Politecnico di Torino

Modeling and control of cyber-physical systems
Project I

Botta Giulia (s310478) – Di Fazio Angela (s312641) – Maggipinto Luigi (s319874)

TASK 1

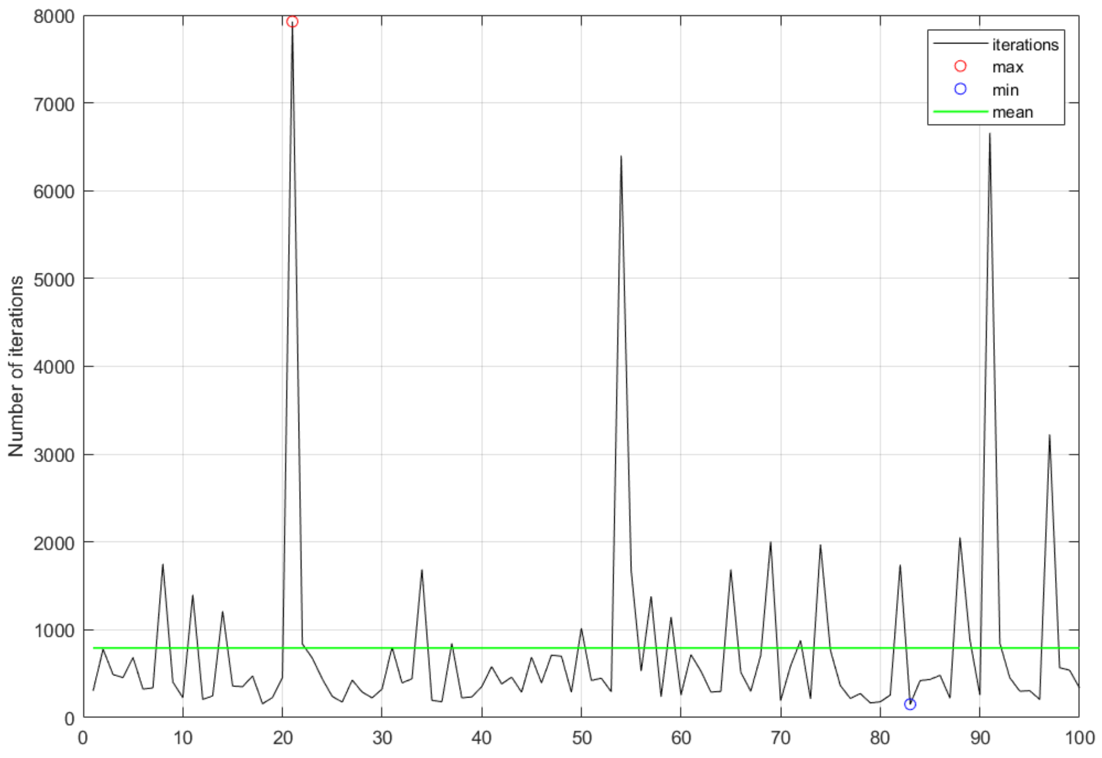
The aim of the first task is to solve a Lasso problem by implementing the iterative shrinkage/thresholding algorithm (IST).

The experiment has been repeated more than 20 times, in order to better analyze the results:

1) The support of x is correctly estimated 18 times on 25 and we reached a great approximation of x, with a 70% of accuracy.

