NON LINEAR CONTROL AND AEROSPACE APPLICATIONS

Lecture notes

Carlo Migliaccio

AA 2023/2024

Contents

Part I

Non linear modeling and control

Chapter 1

Introduction

1.1 Dynamic Systems, Variables, Signals

A **Dynamic System**, roughly speaking, is a collection of interacting object which evolve over the time. Some examples could be: vehicles, mechanical systems, spacecrafts etc.
The *time evolution* of these systems can be described by quantities called **variables**. The functions which describe the evolution over the time of such quantities play an important role and they are called **signals**.
Among all variables there are some of them which are particularly important such as:

- **Input** $u(t)$: variables that influence the time evolution of the system, the so called causes. We can distinguish two types of input:
  1. **Command input**: a human user can choose it;
  2. **Disturbance**: as their presence is not under the control of a human, they can't be chosen.
- **Causes** $y(t)$: measured variables;

1.2 General considerations

In the area of automatic control and dynamic systems, is important to distinguish between:

- **Physical system** (which we call the **plant**);
- The **model**: can be defined as a mathematical description of the variables.

A model is important for several reasons: to make a prediction, filtering, analyze a system, control it... Despite these important aspects, we ought to remember that *models represent an approximation of the real system.* For this reason we have to deal with uncertainty of two types:

1. **Parametric**: caused by approximation of the parameters;

Figure 1.1: Rappresentazione a blocchi di un sistema dinamico

2. **Dynamic**: caused by the fact of overall some details in the description of the model which rarely is exactly known and so easily descripted.

To deal efficiently with *uncertainty* we need **robust** techniques for modeling and control such types of systems.

1.3 State equations

A **dynamic system** can be always described by a **set of first order differential equations**:

$$\dot{x} = f[x(t), u(t), t] \tag{1.1}$$
$$y(t) = h[x(t), u(t), t] \tag{1.2}$$

where:

- $x(t) \in \mathbb{R}^{n_x}$ is the **state**;
- $u(t) \in \mathbb{R}^{n_u}$ is the **input**;
- $y(t) \in \mathbb{R}^{n_y}$ is the **output**;
- n_x is the **order of the system**, is important because it tells us how many variables we need to describe the system.

This description is standard and sufficiently **general** to represent any dynamical system. Moreover it is used by many *control design method.*

Interpretation of the state equation

Since the state equation

$$\dot{x} = f[x(t), u(t), t]$$

is **dynamic** and it can be written in local form as

$$x(t + dt) = x(t) + f[x, u, t]dt$$

it is clear that locally it can have the following meaning: the state in a time $t + dt$ is equal to the current state $x(t)$ to which we add a variation fdt.
Differently the **output equation** is **static** in the sense that it represents a relation between variables at the same time t.
In order to concluding this discussion we say that in general the functions f and g are *vector field.*

1.4 Classifications of dynamical systems

According to the characteristic we observe of a dynamic system there could be several classifications. We can mention the most important:

- **Non Linear systems**: here the functions $f(x, u, t)$ and $h(x, u, t)$ are not linear, the great majority of the systems in nature **are non linear!**

- **Linear systems**: here we can write the equations previosly seen in the form:

$$\dot{x}(t) = A(t)x(t) + B(t)u(t)$$
$$y(t) = C(t)x(t) + D(t)u(t)$$

here *f, h* are linear while the matrices $A(t), B(t), C(t), D(t)$ are possibly time varying; if they are constant we identify a particular (very important) case which is the Linear Time Invariant system (**LTI**);

- **Time varying (Time Invariant) systems** here we have:

$$\frac{\partial f}{\partial t} \neq 0 \quad \frac{\partial g}{\partial t} \neq 0$$
$$(\frac{\partial f}{\partial t} = 0 \quad \frac{\partial g}{\partial t} = 0)$$

The functions f, h (do not) explicitly depend on the time;

- **Continuous time/Discrete Time** in the former case we have that the variable t is a real positive number, that is $t \in \mathbb{R}^+$, in the latter case we have that the variable t (also indicated with k) is a natural number, that is $t \in \mathbb{N}$;

- The last classification is based on the **number of inputs and outputs** (n_y, n_u). Specifically we have:

  1. **SISO** (Single input, Single output) $\rightarrow n_u = 1, n_y = 1$
  2. **MIMO** (Multiple input, Multiple output) $\rightarrow n_u > 1, n_y > 1$
  3. **SIMO** (Single input, Multiple Output)$\rightarrow n_u = 1, n_y > 1$
  4. **MISO** (Multiple input, single Output)$\rightarrow n_u > 1, n_y = 1$

Chapter 2

Stability concepts

Stabilty is a very important notion we would like analyzing of a system in order to study his behaviour over the time.

2.1 Some definitions

Let us consider the system $\mathcal{S}$ which has state equation equal to

$$\dot{x(t)} = f[x(t), u(t), t]$$

with input signal $u(t)$ and **initial conditions** $x(0)$.

Definition *(Solution of a system)* The signal resulting from the system integration is a **function of the time** $x(t), \quad t \geq 0$ which is called the **solution** of the system **corresponding to** the initial state $x(0)$ and the input $u(t)$.

Definition *(Trajectory of a system)* The **set of the points of the state space generated by the solution** is called a **trajectory of the system** corresponding to the initial state $x(0)$ and the input $u(t)$.

Let $x(t)$ be the *solution* of the system with the respecto to the couple (initial condition, input) equal to $(x(0), u(t))$, we call it the **nominal solution**. If we change a bit the initial state to $x^p(t) \neq x(t)$ but with the same input signal we have the **perturbed solution**.

2.1.1 Marginal Stability

Definition *(Marginal Stability)* The solution $x(t)$ is **marginally** (or simply) **stable** if

$$\forall \epsilon > 0, \exists \delta > 0 : \forall x^p(0) : \|x^p(0) - x(0)\| < \delta \Longrightarrow \|x^p(t) - x(t)\| < \epsilon, \quad \forall t \geq 0$$

[Roughly speaking we say that the perturbed solution is always near the nominal solution and so, using other words, for any choice of ϵ, I am able to find a δ such that if I choose a perturbed state within the ball of radius δ for each time t, the perturbed solution "fall" within the ball of radius *epsilon*.]

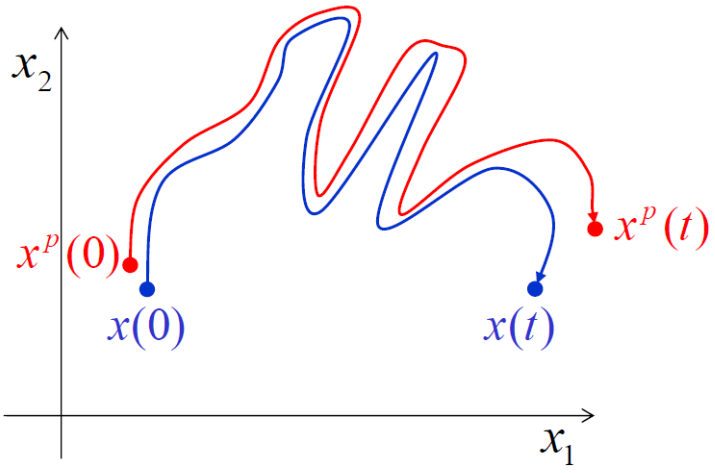


Figure 2.1: Nominal solution and Perturbed solution

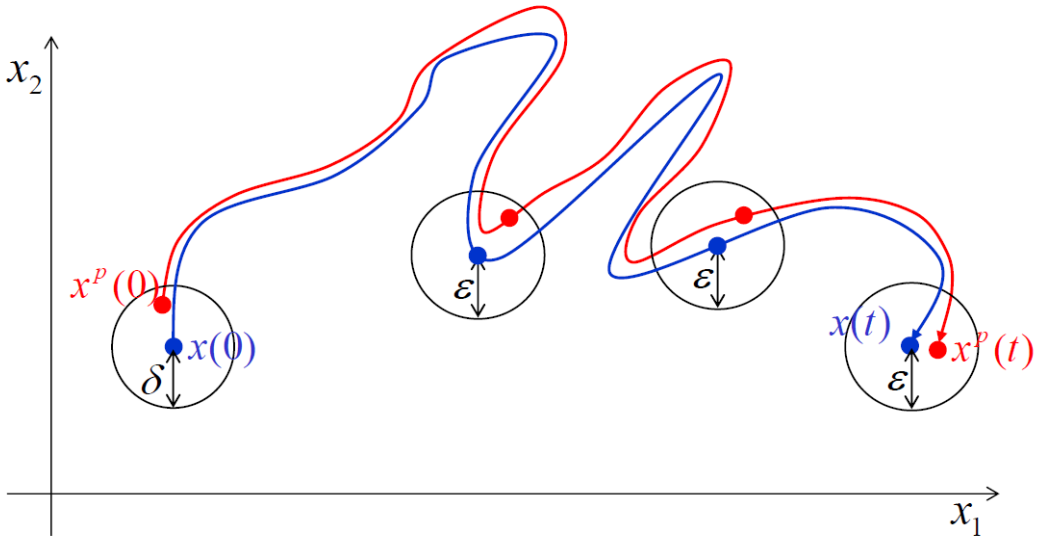


Figure 2.2: Marginal stability

2.1.2 Asimptotical Stability

Definition *(Asimptotical stability)* The solution $x(t)$ is **asimptotically stable** if is stable and $\lim_{t\to\infty}\|x^p(t) - x(t)\| = 0$. Moreover if the convergence is **exponential**, the solution is **exponentially stable**. (This concept is also linked to *modal analysis*.)

2.1.3 Unstability

Definition *(Unstability)* A solution of a system is **unstable** if it is not stable. (Despite I choose a perturbed initial condition which could be even very close to $x(0)$, the perturbed solution $x^p(t)$ **go far away** from $x(t)$.

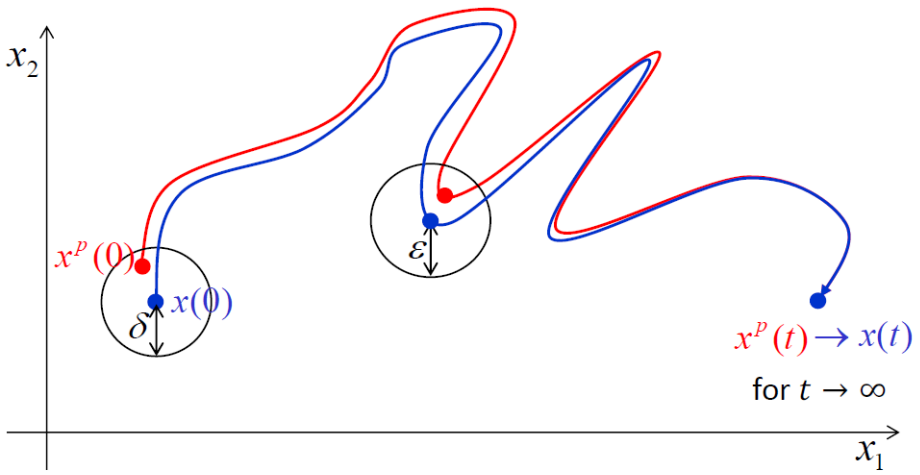


Figure 2.3: Asimptotical stability of a trajectory

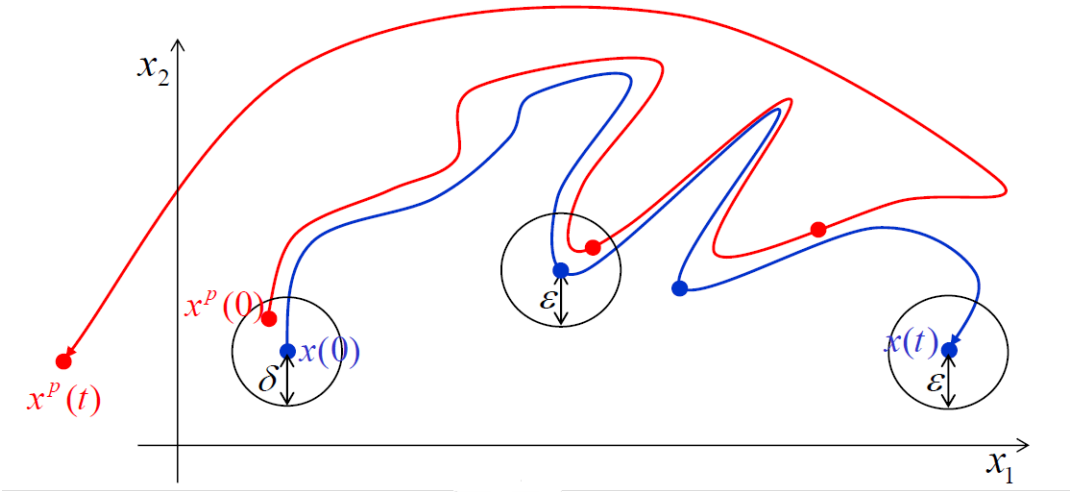


Figure 2.4: Unstable solution

2.2 Equilibrium point of a system

2.2.1 Equilibrium point

Given a system with the state equation $\dot{x} = f(\bar{x}, \bar{u})$, given a constant input $\bar{u}$, $\bar{x}$ is called an **equilibrium point or state** corresponding to the constant input $\bar{u}$, if it is solution of the equation

$$f(\bar{x}, \bar{u}) = 0$$

So: $f(\bar{x}, \bar{u}) = 0 \iff \dot{x} = 0 \iff$ null variation over the time $\iff$ the state remains constant

2.2.2 Stability of the equilibrium point

Definition (Stability) The equilibrium point $\bar{x}$ is stable if:

$$\forall \epsilon, \exists \delta : \forall x(0) : \|x(0) - \bar{x}\| \leq \delta \Longrightarrow \|x(t) - \bar{x}\| \leq \epsilon, \quad \forall t \geq 0$$

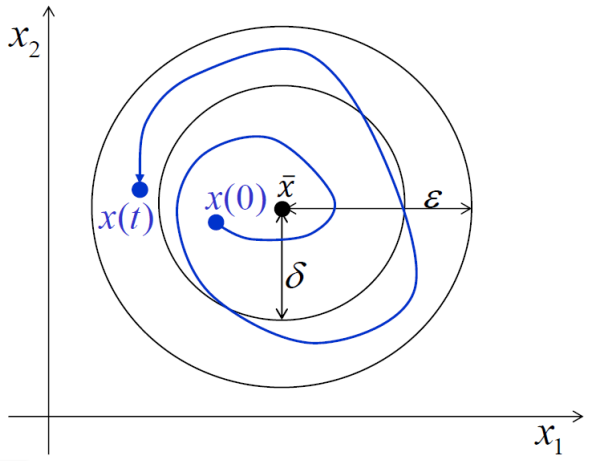


Figure 2.5: Stable equilibrium point

A **simpler explanation** of this concept is that if we move from the equilibrium point choosing a $x(0)$ within a certain ball of radius δ, the equilibrium point itself is stable if the trajectory $x(t)$ remains close to the equilibrium point within another ball of a certain radius ϵ.

To find the equilibrium point is sufficient to solve the equation $f(\bar{x}, \bar{u}) = 0$. Sometimes is quite easy to solve it (e.g. the case of the pendulum), but many times could be not trivial to find an analytical solution.

Definition (Asymptotic Stability) The equilibrium state $\bar{x}$ is **asymptotically stable** if it is stable and it holds that:

$$\lim_{t \to \infty} \|x(t) - \bar{x}\| = 0$$

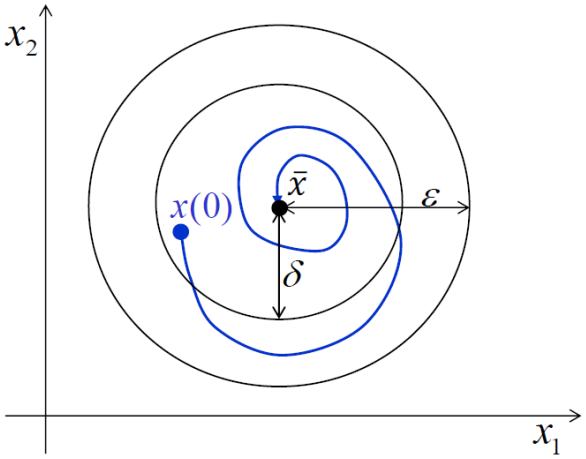


Figure 2.6: Asymptotically stable equilibrium point

Definition (Instability) The equilibrium state $\bar{(x)}$ is unstable if it is not stable. That is, Despite we remain close to the equilibrium point the trajectory goes far away the neighbourhood of the point itself.

Definition (Domain of attraction) **A domain of attraction** is a (generic) set of points such that the trajectories initiated at these point converge to the *equilibrium point*. **The domain**

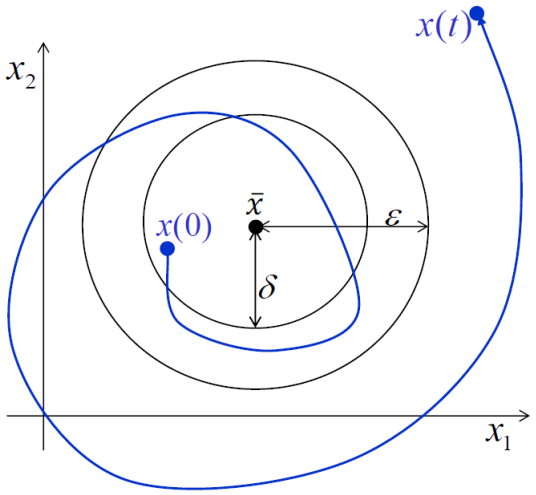


Figure 2.7: Unstable equilibrium point

of attraction is the set of **all points** such that if we start from these point we reach the equilibrium point. If asymptotic stability holds for all initial conditons, then the equilibrium point is **stable in the large** or **globally** stable.

