MODELING AND CONTROL OF CYBER-PHYSICAL SYSTEMS

Lecture notes

Carlo Migliaccio

AA 2023/2024

Contents

Part I

Modeling of Cyber-Physical systems

Chapter 1

Introduction

1.1 Some definitions

Definition (Helen Gill, 2006) "Cyber-Physical systems are physical, biological, and engineered systems whose operations are integrated, **monitored, and/or controlled** by a **computational core**. Components are **networked** at every scale. Computing is deeply embedded into every physical component, possibly even into materials. The computational core is an embedded system, usually demands real-time response, and is most often distributed"

Roughly speaking a CPS is a **collection of devices** that:

1. compute
2. inter-communicate
3. interact with the physical word

Figure 1.1: Three Dimensions of CPSs

At this point, we can distinguish in this scenario two layers for CPS: (1) The **Cyber Layer** which is linked to the computation and communication issues, (2) the **Physical Layer** deals with the interaction of the devices with the physical word.

1.2 Some examples of CPS

Examples of Cyber-Physical systems could be:

- **Automotive vehicles**: in a car you can find hundreds of sensors, actually we have a computational core, are interconnected in some way (eg. CAN), moreover several feedback-control systems can be find;
- **Teams of mobile robots** that aim to get a target. Robots collaborate to achieve a goal that can be: the exchange of information for example;
- **Wireless sensor networks** in order to monitor and area (indoor localization without a GPS)

1.3 Enabling Technologies and related problems

We can wonder: "How can we deploy CPS?". The answer is by using:

1. **Embedded systems**: hardware and software integrated within mechanical and electrical systems
2. **Sensors and Actuators** for monitoring and control purposes
3. **Communication Networks** for example Wireless communication

Despite the **implementation** it's not too much expensis it raises several issue: at first the **Vulnerability** which is linked to the **Safety** of the overall system.

1.4 Mathematical models for CPS

We can modelize a Cyber-Physical System by using:

- **Basic Models**: continuos-discrete time LTI systems
- **Hybrid models**: they can describe the interaction between devices and the physical layer including both continuous and discrete events;
- **Systems under adversarial attacks**: a CPS could be attacked in order to manipulate the exchanged information;
- **Multi-agent systems**: networks of intercommunicating *agents* which may collaborate to reach a *common goal*: **Consensus**.

Hybrid systems

In order to **model the presence of physical events** that can change the dynamics, we can consider **hybrid systems**, also known as **switched linear systems**. We can describe them by using the following formalism:

$$x(k+1) = A_{q_k}x(k) + B_{q_k}u(k)$$
$$y(k) = C_{q_k}x(k) + D_{q_k}u(k)$$

for $k = 0, 1, ...$

- $x(k)$ is the continuous state
- $q_k \in \{1, 2, ..., Q\}$ is the discrete state or mode, if $q_k \neq q_{k+1}$ then at the time k, the dynamics changes, there is a **switch** for the system
- The parameters denoted with $A_{q_k}, B_{q_k}, C_{q_k}, D_{q_k}$ are associated witch the active submodel, the parameters are piecewise constant. For this reason the hybrid systems could be seen as a particular case of LTI systems.

Some examples of hybrid systems could be: a bouncing ball, a robot moving with obstacles (which are items of the physical layer) in a room etc.

Modeling the presence of attacks

An attack in the framework of CPS can be modeled as an additive term either on the actuator or the (distributed) sensors.

$$x(k+1) = Ax(k) + Bu(k) + b(k)$$
$$y(k) = Cx(k) + Du(k) + a(k)$$

Where we indicate with:

- $b(k)$ the attacks on the actuators
- $a(k)$ the attacks on the sensors

We focus only on the term $a(k)$ which can alter the dynamics of the system. We can't model the attacks as a *disturbance/noise* as we could face them by using some techniques of the classical control theory (eg. Loop shaping).
A "good attack" can't be modeled in a proper way like a disturbance, but fortunately as the sensors are distributed, the attacks can be done only on a subset of them. There are in the common case **sparse attacks.** Our reference model to develop the theory is then the following:

$$x(k+1) = Ax(k) + Bu(k)$$
$$y(k) = Cx(k) + Du(k) + a(k)$$

Chapter 2

Secure estimation of CPSs state under adversarial attacks

2.1 State estimation and Observabilty

Let us consider the LTI system (without input):

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

$y(k) \in \mathbb{R}^q$, $x(k) \in \mathbb{R}^n$, $A \in \mathbb{R}^{n,n}$, $C \in \mathbb{R}^{q,n}$

The **State estimation** is the procedure by which we can **recover** the state $x(k) \in \mathbb{R}^n$ of the system from measurements $y(k)$ for $k = 0, 1, 2, ...$ **Note that...** every element of the vector $y(k)$ is a measure from a sensor, and so we have $y_i(k) \in \mathbb{R}$.
We can say that a system is **Observable** if exists a finite time $T \in \{0, 1, ...\}$ such that $x(0)$ can be recovered from the measurements $y(k), \quad k = 0, 1, ..., T-1$.

But why only $x(0)$? Let us compute $y(0), ..., y(T-1)$ to verify this fact:

$$y(0) = Cx(0)$$
$$y(1) = Cx(1) = CAx(0)$$
$$y(2) = Cx(2) = CA^2x(0)$$
$$...$$
$$y(T-1) = ... = CA^{T-1}x(0)$$

At this point by using the vectorial notation, we have:

$$\begin{pmatrix} y(0) \\ y(1) \\ y(2) \\ \vdots \\ y(T-1) \end{pmatrix} = \begin{pmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{T-1} \end{pmatrix} x(0)$$

$$\mathcal{O}_T = \begin{pmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{T-1} \end{pmatrix}$$

$\in \mathbb{R}^{qT,n}$ where if $T = n$ we call $\mathcal{O}_n$ the **Observability matrix**.
When the equation

$$\begin{pmatrix} y(0) \\ y(1) \\ y(2) \\ \vdots \\ y(T-1) \end{pmatrix} = \mathcal{O}_T x(0)$$

has a **unique solution** the system is observable. At this point, we distinguish two different cases:

- $qt < N$ the system is *underdetermined*
- $qt \geq N$ and $rank(\mathcal{O}_T) = n$, then we can (pseudo)invert it. In particular if $qT = n$ then

$$x(0) = \mathcal{O}_T^{-1} y(k)$$

otherwise if $qT > n$ then

$$x(0) = (\mathcal{O}_T^T \mathcal{O}_T)^{-1} \mathcal{O}_T^T y(k)$$

Where we call **Moore-Penrose pseudoinverse** the matrix $\mathcal{O}_T^\dagger = (\mathcal{O}_T^T \mathcal{O}_T)^{-1} \mathcal{O}_T^T$

Theorem 1 *(Kalman, 1960) An LTI system is observable if and only if rank($\mathcal{O}_n$)=n*

In general there are two approaches for the state estimation:

- **static approach**: solving the equation like we have just seen.
- **dynamic approach**: by using a **Luemberger Observer** that allow us to recover the state after a bunch of steps.

2.2 Secure state estimation

In this section we make a try to expand the concept of **Observability and state estimation** when we have attacks on the sensors, and so the additive term $a(k)$ in the output equation.

$$\begin{aligned} x(k+1) &= Ax(k) \\ y(k) &= Cx(k) + a(k) \end{aligned}$$

Assumption The term $a(k) \in \mathbb{R}^q$ is **sparse** in the sense that no more than $h \ll q$ of its elements are non-zero. By using the l_0-norm, $\|a\|_0 \leq h$.

