by a magnitude of 30;
- Aware attack implies a perturbation 0.5-proportional to the sensors' measurements. Given the nature of this kind of attack it must be updated at every iteration: every time the targets move, the measurements do it as well, consequently.

Below, the core of our Matlab implementation of the Sparse Observer algorithm.

Algorithm 2 Sparse observer

```

1: for all  $k = 1, \dots, T$  do
2:    $\hat{z}(k + \frac{1}{2}) = \mathbb{S}_{\tau\Lambda} [\hat{z}(k) + \tau G^T(y(k) - G\hat{z}(k))]$ 
3:    $\hat{x}(k + 1) = A\hat{x}(k + \frac{1}{2})$ 
4:    $\hat{a}(k + 1) = \hat{a}(k + \frac{1}{2})$ 
5: end for

```

% Algorithm of sparse observer

```

grad = G'*(y - G*z);
z_half_next = shrinkage(z + tau*grad, tau*lambda);
x_next = A*z_half_next(1:p, 1);
a_next = z_half_next(p+1:end, 1);
z = [x_next; a_next];

```

For this task we are now introducing another function that we developed: *filter_attacks(x, magnitude)*. This function is used alongside *arrayfun* from the Matlab library in order to accomplish the final refinements for the attack estimate.

- x stands for the single component
- $magnitude$ is the desired threshold

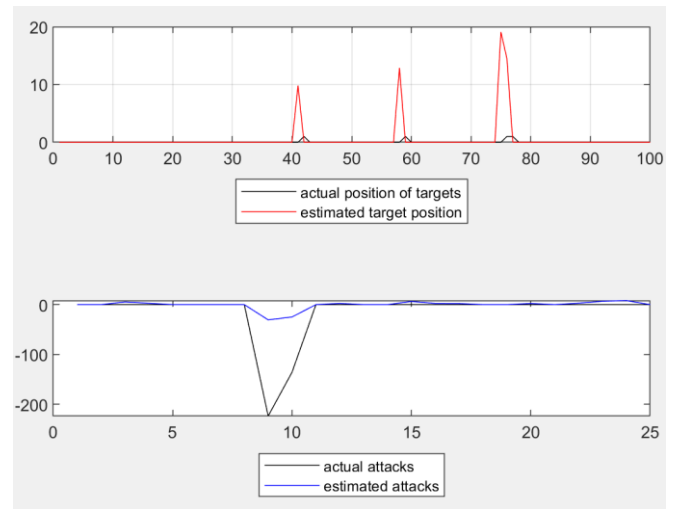
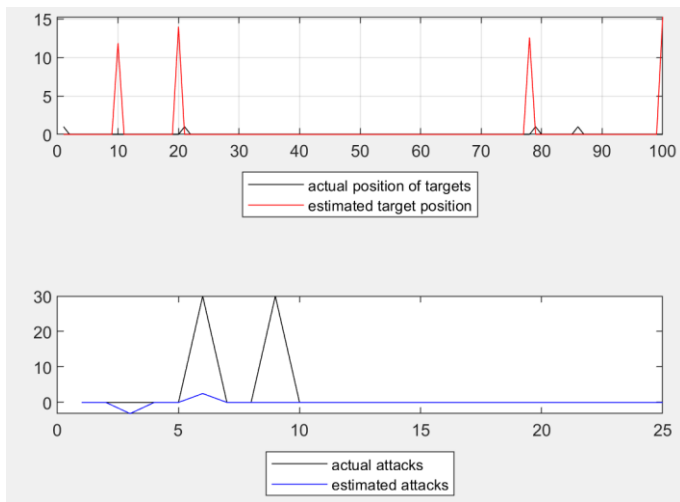
```

function y = filter_attacks(x,magnitude)
    if abs(x) < magnitude
        y = 0;
    else
        y = x;
    end
end

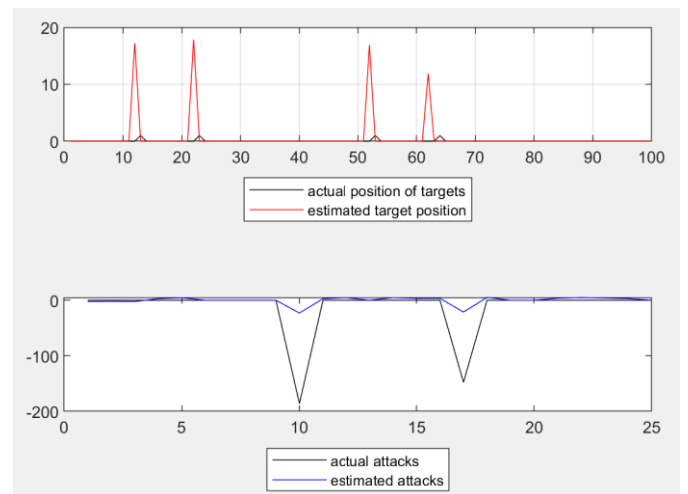
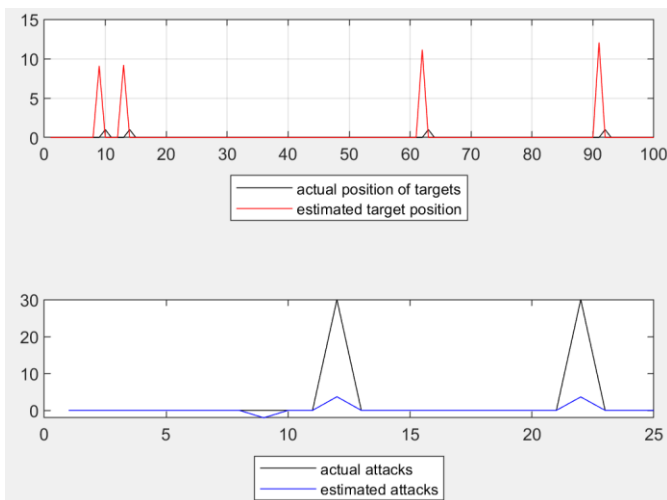
```