2.3 Particular case: Stability of LTI systems

In the case of a Linear Time-Invariant (LTI) system we can write the state solution as $\dot{x}(t) = Ax(t) + Bu(t)$. In this particular case the stability is for the system, not only for a trajectory. For any constant input $\bar{u}$ the equilibrium states can be found **solving respect to** $\bar{x}$ the equation

$$\dot{x} = A\bar{x} + B\bar{u} = 0 \longleftrightarrow A\bar{x} = -B\bar{u}$$

If $det(A) \neq 0$, then the solution is unique otherwise we can have infinite solutions which generate a subplane (e.g. a line). In the case that $\bar{u} = 0$ we obtain a subspace.
If $\lambda_1, ..., \lambda_i$ are the **eigenvalues** of A and $k_1, ..., k_i$ are their **auxiliary geometric multiplicities**, holds that:

Theorem The LTI system is (internally):

1. asymptotically stable stable if and only if $Re(\lambda_i) < 0, \forall i$
2. marginally stable if and only if $Re(\lambda_i) \leq 0, k_i = 1 if Re(\lambda_i) = 0$
3. unstable if and only if $\exists i : Re(\lambda_i) > 0 \quad \text{or} \quad \exists i : Re(\lambda_i) = 0, k_i > 1$

In the case of *non linear systems* the results which has just seen have to be applied to **each equilibrium point**. It is also of interest to analyze a general property of stability which is the **BIBO stability** (Bounded Input - Bounded Output), that refers to any type of system of the form

$$\dot{x} = f[x, u, t]$$
$$y = h[x, u, t]$$

Definition (BIBO stability) A system descripted by the standard state equation is **BIBO stable** if **for any bounded initial condition** and **for any bounded input** such that $\|u(t)\| \leq M_u < \infty, \forall t \geq 0$ the resulting output is also bounded, that is $\|y(t)\| \leq M_y, \forall t \geq 0$
In order to conclude this chapter, we can mention some observation that come from the *modal analysis*:

- asympotic stability $\Longrightarrow$ BIBO stability
- BIBO unstability $\Longrightarrow$ instability or marginal stability

An example of system that is (simply) stable but NOT BIBO stable is the integrator descripted by the equation

$$\dot{x}(t) = u(t), \quad y(t) = x(t), x, y, u \in \mathbb{R}$$

If we applied a **constant input** the result $\mathbf{y(t)}$ is a ramp. Moreover the BIBO stability is also called **external stability**.

Chapter 3

Behaviour of Non Linear systems

The **behaviours** of non linear systems can be differ a lot from the linear one. The first difference we observed was that while for an LTI system we can say that **the system is stable** (stability as a property of the system), for non linear systems we talk about **the stability of a trajectory (or solution)** (stability as a property of a particular solution $x(t)$ of the system). The phenomena which occurs in the case of non linear system are for example:

- multiple isolated equilibrium points (see the example)
- limit cycles
- bifurcations
- finite escape time
- chaos
- ...

3.1 Phase plane portrait

We focus in this part on **Second-Order systems**, that are those for which $n_x = 2$. The **phase plane analysis** is a method for studying the properties of such systems, for higher dimensions we can use a sort of generalization represented by the *Poincare Maps*.
The analysis we conduct reguard the **free evolution** of the system, and it is done in order to understand the most intrinsic properties of the system itself.

Definition (Phase Portrait, in Italian "Ritratto di fase") Let us consider an *autonomous system* with state $x = (x_1, x_2) \in \mathbb{R}^2$: $\dot{x}(t) = f(x)$, but we can write it also as:

$$\dot{x_1} = f_1(x_1, x_2)$$
$$\dot{x_2} = f_2(x_1, x_2)$$

For any initial condition $x(0)$ the integration of the equation generates a solution and consequently a trajectory. If we repeat the integration for different $x(0) = x_0$ we obtain a family of trajectories which is called **phase portrait**.

Example (Mass-Spring-Damper)
Let us consider a **mass-spring-damper** system with $F = 0, \quad \beta = 0, \quad m = 1, \quad k = 1$. The dynamic equation of the system is $\ddot{p} + p = 0$, it has solution: $p(t) = p_0 \cos t$ and $\dot{p}(t) = -p_0 \sin t$.
We have in this case $x_1 = p(t)$ and $x_2 = \dot{p(t)}$.
We can determine a well known relation between the two variables that we can plot in the

phase plane or *state domain*. This equation is: $p^2 + \dot{p}^2 = p_0^2$ which defines a circle of radius p_0 in the phase plane. Varying p_0 we obtain the **phase portrait** representation.

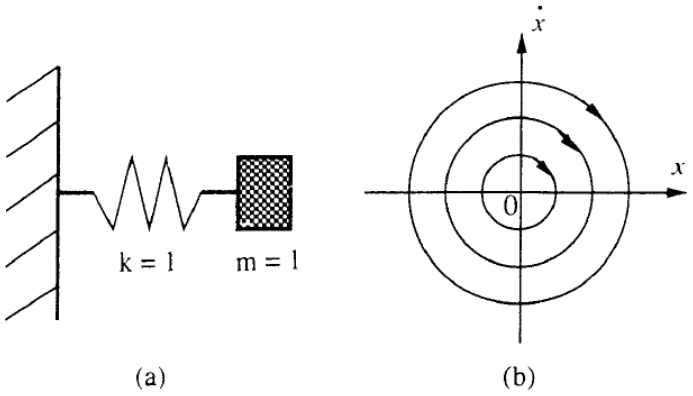


Figure 3.1: Mass-Spring-Damper and its *phase portrait*

3.1.1 Singular Points

Given the system $\dot{x} = f(x)$ with $x \in X \subseteq \mathbb{R}^2$. We define the **slope** of a trajectory at a point $x = (x_1, x_2)$ as:

$$\frac{dx_2}{dx_1} = \frac{f_2(x_1, x_2)}{f_1(x_1, x_2)}$$

At this point we can distinguish **two cases**:

- $f_2(x_1, x_2) \neq 0, f_1(x_1, x_2) \neq 0$ in this case the slope is **well defined** and the trajectories won't intersect in the phase plane;
- $f_2(x_1, x_2) = f_1(x_1, x_2) = 0$ we can't define the slope that is **undetermined**, and so some trajectories intersect at point $x = (x_1, x_2)$. We call this point **singular (or fixed) point**.

Important! As in the case of the *singular points* we assume that $\dot{x_1} = f_1 = 0$ and $\dot{x_2} = f_2 = 0$ follows that the points for which this happen, are also **equilibrium points**. They are stable if all trajectory **converge** to this equilibrium point.

Some considerations

- The Figure 3.2 cannot be the phase portrait of an LTI system: in that case we have either one or infinity equilibrium points. This highlight that in the non linear case there can exist few isolated equilibrium points;
- In the figure there are two equilibrium point: one is *stable* another is *unstable*.
- The figure is the phase portrait linked to the following example:

It is interesting to note that a phase portrait might by drawn even for a **first order system**, the difference is that we would have only one trajectory on which we can choose the initial condition. We show here an example (from book "Applied non linear control" - Slotine, 1996):

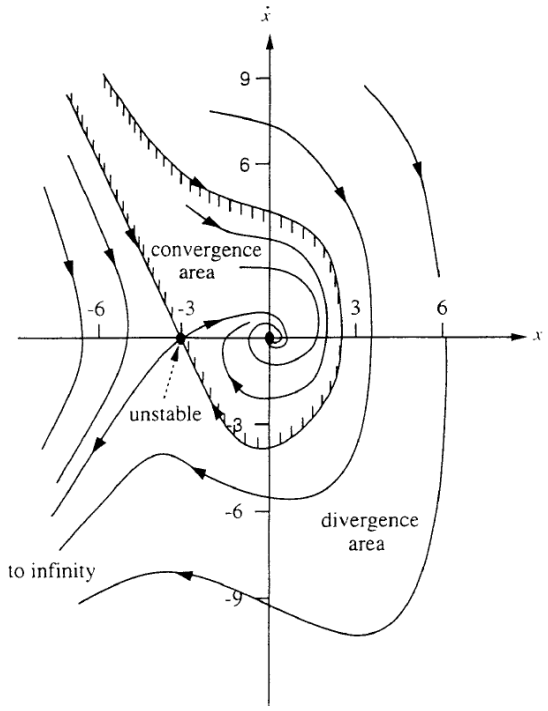


Figure 3.2: The phase portrait of a Non Linear System

Consider the system

$$\ddot{x} + 0.6\,\dot{x} + 3\,x + x^2 = 0$$

whose phase portrait is plotted in Figure 2.2. The system has two singular points, one at $(0, 0)$ and the other at $(-3, 0)$. The motion patterns of the system trajectories in the vicinity of the two singular points have different natures. The trajectories move towards the point $x = 0$ while moving away from the point $x = -3$. □

3.1.2 Methods to draw phase portrait

There are mainly **two methods** for constructing the *phase portrait*:

1. **Analytical/Geometrical methods**: they can be applied only in particular cases;
2. By using **Numerical simulation**: can be applied in any case, moreover they can be generalized to system of order higher than 2.

3.2 Behaviour of LTI systems

Studying the phase portrait of **LTI systems** is important because: i) we can have an idea of their behaviour, ii) We can use them to study locally the (continuous) non-linearities. We are talking about systems of the form $\dot{x} = Ax$ which have quite simple properties:

1. An LTI system has a **unique equilibrium point** if the matrix A is non-singular, such point is stable if the eigenvalues of the A matrix have strictly negative real part, *regardless the initial conditions*;
2. The solutions can be computed analitically in an easy way;

Example 2.3: A first-order system

Consider the system

$$\dot{x} = -4x + x^3$$

There are three singular points, defined by $-4x + x^3 = 0$, namely, $x = 0, -2$, and 2. The phase-portrait of the system consists of a single trajectory, and is shown in Figure 2.3. The arrows in the figure denote the direction of motion, and whether they point toward the left or the right at a particular point is determined by the sign of $\dot{x}$ at that point. It is seen from the phase portrait of this system that the equilibrium point $x = 0$ is stable, while the other two are unstable. □

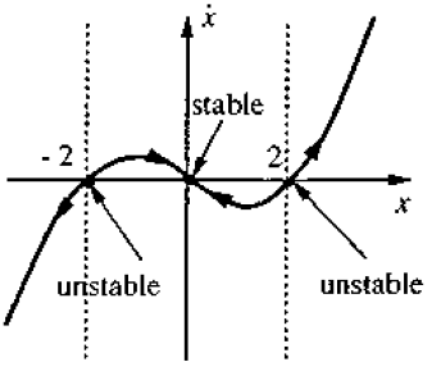


Figure 2.3 : Phase trajectory of a first-order system

Figure 3.3: Example of first order phase portrait

3. In presence of an input $ut(t) \neq 0$ the system becomes of the form $\dot{x(t)} = Ax(t) + Bu(t)$ and we can observe that:
   (a) We can apply the **superposition princple** ;
   (b) The asimptotic stability leads to BIBO stability
   (c) If $u(t) = A\sin(\omega t + \phi)$, the output is a sinusoidal signal of the same frequency and (Magnitude, Phase) depending on the transfer function of the system itself.

In order to give consider the case when $A \in \mathbb{R}^{2,2} \quad x \in \mathbb{R}^2$, the system has got **two eigenvalues** λ_1, λ_2. Actually, we can distinguish the following cases:

1. $\lambda_{1,2}$ **with the same sign (node)**, if they are both negative, all solution converge otherwise all diverge;
2. $\lambda_{1,2}$ **with opposite signs (saddle)**, in this case as we have some positive, some negative eigenvalue $\Rightarrow$ some solutions converge other diverge;
3. $\lambda_{1,2}$ **complex conjugate with non-zero real part (focus)** in this case if $\mathrm{Re}(\lambda_{1,2}) < 0$ the solutions converge, if $\mathrm{Re}(\lambda_{1,2}) > 0$, the solutions diverge.
4. $\lambda_{1,2}$ **complex conjugate with zero real part (center)** some trajectories are **ellipsis** as they are composition of harmonic signals;

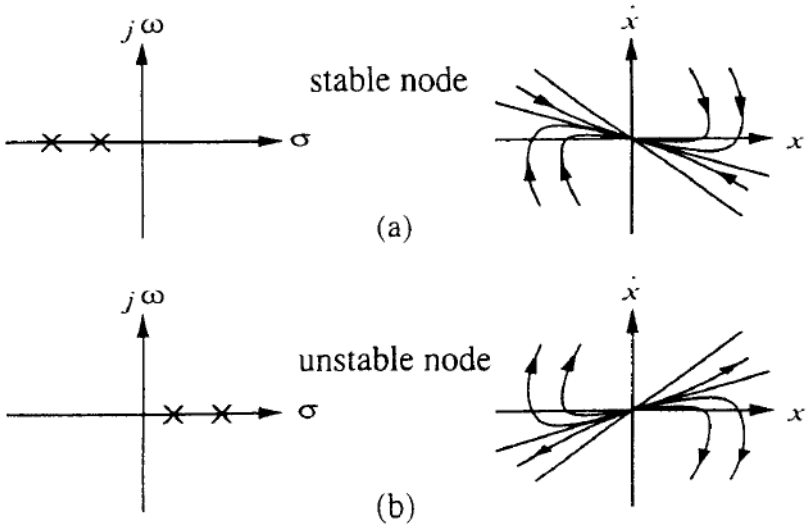


Figure 3.4: Stable/Unstable Node

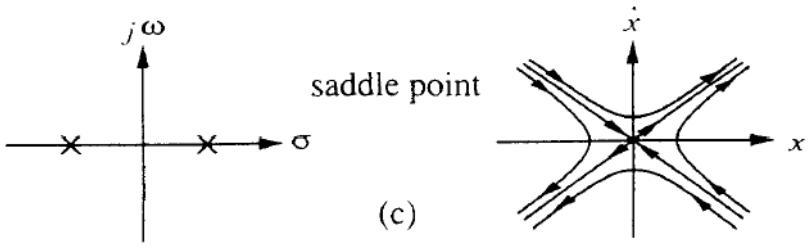


Figure 3.5: Saddle point

3.3 Non linear case

The **behaviour of non linear systems** is much more complex, due to the lack of linearity and superposition principle, they respond to external inputs in a different way. Next we enumerate the most common behaviours: multiple equilibrium points, limit cycles, bifurcations, finite escape time, chaos...

3.3.1 Multiple isolated equilibrium points

Some non linear systems (e.g. the pendulum) have multiple equilibrium points which are isolated, that is in the vicinity there no other equilibrium points. As we said before in the case of LTI systems there only two possibilities: (1) **a single equilibrium point**, (2) infinitely many equilibrium points in the case that $\det(A) = 0$.

3.3.2 Limit cycles

A **limit cycle** is an **isolated closed curve** on which the motion is *periodic*. A famous example is the **Van Der Pol oscillator** which is substantially a mass-spring-damper system with a non linear damper.
Limit cycles are unique properties of non linear systems, the limit cycle has to be both:

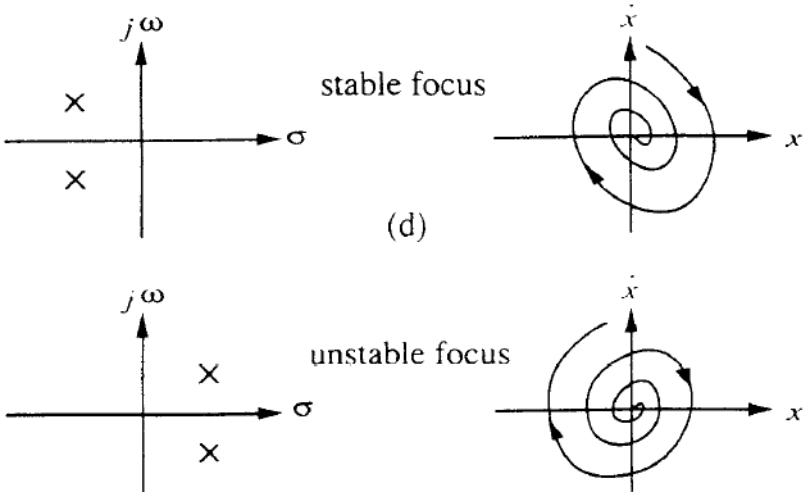


Figure 3.6: Caption

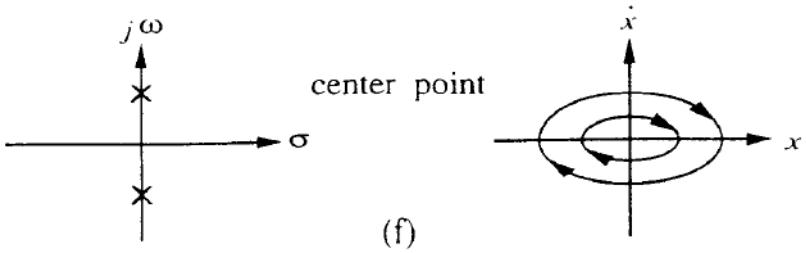


Figure 3.7: Caption

1. **closed**, this feature indicate the periodic motion
2. **isolated**, with nearby trajectories that converge or diverge from it.

Depending on the motion of the trajectories in the vicinity of the limit cycle, it can be called:

- **Stable limit cycle**: all trajectories in the "neighbourhood" converge to it → we have an **ATTRACTOR**;
- **Unstable limit cycle**: all trajectories diverge from the limit cycle → we have a **REPELLOR**;
- **Semi-Stable limit cycle**: some trajectories converge to it, other diverge → we have a **SADDLE**;

The limit cycle of a *Van Der Pol* oscillator is a stable one.
Limit cycles represent an important phenomenon in non linear systems as they can be encountered in many engineering applications and in nature. To give an example in the Aerospace field a limit cycle caused by the interaction between *aerodynamic forces* and *structural vibrations* can be very dangerous. Other times a limit cycle could be desirable! An engineer has to know **how to eliminate them** when undesirable, how to **create/amplify** them when they give some advantages. There are some **simple theorems** that provide a method to individuate them by analyzing some mathematical properties of the system.