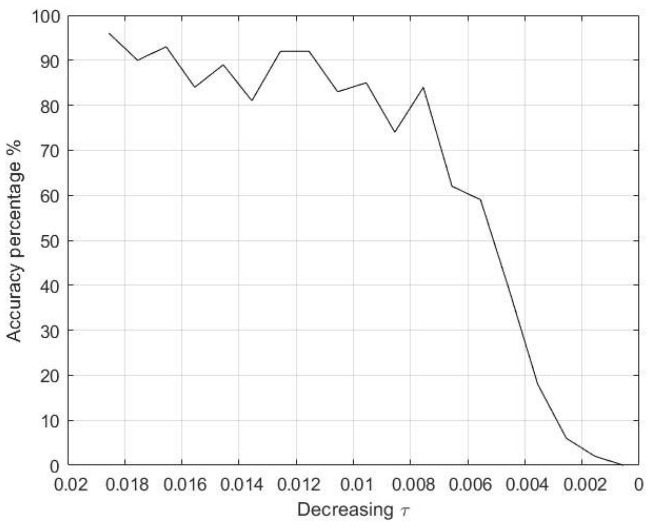
2) We notice that, increasing the parameter q, it is possible to increase and maximize the support recovery, till obtaining the 100% success. At this purpose, we modify our MATLAB code, including a cycle to increase the value of q (from 10 to 25). Since the first iterations with a bigger value of q, we notice a more accurate performance, e.g. 86% with q=10. The support recovery will increase till to q=15/16 (according to the experimental values) that allows to reach the 100% success. From this value of q on, the percentage is 100%.

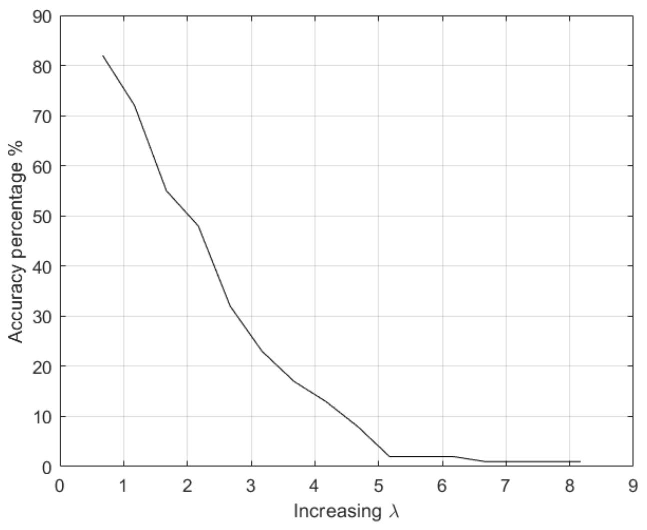
3) In general, the iterations vary a lot according to the random matrix that is generated. The max value that we obtained is 7926, the minimum value 152 and the mean is 793,2.



4) According to this point, we include a cycle decreasing τ and we aimed to observe what would happen to our estimate. For example:
    a. Beginning with τ = 0.0177 and an estimate equal to 80-90%, we found that with final τ = 0.0027 our estimate percentage was below 10%;
    b. With τ = 0.0184 we obtained the same result as before, with final τ = 0.0034.

   So, we can conclude that, by keeping a fixed lambda and decreasing τ, the value to be compared to the two thresholds becomes smaller and, from a certain point on, it will be too small and will be shrunk to zero so the estimation will be less accurate than before.





5) We noticed that, increasing lambda by keeping τ constant, the window between the two threshold of the algorithm becomes bigger so there will be more values shrunk to zero and the estimation will be less accurate than before. For example:
    a. Beginning with λ = 0.5389 and an estimate equal to 80-90%, we found that with final λ =5.0389 our estimate percentage was below 10%;
    b. With λ = 0.4051 we obtained the same result as before, with final λ = 3.4051.

TASK 2

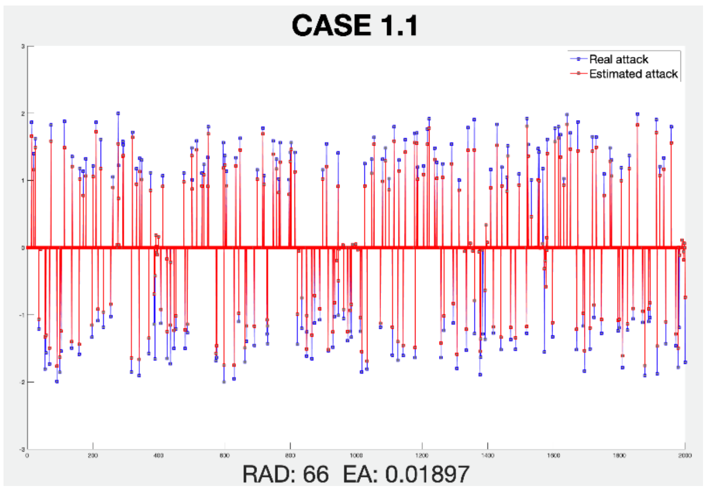
The goal of this task is to estimate the x under sparse sensor attack a, by implementing the same algorithm used in the previous task (IST), in which there weren't the attacks. Here we are considering h=2 attacks; because of the much higher number of sensors (q=20), we can apply the algorithm also to the attacks.