To explore the effect of both types of attacks, experiments were conducted, aiming to uncover key observations: the results of these experiments provide further evidence that corroborates the patterns identified in the previous tasks. Two groups of plots are displayed below: the outcome of the estimation process when unaware attacks are employed compared to the outcome when using aware attacks. The plots reveal that the targets estimate is predominantly precise, but the attack is better recognized in experiments that employed the aware variant.

Aware attacks



Unaware attacks



This could be addressed, accordingly to previous finds, to the comparison of the magnitudes of the attacks against that of the sensor measurements. Since the latters have an order of magnitude of 10^2 , the effect of aware attacks is more pronounced and recognizable compared to the case of a fixed attack of 30. Thanks to the consistent substitution of the sensor measurements provided by the aware attack, the reconstruction algorithm may be able to more easily identify and correct this disturbance through time. On the contrary, the unaware attack gets blended and is more keen on disappearing among the measurements, therefore it is more difficult to isolate and correct.

Task 5

The goal of this project consists in:

- ➔ analysing the performance of a distributed estimation system under attacks, with a particular focus on the detection of unaware attacks,
- ➔ determine the ability of the system to detect and mitigate the impact of the attack by refining the sensor estimates.

We simulate a **distributed** estimation system consisting of 10 sensors that only have visibility on their own measurement and store only their own RSS fingerprints. We assume that the attacker is injecting a random set of malicious measurements into the sensor readings, representing the unaware attack. In this scenario the sensors must estimate the target vector and recognize the possible attack.

To detect and mitigate the impact of the attack we apply a distributed estimation algorithm that uses a stochastic matrix to model the sensor network and iteratively computes the estimate of the unknown target vector. We run several experiments with different levels of attack magnitudes and refine the attacks using “filter_attacks.m”.

Analyzing the eigenvalues of stochastic matrices Q1, Q2, Q3 and Q4 to check which ones solve the consensus problem, we can assert that every matrix has exactly one eigenvalue equal to 1 and all the others smaller in magnitude. Therefore, all the matrices allow us to achieve average consensus.