Tori

When the order of the system is that $n_x > 3$, than the motion in the state space can occur on a Torus which constitute a generalization of non-isolated cycle (pendulum) or limit cycles in

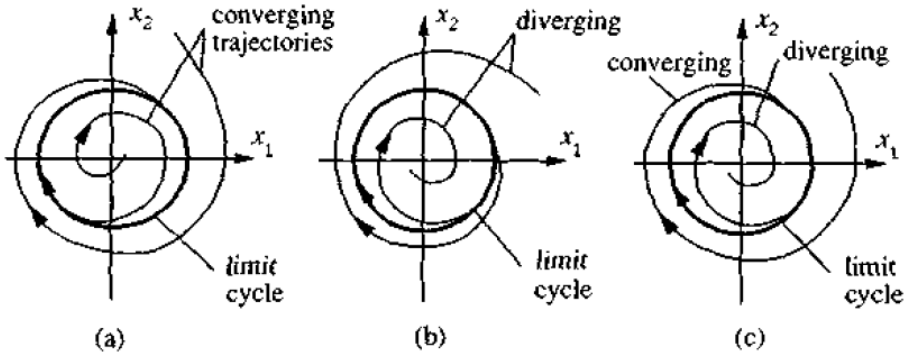


Figure 3.8: Types of **limit cycles**

higher dimension, for this reason, it can be an attractor, repellor or saddle.

3.3.3 Bifurcations

When the **physical parameters** of a system change, can change also the equilibrium points combined with their stability properties. The values of the parameters (α) in general for which this variations occurs are called **critical** or **bifurcation values**.
The **bifurcation**, i.e. the **quantitative change of the parameters** resulting in a **qualitative change of the system** is another property which characterize non linear systems.
Common examples are **pitchfork** and **Hopf** bifurcations.

3.3.4 Finite escape time

Is a phenomenon by which some states diverge in a finite time we call t_esc, that is equivalent to state that $\lim_{t\to\infty} x(t) = \infty$.

3.4 Chaotic systems

For LTI systems, small differences in initial conditions can cause small differences in the output. For non linear systems is different: there are many cases in which a **small difference in the initial conditions** is able to lead to **unpredictable behaviours**, this phenomenon is called **chaos**. It is useful to distinguish chaos from random motion, in the case of *chaos* the problem is deterministic and there is a small uncertainty in the initial conditions.
Chaotic behaviours can be observed in many physical systems (e.g. the **atmosphere model** can be described like this), the most common is the **turbulence** in fluid mechanics.
In the context of **feedback control** it is important to recognize when a system can enter in a chaotic behaviour, and techniques to recover the system itself from it are required. The motion on these systems can occur over geometric entities with **fractal structures** called **strange attractors**, besides you can have **strange repellors** and **strange saddles**. Their dimensions are measured by using the **Housdorff dimensions**.

One of the classical **academic examples** is the *Chua circuit*, which is a quite simple electrical circuit with a non linear resistor whose trajectories are strange attractors. The figure shows two different solutions with different initial conditions.

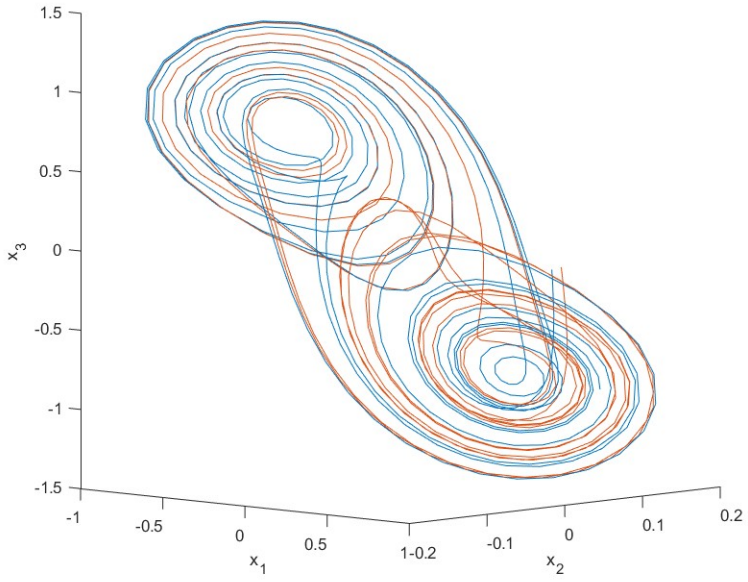


Figure 3.9: Strange attractor for the *Chua Circuit*

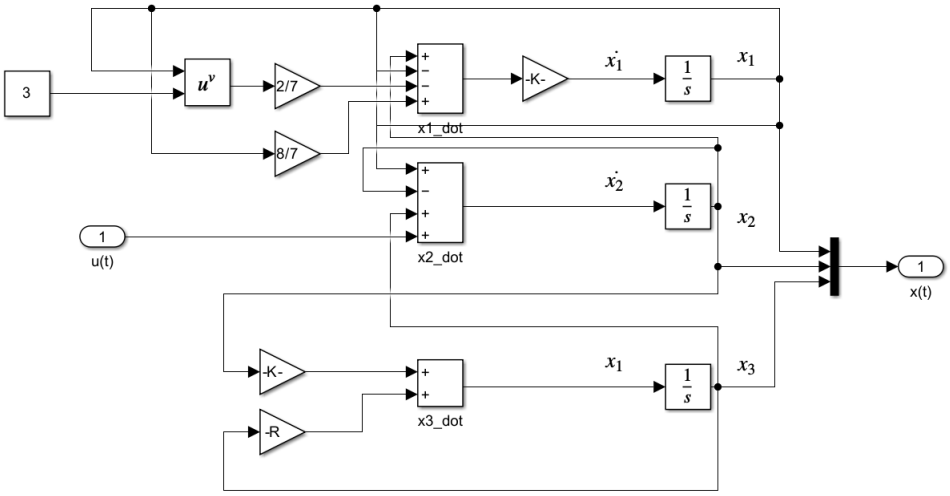


Figure 3.10: Simulink model for the Chua Circuit

Chapter 4

Lyapunov methods for non linear systems

Aleksandr Michajlovic Lyapunov (1857-1918) proposed two methods to study the stability of a **non linear system**. The first is based on observations done on a **linearized system**, the second is based on seeking a particular function that could satisfy some properties.

4.1 Linearization method (LM)

It is often of interest to linearize a system either around an **equilibrium point** or a **trajectory**. Usually it is useful to analyze the system and control it in a proper way.
A non linear system has the form

$$\dot{x(t)} = f[x(t), u(t)]$$
$$y(t) = h[x(t), u(t)]$$

In this scenario we define the following quantities:

$$\delta x(t) = x(t) - \bar{x}$$
$$\delta u(t) = u(t) - \bar{u}$$
$$\delta y(t) = y(t) - h(\bar{x}, \bar{u})$$

The **linearization step** is based on the **Taylor expansion** which states that **in the neighbourhood of an equilibrium point** x_0,

$$f(x) = f(x_0) + \frac{\partial f}{\partial x}(x - x_0)$$

If we apply this result to the non linear system for both functions f and g we obtain:

$$\dot{x} = f(x, u) = \delta\dot{x}(t) = \left.\frac{\partial f}{\partial x}\right|_{(\bar{x},\bar{u})} \delta x + \left.\frac{\partial f}{\partial u}\right|_{(\bar{x},\bar{u})} \delta u$$
$$\delta y(t) = \left.\frac{\partial h}{\partial x}\right|_{(\bar{x},\bar{u})} \delta x + \left.\frac{\partial h}{\partial u}\right|_{(\bar{x},\bar{u})} \delta u$$

Due to the fact that $f(x, u)$, $h(x, u)$ are **vector fields** the derivatives are **Jacobian matrices** which correnspponds to matrices A, B, C, D of a linearized system

$$\delta\dot{x}(t) = A\delta x(t) + B\delta u(t)$$
$$\delta y(t) = C\delta x(t) + D\delta u(t)$$

Where the matrices A, B, C, D are defined as follows:

$$A = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_{nx}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_{xn}}{\partial x_1} & \cdots & \frac{\partial f_{xn}}{\partial x_{nx}} \end{bmatrix}_{(\bar{x},\bar{u})} \qquad B = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \cdots & \frac{\partial f_1}{\partial x_{nu}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_{xn}}{\partial u_1} & \cdots & \frac{\partial f_{nu}}{\partial x_{nx}} \end{bmatrix}_{(\bar{x},\bar{u})}$$

$$C = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_{nx}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_{xn}}{\partial x_1} & \cdots & \frac{\partial f_{xn}}{\partial x_{nx}} \end{bmatrix}_{(\bar{x},\bar{u})} \qquad D = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_{nx}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_{xn}}{\partial x_1} & \cdots & \frac{\partial f_{xn}}{\partial x_{nx}} \end{bmatrix}_{(\bar{x},\bar{u})}$$

The **linearized system** represents an approximation of the non linear system in the neighbourhood of the equilibrium point $(\bar{x}, \bar{u})$

4.1.1 Stability of an equilibrium point

Let $\lambda_1, ..., \lambda_n$ be the eigenvalues of the A matrix previously defined. The **equilibrium point** $(\bar{x}, \bar{u})$ of the system is:

- **Asimptotically stable** if and only if $\mathrm{Re}(\lambda_i) < 0 \quad \forall i$
- **Unstable** if exists an i such that $\mathrm{Re}(\lambda_i) > 0$
- Nothing can be said in case of marginal stability of the linearized system.

This method is very simple, but provide us a way to study only the **local stability**, moreover it does not give any information on stability when the linearized system is only **marginally stable**.

4.2 Lyapunov Direct Method (DM)

The **Direct Method (DM)** is not restricted to local motion. It gives a way to determine the stability properties by constructing an **energy-like function** called the **Lypunov Function**. The method is based on a *physical observation*:

If the **total energy** of a system is dissipated, then the system must settle down to an **equilibrium point**.

For example if we analyze the **mass-spring-damper** system, we note that is impossible to find an analytical solution, moreover the **Linearization Method** does not give any useful information if the motion starts outside the linear range. Using instead the expression of **mechanical energy** we can find more useful information.
The **Lyapunov's Direct Method** is based on a generalization of this idea to more complex systems. Now it is the moment to formalize this concepts and give some useful definitions.

Definition (Positive definite functions) A function $V : \mathbb{R}^n \to \mathbb{R}$ is *locally positive definite*, if for some ball $B_R = \{x : \|x\| \leq R\}$,

$$\begin{aligned} V(x) &= 0, \quad x = 0 \\ V(x) &> 0, \quad x \neq 0, x \in B_R \end{aligned}$$

If $B_R = \mathbb{R}^n$ then the function $V(x)$ is said to be **globally definite positive**.

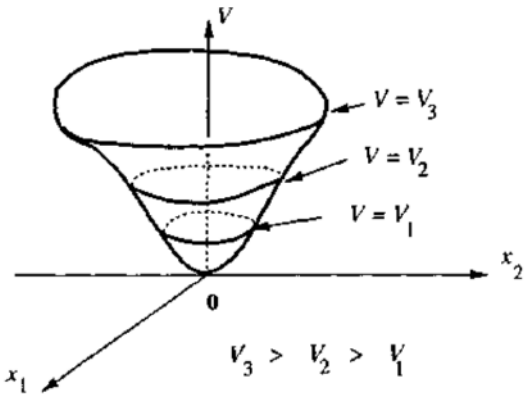


Figure 4.1: Example of a **Positive Definite Function**

Definition (Positive semi-definite functions) A function $V : \mathbb{R}^n \to \mathbb{R}$ is *positive semi-definite*, if for some ball $B_R = \{x : \|x\| \leq R\}$,

$$\begin{aligned} V(x) &= 0, \quad x = 0 \\ V(x) &\geq 0, \quad x \neq 0, x \in B_R \end{aligned}$$

Definition A function $V(x)$ is *negative definite* if $-V$ is *positive definite*. A function $V(x)$ is *negative semidefinite* if $-V$ is *positive semi-definite*.

We are ready to give the definition of what is a **Lyapunov function**, two theorems based on the existence of such function can be used to recover the **stability properties** of a non linear system.

Definition (Lyapunov function) A functionm $V : \mathbb{R}^n \to \mathbb{R}$ is a *Lyapunov function* if we can say that **for some ball** B_R,

1. $V(x)$ is positive definite and **has continuous partial derivatives**,
2. $\dot{V}(x)$ is negative semi-definite

The derivative $\dot{V}(x)$ respect to time can be computed by using the **chain rule**:

$$\dot{V}(x) = \frac{\partial V}{\partial x}\frac{\partial x}{\partial t} = \nabla V(x)\dot{x} = \nabla V(x) f(x)$$

Using the fact that for an autonomous system we have $\dot{x}(t) = f(x)$

Without loss of generality we can assume that $\bar{x} = 0$ is an equilibrium point, about the

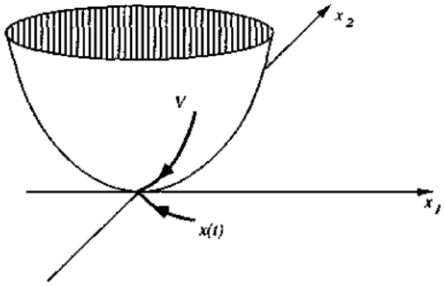


Figure 4.2: Example of a **Lyapunov Function**

stability we have these theorems:

Theorem (local stability) If the system

$$\dot{x}(t) = f[x(t), u(t)]$$
$$y(t) = h[x(t), u(t)]$$

-admits a **Lyapunov Function** for some ball B_R, then the equilibrium point $\bar{x} = 0$ is (marginally) stable).
-If moreover the function $\dot{V}(x)$ is **negative definite** in B_R then the stability is **asymptotic**.

Theorem (global asymptotic stability) If a system admits a Lyapunov function for $B_R = \mathbb{R}^n$ and
-$\dot{V}(x)$ is negative definite in $\mathbb{R}^n$
-$V(x)$ is **radially unbounded**, that is $\|x\| \to \infty \Rightarrow V(x) \to \infty$
The equilibrium point $\bar{x} = 0$ is **globally asysmptotically stable** for the system.

Such theorems are not constructive, in the sense that they do not tell us how to build a **Lyapunov function**, there are some methods that do not allow us to find in general the functions, but only in some cases. Some common methods are:

- **Physical insight approach**, when it is possible to use physical observations a very powerful analysis can be conducted for these systems;
- **Krasovskii's Method**, it gives **sufficient conditions** based on the Jacobian of the matrix A of the linearized system.
- **Variable Gradient Method** where a particular form for the gradient of the function $V(x)$ is assumed, then we compute the $V(x)$ by integrating such gradient.

4.3 An Example: The pendulum

In this section are shown an example of *Lyapunov function* for the pendulum case, such function satisfies the properties in the theorem above mentioned. It is:

$$V(x) = 2(1 - \cos(x_1)) + \frac{x_2^2}{2} + \frac{(x_1 + x_2)^2}{2}$$

In particular we observe that:

1. $V(x)$ is a Lyapunov Function in a certain ball B_R

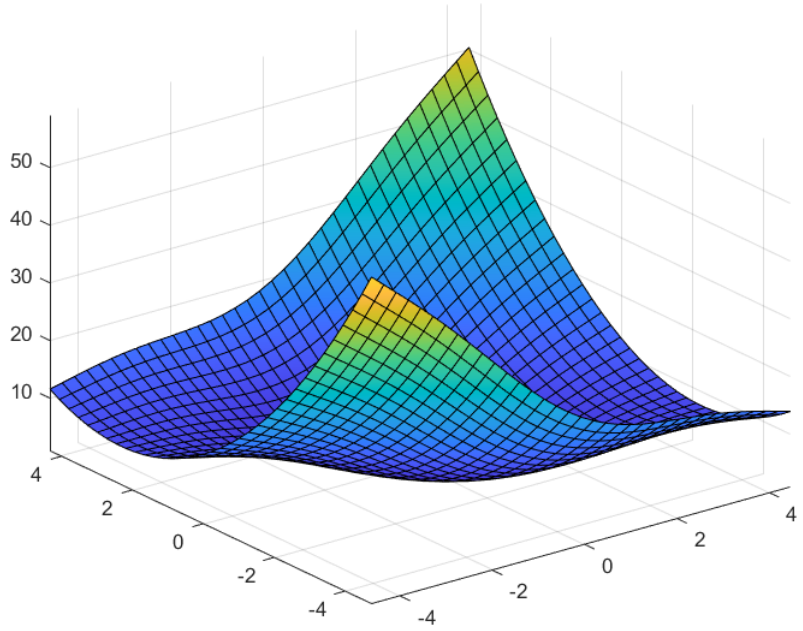


Figure 4.3: $2(1-\cos(x_1)) + \frac{x_2^2}{2} + \frac{(x_1+x_2)^2}{2}$

2. $\dot{V}(x)$ is **negative definite** in a certain ball B_R centered in $\bar{x} = 0$.

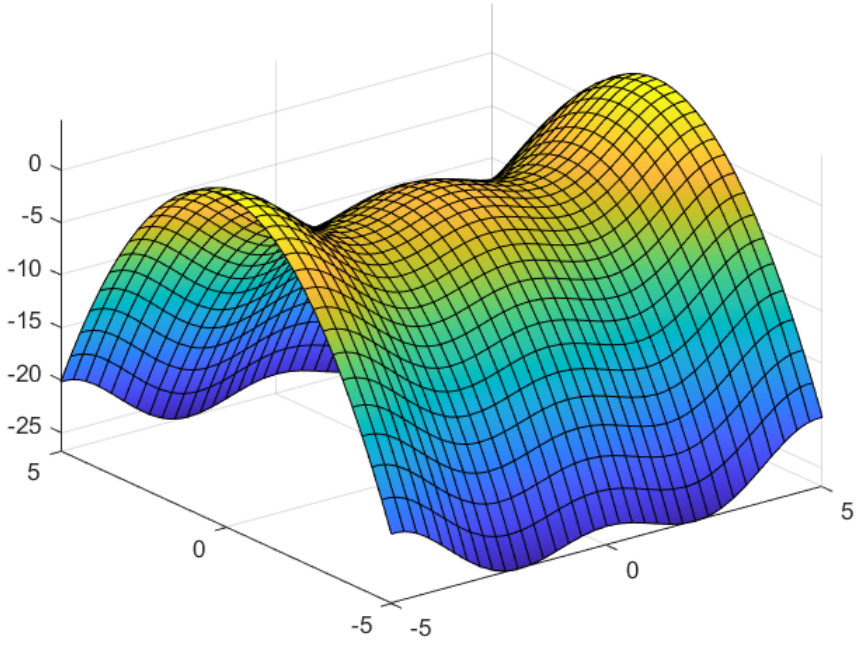


Figure 4.4: $\dot{V}(x)$

Chapter 5

Introduction to Non Linear Control

In general the **control theory** deals with the formulation of a **control law** $u(t)$ above a given physical system with a given dynamics. The system C, the so called **Controller**, should be applied on the real plant such that $y = r$ for a set of signals of interest. The command $u(t)$ is defined into the system C in a proper way.