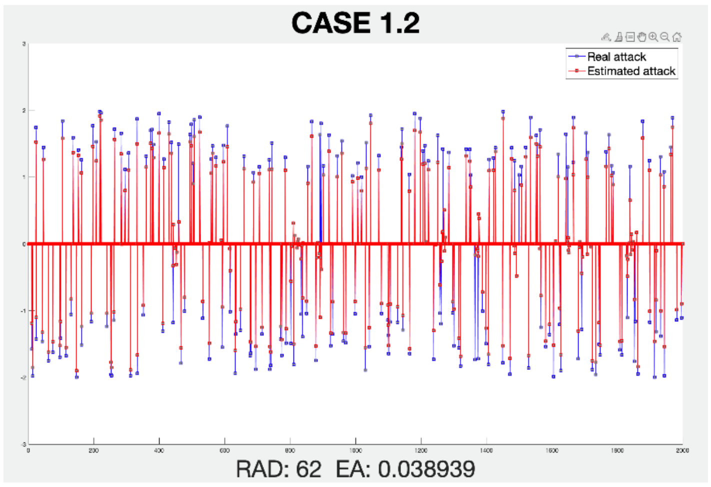
In order to complete this task and analyze the results, we have to distinguish 4 different cases:

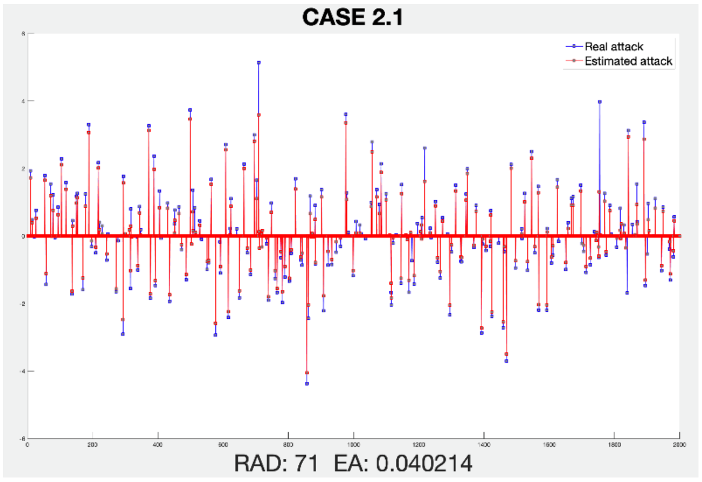
1. The attacks are "Unaware"
    1.1. The noise eta is equal to 0
    1.2. The noise eta is normally distributed
2. The attacks are "Aware"
    2.1. The noise eta is equal to 0
    2.2. The noise eta is normally distributed