We refer to **Secure state estimation** when we want recover the state $x(0)$ from sensors' measurements $y(k), k = 0, 1, ...$ and **unknown attacks** $a(k)$. (As we said we are not able to model attacks in a proper way, we wouldn't have a different theory and several techniques to face the problems related to them).

Let us anylize the mentioned problem by spotting the differences that occurs in the case of attacks:

$$\begin{aligned} y(0) &= Cx(0) + a(0) \\ y(1) &= Cx(1) = CAx(0) + a(1) \\ y(2) &= Cx(2) = CA^2x(0) + a(2) \\ &... \\ y(T-1) &= ... = CA^{T-1}x(0) + a(T-1) \end{aligned}$$

Then

$$\begin{pmatrix} y(0) \\ y(1) \\ y(2) \\ \vdots \\ y(T-1) \end{pmatrix} = \begin{pmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{T-1} \end{pmatrix} x(0) + \begin{pmatrix} a(0) \\ a(1) \\ a(2) \\ \vdots \\ a(T-1) \end{pmatrix}$$

where we have qT equations for $n + qT$ unknowns! That would indicate potentially **infinitely many solutions**.
Despite this observation if we add the hypotesis that the vector containing the attack is **sparse**, in some situations we could have a unique solution to this problem.

Definition 1 *We say that h errors are **correctable** after T steps if its possible to recover any $x(0)$ given $y(0), ..., y(T-1)$ under the condition $\|a\|_0 \leq h$, for each $k = 0, ..., T-1$.*

This corresponds to ask that the problem

$$\begin{pmatrix} y(0) \\ y(1) \\ y(2) \\ \vdots \\ y(T-1) \end{pmatrix} = \begin{pmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{T-1} \end{pmatrix} x(0) + \begin{pmatrix} a(0) \\ a(1) \\ a(2) \\ \vdots \\ a(T-1) \end{pmatrix} \quad \text{s.t.} \quad \|a\|_0 \leq h, k = 0, ..., T-1$$

had a **unique solution**.
Now we have to face with two problems:

1. Under **which conditions** can I solve the proposed problem?
2. **How can I solve** it?

2.3 Well-Position of the problem (Static case)

Let us analyze the problem in a very particular case, that is when the system we want to ovbserve is **static**. That is the matrix $A \in \mathbb{R}^{n,n}$ is the identity matrix $\mathbb{I}_n$. This corresponds to state that $x(k+1) = x(k) = x$ (remember that we have no input $u(t)$).
In this case the observability matrix has a very simple form, in particular it is equal to the matrix $C \in \mathbb{R}^{q,n}$. We have that the problem becomes:

$$y = Cx + a \quad \text{s.t.} \quad \|a\|_0 \leq h$$

It is useful to give these formal definitions:

Definition 2 (h-sparsity) *A vector is said to be **h-sparse**, if $\|a\|_0 = h$*

Definition 3 (Support) *Given $a \in \mathbb{R}^q$ we denote with $Supp(a) = \{i : a_i \neq 0\}$, the set of i such that the i-th component of the vector is a non-zero number.*

How the attack could be done in order to not to be 'detectable'? Let's consider an $a = Cw, \quad w \in \mathbb{R}^n, w \neq 0$ and assume it is that $\|Cw\|_0 \leq h$ (for the assumption we made). We sobstitute in the equation and obtain:

$$y = Cx + Cw = C(x + w) \quad \text{s.t.} \quad \|Cw\|_0 \leq h$$

Actually, if such w exixts, the **attack is feasible** and so not detectable. This property depends strongly on the property of the matrix C.
Fortunately we have a **proposition** that provide us with an equivalence to determine the **resilience** of the system to h attacks.

Proposition 1 (Correctability) *h errors are correctable (or equivalently **the system is resilient against h attacks**) after T steps if and only if for all $z \in \mathbb{R}^n$, $\|\mathcal{O}_T z\|_0 > 2h$.*

$$\textbf{\textit{System correctable}} \iff \forall z \in \mathbb{R}^n, \quad \|\mathcal{O}_T z\|_0 > 2h$$

In the static case the Proposition 1 becomes:

$$\textbf{System correctable} \iff \forall z \in \mathbb{R}^n, \quad \|Cz\|_0 > 2h$$

Proof 1 *As this is a characterization property we have to proof the proposition in the two direction $\Leftarrow$ and $\Rightarrow$. We give this proof for the static case, but the procedure for the more general case is analogue.*

- ***[Proof of $\Leftarrow$]*** *Assuming that $\forall z \in \mathbb{R}^n$ we have $\|Cz\|_0 > 2h$ we want to demonstrate that h errors are correctable, that is the problem has a unique solution. As in many situation we want to demonstrate the uniqueness of something, we can go on **by contraddiction**. In particular assume that $y = Cx + a$ s.t. $\|a\|_0 \leq h$ has got two different solutions:*

$$\begin{pmatrix} x' \\ a' \end{pmatrix} \text{ and } \begin{pmatrix} x'' \\ a'' \end{pmatrix}$$

*then we have that $y = Cx' + a' \quad s.t. \quad \|a'\|_0 \leq h$ but also $y = Cx'' + a'' \quad s.t. \quad \|a''\|_0 \leq h$. $Cx' + a' = Cx'' + a'' \iff C(x' - x'') = a'' - a'$ But due to the fact that a' and a'' are h-sparse we can obtain by substracting them a vector which is **at most** 2h-sparse. We have finished because we found a contradiction , that is, $\exists z \in \mathbb{R}^n, \|Cz\|_0 \leq 2h$. In this case $w = a' - a''$*

- ***[Proof of $\Rightarrow$]*** *Assuming that h errors are correctable, we want to demonstrate that $\forall z \in \mathbb{R}^n$ we have $\|Cz\|_0 > 2h$. Similarly the former case, we can demonstrate the property by contradiction assuming that $\exists w \in \mathbb{R}^n \quad \|Cw\|_0 \leq 2h$. We can write such Cw like a sum of two (at most) h-sparse vectors g_1 and $g_2 \rightarrow Cw = g1 + g2$. But we can also say that $Cw = g_1 + g_2 \longleftrightarrow Cw - g_1 = g_2 = C0 + g_2$. In this way we are saying that we have **two distinct solutions**, this is in contradiction with our hypotesis.* `QED`

The proposition that has just been proved is very powerful, but how are we able to try for all z that the statement is valid? In the great majority of the situations is easier to provide a **counter-example**.
Now we are going to do some example and then to provide (at least) a necessary condition for well-position of the problem of correctability.

2.3.1 Some examples

Example 0: a "naive" example

Consider that we have $C = \mathbb{I}_n$, $q = n$, $h = 1$, $n = 3$. In this case the output equation is:

$$\begin{aligned} y_1 &= x_1 \\ y_2 &= x_2 \\ y_3 &= x_3 \end{aligned}$$

As we can see the system is **perfectly observable** because each measurement of y gives an element of the state, but suppose that we know that there is an attack ($h = 1$), but we don't know where. If one of the measurement is corrupted, there is no way to recover the state of the system $\Rightarrow$ C is not correcting h errors $\Rightarrow$ the system is not resilient to h attacks. In an intuitive way we can say that we could add more sensors to improve the situation, it is 'the path' that follows the next example.