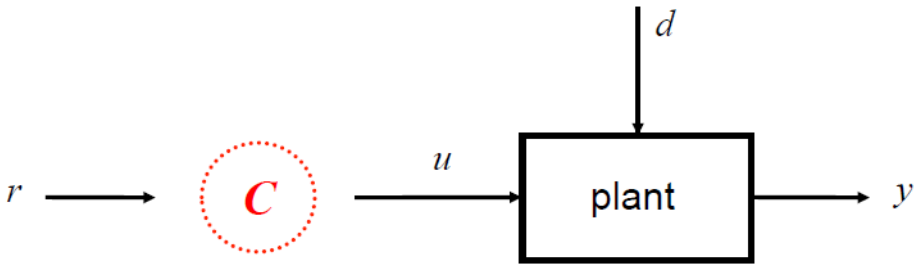


Figure 5.1: Classical structure of a controlled plant

In this field topics of interest are: (a) The nonlinear stabilization of the real plant, (b) the nonlinear Tracking, (c) The specification of Desired Behaviour of the System.

5.1 Non linear Stabilization and Tracking

In the case that the **tasks of a Control System** involve large range or high speed motions, non linear behaviours and effects can be a problem, for this reason **non linear control** is necessary to achieve some objectives.
Generally speaking, the tasks of a control system can be divided into two categories:

- **Stabilization** (or Regulation), here a control system called a *stabilizer* (or regulator), is designed so that the **closed-loop** system is stable around an equilibrium point (examples: temperature regulation, altitude control...);
- **Tracking** (or Servo), here the design objective is to build a controller, called the *tracker*, which has the task to ensure that the system could track a desired **time-varying trajectory** (examples: making a robot hand draw straight lines or circles, making an aircraft fly along a specified path...)

5.2 Specifying the desired behaviour

In linear control, the desired behaviour of a control system can be specified either in the time-domain by providing specifications like rise time, overshoot and settling time) or in the frequency domain in terms of regions in which the *loop transfer function* (usually indicated with $L(s)$) should lie at high and low frequencies.
However, **systematic specifications** for nonlinear systems (except those equivalent to linear system) are not obvious: at first a frequency-domain specification is not available, the response of the system to one command does not reflect its response to another command. As a result, for nonlinear system, one often try to find some **qualitative** specifications in the operating region of interest. Some common characteristics are: Stability (guaranteed for the nominal model), Robustness, Cost, Accuracy and speed of response.

5.3 A general procedure for Control Design

Given a physical system to be controlled, one typically goes through the following standard procedure:

1. specify the desired behaviour, and select the actuators and sensors;
2. model the physical plant by a set of differential equations;
3. design a control law for the system;
4. analyze and simulate the resulting control system;
5. implement the control system in hardware;

5.4 Available methods for nonlinear controllers

As in the analysis, we have not a general method to **design a nonlinear controller**, even in this case we have a collection of techniques which are either alternative or complementary and each applicable to a class of nonlinear control problems.

5.4.1 Trial and error

It is based on the analysis method provided before, like the *phase plane method* and the *Lyapunov analysis*. The design strategy deploys this technique in a way similar to the loop-shaping. This technique may work sometimes but in complicated situation it fails.

5.4.2 Feedback linearization

As discussed in the previous paragraphs, one of the main steps for design a controller (the first in particular) is to derive a meaningful model for the plant. **Feedback linearization** deals with techniques for *transforming the original system models into equivalent models of a different simpler form.* Can be used in a proper way as a strategy for design nonlinear controller. The **basic idea** is to linearize (fully or in part) the nonlinear plant and then apply the well-known techinques to complete the control design.
It is appliable to a number of practical cases. As a *drawback* it does not guarantee robustness against **disturbances** and **parameters uncertainty**.

5.4.3 Robust control

In the great majority of the control techniques the controller is designed according to the **nominal model of the plant/physical system**. In pure **model-based** design strategies uncertainties are neglected. In **robust nonlinear control** (eg. *Sliding Mode Control*), on the other hand, the controller is designed by exploiting both nominal plant and some characterization of the model uncertainties.

5.4.4 Adaptive control

Adaptive control is a technique to deal with **uncertain or time-varying systems**. Despite these alternatives, nowadays, adaptive control techinques are used when one have to face with system with *known dynamic structure* but **unknown constant or time-varying parameters**.
There are quite strong results for linear systems concerning this strategy, but also for nonlinear systems with **measurable state** and **linearly parametrizable dynamics** have been deployed some methods. When it is possible to design such type of controller it represents an alternative or a complementary part for the robust control design.

5.4.5 Gain scheduling

Gain scheduling is a method which make a try to apply well-known linear techniques to control nonlinear systems. It was developed for the trajectory control of aircraft.
The **idea behind** this technique is to collect a **set of operating points** which may cover the range of the system operation. Then, at **each of these points**, the designer makes a linear time variant approximation to the plant dynamics and build a **linear controller** for each linearized plant. What happen between two operating points? The parameters are interpolated, or **scheduled**, providing a global compensator. Maybe Gain Scheduling is the simplest method for designing nonlinear controllers, but the main problem is that one have only limited guarantees in stability in non linear operations; another drawback is the necessity of computing a cospicuous number of linear controllers.

Chapter 6

Method(1): Feedback linearization

Feedback linearization is an appproach to nonlinear control whose **central idea** is to algebraically transform a nonlinear system dynamics into a linear one, so that the well-known strategies for linear controllers could be applied. This technique differs from *Jacobian linearization*, in the sense that the **linearization** is achieved by **exact** state transformation and feedback, instead of *linear approximation* of the dynamics.

This idea of simplifying the form of the system's dynamics by choosing a different **state representation** is not unfamiliar, it is well known that in mechanics the description could vary a lot according to the choice of the **reference frame** or **coordinate system**.

In the introductory discussion of feedback linearization, commonly one give the example of designing a control law for controlling the **fluid entering into a tank**. Given the dynamics, it is quite simple to show that can be chosen a particularly simple input to delete the non linearities of the plant in a way that the system resulting in a simple **integrator**.
In more complex situations, than the tank system, it is required to have some standard techniques to linearize the system. In this context we can mention:

1. **input-state linearization** where linearization is performed via state feedback;
2. **input-output** linearization, here linearization is achieved by using the feedback of the output.

To this aim it is useful to introduce the concept of **Lie derivatives** which is an elegant way to express the well-known **chain rule**.

6.1 Lie derivatives

This definitions are given in order to use a more compact notation for the following paragraphs.

Definition A function $f(x)$ is said to be **smooth** if it has continuous partial derivatives of any required order.

Definition Let $h : \mathbb{R}^n \to \mathbb{R}$ a smooth function, and $f : \mathbb{R}^n \to \mathbb{R}^n$ a *vector field* of $\mathbb{R}^n$. The **Lie Derivative of** h **with respect to** f is a scalar function defined by $L_f h \doteq \nabla h f \in \mathbb{R}$

The meaning of **Lie derivative** is doing a derivative of h in the direction provided by f. The

computation of such derivatives can be done in a **recursive way**, as follows:

$$L_f^0 h = h$$
$$L_f^i h = L_f(L_f^{i-1}h) = \nabla(L_f^{i-1}h)f, \quad i = 1, 2, ...$$

6.2 Input-Output linearization

Now we consider the **SISO nonlinar system** of the form

$$\begin{aligned} \dot{x} &= f(x) + g(x)u \\ y &= h(x) \end{aligned} \tag{6.1}$$

where $x \in \mathbb{R}^{n_x}$ is the state, $u \in \mathbb{R}$ is the command input, $y \in \mathbb{R}$ is the output, and f, g, h are smooth functions.
Such system is said to be *affine in u*. We assume that the state is accessible and so measurable, otherwise an observer has to be employed.
Input-output linearization approach consists in differentiate repeatedly, until the input u appears, then design the control law $u(t)$ to cancel the nonlinearity.
We can start by taking the output equation $y = h(x)$ and differentiate it:

$$\dot{y} = \frac{\partial h}{\partial x}\dot{x} = \nabla h(x)\dot{x} = \nabla h(x)(f(x) + g(x)u) = \tag{6.2}$$
$$\nabla h(x)f(x) + \nabla h(x)g(x)u = L_f h + L_g h(x)u \tag{6.3}$$

If $L_g h(x) \neq 0$ at some point $x = \bar{x} \in \Omega_x \subseteq \mathbb{R}^n$ then in Ω_x the control law is:

$$u = \frac{1}{L_g h(x)}(-L_f h(x) + v) \tag{6.4}$$

This transforms the output equation in $\dot{y} = v$. Otherwise we differentiate again obtaining:

$$\ddot{y} = L_f^2 h(x) + L_g L_f h(x)u \tag{6.5}$$

Until the multiplicative term of u is not null, you have to differentiate again and again obtaining:

$$y^{(i)} = L_f^{(i)} h(x) + L_g L_f^{(i-1)} h(x)u \tag{6.6}$$

Then, for some γ the term $L_g L_f^{\gamma-1} h(x)u \neq 0$ the control law and the resulting system will be respectively

$$u = \frac{1}{L_g L_f^{\gamma} h(x)}(-L_f^{\gamma-1} + v) \tag{6.7}$$
$$y^{(\gamma)} = v \tag{6.8}$$

The equation (6.8) can be written in the state space form, definining the **state vector**

$$\mu = (\mu_1, ..., \mu_\gamma) \doteq (y, \dot{y}, ..., y^{(\gamma-1)}) \tag{6.9}$$

The resulting state equations are in the so-called *companion form*:

$$
\begin{aligned}
&\dot{\mu} = A\mu + Bv \\
&y = \mu_1 \\
&A = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 1 \\ 0 & 0 & \dots & 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}
\end{aligned}
$$

These equations describe only a part of the system dynamics, called the **external dynamics**, which is the part that one can control. This is linked with another important information of the non linear system that is the **relative degree**.

6.2.1 Relative degree

Now we give the definition of **relative degree**:

Definition The integer $\gamma \leq n$ is the **relative degree** of the system (6.1) in Ω.

This is a generalization of the concept of relative degree for LTI systems (the difference between the denominator degree and numerator degree of the transfer function $H(s)$. In the case that $\gamma = n$, input-output linearization corresponds to input-state linearization.
It could happen that the term $L_g L_f^{\gamma-1} h(x)$ is zero at $\bar{x}$ but non zero in the neighbourhood of $\bar{x}$. In this case the relative degree is *undefined* in $\bar{x}$.

6.2.2 Normal form

Once we have applied the control law, we can define the **normal state** (μ, ψ) in the following way:

$$
\mu = (\mu_1, ..., \mu_\gamma) \doteq (y, \dot{y}, ..., y^{\gamma-1}) \in \mathbb{R}^\gamma \qquad \text{external dynamics} \tag{6.10}
$$

$$
\psi \in \mathbb{R}^{n-\gamma} \qquad \text{internal dynamics} \tag{6.11}
$$

The nonlinear system (6.1) can be written in the so called *normal form*:

$$
\begin{aligned}
&\dot{\mu} = \begin{bmatrix} \mu_2 \\ \vdots \\ \mu_\gamma \\ a(\mu, \psi) + b(\mu, \psi)u \end{bmatrix} \qquad \begin{aligned} a(\mu, \psi) &\doteq L_f^\gamma h(x) \\ b(\mu, \psi) &\doteq L_g L_f^{\gamma-1} h(x) u \end{aligned} \\
&\dot{\psi} = w(\mu, \psi) \\
&y = \mu_1
\end{aligned} \tag{6.12}
$$

While the **external dynamics** is well defined and one can control it, find explicitly the **internal dynamics** could require to solve Partial Differential Equations (PDE), however in some cases it is possible to study the **stability of the internal dynamics** theoretically by using the *zero dynamics* otherwise simulations have to be used.

6.2.3 Zero dynamics

The internal dynamics of the system (6.1) is not controlled, it can be therefore either bounded or unbounded. We introduce the concept of **zero dynamics** which represents a simplification of the internal dynamics.

Definition (Zero dynamics) The zero-dynamics of the system 6.1 in Ω is defined by

$$\begin{aligned} \dot{\mu} &= 0 \\ \psi &= w(0, \psi) \end{aligned} \tag{6.13}$$

with initial conditions $\mu(0) = 0$ and $\psi(0) = \psi_0$

Definition The system 6.1 is *locally asymptotically minimum phase at* $\bar{x}$ if $\psi = 0$ is a locally asymptotically stable equilibrium point of the system (6.13)

6.2.4 First control problem: Regulation

In the introduction, we have seen that the most common problem in the field of automatic control are: regulation and tracking.
In order to obtain a **regulation** of the system (6.1) is sufficient to define the control law:

$$\begin{aligned} u &= \frac{1}{L_g L_f^{\gamma-1} h(x) u}(-L_g h(x) + v) \quad \text{feedback linearization} \\ v &= -K\mu \qquad\qquad\qquad\qquad\qquad\qquad \text{linear control} \end{aligned} \tag{6.14}$$

where $K = [k_1, ..., k_\gamma] \in \mathbb{R}^\gamma$ is such that $A - BK$ is asymptotically stable.

Theorem Assume that the system (6.1) has a locally asymptotically stable zero dynamics. Then, the *closed loop* defined by (6.1) and (6.14) is *locally asymptotically stable.*

6.2.5 Second control problem: Tracking

Now let consider the problem that the system output is required to track a desired signal $r(t)$, this signal has to be **smooth** and **bounded**. Let $\mu_r \doteq (r, \dot{r}, ..., r^{\gamma-1})$, we define the **tracking error** as

$$\tilde{\mu} \doteq \mu_r - \mu$$

We would like to make this error small and possibly to force it to converge to zero. Even in this case we define:

$$\begin{aligned} u &= \frac{1}{L_g L_f^{\gamma-1} h(x) u}(-L_g h(x) + v) \quad \text{feedback linearization} \\ v &= -K\tilde{\mu} + r^{(\gamma)} \qquad\qquad\qquad\qquad\qquad\qquad \text{linear control} \end{aligned} \tag{6.15}$$

where $K = [k_1, ..., k_\gamma] \in \mathbb{R}^\gamma$ is such that $A - BK$ is asymptotically stable.

Theorem Assume that the solution of the equation

$$\dot{\psi}_r = w(\mu_r, \psi_r), \quad \psi_r(0) = 0$$

exists, and is bounded and asymptotically stable. The, $\tilde{\mu}(t)$ converges exponentially to 0, as $t \to \infty$

6.2.6 Control scheme

The aim of this section is to give a general control scheme for physical systems/plants controlled by using the *feedback linearization*.

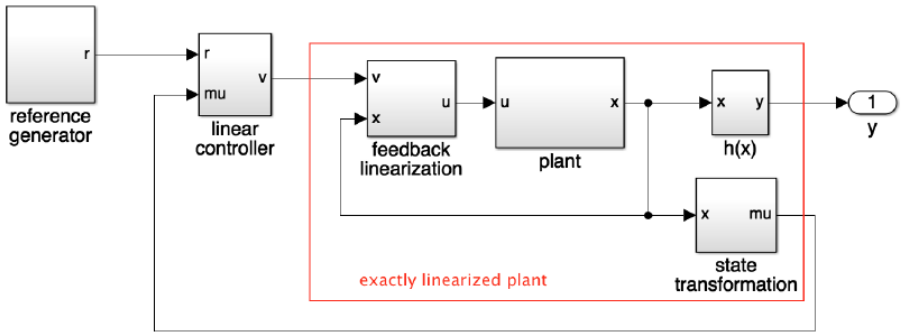


Figure 6.1: Feedback linearization control scheme

Now we are going to describe the blocks of the figure (6.1):

- **Plant**: is a nonlinear system affine in the input u of the form $\dot{x} = f(x) + g(x)u$;
- **State transformation**: is obtained by applying $\mu = (y, \dot{y}, ..., y^{(\gamma-1)}) = (h(x), L_f h(x), ..., L_f^{\gamma-1} h(x))$
- **Feedback linearization**: is obtained implementing the equation 6.15;
- The **linear controller** can be implemented by using any technique (eg. pole placement)

6.3 Discussion

It is useful pay attention that:

1. The results which has been found for Feedback linearization hold in *some region* of the state space $\Omega \subseteq \mathbb{R}^n$, in particular where the multiplicative term $L_g L_f^{\gamma-1} h(x)$ is non-zero. Global results are available but we have to do stronger assumptions.

2. We have seen that there are relevant differences with respect to **Conventional linearization**: in FL the linearization is exact on the whole region Ω. Even in the case the CL worked, the Feedback linearization gives better results;

3. The Feedback linearization technique does not take into account problems related to the uncertainty of the nominal plant and parameters: issue linked to the robustness are provided instead by other techniques like *sliding mode control*.

Chapter 7

Method(2): Sliding mode control

Sliding mode control (SMC) is a well-estabilished method for control of nonlinear systems, and is characterized (like Feedback linearization) by solid theoretical foundations. An interesting feature of SMC is the robustness against **imprecise knowledge of the plant to control**.
General Approach:

1. A **sliding surface** is defined. This surface is a subset of the state domain where the system can have **good properties**.

2. A feedback law is designed in order to bring the plant trajectory towards the sliding surface and once there, to stay close to this surface.