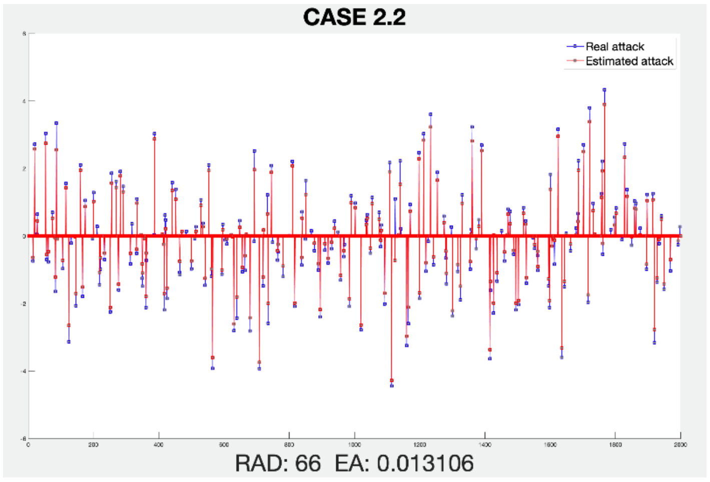
After repeating the experiment for 100 runs, we have to check the "Rate of attack detection" and the "Estimation accuracy": the first indicates the number of times where the estimated attacks are nearly similar to the real attacks, with an uncertainty of 0.3 (chosen by us); the second aims at defining the accuracy of the state x, by the computation of the L2 norm with the x estimated.
Regarding the MATLAB code, we perform the *Unaware attack* choosing the attacks randomly in the required range, and we perform the *Aware attack*, after choosing randomly the support of h values of y and multiplying these values for ½. In this way we obtain the h required attacks.

The following graphs represent the 4 different cases:









These graphs depict the Real attacks (using blue lines and squares) and the Estimated ones (using red lines and squares). Comparing these results, we can easily notice that the RAD of Case 1.1 is higher than the one of Case 1.2, the same can be said if we compare Case 2.1 to Case 2.2. This happens due to the presence of the variable eta that leads the attack detection to be less accurate. In addition we can say that the *Aware attacks* are recovered more precisely than the *Unaware ones*, in terms of RAD.

Eventually, concerning EA's results, we can state that they are all similar and all low (in the order of 10^-2), highlighting that the estimation of x is quite accurate with or without attacks.

TASK 3

In this task we are given the matrix D (6 rows and 7 columns) that is composed as the measurements of a target in each of the 7 cells taken by the 6 sensors present; the vector y (dimension: 6x1 = one data for each sensor) which is the measurement of the sensors for the targets to be estimated. Our aim is to estimate the position of the present targets using the IST algorithm: vector x of dimension equal to 1x7 = one data for each cell.

1) Without attacks: the algorithm gives as estimated position of the target the seventh cell. In the first case with the data taken as they are, the estimate of the cell is made considering the largest component of the vector x, in the second scenario instead the vector x clearly shows that the target is present in the seventh cell.

MATRIX	**LAMBDA**	**ATTACK**	**X**							**CELL**
D not normalized	10^{-3}	no	0	0.1425	0.1102	0	0	0	0.7237	7th
D normalized	1	no	0	0	0	0	0	0	6.2351	7th

2) In this second table are shown the results of the algorithm in presence of attacks: the attack is considered each time on a different sensor and is equal to $\frac{1}{5}$ of the measurement of the sensor. We can see that in the majority of the cases, the estimated cell is always the seventh one, in some other cases the estimated cell is the third one. Looking at the columns of the matrix D and at the values of the y vector we can see that the most similar column to the data in the vector measurement is the 7th one. At the same time, we can see that adding the attack to the 2nd and 3rd sensors, the most similar column to the measured data is the 3rd one.

MATRIX	**LAMBDA**	**ATTACK**	**X**							**CELL**
G=[D I] not normalized	10^{-3}	1st sensor (value = 12.4)	0	0.0202	0.2877	0	0	0.0472	0.6374	7th
G=[D I] normalized	1	1st sensor (value = 12.4)	0	0	0	0	0	0	6.1904	7th
G=[D I] not normalized	10^{-3}	2nd sensor (value = 11.6)	0	0	0.6052	0	0	0	0.3766	3rd
G=[D I] normalized	1	2nd sensor (value = 11.6)	0	0	4.3573	0	0	0	3.7860	3rd
G=[D I] not normalized	10^{-3}	3rd sensor (value = 8.2)	0	0	0.9260	0.0182	0	0	0	3rd
G=[D I] normalized	1	3rd sensor (value = 8.2)	0	0	6.3919	0	0	0	0	3rd
G=[D I] not normalized	10^{-3}	4th sensor (value = 9.2)	0	0	0.3635	0	0	0.2355	0.4403	7th
G=[D I] normalized	1	4th sensor (value = 9.2)	0	0	3.3701	0	0	0	0	3rd
G=[D I] not normalized	10^{-3}	5th sensor (value = 12.8)	0	0.0040	0	0.3680	0	0	0.7621	7th
G=[D I] normalized	1	5th sensor (value = 12.8)	0	0	0	1.0070	0	0	4.8899	7th
G=[D I] not normalized	10^{-3}	6th sensor (value = 12.6)	0	0.1534	0.0738	0	0	0	0.9578	7th
G=[D I] normalized	1	6th sensor (value = 12.6)	0	0	0	0	0	0	6.9645	7th