Example 1: add more sensors (increase q)

At this point we change the (sensing) matrix C by adding new measurements, so

$$C = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

What has changed here is that we have **two more** sensors. We have a duplication of C_1 and C_2 (we indicate with C_i the i-th row of C). If $y_1 = C_1x = y_4 = C_4x$ and $y_2 = C_2x = y_5 = C_5x$ than we can state that the attack is on the sensor 3. But **what if either** $y_1 \neq y_4$ **or** $y_2 \neq y_5$? We have no way to know which sensor has been attacked. If I switched off this couple of devices I can't observe anymore one of the component of the state vector.
This was only an **intuitive way** to understand this fact. How can we formalize it? We can say that the couple (C_1, C_2) has a **non trivial kernel**. That is the equation:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} z = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

has a non zero solution. In particular a solution could be $z = \begin{pmatrix} 0 \\ 0 \\ \alpha \end{pmatrix}$ such z produces

$$Cz = \begin{pmatrix} 0 \\ 0 \\ \alpha \\ 0 \\ 0 \end{pmatrix} \quad \alpha \in \mathbb{R}$$

which due to the fact which is 1-sparse, is **in contradiction** with the proposition.
Conclusion: **even this matrix does not correct h=1 error $\Rightarrow$ the system is not resilient**!

Example 2: more mixed measurements

Freezing the other parameters, let us change again the matrix C in the following way:

$$C = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & -1 & 1 \end{pmatrix}$$

It is quite clear that there is a triple of rows of C which are linearly dependent, specifically $C_5 = C_4 - 2C_2 \Rightarrow$ the triple has a non trivial kernel. By doing simple algebraic steps we desume that a solution could be $z = \begin{pmatrix} \alpha \\ 0 \\ -\alpha \end{pmatrix}$ which generates a 2-sparse Cz we have again a contradiction! Conclusion: **C is not resilient**.

Example 3: linearly independent measurements

Now we provide the following C matrix:

$$C = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 2 & -1 \end{pmatrix}$$

In this case all the triples are linearly independent, this cause the relative kernel to be trivial $\rightarrow$ there is no way to produce a counter-example. Conclusion: this C matrix corrects h=1 error.

2.3.2 A necessary condition for well-position

Let $C \in \mathbb{R}^{q,n}$, $rank(C) = n$ and $q > n$, it is quite clear that any subset Ω composed by $n-1$ rows has got a non trivial kernel. This implicates that the sparsity is not larger than $q-(n-1)$, that is $2h < q-(n-1)$ or equivalently $h \leq q-n$. From which I can recover the following inequality:

$$q \geq 2h + n$$

It is immediate to understand that if I want to correct $h = 1$ error with $n = 3$, we need at least of $q = 5$ sensors.

This fact bring us to state that:

- Large q is not sufficient for **resilience**
- On the other hand there is a **minimum q** which I need to correct a certain number h of errors.

2.4 Reformulation of $y = Cx + a \quad$ s.t. $\quad \|a\|_0 \leq h$

The second question to answer is:

How can we solve the problem $y = Cx + a \quad$ **s.t.** $\quad \|a\|_0 \leq h$**?**

The problem can be reformulated as follows:

$$\min_{x\in\mathbb{R}^n, a\in\mathbb{R}^q} \|a\|_0 \quad \text{s.t.} \quad y = Cx + a$$

It can be proved that **if the system is resilient to h attacks the solution to this problem corrects h errors**. Unfortunately, even this form of the problem has its drawbacks:

- As the problem is a combinatorial one, is **not feasible** (NP-Hard)
- The **objective function** which contains an ℓ_0 norm is **not convex, non continuous and non differentiable in 0**

The solution is going in the direction of **convex relaxation**:

1. In the objective function, we choose to relax $\|a\|_0$ with its **best CONVEX approximation** $\|a\|_1$; it is CONVEX, CONTINUOUS but NON DIFFERENTIABLE IN 0 (we can face this problem);
2. In the real world, we must take into account the noise that could appear in the equation $y = Cx + a$ and so it becomes $y \approx Cx + a$, this leads to the **Least Squares (LS) problem**

$$\min \frac{1}{2}\|y - Cx - a\|_2^2 = \min \frac{1}{2}\left\| y - \begin{pmatrix} C & I \end{pmatrix} \begin{pmatrix} x \\ a \end{pmatrix} \right\|$$

Combining this two approximations, due to the fact we want to minimize both the terms, the **resulting problem to solve** is:

$$\min_{x\in\mathbb{R}^n, a\in\mathbb{R}^q} \frac{1}{2}\left\| y - \begin{pmatrix} C & I \end{pmatrix} \begin{pmatrix} x \\ a \end{pmatrix} \right\| + \lambda\|a\|_1, \quad \lambda > 0$$

2.4.1 Why ℓ_1-regularization promotes sparsity?

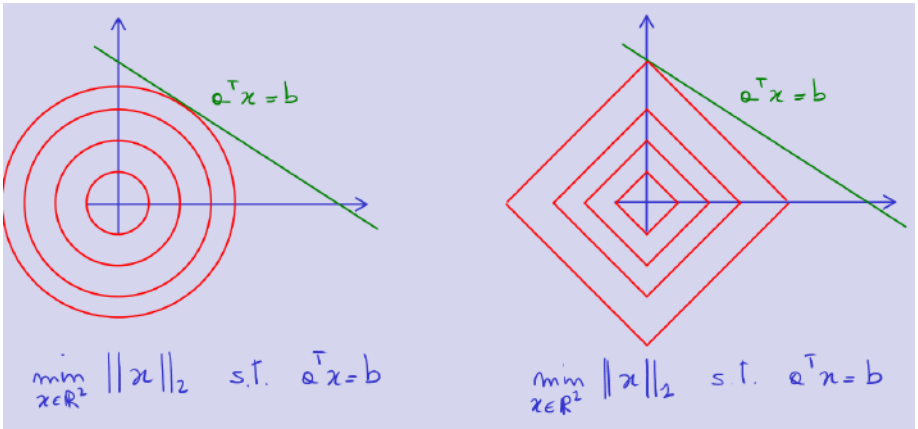

Figure 2.1: ℓ_p-norms and sparsification

In the approximation we have just made, we have considered the ℓ_1-norm. But why it is the best we could do? Let us consider to solve the problem

$$\min_{x\in\mathbb{R}^2}\|x\|_2 \quad \text{s.t.} \quad a^T x = b$$

The term $\|x\|_2 = k, k \in \mathbb{R}$ is a circle in the plane. We start from $k = 0$, and we increase it until we reach the line to satisfy the (linear) constraint. The solution we obtain is certainly not sparse because it is of the form $x^* = (x_1^*, x_2^*)$ both non-zero real number.
On the other hand if we consider the problem:

$$\min_{x\in\mathbb{R}^2}\|x\|_1 \quad \text{s.t.} \quad a^T x = b$$

we note that the solution we can find (by using the same method as before) is sparse. We have seen in an intuitive way that ℓ_2-norm takes the components of the solution close to zero in a "democratic way", in other words without 'reset' any component. At the opposite the ℓ_1-norm push to zero **only some components** making a selection among them $\longleftrightarrow$ this promotes **"sparsification"**.