The SMC control law is similar to the FL control law, but the approaches are quite different. In the case of control by using Feedback linearization of **Chua circuit** we have seen that the nonlinear resistor $\rho(x_1)$ brings the system to have a chaotic behaviour. Although to design the controller we have used a *smooth approximation* of the nonlinearity that prevent us to use a piece-wise function which is not differentiable. If we try to apply the control to the real plant, we can see that the tracking error is very large. This is due to the fact the FL does not take into account uncertainties or approximations of the plant. We will see at the end of this chapter that the tracking error of the controlled system after that a Sliding mode controller has been applied is insignificant with respect to the Feedback linearization one. In the figure below we can note this fact:

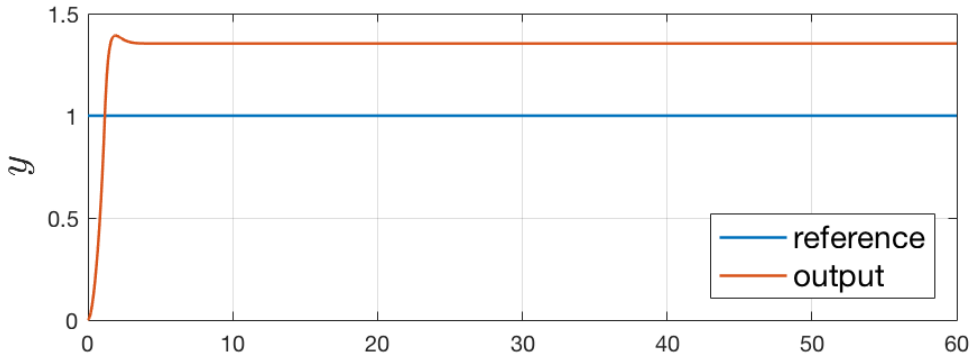


Figure 7.1: Tracking error in presence of uncertainties (FL)

We have the same setting as Feedback linearization, the (nonlinear) system is in the form:

$$\begin{aligned} \dot{x} &= f(x) + g(x)u \\ y &= h(x) \end{aligned} \tag{7.1}$$

which can be lead to the *normal form* that distinguishes: the external dynamics described by the state transformation $x \to \mu$ and the internal dynamics described by $\dot{\psi}$ assumed to be locally asymptotically stable.
Suppose now that the system output $y(t)$ is required to track a reference signal $r(t)$. In a way similar to the FL, we define the *tracking error* as $\tilde{y} \doteq r - y$. Even in this case we want to make it "small" and force it to converge to zero.

Now a set of definitions are given as ingredient to introduce the **Sliding mode control** method.

Definition (Sliding surface) The **sliding surface** is

$$S(t) \doteq \{x \in \mathbb{R}^n : \mathbf{s}(x,t) = 0\}$$

where the function $\mathbf{s} : \mathbb{R}^{n+1} \to \mathbb{R}$ is defined as follows

$$\begin{aligned}\mathbf{s}(x,t) = \tilde{y}^{(\gamma-1)} + k_\gamma \tilde{y}^{(\gamma-2)} + ... + k_2 \tilde{y} = \\ = \tilde{\mu}_\gamma + k_\gamma \ \tilde{\mu}_{\gamma-1} + ... + k_2 \ \tilde{\mu}_1\end{aligned}$$

the coefficients $k_i \in \mathbb{R}$ are chosen so that the roots of the polynomial $P(\lambda) = \lambda^{\gamma-1} + k_\gamma \lambda^{\gamma-2} + ... + k_2$ have **negative real part**.

7.1 Behaviour on the sliding surface

Property 1: If the trajectory of the system is confined to the sliding surface, then the tracking error $\tilde{y} \to 0$, in a way that depends on the root of $P(\lambda)$.

Due to the fact that the system has a nice behaviour if the **Property 1** is verified, we would like to find a control law such that the sliding surface S could be:

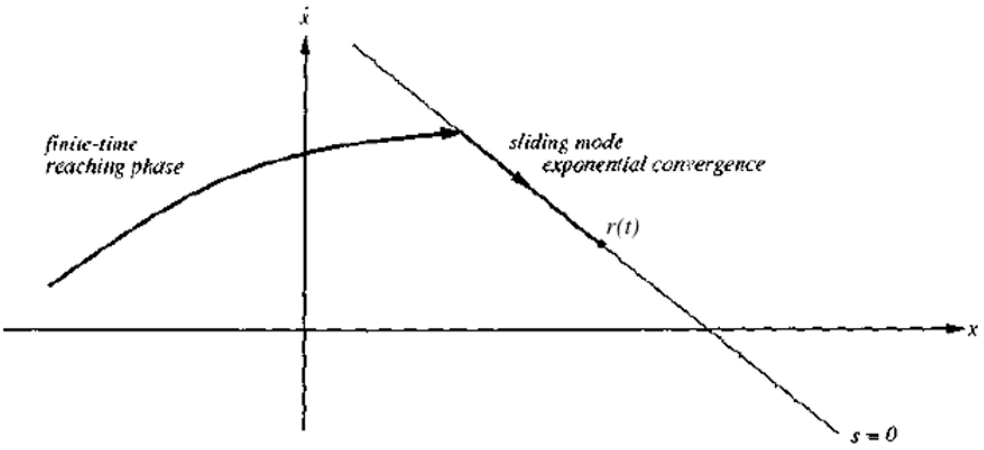
- **Invariant**: if the trajectory is on S, it remains on it; that is $x(\tau) \in S(\tau) \Rightarrow x(t) \in S(t), \forall t \geq \tau$;
- **Attractive**: if the trajectory is outside of the sliding surface S, it is forced to move toward it in a finite time.

The motion of the system on this surface is called **sliding mode**, from this fact is derived the name **Sliding mode control**.

7.1.1 Invariance of the Sliding surface

If in a certain time τ the trajectory of the system lie on the sliding surface, then as defined before $x(\tau) \in S \Rightarrow \mathbf{s}(x(\tau), \tau) = 0$. In order to guarantee the invariance of the surface we have to put $\dot{\mathbf{s}} = 0 \quad \forall t$.

$$\dot{\mathbf{s}} = \tilde{y}^{(\gamma)} + k_\gamma \tilde{y}^{(\gamma-1)} + ... + k_2 \dot{\tilde{y}} = 0 \iff r^{(\gamma)} - a(x) - b(x)u + k_\gamma y^{(\gamma-1)} + ... + k_2 \dot{\tilde{y}} = 0$$



where we used $\tilde{y}^{(\gamma)} = r^{(\gamma)} - y^{(\gamma)} = r^{(\gamma)} - a(x) - b(x)u$.
Solving the equation for u we obtain the following control law

$$u_s = \frac{1}{b(x)}\left(r^{(\gamma)} - a(x) + k_\gamma \tilde{y}^{(\gamma-1)} + ... + k_2\dot{\tilde{y}}\right) \tag{7.2}$$

Property 2: using this law, the sliding surface is an invariant set.

7.1.2 Attractiveness of the Sliding surface

Suppose that at a certain instant τ the trajectory is not on the sliding surface. We would like to bring it toward $S(t)$ in a finite time.
The introduction of a discontinuous term to u_s make S attractive. Then, the *complete control law* is the following, let $k_1 > 0$:

$$u_s = \frac{1}{b(x)}\left(r^{(\gamma)} - a(x) + k_\gamma \tilde{y}^{(\gamma-1)} + ... + k_2\dot{\tilde{y}} + k_1 \text{sign}(\mathbf{s}(x,t))\right)$$

The motivation of adding a discontinuous term is the fact that in this way the derivative of the sliding function

$$\dot{\mathbf{s}}(x,t) = r^{(\gamma)} - a(x) - b(x)u + k_\gamma \tilde{y}^{(\gamma-1)} + ... + k_2\dot{\tilde{y}} = \tag{7.3}$$

$$= -k_1 \text{sign}(\mathbf{s}(x,t)). \tag{7.4}$$

Property 3: $\mathbf{s}(x,t)\ \dot{\mathbf{s}}(x,t) < 0, \quad \forall x,t$, which implies that $S(t)$ is attractive (the sliding function $\mathbf{s}$ converges to zero in finite time). In particular:

- $\mathbf{s}(x,\tau) > 0 \Rightarrow \dot{\mathbf{s}} = -k_1$

$$\int_\tau^t \dot{\mathbf{s}}\ dt = \int_\tau^t -k_1\ dt \iff \mathbf{s}(t) - \mathbf{s}(\tau) = -k_1(t-\tau) \iff \quad \mathbf{s}(t) = \mathbf{s}(\tau) - k_1(t-\tau)$$

- $\mathbf{s}(x,\tau) < 0 \Rightarrow \dot{\mathbf{s}} = k_1$

$$\int_\tau^t \dot{\mathbf{s}}\ dt = \int_\tau^t k_1\ dt \iff \mathbf{s}(t) - \mathbf{s}(\tau) = k_1(t-\tau) \iff \quad \mathbf{s}(t) = \mathbf{s}(\tau) + k_1(t-\tau)$$

In both cases, $\mathbf{s}(t) \to 0$ in finite time $\Rightarrow x(t) \to S(t)$ in finite time applying the input with discontinuous term.

7.1.3 Chattering

The presence of the discontinuous term may cause a phenomenon called **chattering** which consists of high frequency oscillations around the sliding surface. To avoid this problem we can use a **sigmoid function** $\sigma(\eta \mathbf{s})$, where η is a design parameter which defines the slope of the sigmoid itself. The function $\sigma(\eta \mathbf{s}) = \tanh(\eta \mathbf{s})$ is a typical choice. The control law becomes:

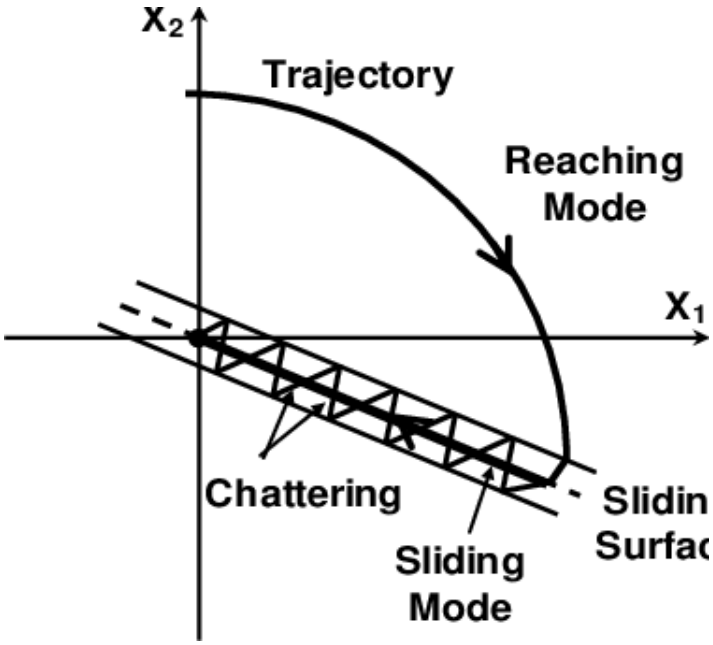


Figure 7.2: Chattering phenomenon

$$u_s = \frac{1}{b(x)}\left(r^{(\gamma)} - a(x) + k_\gamma\ \tilde{y}^{(\gamma-1)} + ... + k_2\ \dot{\tilde{y}} + k_1\sigma(\eta \mathbf{s})\right)$$

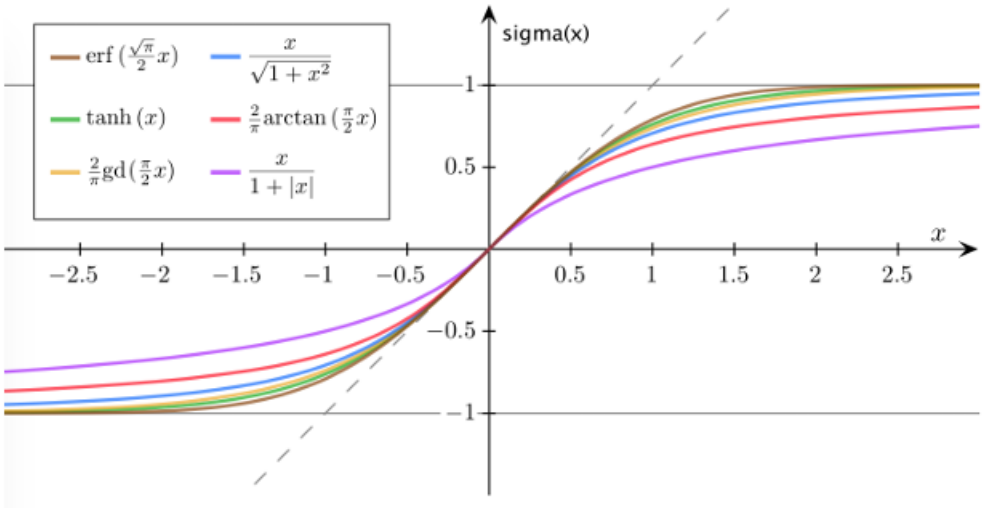


Figure 7.3: Alternatives to sign(**s**)

7.2 Comparison with feedback linearization

The two approaches of **Feedback linearization** and **Sliding mode control** have a common general control law given by:

$$u = \frac{1}{L_g L_f^{\gamma-1} h(x)}(-L_f^{\gamma} h(x) + v) = \frac{1}{b(x)}(-a(x) + v)$$

The techniques have the v which differs of a term. More specifically, let $\tilde{\mu}_i = r^{(i)} - y^{(i)} = \tilde{y}^{(i)}$

- In the feedback linearization: $v = r^{(\gamma)} + k_\gamma \ \tilde{\mu}_\gamma + ... + k_1 \ \tilde{\mu}_1$
- In the sliding mode control: $v = r^{(\gamma)} + k_\gamma \ \tilde{\mu}_\gamma + ... + k_1 \ \sigma(\eta \mathbf{s})$

The term which is different increases the robustness of the controller.

7.3 Control Scheme

Even the general control scheme is quite similar to the feedback linearization, the only difference is that instead of the linear controller there is a block that provided $r^{(\gamma)}$ and $\tilde{\mu}$ gives the sliding control law $v = r^{(\gamma)} + k_\gamma \ \tilde{\mu}_\gamma + ... + k_1 \ \sigma(\eta \mathbf{s})$

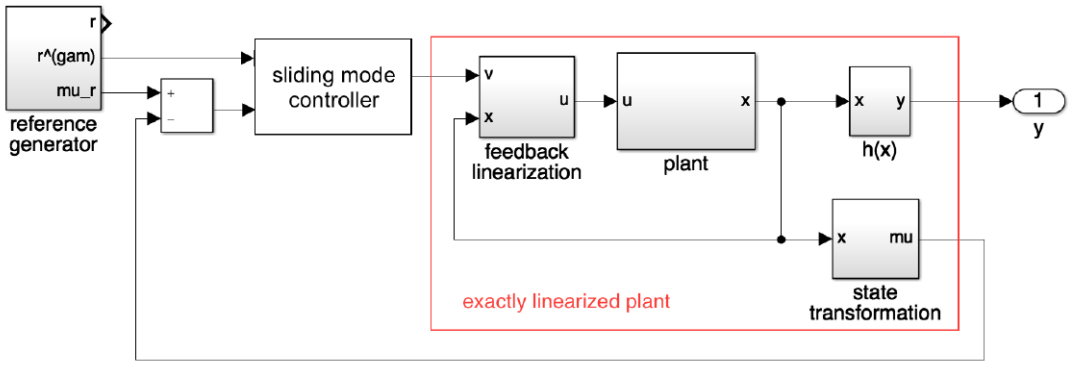


Figure 7.4: Sliding control mode general control scheme

7.4 Robustness properties

Now we consider the case when the system to control in not exactly known. In normal form we can describe the fact in this way:

$$y^{(\gamma)} = a(x) + \Delta a(x) + b(x)u$$

Where $\Delta a(x)$ is a smooth function of $\mathbb{R}^{n_x}$ but it is unknown, moreover we know that $\|\Delta a(x)\| \leq \bar{\Delta}, \ \forall x \in \mathbb{R}^n$. In this way the time derivative of the sliding function is:

$$\dot{\mathbf{s}}(x,t) = -\Delta a(x) - k_1 \ \sigma(\eta \mathbf{s})$$

In the context of uncertainty in the model of the plant it holds that:

Property 4: If $k_1 > \bar{\Delta}$, then the sliding surface is attractive.

The figure below shows the *step response* for a sliding mode controlled system in which uncertainty appears. In particular for control design we use the approximated $\hat{\rho}(x_1)$, while the plant has the real nonlinearity which is a piece-wise function $\rho(x_1)$. The tracking error is smaller than the case of a controller built by using the feedback linearization technique.

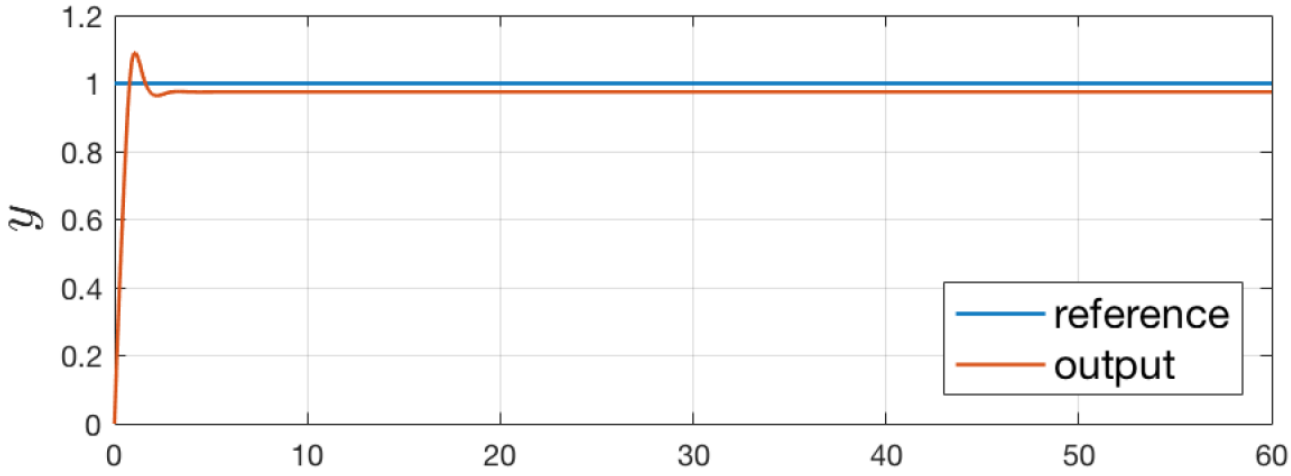


Figure 7.5: Step response: sliding mode controller

7.5 Guidelines for the choice of the SM parameters

- The parameters k_i, $i = \gamma, \gamma - 1, ..., 2$ are the coefficients of the polynomial $P(\lambda)$. Useful indications can be:
  - All the roots must have negative real part;
  - The speed of convergence is linked to the value of the k_i
  - At the same time they cannot be too much large this would result in too high command activity
- The parameter $k_1 > 0$ allows us to **increase the closed-loop robustness**: The larger is k_1 the more robust is the closed-loop system, but too large value of this parameter results in significant **chattering phenomena** and **high command activity**;
- Using the hyperbolic tangent function instead of sign requires to choose the additional parameter η. A large value of η can be taken initially so that $\tanh \simeq \text{sign}$, then η can be gradually reduced in order to reduce the chattering pheonomena without compromising too much **performance** and **robustness**.