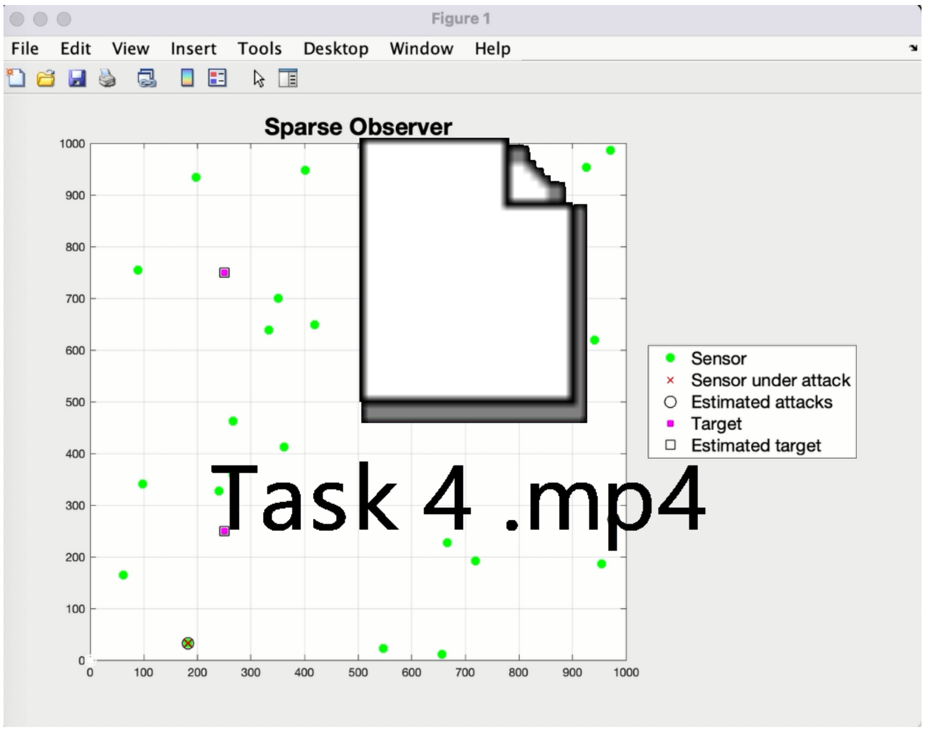
TASK 4

In this task we interface with a localization problem, and to deal with it we use a dynamic approach (where the estimate of x is provided in real time) in which we create a Sparse observer: this is implemented by the online gradient algorithm, that brings us to predict the x(k+1) state and the a(k+1) attack.

As we did in Task 2, we apply the same IST algorithm, with the implementation of the gradient step, and we distinguish 2 kinds of attacks: "*Aware*" and "*Unaware*". In addition, at the end of the algorithm we refine the attacks, by deleting all the estimated ones with magnitude below 2.

The goal of this task is to estimate the j given targets and the h attacks (the targets are moving in a square room with p cells), in such a way that the j largest components of the x, obtained from the IST algorithm, match with the j true targets. Same thing for the attacks.