[Note that we have just showed a trivial way to understand why we have chose one regularization instead of another. The topic would require a more accurate and rigorous explanation, which we don't care.]

2.4.2 Some observations

The problem has a connection with LASSO

The problem we found to solve after applying **relaxation** is very similar to the problem of **LASSO** (Least Absolute Shrinking and Selection Operator) in fact in its original formulation we have:

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

but this give an entire sparse solution. In our problem only a piece of the solution is sparse, whose linked to attacks a, so we can see our problem as a "**Partial LASSO**" because we have **no regularization** on the term x of the solution $\begin{pmatrix} x \\ a \end{pmatrix}$.

The problem has a connection with "Compressed Sensing"

The problem

$$\min_{x\in\mathbb{R}^n, a\in\mathbb{R}^q}\frac{1}{2}\left\| y - G\begin{pmatrix} x \\ a \end{pmatrix}\right\| + \lambda\|a\|_1, \quad \lambda > 0, \quad G = \begin{pmatrix} C & I \end{pmatrix}$$

as the G matrix has more columns than rows, is a **fat matrix** and without regularization the problem has infinitely many solutions. We can consider the y a **linear compressed measurement** vector. The problem of **recover a sparse vector from compressed linear measurement** has a strong connection with the COMPRESSED SENSING problem. Also here we have a **"Partial Compressed Sensing"** as only a part of the solution is sparse.

After a "long journey" we have understood that for the **secure state estimation of CPS under adversarial attacks** we have to solve a **Partial LASSO** problem. For this reason we seek a way to solve it $\Rightarrow$ Iterative Algorithms

2.5 IST: an algorithm for LASSO

Premise *In this course we will not see any black box technique, we will go through the direction of* ***iterative algorithms*** *to solve the LASSO problem.*
In the framework of ***Convex Optimization*** *one of the most used algorithms (also in Data Science/Machine Learning...) is the* ***Gradient Descent Algorithm****. But in the LASSO we have a ℓ_1-regularization, which is not differentiable in 0. We can't even avoid to consider it because we want to reach the (global)* ***minima****. We wonder if using an approximation could be the solution (eg.* ***subgradients****), however again this not produce a sparse solution.*

Once the proper promises have been done, we can introduce this algorithm for solving the **(sparse) optimization problem** of LASSO. The algorithm is called **Iterative Shrinkage/Thresholding (IST)** and it is a variant of the *Descent Gradient Algorithm*. After the definition and a general description, we are goingo to list the steps of the algorithm itself. Initially we consider the original LASSO problem

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

in which we call $F(x) = \|Ax-y\|_2^2$ the Least-Squares functional. Moreover we know that $\nabla F(x) = A^T(Ax-y)$.

Definition (Shrinkage/Thresholding operator) The **Shrinkage/Thresholding operator** $\mathbb{S}_\alpha$ is a component wise operator $\mathbb{S}_\alpha : \mathbb{R}^p \to \mathbb{R}^p, \alpha > 0$. For any $x_i \in \mathbb{R}$:

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

IST Algorithm

1. Initialization: $x_0 \in \mathbb{R}^p$, e.g. $x_0 = 0$
2. For $k = 0, ..., T_{max}$

$$x(k+1) = \mathbb{S}_{\lambda\tau}\left[x(k) - \tau\nabla F(x(k))\right]$$

The parameter τ has to be "small enough" so that the algorithm works as desidered. It has been demonstrated that the algorithm converge to the minimum of the LASSO functional $\frac{1}{2}\|Ax-y\|_2^2 + \lambda\|x\|_1$. The **iterative nature** and simplicity of this method, makes it adaptable also for: **dynamic** and **distributed** systems.

2.5.1 Derivation of IST Algorithm

Let us do a quick RECAP of the latest notions we introduced...
Generally speaking, when we have some measurements $y = A\tilde{x} + \eta$ where $\tilde{x}$ is the **true vector** to recover, we are able to find an **estimate** of it by solving the problem

$$x^* = \arg\min_{x\in\mathbb{R}^n} \frac{1}{2}\|y-Ax\|_2^2$$

also known as the **Least Squares (LS)** problem, where the obtained x^* is the estimate of the true vector. How can we solve this problem?
The **Gradient Descent** method is one of the most used in this field to retrieve a solution for LS problem. This algorithm step by step proceeds in the direction of the gradient, moving from the previous point of a **step size** called τ (small enough). Then we have

$$x(k+1) = x(k) - \tau \nabla F(x), \quad F(x) = \|y - Ax\|_2^2 \quad \text{(LS functional)} \tag{2.1}$$

for $k = 0, 1, 2, ...$
We can observe that the expression (2.1) is a Discrete Time Linear Time Invariant System (DT LTI), therefore we can study its properties by using the known results from the Theory.

Differently from the case which has just mentioned, in the LASSO problem we have the ℓ_1-norm too, because we are interested in finding a **sparse solution** (in our framework of **CPS under adversarial attacks** this is an important requirement). It is good to remind that the LASSO is a **convex, non-differentiable** problem.
Then, how can we solve it? Its solution can be retrieved by using any convex optimization solver (`cvx,lasso, ...`). On the other hand, the *Gradient Descent* algorithm is not effective! (Gradient of a non differentiable functional?? We can't!). Our **approach** is using the **ISTA** Algorithm of which it is given a **possible interpretation**.

Let the functional $F(x) = \frac{1}{2}\|y - Ax\|_2^2 + \lambda\|x\|_1$ be the LASSO functional. We define now a **surrogate functional** which will guide our *derivation of ISTA* as

$$\mathcal{R}(x, b) = F(x) + \frac{1}{2\tau}\|x - b\|_2^2 - \frac{1}{2}\|Ax - Ab\|_2^2, \quad \tau > 0 \tag{2.2}$$

where x is my 'standard' variable, while b is an auxiliary variable. By using a little bit of linear algebra we note that:

$$\frac{1}{2\tau}\|x - b\|_2^2 - \frac{1}{2}\|Ax - Ab\|_2^2 = \frac{1}{2\tau}\|x - b\|_2^2 - \frac{1}{2}\|A\|_2^2\|x - b\|_2^2 = \frac{1}{2}\left(\frac{1}{\tau} - \|A\|_2^2\right)\|x - b\|_2^2$$

and this quantity is non-negative if and only if $\frac{1}{\tau} > \|A\|_2^2 \Rightarrow \tau < \|A\|_2^{-2}$. This provides a guide to choose the τ and ensures that the **surrogate part** of the functional (2.2) is never negative, moreover it can immediately be noted that it is null if and only if $x = b \Rightarrow F(x) = R(x, x)$. That is the global minimum of $F(x)$ is the same of the global minimum of $R(x, x) \Rightarrow F(x^*) = R(x^*, x^*)$. At this point: our aim is to develop a **descent algorithm** for $F(x)$ which we will see that it is eased by $R(x, b)$.