7.6 Discussion

Sliding mode control is a technique which has strong theoretical behaviuor, is robust to model uncertainties and disturbances of a certain type for the mentioned reasons this method give **high control performances**.
On the other hand, as usual, there are some **drawbacks**:

- The model is assumed to be affine in u, it cannot have whatever structure;
- When the internal dynamics is stable everything works well, otherwise we can't approach the control problem; sometimes it is sufficient to change the output to make the same problem easier and so tractable;
- in general there is an **high command activity**.

Possible extensions of the concepts exposed here are possible for systems which have multiple input and multiple output (MIMO).

Chapter 8

Method(4): Non linear Model predictive control (NMPC)

In this chapter it will be illustrated a technique of control called **Nonlinear Model Predictive Control**. Other techniques we treated did not need some specific math tools in order to be explained and understood. This time we need some instruments that guide us to the study of this powerful and general control method.

8.1 Math tools

8.1.1 Vector and signal norms

Vector spaces

Vector spaces

A **linear vector space** F over a field $\mathbb{A}$ is a *set* equipped of two operations:

1. **Vector addition**: $\forall f, g \in F \Leftarrow f + g \in F$
2. **Scalar multiplication**: $\forall \alpha \in \mathbb{R},\ f \in F \Leftarrow \alpha f \in F$

Let us put together these two properties obtaining:

$$\forall \alpha, \beta \in \mathbb{R},\ \forall f, g \in F \leftarrow \alpha f + \beta g \in F$$

Norm: general definition

A **norm** is a function $\|\diamond\| : \mathbb{R}^n \to \mathbb{R}$ which satisfies the following properties:

1. $\|x\| \geq 0$
2. $\|\alpha x\| = \alpha \|x\|$
3. $\|x + y\| \leq \|x\| + \|y\|$ (triangular inequality)
4. $\|x\| = 0 \iff x = 0$

Vector norm

Let us consider a vector $f = (f_1, ..., f_n)^T \in \mathbb{R}^n$, we can define some norm functions in particular ℓ_1, ℓ_2 and ℓ_∞ norms:

- ℓ_1-norm is defined as: $\|f\|_1 = \sum_{i=1}^n |f_i|$
- ℓ_2-norm is defined as: $\|f\|_2 = \sqrt{\sum_{i=1}^n f_i^2}$
- ℓ_∞-norm is defined as: $\|f\|_\infty = \max_{i=1,...,n} |f_i|$
- ℓ_2 weighted norm is defined as: $\|f\|_{Q,2} = \sqrt{f^T Q f} = \sqrt{\sum_{i=1}^n qi f_i^2}$

The last type of norm is used when one want to give more or less importance to a certain component in the vector.

In MATLAB the command in order to compute a norm of a vector f properly defined is `norm(f)`, by default it is the ℓ_2-norm if not specified, otherwise one should use the commands `norm(f,1)` to indicate the ℓ_1-norm and `norm(f,inf)` in order to indicate the ℓ_∞-norm.

Signal norm

Another important type of norm is that of a **signal** which is defined in a similar way with respect to the vector case. We consider both the cases of *scalar function* and *vector valued* functions. The main difference wrt the vectors is the fact that we have integrals instead of summations, a function can be seen as a vector composed by an **infinite number of elements**.

First case: $f : \mathbb{R} \to \mathbb{R}$. We have the following definitions:

$$\|f(t)\|_1 = \int_0^\infty |f(t)|\, dt \quad (L_1 - \text{norm}) \tag{8.1}$$

$$\|f(t)\|_2 = \int_0^\infty [f(t)]^2\, dt \quad (L_2 - \text{norm}) \tag{8.2}$$

$$\|f(t)\|_\infty = \sup_{t \in \mathbb{R}^+} |f(t)| \quad (L_\infty - \text{norm}) \tag{8.3}$$

In MATLAB it is a little bit more complicated to compute a norm of a function because at first we cannot treat infinite values with the computer. A way to overcome this fact is to define a variable `dt` as a *sample step* and evaluate the function in a large number of points, then we can approximately calculate the norm by using the 'standard' command `norm`. Then, we have for example:

```
dt=0.0001;
t=linspace(0,100, dt);
f=sin(t);

L2   = norm(f,2)*sqrt(dt);        %L_2 norm
L1   = norm(f,1)*dt;              %L_1 norm
Linf = norm(f,inf);               %L_inf norm
```

The multiplicative terms `sqrt(dt)` and `dt` are been added in order to apply the definition. By doing the above calculations we have approximated the integrals.

Second case: $f : \mathbb{R} \to \mathbb{R}^n$. This is the case of **vector-valued** functions in which $\forall t \in \mathbb{R}^+$ we have a vector $f(t) \in \mathbb{R}^n$. The definitions previously given are modified as follows (basically we

have norms instead of absolute values due to the fact that for each t, you have a vector instead of a scalar):

$$\|f(t)\|_1 = \int_0^\infty \|f(t)\|_1 \, dt \quad (L_1 - \text{norm}) \tag{8.4}$$

$$\|f(t)\|_2 = \sqrt{\int_0^\infty \|f(t)\|_2^2 \, dt} = \sqrt{\int_0^\infty f(t)^T f(t) \, dt} \quad (L_2 - \text{norm}) \tag{8.5}$$

$$\|f(t)\|_2 = \sup_{t \in \mathbb{R}^+} \|f(t)\|_\infty \quad (L_\infty - \text{norm}) \tag{8.6}$$

$$\|f(t)\|_{Q,2} = \sqrt{\int_0^\infty f(t)^T \, Q \, f(t) \, dt} \quad (L_1 \text{ weighted norm}) \tag{8.7}$$

An observation can be done on (8.3) and (8.6): the sup generalizes the concept of maximum, in particular a function might not assume the value of the maximum.
In MATLAB the **weighted ℓ_2-norm** can be computed as follows:

```
dt=0.0001;
t=linspace(0,100, dt);
f=[exp(-3*t); sin(3*t)];        %vector valued function
Q=diag(1,2,3);

L2 = sqrt( sum ( sum (Q*f.^2) * dt ) );
%           ^     ^
%           |     |
%         \int   l2-norm
```

8.1.2 Optimization

Optimization is a branch of Maths that deals with **finding the best solution to a problem** which is *finding the minimum* of a function. Optimization acts a fundamental role in many engineering field and also in **control design** is important: when we design a controller we want to: (i) minimize the tracking error; (ii) minimize the command effort as the actuators has got limited working ranges; (iii) minimize the effect of disturbances. In our case we introduce it, in order to apply it to *Model Predictive Control*.

Basic notions

In the **univariate function** $f : \mathbb{R} \to \mathbb{R}$ the problem of minimization is the following:

$$x^* = \arg\min_{x \in \mathbb{R}} f(x) \tag{8.8}$$

$$f(x^*) = \min_{x \in \mathbb{R}} f(x) \tag{8.9}$$

The (8.8) is called the **minizer** while the (8.9) is called the **minimum**. The x is the **decision variable**, while $f(\diamond)$ is called either the **objective function** or **cost function** or **loss function** (used in AI).
In a univariate problem there could be: *one global minimum*, *several global minima*, *infinite*

global minima. The optimization problems usually require that the solution (minimizer) observes some **constraints**, in this case we are talking about **constrained minimization**. As the constraints could be more than one, an alternative notation for the problem (8.8) is:

$$\begin{aligned} &\min\ f(x) \\ &\text{subject to } x \in \mathbb{R} \end{aligned} \tag{8.10}$$

Where *subject to* is often abbreviated as **s.t.**
In order to start talking about optimization we considered the **univariate** case in which the decision is taken according a **single variable**, in engineering problems the decision variable can be hundreds or thousands! The graphical representation becomes not possible at all, but the concepts we exposed previously remain more or less the same but more general. We are talking about in this case of **multivariate functions**. In particular:

- The **objective function** $f(x)$ is such that $f : \mathbb{R}^n \to \mathbb{R}$
- The **decision variable** is a vector $x \in \mathbb{R}^n$
- Also the **minimizer** is a vector $x^* \in \mathbb{R}^n$

An Optimization problem is said to be **feasible** if there is at least a solution, otherwise it is said to be **unfeasible**. We can define in this way a **feasible set** as:

$$\mathcal{X}^* = \{x^* \in \mathbb{R}^n \ : p = f(x^*)\} \tag{8.11}$$

where p is the minimum. We can immediately understand that when an optimization problem is *unfeasible* is such that $\mathcal{X}^* = \emptyset$.
Surely a nice class of optimization problems is the **convex optimization** class.

Minimization

In some simple cases it is possible to find an analytical solution to an optimization problems, on the other hand in the great majority of the real optimization problems it is not possible to find in a simple way an analytical solution.
In these cases, otherwise, a **numerical solution** can be found by using an iterative algorithm which starts at a certain point x^O, and based on the **gradient function**, other points are visited. There are some functions which have "nice properties", the **convex functions** their local minimum are global mimimum. It is remarkable that the *level curves* of a convex function define **convex sets**.

8.2 Introduction to NMPC

Nonlinear Model Predictive Control (NMPC) is a general control methods which differently than the others that has been treated, provide us with a way for taking into account both the aspects related to:

- **Constraints** which are applied either on input, state or output;
- **Trade off between performance and command activity** which is managed in a systematic way how will be seen.

The **Approach** of this control technique is the following. At each time step:

1. A prediction **over a given time horizon** is performed using a model of the plant;
2. The command input is chosen as the one yielding the "best" prediction, that is the closer to the specified behaviour and this is done by solving an **online optimization algorithm**

This time the mathematical form of the plant is not assumed to be **affine** like in the Feedback linearization and Sliding Mode control, but it is in general a MIMO system characterized by

$$\begin{aligned} \dot{x} &= f(x,u) \\ y &= h(x,u) \end{aligned} \tag{8.12}$$

where x, y and u are as usual the state, the output and the input. A generalization to time varying systems could be done without particular efforts.
The state is measured in **real time**, with a proper sampling time T_s. The measurement are:

$$x(t_k),\ t_k = T_s k,\ k = 0, 1, ...$$

If the state cannot be measured an observer has to be employed.

8.3 First step: Prediction

At each time $t = t_k$ both the **state** $\hat{x}(\tau)$ and the **output** $\hat{y}(\tau)$ are predicted over the time interval $[t, t + T_p]$ by integrating (8.12). The time $T_p \geq T_s$ is called the **prediction horizon**. For each $\tau \in [t, t + T_p]$, the quantity $\hat{y}(\tau)$ (the **predicted output**) depends on:

- The **initial state** $x(t)$
- The **input signal** $u(t : \tau)$ which denotes a generic input signal in the interval $[t, \tau]$, this is an **open-loop** signal in the sense that it depends on the initial state $x(t)$, but not on $x(\tau)$ that is the "current" state.

8.4 Second step: Optimization

At each time $t = t_k$ we find an **input signal** $u(t : \tau) = u^*(t : \tau)$ such that the prediction

$$\hat{y}(x(t), u^*(t : \tau)) \equiv \hat{y}(u^*(t : \tau))$$

has the desired behaviour for all $\tau \in [t, t + T_p]$. We seek a signal that for each sub-interval $[t, \tau]$ gave the desired behaviour. This concept is formalize by defining the objective function

$$J(u(t : t + T_p)) = \int_t^{t+T_p} \left(\|\tilde{y}(\tau)\|_Q^2 + \|u(\tau)\|_R^2 \right) d\tau + \|\tilde{y}(t + T_p)\|_P$$

the term $\tilde{y}(\tau) = r(\tau) - \hat{y}(\tau)$ is the **predicted tracking error**, $r(\tau) \in \mathbb{R}^{n_y}$ is a **reference to track**, while $\| \diamond \|_X$ is the **weighted norm**. The input $u^*(t : t + T_p)$ is chosen as one **minimizing** the objective function $J(u(t : t + T_p))$, in which:

- The term $\|\tilde{y}(\tau)\|_Q^2$ is for minimizing the tracking error over a finite interval;
- The term $\|\tilde{y}(t + T_p)\|_P^2$ gives importance to the final tracking error;
- Finally the term $\|u(\tau)\|_R^2$ allows us to manage in a systematic way the performance beetween command activity and performances.

The predicted tracking error $\tilde{y}(\tau)$ depends on the *predicted output* which is derived from the integration of (8.12), so the **minimization of the functional** J is therefore subject to the constraints:

$$\dot{\hat{x}}(\tau) = f(\hat{x}(\tau), u(\tau)),\ \hat{x}(t) = x(t), \tau \in [t, t+T_p]$$
$$\hat{y}(\tau) = h(\hat{x}(\tau), u(\tau))$$

Other constraints can be present on:

- The output or the state (eg. collision avoidance, obstacles) so that we have $\hat{x}(\tau) \in X_c$ and $\hat{y}(\tau) \in Y_c$;
- The input (eg. input saturation) so that we have $u(\tau) \in U_c$;

The image below **summarizes** the steps of NMPC which are done for each $t = t_k$ many feasible solutions $u(t : t+Tp)$ are provided, in the **prediction step**, relative predicted output and state are computed by using a model of the plant, for each of these choices I compute the quantity $J(u(t : t+T_p)$, finally in the **optimization step**, I select the best choice $u^*(t : t+T_p)$.

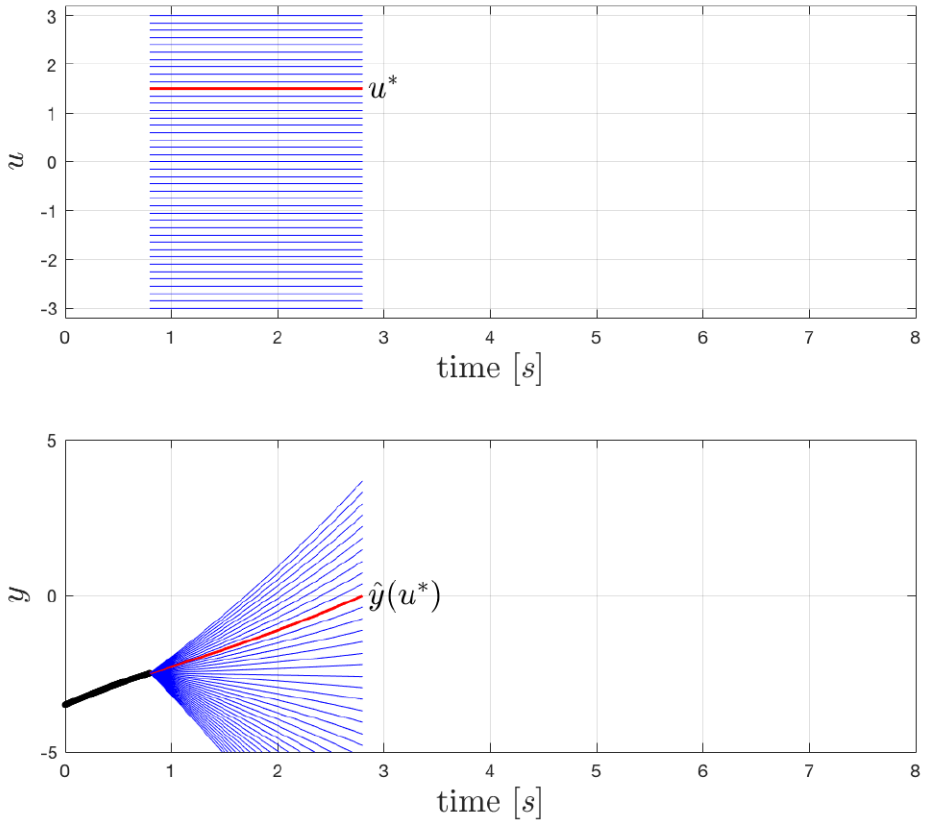


Figure 8.1: Example of a step of Prediction and Minimization

For each $t = t_k$ and $\tau \in [t, t+T_p]$ the following *optimization problem* is solved:

$$
\begin{aligned}
& u^*(t : t + T_p) = \arg\min_{u(\cdot)} J(u(t : t + T_p) \\
& \text{subject to:} \\
& \dot{\hat{x}}(\tau) = f(\hat{x}(\tau), u(\tau)),\ \hat{x}(t) = x(t) \quad (\text{initial conditions}) \\
& \hat{y}(\tau) = h(\hat{x}(\tau), u(\tau)) \\
& \hat{x}(\tau) \in \mathcal{X}_c,\ \hat{y}(\tau) \in \mathcal{Y}_c,\ u(\tau) \in \mathcal{U}_c
\end{aligned}
\tag{8.13}
$$

where T_s is the sampling time, $T_p \geq T_s$ is the **prediction horizon**. The exposed optimization problem is in general **non-convex** moreover must be solved online. Usually such optimization problem is solved by using efficient numerical algorithms.