Instead of reporting the values of the data obtained, we built an animated graph, displayed below, to represent the different states of the sparse observer:



As we can see, after T=50 iterations of the algorithm, all the targets and all the attacks are estimated. This particular run, for example, shows the *Aware attacks*, but we obtain essentially the same results also with *Unaware attacks*: more precisely by executing the program 10 times, we achieved full success (all the attacks and all the target estimated correctly) 9 times out of 10 considering the Aware attacks, and 8 times out of 10 with the Unaware attacks.

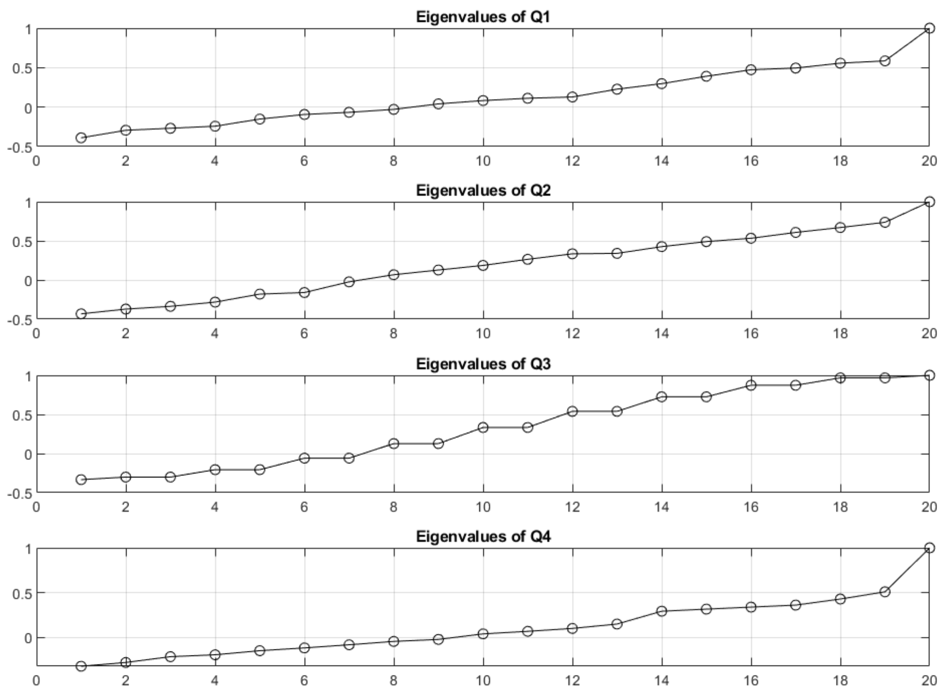
TASK 5

The objective of this task is to give the secure estimate of Task 2 in-network, using the distributed IST (DIST) algorithm.

We consider a (non-sparse) $\tilde{x}$ in presence of sparse sensor attacks and a distributed setting. The experiment has been made on 4 different and stochastic matrices, that represent 4 possible network topologies.

After repeating the experiment more than 20 times, we obtained the following results:

1) The consensus is well reached by the matrices 1, 2 and 4, while the matrix 3 does not completely reach it. If we point out the eigenvalues of the four matrices, we can easily check that for the matrices that reach the consensus there is only one eigenvalue equal to one and the other are far from this value. On the contrary the third matrix has more than one eigenvalue near to the value 1.



In the matrices that reach the consensus, the nodes obtain almost the same estimate.

2) It can be noticed that the matrices 1, 2 and 4, the same that reach the consensus, may estimate better the attacks (either 14 or 13 out of 20), while the third matrix is quite far from this result (3 out of 20).
3) The estimation accuracy is the same for the matrices 1, 2 and 4, while the third one has the biggest value, in fact the estimated x are far from each other.

0.054702	0.057655	0.113525	0.053619

4) About the convergence time, we didn't find a true correlation to the eigenvalues. We noticed that the Q3 matrix is much slower than the other three and we can justify this behavior saying that it takes more iterations because it does not achieve the consensus exactly. The Q2 matrix is the fastest one but it is also the one with the highest norm value (taking into account only 1-2-4) and the one that estimates 13 attacks either than 14 as the Q1 and Q4 do. About the two other matrices we can say that their convergence time is similar as their results are.