It is not trivial to find a closed form for the solution of minimizing the surrogate functional considering both variables x, b together. Therefore, we move in the direction of an **alternating minimization** (alternating $\Rightarrow$ in turn). This is both **feasible** and **leads to the minimum** of $F(x)$. More clearly:

$$\begin{cases} \min_{b \in \mathbb{R}^n} \mathcal{R}(x, b), & \text{keeping } x \text{ as a constant} \\ \min_{x \in \mathbb{R}^n} \mathcal{R}(x, b), & \text{keeping } b \text{ as a constant} \end{cases}$$

First step: minimizing with respect to b

It is more trivial because we have seen recently that the *surrogate part* depends on b, and it is minimum if $x = b$, then

$$x = \arg\min_{b \in \mathbb{R}^n} \mathcal{R}(x, b)$$

Second step: minimizing with respect to x

The problem here is find $\arg\min_{x\in\mathbb{R}^n} \mathcal{R}(x,b)$. In this case we have to expand the functional by expressing the squared norm explicitly, to reach a conclusion.

$$\begin{aligned}
\arg\min_{x\in\mathbb{R}^n} \quad & \mathcal{R}(x,b) = \arg\min_{x\in\mathbb{R}^n} \quad \frac{1}{2}\|Ax\|_2^2 + \frac{1}{2}\|y\|_2^2 - yA^Tx + \lambda\|x\|_1 + && (2.3)\\
& \frac{1}{2\tau}\|x\|_2^2 + \frac{1}{2\tau}\|b\|_2^2 - \frac{1}{\tau}b^Tx + && (2.4)\\
& -\frac{1}{2}\|Ax\|_2^2 - \frac{1}{2}\|Ax\|_2^2 - b^TA^TAx = && (2.5)\\
\arg\min_{x\in\mathbb{R}^n} \quad & \lambda\|x\|_1 + \frac{1}{2\tau}\|x\|_2^2 - x^T\left(\frac{1}{\tau}b + A^Ty - A^TAb\right) = && (2.6)\\
& \tau\lambda\|x\|_1 + \frac{1}{2}\|x\|_2^2 - x^T[b + \tau A^T(y - Ab)] + \frac{1}{2}\|b + \tau A^T(y - Ab)\|_2^2 = && (2.7)\\
\arg\min_{x\in\mathbb{R}^n} \quad & \tau\lambda\|x\|_1 + \frac{1}{2}\|x - q\|_2^2 = \arg\min_{x\in\mathbb{R}^n} \quad \sum_{i=1}^{n}\left(\tau\lambda|x_i| + \frac{1}{2}(x_i - q_i)^2\right) && (2.8)
\end{aligned}$$

In (2.7), by multiplying all terms in (2.6) by τ and by adding a constant term $q = b + \tau A^T(y - Ab)$ nothing change. Moreover the final step (2.8) is separable, in the sense that If I want minimize the sum, I can proceed **component wise**. Therefore

$$\arg\min_{x\in\mathbb{R}^n} \quad \sum_{i=1}^{n}\left(\tau\lambda|x_i| + \frac{1}{2}(x_i - q_i)^2\right) \Longleftrightarrow \arg\min_{x_i\in\mathbb{R}} \quad \tau\lambda|x_i| + \frac{1}{2}(x_i - q_i)^2 = (...) = \tag{2.9}$$

$$= \begin{cases} q_i - \tau\lambda & \text{if} \quad q_i > \tau\lambda \\ q_i + \tau\lambda & \text{if} \quad q_i < -\tau\lambda \\ 0 & \text{if} \quad q_i \in [-\tau\lambda, \tau\lambda] \end{cases} \tag{2.10}$$

but this is the operator we formerly introduced as $\mathbb{S}_{\tau\lambda}(q)$ which pushes the component of the vector q close to zero. We can reach a **conclusion**.
Given $x_0 = 0$ for any $k = 0, 1, ...$

1. $b(k+1) = \arg\min_{b\in\mathbb{R}^n} \mathcal{R}(x(k), b) = x(k)$
2. $x(k{+}1) = \arg\min_{x\in\mathbb{R}^n} \mathcal{R}(x, b(k{+}1)) = \arg\min_{x\in\mathbb{R}^n} \mathcal{R}(x, x(k)) = \mathbb{S}_{\tau\lambda}[x(k) + \tau A^T(y - Ax(k))]$

Then,

$$x(k+1) = \mathbb{S}_{\tau\lambda}[x(k) + \tau A^T(y - Ax(k))] \tag{2.11}$$

which is our **Iterative Shrinkage/Thresholding algorithm (ISTA)**. Then, at the end of this discussion we have seen that the algorithm on which we focus comes from an **alternating minimization** of the surrogate functional $\mathcal{R}(x, b)$.

Theorem (1) ISTA is a **descent algorithm** for LASSO, then $F(x(k+1)) \leq F(x(k))$

Proof The proof of the theorem above is very simple:

$$\begin{aligned}
F(x(k)) = & \mathcal{R}(x(k), x(k)) \geq && \text{(minimizing over x)}\\
& \mathcal{R}(x(k+1), x(k)) \geq && \text{(minimizing over b)}\\
& \mathcal{R}(x(k+1), x(k+1)) = F(x(k+1), x(k+1))
\end{aligned}$$

Theorem (2) ISTA converges to the minimum of LASSO.

2.5.2 Curiosity

The equation (2.11) combines a linear part (the argument of the operator $\mathbb{S}$) and a **non linear part** that is the operator itself, being defined in a piece-wise fashion. The resulting system is a **Discrete Time Non Linear Time Invariant system** which is not trivial to treat.
This reminds a little the structure of a **neural network** where we have a **linear part** and a **non linear part** constituted by the **activating function**. For this reason the **evolution of ISTA** can be seen as a **Neural Network** application.

2.5.3 Application of ISTA on CPS framework

We have seen recently that the problem of Cyber-Physical system under attacks can be formulated as follows:

$$\min_{x\in\mathbb{R}^n, a\in\mathbb{R}^q} \frac{1}{2}\left\| y - G\begin{pmatrix} x \\ a \end{pmatrix} \right\| + \lambda\|a\|_1 \tag{2.12}$$

that is a **partial LASSO** because only a part of the found solution is sparse, whose related to the (sparse) attacks. What about ISTA for Partial LASSO? I have no limitations about the choice of the coefficient λ, in particular I can get $\lambda = 0$ for the elements of the solution which are not interested by sparsity as it is the state of the system. Then,

$$\lambda\|a\|_1 = \lambda\left\| \begin{pmatrix} x \\ a \end{pmatrix} \right\|_1 = 0|x_1| + ... + 0|x_n| + \lambda|a_1| + \lambda|a_q| \tag{2.13}$$

This changes nothing, but it is essential that we made this variation in a way that the derivated descent algorithm could fit the "Partial LASSO" problem used for CPS purposes.

Chapter 3

Localization by RSS fingerprinting

3.1 Introduction

Localization is the problem related to the **estimation** of the position of a **target**. Other problems are those related to the **detection** (whether a target is present or not) and **tracking** that is we track the position of a **moving target**.
Nowadays the focus is in particular on **Indoor Localization** whose mathematical modeling is quite challenging due to the presence of *multipath* and *reflecting surfaces*. For these reasons there is not an available **unified approach**.
GPS is not usable for indoor localization, the so called **WSN (Wirelless Sensor Networks)** are used for this purpose. A WSN is essentially a network of devices equipped with sensors. We can find two main approaches:

1. **Triangulation and Trilateration**: in this case we assume that the target is a **transmitting device** that broadcasts a signal. The former method deploy the **direction of arrival** of such signal, the latter use instead the **distance** from the source's signal (*id est* the target). For this approach, in the case that the measurement are exact, 3 non-aligned sensors are sufficient, alternatively the higher number of sensors, the higher precision reached.