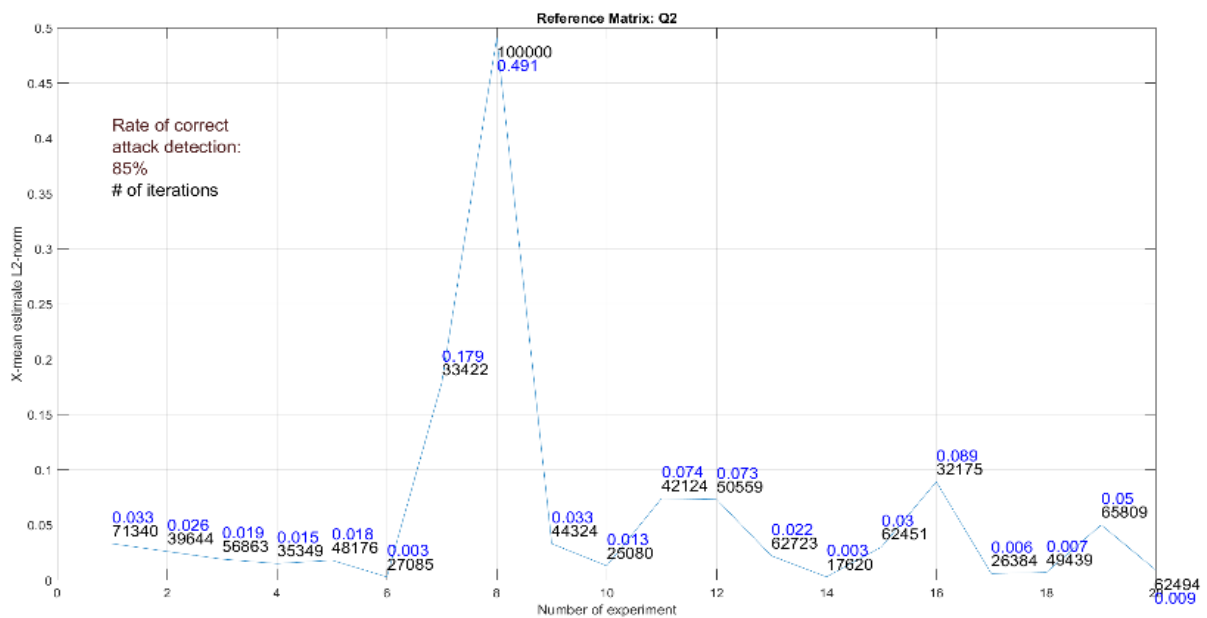
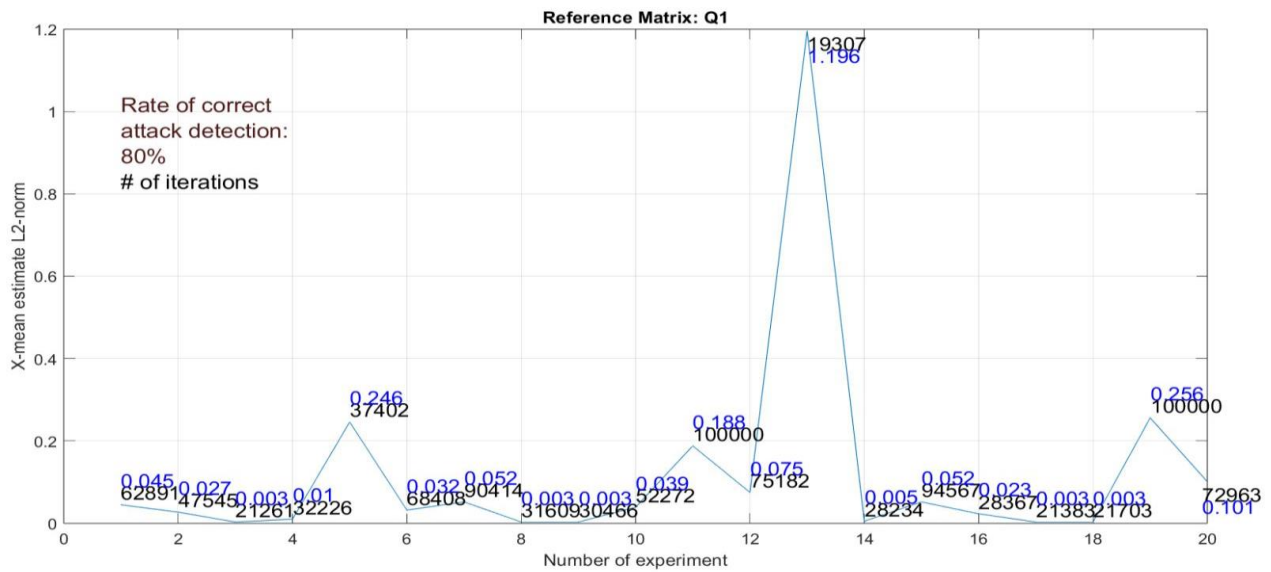
	Essential spectral radius	Medium number of iterations for convergence
Q1	0.5857	51810
Q2	0.7373	47653
Q3	0.9674	53814
Q4	0.5103	46496