8.5 Reciding Horizon

Suppose that at a certain time $t = t_k$, the problem (8.13) is solved and a $u^*(t : t+T_p)$ is found, as we mentioned this is an open-loop input,in fact it depends on $x(t)$, but not on $x(\tau), \tau > t$. For this reason If we applied this input in the whole interval $[t, t + T_p]$, we would not have be able to **increase the precision** or **adapt to a varying scenario**. At this point the **NMPC feedback control algorithm** is obtained applying the so-called *receding horizon strategy*:

1. At each time $t = t_k$
   - The problem (8.13) is solved
   - Apply only the first input value $u(\tau) = u^*(t = t_k)$ and keep it constant for each $\tau \in [t_k, t_{k+1}]$
2. Repeat the preceding steps for $t = t_{k+1}, t_{k+2}, ...$

8.6 Closed-loop scheme

As it has been done in the explanation of the previous techniques of Nonlinear control, now we show the control scheme for a plant controlled by using the NMPC technique. The **plant** is in the form (8.12), the block **NMPC** contains an on-line algorithm which solve the optimization problem and apply the *reciding horizon strategy*.

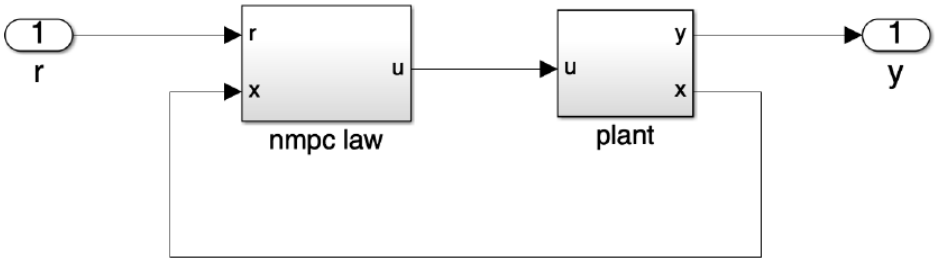


Figure 8.2: Control Scheme for NMPC

8.7 NMPC design

8.7.1 Choice of the parameters

- $\mathbf{T_s}$, sufficiently small in order to respect the Nyquist-Shannon Theorem, but not too small this would result in slow computation; sometimes cannot be chosen.
- $\mathbf{T_p}$ it is determined through a trial and error procedure in simulation keeping in mind that a "large" T_p increases the **closed-loop stability**, a "too large" T_p would result in reducing in the **short-time tracking accuracy**.

8.7.2 Choice of the weight matrices

The **initial choice** of the **weight matrices** Q, R, P can be diagonal non negative such that

$$Q_{ii} = \begin{cases} > 0 & \text{in the presence of requirements on} y_i \\ = 0 & \text{otherwise} \end{cases}$$

$$R_{ii} = \begin{cases} > 0 & \text{in the presence of requirements on} u_i \\ = 0 & \text{otherwise} \end{cases}$$

$$P_{ii} = \begin{cases} > 0 & \text{in the presence of requirements on} y_i \\ = 0 & \text{otherwise} \end{cases}$$

The weight matrices can be tuned by **trial/error procedure** considering that:

- Increasing $Q_{ii}, P_{ii} \Rightarrow$ decreasing of the energy of $x_i, y_i \Rightarrow$ reducing **oscillations** and **convergence time**;
- Increasing $R_{ii} \Leftarrow$ decreasing of the energy of $u_i \Leftarrow$ reducing **command effort** and **energy consumption**.

8.8 Discussion

Through our discussion we have mentioned yet, which are the advantages and disadvantages of such type of approach. But it is not bad repeat them:

+ NMPC is a general and flexible methodology which can face complex MIMO systems;
+ The formulation of the problem is intuitive since it is based on optimality concepts;
+ The classical trade-off between input effort and performances is treated in a systematic way through the weighted norms;
- High computational cost due to the **online resolution** of an optimization algorithm;
- In Nonlinear MPC, often the optimization problem is not convex $\Longrightarrow$ local minima may be found;
- Like all methods, if the internal dynamics is unstable we cannot manage the control problem, but this time the problem is in the mathematical model.

Moreover, it is interesting focusing the attention on the differences between NMPC and MPC, in terms of Convexity, Nonlinear plant control problem, and Non-convex constraints.

Convexity

- ☐ MPC → if the contraints are convex, finding a global minimum is guaranteed;
- ☐ NMPC → only local optima can be found.

Nonlinear plant

- ☐ MPC → requires the linearization of the dynamics on a given set of working point, this is even more complicated when one have a tracking problem.
- ☐ NMPC → can be applied directly on the nonlinear plant without any sort of linearization.

Non-Convex contraints

- ☐ MPC → it is required the convexification of the contraints;
- ☐ NMPC → the inequality can be expressed simply by using inequalities.

What inside the NMPC block?

In the Figure (8.2) inside the **NMPC block**, there is an optimization algorithm that at each time t_k solve (online) the problem related to the minimization of the functional (8.13). In `MATLAB` there are some functions can be used to this aim:

- `fminunc()`, whose aim is finding the minimum of a multivariable function with no constraints; there are two possible algorithms you can choose: **Quasi-Newton method** and **Trust region method**.
- `fmincon()`, whose aim is finding the minimum of **constrained nonlinear multivariate function**; here there more possible choices like **SQP** and some **interior point methods**.

Chapter 9

Kalman Filters

Figure 9.1: Rudolf Kalman (1930-2016)

In many real-world applications not all the state variables are measurable: sometimes either sensors may be expensive or there is no available space. In other situations could be useful having more than one sensor for the same state variable for redundancy, even though a variable is measured relevant **disturbances** and **noises** could appear.

In all these situations **Observers and Filters** are employed, they are devices which implement some **algorithms** in order to estimate the state variables from the measurement of input to the system, and the output from the system. The difference between Observer and Filter is that in the first case no assumptions are available on the disturbance/noise characteristics, in the latter case we call *filter* a device which takes into account also the presence of measurement noise and/or the presence of disturbances. This aspect in real-world situations ought to be taken into account. Widely used observers are:

1. **Linear Kalman Filter**;
2. **Extended Kalman Filter**.

[They were introduced by Kalman starting from the following scientific paper: Kalman, R.E., A new approach to linear filtering and prediction problems, in Journal of Basic Engineering, vol. 82, n. 1, 1960].

9.1 Linear Kalman Filter

This has been developed even in the continuous time (CT) that in the discrete time (DT). The **Kalman Filter** description in discrete time is simpler and more suitable for *on-line implementation* where sampled data (in discrete time) are used. Moreover the discrete time description is useful for the implementation of such method on microprocessors. For this aim consider a **Discrete time Linear Time Varying (LTV)** system described by the equations:

$$\begin{aligned} x_{k+1} &= F_k x_k + G_k u_k + d_k \\ y_k &= H_k x_k + d_k^y \end{aligned} \tag{9.1}$$

wher $k \in \mathbb{Z}$ is the time index, while d_k and d_k^y are respectively the **process disturbance** and **measurement noise**. The state x_k and the disturbances are not known, while the input u_k and the output y_k are measured.

The **objective** of Kalman's algorithm is to estimate possibly in an accurate way the state $\hat{x}_k$ of x_k from the **current** and **past** measurements of the input and the output.

Like also other observer algorithms, the Kalman filter is based on the following two operations:

1. **Prediction**. At time $k-1$ a **prediction** x_k^p is computed by using the **system model**

$$x_k^p = F_{k-1}\hat{x}_{k-1} + G_{k-1}u_{k-1} \tag{9.2}$$

2. **Update of the estimate**. At time k, the prediction previously done is *corrected* using the output y_k in order to give a *more accurate* estimate of the state. Technically speaking:

$$\begin{aligned} \hat{x}_k &= x_k^p + K_k \delta y_k \\ \delta y_k &= y_k - y_k^p = y_k - H_k y_k^p \end{aligned} \tag{9.3}$$

where K_k is chosen in order to minimize the **variance** of the *estimation error norm* $\mathbb{E}\left[\|x_k - \hat{x}_k\|_2^2\right]$ when the variance of the estimation error the error itself becomes small, y_k^p is the **predicted output**, i.e. the output computed using the system model (H_k) and the **predicted state** x_k.

In order to introduce the algorithm is useful giving some definitions:

- x_k^p: prediction of x_k at time $k-1$
- $\hat{x}_k$ is the **estimate** of x_k at time $\delta x_k = x_k - x_k^p$ is the *state prediction error*
- $\tilde{x}_k \doteq x_k - \hat{x}_k$ is the *state estimation error*
- $P_k^p \doteq \mathbb{E}[\delta x_k \delta x_k^\mathsf{T}]$ is the *covariance matrix of the prediction error*
- $P_k \doteq \mathbb{E}[\tilde{x}_k \tilde{x}_k^\mathsf{T}]$ is thr *covariance matrix of the estimation error*
- $Q_d \doteq \mathbb{E}[d_k d_k^T]$ is the *covariance matrix of* d_k
- $R_d \doteq \mathbb{E}[d_k^y d_k^{y\mathsf{T}}]$ is the *covariance matrix of* d_k^y

Before starting with the algorithm some **preliminary operations** have to be done:

1. **Design of $\mathsf{R_d}$ and $\mathsf{Q_d}$** typically based on the probabilistic available information on the noise/disturbance, commonly the are chosen to be diagonal with the variance of d_k and d_k^y on the diagonal. Most of the times they require tuning because not always one have precise information on the noise structure;
2. **Initialization** by setting:
   - $\hat{x}_0 = 0$, the initial estimated state;
   - P_0 the initial covariance matrix of the estimation error (tipically assumed to be the identity I).

After the discussion on these important details, we are ready to present the Kalman Algorithm:

Kalman Filter algorithm

1. Prediction:

$$x_k^p = F_{k-1}\hat{x}_{k-1} + G_{k-1}u_{k-1}$$
$$P_k^p = F_{k-1}P_{k-1}F_{k-1}^\mathsf{T} + Q_d$$

2. Update:

$$S_k = H_k P_k^p H_k^\mathsf{T} + R_d$$
$$K_k = P_k^p H_k^\mathsf{T} S_k^{-1}$$
$$\delta y_k = y_k - H_k x_k^p$$
$$\hat{x}_k = x_k^p + K_k \delta y_k$$
$$P_k = (I - K_k H_k) P_k^p$$

The presented algorithm uses some assumptions on the stochastic part of the problem that is related to the noise/disturbance. In particular, without entry too much into details, the noises are assumed to be iid (independent and identically distributed), *cross-uncorrelation* of noise and input-noise is required. Finally an important requisite which is always needed when an observer has to be employed is the **global observability**.
There is the important following theorem of which the proof is omitted, that state:

Theorem
The linear Kalman Filter built as presented guarantees an estimation error with **zero mean** and **minimum variance**:

$$K_k = \arg\min_{\mathcal{K}} \mathbb{E}\left[\|x_k - \hat{x}_k\|_2^2\right]$$

9.1.1 Linear Time Invariant case (LTI)

Let us consider now the case in which the matrices F, G, H are linear time invariant (LTI), that is they do not depend on the time k. The system becomes:

$$\begin{aligned} x_{k+1} &= Fx_k + Gu_k + d_k \\ y_k &= Hx_k + d_k^y \end{aligned} \tag{9.4}$$

where the quantities d_k, d_k^y are the noises.

It can be proven that if the covariance matrices Q_d, R_d of the process noise and measurement noise respectively are **definite positive**, that is $Q_d, R_d \succ 0$ and (F, H) is observable, then

$$\lim_{k\to\infty} P_k^p \to \bar{P}$$

This implies that also P_k, S_k, K_k converges to constant values P, S, K as $k \to \infty$. Using these information we can derive the asymptotic equations for the following matrices:

$$\begin{aligned}
\bar{P} &= FPF^\mathsf{T} + Q_d \\
S &= H\bar{P}H^\mathsf{T} + R_d \\
K &= \bar{P}H^\mathsf{T}S^{-1} \\
P &= (I - KH)\bar{P}
\end{aligned}$$

By eliminating P and S we can write the following two equations:

$$\bar{P}F(\bar{P} - \bar{P}H^\mathsf{T}(H\bar{P}H^\mathsf{T} + R_d)^{-1}H\bar{P})F^\mathsf{T} + Q_d \tag{9.5}$$

$$K = \bar{P}H^\mathsf{T}(H\bar{P}H^\mathsf{T} + R_d)^{-1} \tag{9.6}$$

The first of these two equations is called *discrete algebraic Riccati equation*, the second is the filter gain matrix. Then we have the predicted and estimated state equations:

$$\begin{aligned}
x_k^p &= F\hat{x}_{k-1} + Gu_{k-1} \\
\hat{x}_k &= x_k^p + K(y_k - Hx_k^p)
\end{aligned}$$

We have to replace the definition of x_{k-1} in the first equation, in this way we will obtain:

$$\begin{aligned}
x_k^p = F[x_{k-1}^p + K(y_{k-1} - Hx_{k-1}^p)] + Gu_{k-1} = \\
F[x_{k-1}^p + Ky_{k-1} - KHx_{k-1}^p] + Gu_{k-1} = \\
Fx_{k-1}^p + FKy_{k-1} - FKHx_{k-1}^p + Gu_{k-1} \overset{L \doteq FK}{=} \\
(F - LH)x_{k-1}^p + Ly_{k-1} + Gu_{k-1}
\end{aligned}$$

The equations describing the *prediction* and the *estimation* of the state, at the end of the day are:

$$x_k^p = (F - LH)x_{k-1}^p + Ly_{k-1} + Gu_{k-1} \tag{9.7}$$

$$\hat{x}_k = x_k^p + K(y_k - Hx_k^p) \tag{9.8}$$

The LTI case is a subcase of LTV, the properties on the error are required also here, the matrix K previously defined ensures the minimum variance of the estimation error. The `Matlab` function `kalman` gives L and K.

9.2 Extended Kalman Filter (EKF)

Here we want to define an algorithm for the estimation for the nonlinear case. For this purpose we define the following *discrete-time nonlinear system*:

$$\begin{aligned}
x_{k+1} &= f(x_k, u_k) + d_k \\
y_k &= h(x_k) + d_k^y
\end{aligned}$$

The matrices F_k and H_k are obtained by linearizing the nonlinear functions f and h along the estimated trajectory. In particular:

- $F_k \doteq \frac{\partial f}{\partial x}(\hat{x}_k, u_k)$: which is substancially the Jacobian of f computed at $(\hat{x}_k, u_k)$
- $H_k \doteq \frac{\partial h}{\partial x}(\hat{x}_k)$: which is the Jacobian of h computed at $\hat{x}_k$

Notation, Definitions, design of Q_d and R_d and initialization are the same than in the linear case, also the algorithm has not big differences.

Extended Kalman Filter algorithm

1. Prediction:

$$x_k^p = f(\hat{x}_{k-1}, u_{k-1})$$
$$P_k^p = F_{k-1} P_{k-1} F_{k-1}^\mathsf{T} + Q_d$$

2. Update:

$$S_k = H_k P_k^p H_k^\mathsf{T} + R_d$$
$$K_k = P_k^p H_k^\mathsf{T} S_k^{-1}$$
$$\delta y_k = y_k - h(x_k^p)$$
$$\hat{x}_k = x_k^p + K_k \delta y_k$$
$$P_k = (I - K_k H_k) P_k^p$$

There are some **advantages** in using an EKF. At first the algorithm in a relatively simple way provides a way to estimate the state variables for a nonlinear system and it works well in many real-world applications. There are also **drawbacks** for instance EKF is not optimal, due to the fact some approximation are used the stability is NOT guaranteed, at the end it is remarkable that there is *high sensibility to the initial conditions*.

9.2.1 Discussion

Prediction and correction

We have seen in EK and EKF algorithms that two steps are required: (i) the **prediction** is made by using the model of the system in a preliminar way; (ii) It is then important to **CORRECTS** the prediction by using the measurement of the output y_k, which is very important because increases the accuracy of the overall filter stability providing the possibility to **filter** measurement noise.

Current/Delayed versions

This is a more technical detail which reguards the implementation of the filter in embedded systems. In particular there are two possible approach:

- Current version. It uses all the measurement up to time k, the prediction equation is used

$$\hat{x}_k = x_k^p + K_k \delta y_k$$

- Delayed version. It uses all the measurements (samples) up to time $k-1$, for this reason the estimation equation is used

$$x_k^p = f(\hat{x}_{k-1}, u_{k-1})$$

The latter is easier to implement because often it is not possible to measure and compute the state at the same time.

9.3 Application: Spacecraft attitude determination

In the *aerospace field* Kalman filters are very used as a **virtual sensor**, in many situation there is no possibility to use the sensors because they are expensive or simply there is no space to install them. Then, the **estimation** acts a crucial role for the exposed reason. One of the application in aerospace field is the **attitude determination**.
Here the state equations are in the following form:

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{A}(\mathbf{x})\mathbf{x} + \mathbf{B}\mathbf{u} + \mathbf{B}\mathbf{d}^u \\ y &= h(\mathbf{x}) + \mathbf{d}^y\end{aligned}$$

$$\mathbf{x} \doteq \begin{bmatrix} \mathfrak{q} \\ \boldsymbol{\omega} \end{bmatrix}, \quad \mathbf{A}(\mathbf{x}) \doteq \begin{bmatrix} 0 & \frac{1}{2}\mathbf{Q} \\ 0 & -\mathbf{J}^{-1}\boldsymbol{\omega} \times \mathbf{J} \end{bmatrix}, \quad \mathbf{B} \doteq \begin{bmatrix} 0 \\ \mathbf{J}^{-1} \end{bmatrix}.$$

This is a **continuous time** system, we use the **forward Euler method** in order to discretize it by obtaining:

$$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{F}(\mathbf{x_k})\mathbf{x_k} + \mathbf{G}\mathbf{u_k} + \mathbf{d_k} \\ \mathbf{y}_k &= h(\mathbf{x_k}) + \mathbf{d}_k^y\end{aligned}$$

Where $\mathbf{F}(\hat{\mathbf{x}}_\mathbf{k}) \doteq \mathbf{I} + \tau\mathbf{A}(\mathbf{x_k})$, $\mathbf{G} \doteq \tau\mathbf{B}$ and $\mathbf{d}_k = \tau\mathbf{B}\mathbf{d}_k^u$ and τ is the *sampling time*. The equations are **quasi-LTV**: the matrix $\mathbf{F}(\hat{\mathbf{x}}_\mathbf{k})$ depends on the state. In the standard LTV system the matrix $\mathbf{F_k}$ is known and it is independent on the state. Here $\mathbf{F}$ is not exactly known and depends on the state. At the end of the day the EKF matrices can be taken like this:

$$\begin{aligned}&\mathbf{F}_k \doteq F(\hat{x}_k) \quad \text{or} \quad \mathbf{F}_k \doteq \frac{\partial f}{\partial \mathbf{x}}(\hat{x}_k, \mathbf{u}_k) \\ &\mathbf{H}_k \doteq \frac{\partial h}{\partial \mathbf{x}}(\hat{\mathbf{x}}_k)\end{aligned}$$

In general the two $\mathbf{F}$ matrices are similar but not equal.