2. **Fingerprinting methods** refers to techniques that match the **fingerprint** of some characteristics of a signal which is **location-dependent**. For our purposes we consider the **Received Signal Strength** or **RSS**.

3.2 RSS-fingerprinting: general description

In general for all **Fingerprinting method**, we identify **two phases**:

1. **Training phase** in which we collect the fingerprints of a scene;

2. **Runtime phase** in which we match the online measurement with the closest a-priori location fingerprints.

As we said, we focus our attention on *RSS-fingerprinting*.

3.2.1 (Phase 0) Initialization

Given the room where the localization has to be done, we deploy the WSN in some way (For example: grid deployment, or random (uniform) deployment, split the room into p cells. **Localize the target** $\longrightarrow$ detect in which cell the target is placed.

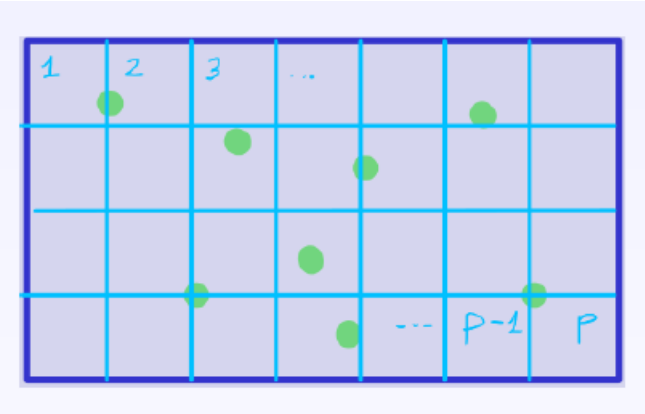

Figure 3.1: (Phase 0) - Initialization

3.2.2 (Phase 1) Training phase

We put the target in each cell. According to the fact that the target itself broadcasts a signal, **each sensor** measures and stores the RSS, and create a **signature map** or **dictionary**. More specifically, the dictionary can be represented by a matrix D, in which each entry $D_{i,j}$ denotes the RSS-measurement the sensor i takes when the target is in the j-th cell.
Each sensor builds its own dictionary, and the WSN builds an **overall dictionary** $D \in \mathbb{R}^{q,p}$. This phase takes some time and requires that the nodes of the network saved the information, it makes the runtime phase more accurate.

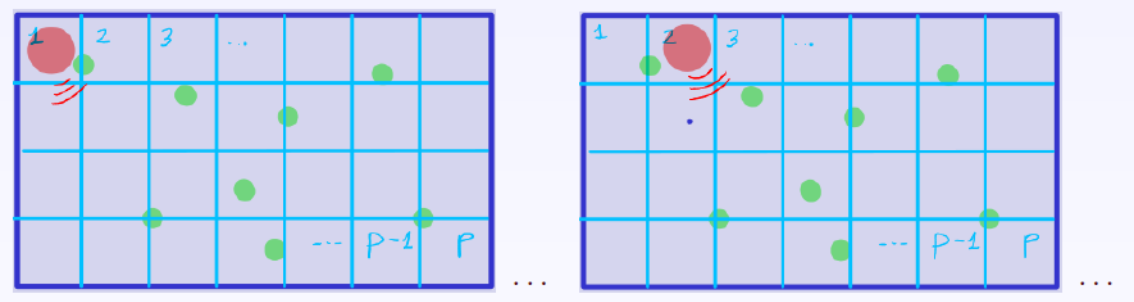

Figure 3.2: (Phase 1) - Training phase

3.2.3 (Phase 2): Runtime phase

Is the phase in which the localization is performed. Each sensor takes a measurement y_i, the j-th column of the dictionary indicates the j-th cell of the room in which the target is located! If there is one target, then one of the columns of the dictionary would correspond to the measurement (in theory), then the target is located in the j-th cell where j is such that $y = D_j$, with D_j the $j - th$ column of the dictionary but usually, we can have *multiple target* in the same room moreover it's very difficult that $y = D_j$ due to the presence of noise on sensors.

Then, in this scenario we have that the vector of measurements y is a sum of a subset of the columns of the matrix D, to which we must add an additional term related to the noise. We have:

$$y = Dx + \text{noise} \tag{3.1}$$

Where $x \in \{0, 1\}^p$ and $x_i = 1$ when in the i-th cell there is a target.
A further observation which can be done is that, in the real problems of *indoor localization*

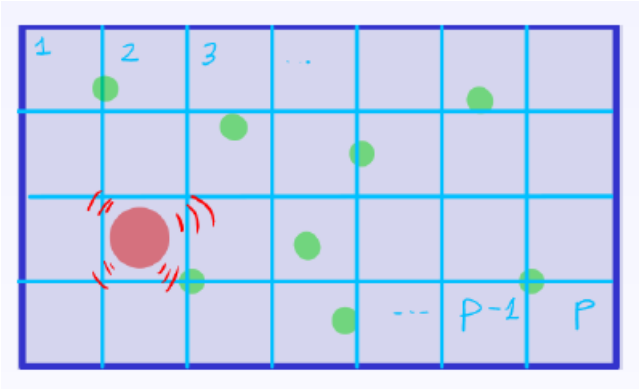

Figure 3.3: (Phase 2) - Runtime phase

the number of the targets one would like to localize is **much smaller** than the number of the cells, this leads to the sparsity of the solution $x \in \{0,1\}^p$. Summarizing: (i) We want to find a solution to the problem (3.1), (ii) We know that such solution is **sparse**. Then,

$$x^* = \arg \min_{x \in \{0,1\}^p} \frac{1}{2}\|y - Dx\|_2^2 + \lambda \|x\|_1 \tag{3.2}$$

This type of formulation leads to a **mixed integer** combinatorial problem, that again is NP-hard, so non-tractable. We can relax the constraint $x \in \{0,1\}^p$ into $x \in \mathbb{R}^p$, obtaining:

$$x^* = \arg \min_{x \in \mathbb{R}^p} \frac{1}{2}\|y - Dx\|_2^2 + \lambda \|x\|_1 \tag{3.3}$$

This is the problem of LASSO in the original formulation. It is important to remember that: the solution of the problem (3.3) is not the original x due to: (i) the presence of the noise, (ii) a bias introduced by the ℓ_1-regularization term which in part promotes sparsity on the other hand inserts an error. One can use of course the **IST Algorithm** to find a solution.

3.3 Localization under sparse sensor attacks

Assume that, in a realistic way, the training phase is *attack-free*, therefore the matrix D is attack free. Let us focus the attention again in the runtime phase.
What if the sensors were under adversarial attacks? One can apply the theory we developed in the former chapter! Then, we assume that the sensors under attacks are much smaller than the total number q of them. In this context the equation (3.1) becomes:

$$y = Dx + a + noise \tag{3.4}$$

Then, the formulation of problem changes as follows:

$$x^* = \arg \min_{x \in \mathbb{R}^p} \frac{1}{2}\|y - Dx - a\|_2^2 + \lambda_1 \|x\|_1 + \lambda_2 \|a\|_1 \tag{3.5}$$

where we are using different weights λ_1, λ_2 to give more or less importance to the term related to the solution x and to the attack a. According to this novel aspect, the risen problem is a **weighted LASSO**. **IST algorithm** continues to work, as we saw in the case of presence of *non-sparse* part of the solution in the case of SSE under adversarial attacks.