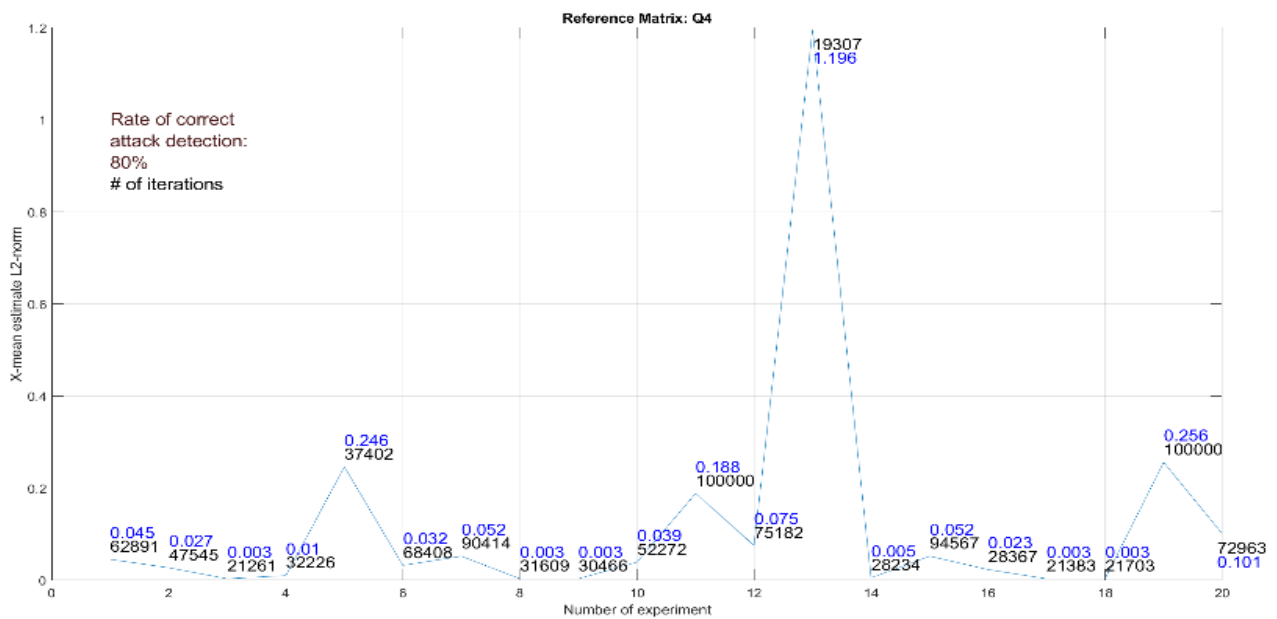
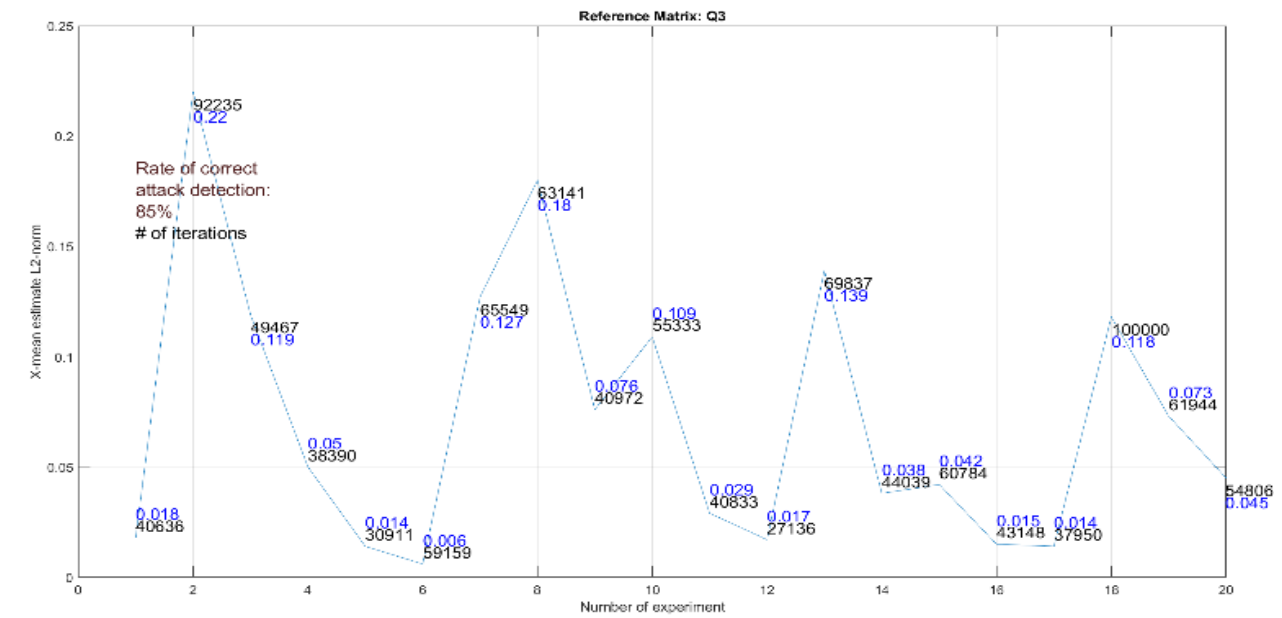
Moreover, investigating the essential spectral radius (*esr*) of each stochastic matrix (table above) we would expect to observe that its magnitude is inversely proportional to the number of iterations needed for convergence. The greater the *esr* is, the faster the algorithm should reach consensus. In this case, instead, we cannot clearly notice a relation between the two values because we may not be considering other mathematical insights that might be influencing the convergence time.

However, a significant aspect to consider is the eigenvalues spectrum: the average distribution of all the eigenvalues of the matrix can suggest how fast the convergence could be reached. In fact, when the distribution is tightly clustered around a 0, the speed of convergence increases.

We can notice that in case of matrix Q4, even if the essential spectral radius is the smallest, several eigenvalues are close to 0 and the velocity of convergence is indeed the highest.

In the following graphs, we depict for each matrix the rate of correct attack detection (red comment), the square of the L2-norm of the difference between true vector x and its estimate (blue labels) after reaching consensus between the sensor and the number of iterations needed to break out of the algorithm (black labels).





The rate of correct attack detection is high for any matrices we employ (always greater than 80% and up to 90%). Therefore, the algorithm can successfully identify which sensors are corrupted.

Furthermore, except some isolated cases, the estimation of the state is quite accurate: indeed, the estimation error is sufficiently low as the plots suggest (in the order of magnitude of $10^{-1} \sim 10^{-2}$).