3.4 Other approaches to Localization

The approach we have just seen is not the only one. Next paragraphs are in order to give some alternatives which differ from the first we have presented in computational complexity, time of convergence and so on.

3.4.1 k-Nearest Neighbour (K-NN)

Assume to know that there is only **one target**, given the vector $y \in \mathbb{R}^q$, we could find the $j-th$ column of the dictionary D that is the **nearest** with respect to y. The localization problem in this specific scenario becomes:

$$\hat{j} = \arg \min_{j=1,\dots,p} \|D_j - y\|_2^2 \tag{3.6}$$

where D_j is the j-th column of D. Note that in the problem there is not the factor $\frac{1}{2}$ but from the moment we have a minimization problem nothing changes.
RSS is additive so we can localize more than one target in the *runtime phase*. Suppose that the targets are $k = 2$, the vector y can be seen as a sum of the columns, so the problem (3.6) gets transformed in:

$$(\hat{j_1}, \hat{j_2}) = \arg \min_{(j_1,j_2)=1,\dots,p} \|D_{j_1} + D_{j_2} - y\|_2^2 \tag{3.7}$$

In general, we have to check a number of configurations that is equal to $\binom{p}{k} \rightarrow$ **NP-Hard**. Note that: this approach can be used if we have small p, k otherwise other techniques ought to be used in order to promote efficiency.

3.4.2 Linear Regression (noise-free case)

If we have multiple targets, in absence of noise the localization problem could be formulated as a **binary linear regression**. We have to solve in x the equation $y = Dx$ and add the constraints about the x domain and sparsity. Then, it is obtained:

$$\begin{aligned} & Dx = y \\ & \text{s.t. } x \in \{0,1\}^p, \ \sum_{j=1}^{p} x_j = k \end{aligned} \tag{3.8}$$

Even in this case it has been risen a mixed-integer combinatorial problem $\rightarrow$ NP-Hard. The choice should be taken in an accurate way according to the dimension of the problem.

3.5 Some comments on the setting

These algorithms assume that there is a **Fusion Center** where data from the sensors are collected. In this way: the Fusion Center stores the whole dictionary D and the runtime measurements, so that it can run the localization algorithm which is nothing but one of the exposed methods.

Chapter 4

Dynamic Secure State Estimation of CPSs under adversarial attacks

It has been explained in the previous chapter that for a system

$$\begin{aligned} x(k+1) &= Ax(k) \\ y(k) &= Cx(k) \end{aligned} \tag{4.1}$$

if one collects n measurements $y(i), i = 1, ..., n$, we can recover the state $x(k)$ at each k, if we are able to find $x(0)$ and then invert the equation

$$y = \mathcal{O}_n x(0)$$

the Theorem by Kalman, states that this is possible, that is the system is **observable**, if and only if $\text{rank}(\mathcal{O}_n) = n$. This is the static-batch approach to the **state estimation problem**, on the other hand - as an alternative technique - if the system is observable we can estimate $x(k)$(then $x(0)$) dynamically by constructing a device called the **Observer**, in the deterministic case it is called **Luemberger Observer**.

4.1 Review of Luemberger Observer

A copy of the system (4.1) is made with the only difference of adding a correction term. Starting from this point we use $\hat{x}(k)$ to indicate the **estimate of the state at the time** k (discretized time), and $\hat{y}(k)$ is the output computed by using the estimate. The Luemberger Observer has the following equations:

$$\begin{aligned} \hat{x}(k+1) &= A\hat{x}(k) - L[\hat{y}(k) - y(k)] \\ \hat{y}(k) &= C\hat{x}(k) \end{aligned} \tag{4.2}$$

The quantity $e(k) = \hat{x}(k) - x(k)$ is the error of the estimate at time k, the aim is to design an online algorithm which could make $e(k) \to 0$.
In order to understand the role of the matrix $L \in \mathbb{R}^{n,q}$, called the **observer gain matrix**, we can write:

$$\begin{aligned} e(k+1) = \hat{x}(k) - x(k) &= A\hat{x}(k) - L[\hat{y}(k) - y(k)] - Ax(k) = && (4.3) \\ &= A\hat{x}(k) - LC\hat{x}(k) - LCx(k) - Ax(k) = && (4.4) \\ &= A[\hat{x}(k) - x(k)] - LC[\hat{x}(k) - x(k)] = && (4.5) \\ &= (A - LC)[\hat{x}(k) - x(k)] = && (4.6) \\ &= (A - LC)\ e(k) && (4.7) \end{aligned}$$

The resulting equation $e(k+1) = (A - LC)\, e(k)$ describes an LTI discrete time dynamical system in which the state matrix is represented by $A - LC$. From the theory of dynamical systems, we know that the system is asymptotically (internally) stable if after a certain time k, it is verified that $e(k) \to 0$, the result we are seeking, it is verified when the *eigenvalues* of $A - LC$ are in the unitary circle.

Regarding the matrix A it is not very interesting to track $x(k)$ if it is asymptotically stable because in this situation $\lim_{k\to\infty} x(k) = 0$. So A is required to be stable but not asymptotically (some authors refers to this type of stability as **marginal stability**).

Theorem If the system is observable, then there exists L such that $A-LC$ is asymptotically stable. (We can drive the error to zero in order to track the state of the system).

4.2 State estimation by Least-squares approach

At a certain time k, given the current measurement $y(k) = Cx(k)$, we might estimate $x(k)$ by solving the following problem:

$$\hat{x}(k) = \arg\min_{x\in\mathbb{R}^n} \frac{1}{2}\|y(k) - Cx\|_2^2 \tag{4.8}$$

if $q > n$ and C is full rank. We call $\mathcal{F}(x) = \frac{1}{2}\|y(k) - Cx\|_2^2$ the Least-Squares functional. There is a problem: the (pseudo)inversion of the matrix C, could be non-trivial for a medium-large dimensional problem. Even the classical Gradient Descent Algorithm would be too slow!
A solution is: at each k, we run a **single step** of gradient descent resulting in an **Online gradient descent**.

Online Gradient Descent (OGD)
Given the measurement $y(k) = Cx(k)$ and $\hat{x}(k)$ computed before time k

$$\hat{x}^+(k) = \hat{x}(k) - \tau\nabla\mathcal{F}(\hat{x}(k)) = \hat{x}(k) - \tau C^T[C\hat{x}(k) - y(k)] \tag{4.9}$$
$$= \hat{x}(k) - \tau C^T\,[\hat{y}(k) - y(k)] \leftarrow \text{estimate of } x(k) \tag{4.10}$$
$$\hat{x}(k+1) = A\hat{x}^+(k) \leftarrow \text{prediction of } x(k+1) \tag{4.11}$$
$$\hat{y}(k) = C\hat{x}(k) \tag{4.12}$$

By merging estimate and prediction we obtain:

$$\hat{x}(k+1) = A\hat{x}(k) - \tau AC^T[\hat{y}(k) - y(k)] \tag{4.13}$$

It can be noted that there is a certain similarity of the system (4.13) and the (4.2), in particular the OGD is a Lumberger Observer with $L_g = \tau AC^T$.
Since we have fixed, in a certain way, the matrix L_g, one would wonder when

$$A - L_gC \tag{4.14}$$

is asymptotically stable. We can rewrite it as $A(I - \tau C^TC)$. If we choose $\tau < \frac{1}{\|C\|_2^2}$ then we obtain that $\|I - \tau C^TC\| \leq 1$. Finally, two cases are to be considered:

- If $\|I - \tau C^TC\| = 1$, and A is *marginally stable*, then $A - L_gC$ is marginally stable;
- If $\|I - \tau C^TC\| < 1$, and A is *marginally stable*, then $A - L_gC$ is **asymptotically stable**.

Until this moment, we have presented these results for an LTI DT dynamical system in which there are not attacks. What about **Dynamic Secure State Estimation**?

4.3 Dynamic SSE with constant attack

Recalling that a CPS under adversarial attacks on the sensors can be described by the system:

$$\begin{aligned} x(k+1) &= Ax(k) \\ y(k) &= Cx(k) + a(k) \end{aligned} \tag{4.15}$$

We have seen in the Second Chapter that in this case the problem of observability results in:

$$\begin{pmatrix} y(0) \\ \vdots \\ y(T-1) \end{pmatrix} = \mathcal{O}_T x(0) + \begin{pmatrix} a(0) \\ \vdots \\ a(T-1) \end{pmatrix} \tag{4.16}$$

We have solved this problem for the static case in which we have seen the IST Algorithm, but in the case in which A is not the identity matrix, the problem is not trivial to solve!
If an assumption is done on the 'shape' of the attacks the problem could be well posed, in particular we should assume that the attacks are constant and equal to a vector $a \in \mathbb{R}^q$, at this point the problem (4.16) results in:

$$\begin{pmatrix} y(0) \\ \vdots \\ y(T-1) \end{pmatrix} = \mathcal{O}_n x(0) + \begin{pmatrix} a \\ \vdots \\ a \end{pmatrix} = \begin{pmatrix} C & I \\ CA & I \\ \vdots & \vdots \\ CA^{T-1} & I \end{pmatrix} \begin{pmatrix} x(0) \\ a \end{pmatrix} \tag{4.17}$$

where the matrix

$$\mathcal{O}'_T = \begin{pmatrix} C & I \\ CA & I \\ \vdots & \vdots \\ CA^{T-1} & I \end{pmatrix} \tag{4.18}$$

is an **augmented observability matrix**. Is this CPS observable?

In order to clarify this aspect, let us consider a couple of measurements for $k = 0, 1$:

$$y(0) = Cx(0) + a \tag{4.19}$$
$$y(1) = Cx(1) + a = CAx(0) + a \tag{4.20}$$

We might manipulate algebraically these equations in order to eliminate the attack a which is assumed to be constant. For example, one can subtract the (4.20) from the (4.19), and it will be obtained

$$y(1) - y(0) = CAx(0) + a - Cx(0) - a = \tag{4.21}$$
$$[CA - C]x(0) = \tag{4.22}$$
$$C[A - I]x(0) \tag{4.23}$$

Moreover, let us suppose that $q = n$ so that the matrix $C[A-I] \in \mathbb{R}^{n,n}$. If such matrix would be invertible, we could recover $x(0)$ without problem by inverting the equation (4.23). One might think that I could go further in the computation of $y(k)$ and by using such manipulations, I can eliminate the attack and recover without problems the state.
BUT in many situations the square matrix $A - I$ is not invertible, we have seen that is reasonable that in the Spec(A) (set of the eigenvalues of A) there is an eigenvalue $\lambda_i = 1$ for some i. This imply that the matrix $A - I$ has a **null eigenvalue**, that is the same to confirm

that the matrix is **not full rank** and for this reason **non invertible**. One wonder if we might have a generalization of this concept. It is possible by analysing the **kernel of** $\mathcal{O}'_T$. To this aim, again, we will exploit some algebraic tricks. This time we subtract couple of rows of the matrix (4.18) from the bottom to the top, obtaining:

$$\begin{pmatrix} C & I \\ C(A-I) & I \\ \vdots & \vdots \\ CA^{T-3}(A-I) & 0 \\ CA^{T-2}(A-I) & 0 \end{pmatrix} \begin{pmatrix} x \\ a \end{pmatrix} \tag{4.24}$$

This is nothing but the system (4.16) rewritten in a different form. Let us neglect at the moment the first row of the rewritten matrix. It is recognizable the matrix $\mathcal{O}_{T-1}$, however due to the fact of being multiplied by $A - I$ the linear system

$$\mathcal{O}_{T-1}(A-I)x = 0$$

is underdetermined and has got infinitely many solutions. Despite $\mathcal{O}_{T-1}$ is full rank (it is a minimum requirement because if the system without attack is not observable, I cannot recover the state with attacks!) we do not have a trivial kernel because of $A - I$ which is not full rank. Moreover if we add the first equation we obtain the total system

$$\begin{cases} (A-I)x = 0 \\ Cx + a = 0 \end{cases}$$

We are ready to give the following proposition:

Proposition If the matrix A has an eigenvalue $= 1$, the dynamic CPS with constant attack is not observable.

The fact that an eigenvalue of A might be equal to one, is quite common from the moment we do not desire the situation in which the state tends to zero when $k \to \infty$.
Then, the proposition states that in general a CPS under attacks **is not observable** even if the attack is **constant**. However, we have not exploited yet the information about the **sparsity of the attacks** which allows us to develop a so-called **SPARSE OBSERVER**.
Before giving the final result is useful to give a little of notation:

$$z(k) = \begin{pmatrix} x(k) \\ a(k) \end{pmatrix} \quad \hat{z}(k) = \begin{pmatrix} \hat{x}(k) \\ \hat{a}(k) \end{pmatrix} \quad \hat{z}^+(k) = \begin{pmatrix} \hat{x}^+(k) \\ \hat{a}^+(k) \end{pmatrix} \quad G = \begin{pmatrix} C & I \end{pmatrix}$$

We want to solve the problem of recover the state of the dynamic CPS under constant attacks that is to solve:

$$\min_{x \in \mathbb{R}^n, a \in \mathbb{R}^q} \frac{1}{2} \|y(k) - Gz(k)\|_2^2 + \lambda \|a\|_1$$

It can be demonstrated that after a sufficient number T of steps the solution is given by the following algorithm:

SPARSE OBSERVER

Given $y(k) = Gz(k)$ and $\hat{z}(k)$,

$$\begin{aligned}
\hat{z}^+(k) &= \hat{z}(k) - \tau G^T[G\hat{z}(k) - y(k)] && \leftarrow \text{estimate of } z(k) \\
\hat{x}(k+1) &= A\hat{x}^+(k) && \leftarrow \text{prediction of} x(k+1) \\
\hat{a}^(k+1) &= \mathbb{S}_{\tau\lambda}[\hat{a}^+(k)] && \leftarrow \text{"sparsify" the attacks} \\
\hat{y}(k) &= G\hat{z}(k)
\end{aligned}$$

What are the differences from the previous version? The matrix A is not the identity matrix, the state of the CPS **changes** $\Rightarrow$ in general $x(k) \neq x(k+1)$.

Chapter 5

Distributed Consensus based algorithms for CPSs

5.1 Toward the Fusion Center removal

5.2 The Consensus algorithm

5.3 Uses of the Consensus algorithm

Part II

Control of Cyber-Physical systems

Chapter 6

System identification: an introduction

Chapter 7

Set-membership Identification

Chapter 8

State variable feedback control (SVFB)

Chapter 9

Formation contorl