EKF algorithm (for attitude determination)

1. Prediction:

$$\begin{aligned}x_k^p &= \mathbf{F_{k-1}}\hat{\mathbf{x}}_{k-1} + \mathbf{G}u_{k-1} \\ \mathbf{P}_k^p &= \mathbf{F}_{k-1}\mathbf{P}_{k-1}\mathbf{F}_{k-1}^\mathsf{T} + \mathbf{Q}_d\end{aligned}$$

2. Update:

$$\begin{aligned}\mathbf{S}_k &= \mathbf{H}_k\mathbf{P}_k^p\mathbf{H}_k^\mathsf{T} + \mathbf{R}_d \\ \mathbf{K}_k &= \mathbf{P}_k^p\mathbf{H}_k^\mathsf{T}\mathbf{S}_k^{-1} \\ \delta\mathbf{y}_k &= \mathbf{y}_k - h(x_k^p) \\ \hat{\mathbf{x}}_k &= \mathbf{x}_k^p + \mathbf{K}_k\delta\mathbf{y}_k \\ P_k &= (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^p\end{aligned}$$

Part II

Aerospace applications

Chapter 10

Introduction

This chapter is aimed to give a basic introduction to the **Aerospace applications** part of this course, whose focus will be particularly on spacecraft and more specifically **satellites**.

Figure 10.1: A satellite on MEO

There are many fields in which satellites are used: Earth observation, scientific research, ICT field, other planet observation and so on.
Their size is very variable: from few chilos to many tons. During its life a satellite can: reaching a target orbit for a mission for example otherwise a satellite can be exploited to check out some space instrumentation.

A satellite can work in one of the three types of orbit. The orbits can be classified in:

- Low Earth Orbit **(LEO)**: 180-2000 km (above the Earth surface)
- Medium Earth Orbit **(MEO)**: 2000-35000 km
- Hight Earth Orbit **(HEO)**: $\geq$35000 km

The topics of this part are divided in this way:

1. **Attitude dynamics and control**

    - rotations, attitude kinematics and dynamics
    - attitude control

2. **Orbital dynamics and control**:

    - orbital dynamics;
    - orbital control.

The first part of each chapter will be in order to introduce some theoretical aspect which are fundamental to understand and control a spacecraft. In the part reguarding the **spacecraft attitude** some results about mathematical tools for rotations are given.

Chapter 11

Attitude kinematics and dynamics

11.1 Rotations

Rotations are fundamental to dictate an **orientation** (often called **attitude**) to a spacecraft or an aircraft. This is in order to *capture the solar energy*, *make some communication* by using antennas positioned on the Earth surface.
In general a spacecraft can be seen as a *rigid body*, whose motion can be described mathematically using a combination of **rotation** and **translation**.
The objective of this section is to provide a **set of math tools** in order to sistematically describe this part of the motion. In particular, will be discussed the following techniques:

1. Direction cosine matrices (DCM);
2. Euler angles;
3. Angle-axis representation;
4. Quaternions;

All these four techniques are linked in some way, how it will be seen.

11.1.1 Reference frames (RF)

Talking about rotations is fundamental to give the definition of **reference frame**, from the moment that a rotation is described between **two different reference frames**.

Definition(Reference Frame)
An (orthogonal) **frame of reference** $\mathcal{R} = (O, \mathbf{i}, \mathbf{j}, \mathbf{k})$ is composed by an origin O and a set of three **unit vectors** $\{\mathbf{i}, \mathbf{j}, \mathbf{k}\}$ that are *mutually orthogonal*.

Given a reference frame $F = \{O, \mathbf{i}, \mathbf{j}, \mathbf{k}\}$ a vector $\mathbf{r} \in \mathbb{R}^3$ can be expressed in two ways:

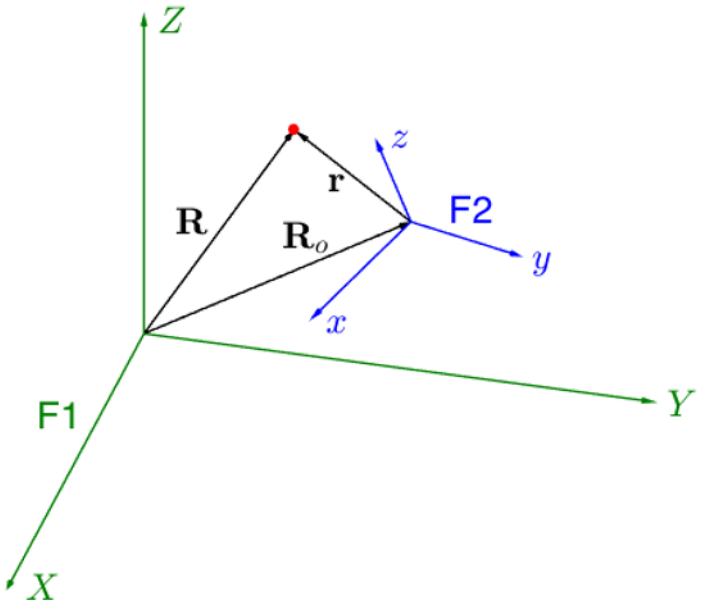
- As a **physical vector**: it is an abstract concept according to which a vector can be expressed as a linear combination of the unit vector, that is: $r = x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$
- As a **coordinate vector**: it is a column vector linked to the specific reference frame $F_\diamond$, that is

$$\mathbf{r} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Then, a generic point/vector in a reference frame has a different representation in another reference frame. The relation beetween two reference frames $F1 = \{O, \mathbf{i}, \mathbf{j}, \mathbf{k}\}$ and $F2 = \{O, \mathbf{I}, \mathbf{J}, \mathbf{K}\}$, introduces to the topic of **Direction cosine matrices (DCM)**.

11.1.2 Direction cosine matrices (DCM)

In order to introduce a way to link the entities in two reference frames $F1$ and $F2$, let us give a graphical representation of them:



In the above figure the following quantities are shown and a generic particle (in red) is drawn:

$$\begin{aligned} &\mathbf{R} = X\mathbf{I} + Y\mathbf{J} + Z\mathbf{K} && \text{Position of the particle in F1} \\ &\mathbf{r} = x\mathbf{i} + y\mathbf{j} + z\mathbf{k} && \text{Position of the particle in F2} \\ &\mathbf{R_O} = X_O\mathbf{I} + Y_O\mathbf{J} + Z_O\mathbf{K} && \text{Position in F1 of the O of F2} \end{aligned} \tag{11.1}$$

From the geometry follows that $\mathbf{R} = \mathbf{R_O} + \mathbf{r}$. One wonder at this point **what is the relation between X, Y, Z and x, y, z**, using the given former properties we can write:

$$\begin{aligned} X &= \mathbf{R}\cdot\mathbf{I} = (\mathbf{R_O} + \mathbf{r})\cdot\mathbf{I} = X_O + x\mathbf{I}\cdot\mathbf{i} + y\mathbf{I}\cdot\mathbf{j} + z\mathbf{I}\cdot\mathbf{k} \\ Y &= \mathbf{R}\cdot\mathbf{J} = (\mathbf{R_O} + \mathbf{r})\cdot\mathbf{J} = Y_O + x\mathbf{J}\cdot\mathbf{i} + y\mathbf{J}\cdot\mathbf{j} + z\mathbf{J}\cdot\mathbf{k} \\ Z &= \mathbf{R}\cdot\mathbf{K} = (\mathbf{R_O} + \mathbf{r})\cdot\mathbf{K} = Z_O + x\mathbf{K}\cdot\mathbf{i} + y\mathbf{K}\cdot\mathbf{j} + z\mathbf{K}\cdot\mathbf{k} \end{aligned}$$

Expressed in matrix form:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} X_O \\ Y_O \\ Z_O \end{bmatrix} + \begin{bmatrix} \mathbf{I}\cdot\mathbf{i} & \mathbf{I}\cdot\mathbf{j} & \mathbf{I}\cdot\mathbf{k} \\ \mathbf{J}\cdot\mathbf{i} & \mathbf{J}\cdot\mathbf{j} & \mathbf{J}\cdot\mathbf{k} \\ \mathbf{K}\cdot\mathbf{i} & \mathbf{K}\cdot\mathbf{j} & \mathbf{K}\cdot\mathbf{k} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \mathbf{T} \doteq \begin{bmatrix} \mathbf{I}\cdot\mathbf{i} & \mathbf{I}\cdot\mathbf{j} & \mathbf{I}\cdot\mathbf{k} \\ \mathbf{J}\cdot\mathbf{i} & \mathbf{J}\cdot\mathbf{j} & \mathbf{J}\cdot\mathbf{k} \\ \mathbf{K}\cdot\mathbf{i} & \mathbf{K}\cdot\mathbf{j} & \mathbf{K}\cdot\mathbf{k} \end{bmatrix}$$

The dot products $\mathbf{I}\cdot\mathbf{i}$, $\mathbf{I}\cdot\mathbf{j}$... represents the **direction cosine angles** which the axis of one reference frame forms with respect to the other. $\mathbf{T}$ is the *direction cosine matrix*(**DCM**). In literature the DCM can be expressed as:

$$\mathbf{T} = \begin{bmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{bmatrix} = [T_{ij}]$$

and from the moment that translation and rotation can be studied indipendently, without loss of generality we can assume $\mathbf{R}_O = 0$, then the $\mathbf{T}$ becomes a **transformation matrix** and in

particular

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \mathbf{T} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Furthermore, there are *two different interpretation:*

- **Coordinate transformation** from F2 to F1 another term used to indicate this concept is *alias*;
- **Rotation** of vectors in a given fixed frame, so from F1 to F2, where F2 is the fixed one, another term to indicate this is *alibi*.

Some examples

Let us consider some examples in 2D, the DCM assumes the form:

$$\mathbf{T} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & cos\theta \end{bmatrix}$$

From the moment that $z = 0$, the third row and the third column of the complete matrix $\mathbf{T}$ disappear.

(#1) 2D-Coordinate transformation (alias) F2→F1

Let us assume that a reference frame F2 is rotated with respect to F1 of an angle $\theta = 0.15\pi$, and a particle in F2 is given by

$$\mathbf{r} = 0.5\mathbf{i} + 0.3\mathbf{j} = \begin{bmatrix} 0.5 \\ 0.3 \end{bmatrix}$$

The particle coordinates in F1 are obtained through the following transformation

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \mathbf{Tr}$$

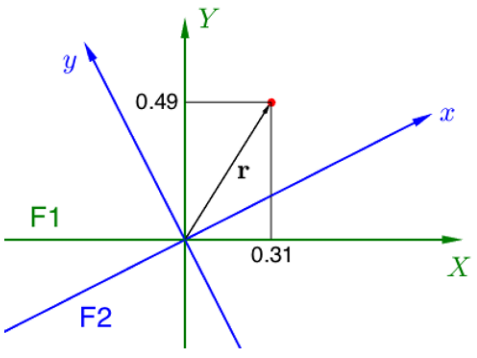


Figure 11.1: 1st interpretation: Coordinate transformation

(#2) 2D-Rotation (alibi) F1→F2

This time, let us consider a **vector r** that in the reference frame F2 is represented by

$$\mathbf{r} = \begin{bmatrix} 0.5 \\ 0.3 \end{bmatrix}$$

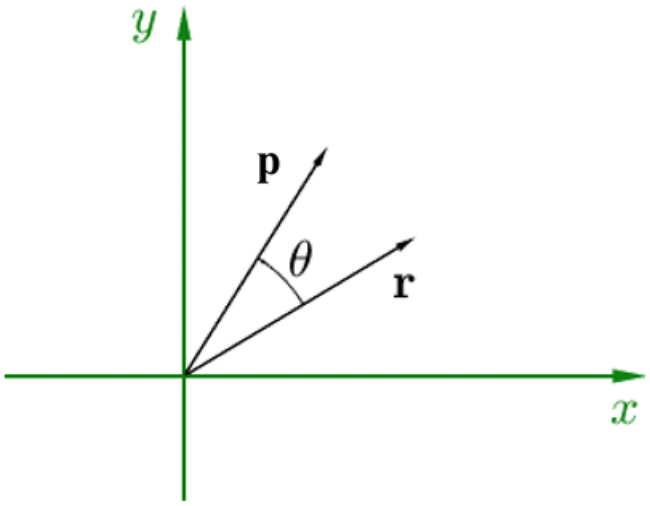


Figure 11.2: 2nd interpretation: rotation in a fixed frame

We can obtain a **rotated vector r′** in the same reference frame(fixed) applying the transformation **T**:

$$\mathbf{r}' = \mathbf{T} \cdot \mathbf{r}$$

(#3) Rotation of a square in 2D The concept of rotation can be generalized to a 2D-figure like a square for example. A square in 2D can be represented by a matrix $\mathbf{S} \in \mathbb{R}^{2,4}$, which contains by column the (2D)coordinates of the vertices.
If we want rotate a square of an angle θ we can apply the DCM (computed in the angle θ) obtaining a **rotated square S′** as follows:

$$\mathbf{S}' = \mathbf{T} \cdot \mathbf{S}$$

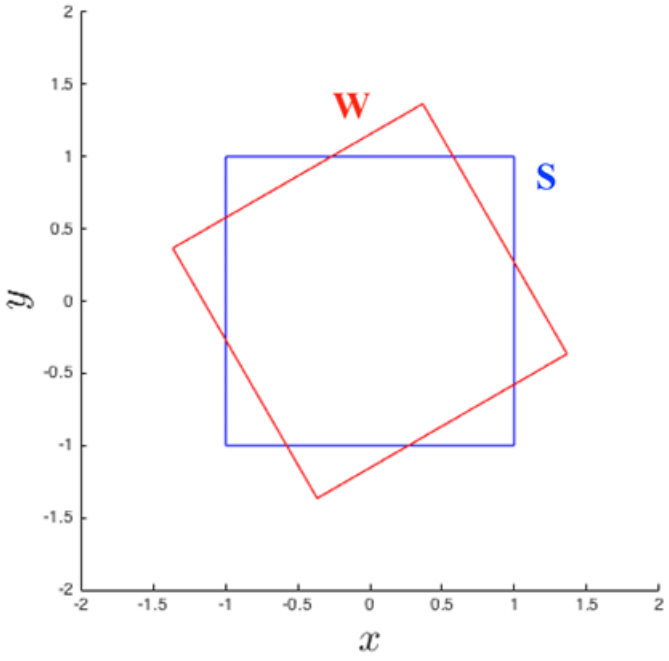


Figure 11.3: Rotation of a square

11.1.3 Euler angles

In **three dimensions** we define the *elementary rotation matrices*

$$\mathbf{T}_1(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & cos\phi \end{bmatrix} \quad \text{Rotation about } X \text{ (or x) of } \phi$$

$$\mathbf{T}_2(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \quad \text{Rotation about } Y \text{ (or y) of } \theta$$

$$\mathbf{T}_3(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ sin\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{Rotation about } Z \text{ (or z) of } \psi$$

The interesting thing is that **any rotation in 3D** can be expressed as a product of $\mathbf{T}_1$, $\mathbf{T}_2$, $\mathbf{T}_3$, moreover the three angles ϕ, θ, ψ are the so-called **Euler angles**.
There are 12 possible combinations of the three elementary matrices of rotations (with non sequentially repeated indexes) they can be divided in **two groups**:

- **6 Tait-Bryan rotations** the most used are 123 and 321
- **6 proper Euler rotations** the most common is 313

As they are very used their expression is reported below:

Tait-Bryan $\mathbf{T}_{123}$

$$\mathbf{T}_{123} = \begin{bmatrix} \cos\theta\cos\psi & -cos\theta\sin\psi & \sin\theta \\ \cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & \cos\phi\cos\psi - \sin\phi\sin\theta\sin\psi & -\sin\phi\cos\theta \\ \sin\phi\sin\psi - \cos\phi\sin\theta\cos\psi & \sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi & \cos\phi\cos\theta \end{bmatrix}$$

Tait-Bryan $\mathbf{T}_{321}$

$$\mathbf{T}_{321} = \begin{bmatrix} \cos\theta\cos\psi & -\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix}$$

Proper Euler $\mathbf{T}_{313}$

$$\mathbf{T}_{313} = \begin{bmatrix} \cos\phi\cos\psi - \sin\phi\cos\theta\sin\psi & -\cos\phi\sin\psi - \sin\phi\cos\theta\cos\psi & \sin\phi\sin\theta \\ \sin\phi\cos\psi + \cos\phi\cos\theta\sin\psi & -sin\phi\sin\psi + \cos\phi\cos\theta\cos\psi & -cos\phi\sin\theta \\ \sin\theta\sin\psi & \sin\theta\cos\psi & \cos\theta \end{bmatrix}$$

11.1.4 Rotation matrices general properties

The following are **general properties** related to the rotation matrices. **Notation**: we indicate with $\mathbf{T}_\diamond$ any composition of the elementary rotation matrices (with non-sequentially repeated elements).
Matrices $\mathbf{T}_\diamond$ have the following properties:

- ✦ They are **linear transformations**, from the product of linear matrices;
- ✦ They are **orthogonal matrices**, that is matrices whose columns are composed of real entries and are **orthonormal**. The most important property is:

$$\mathbf{T}_\diamond^T = \mathbf{T}_\diamond^{-1}, \quad \mathbf{T}_\diamond^T \mathbf{T}_\diamond = \mathbf{T}_\diamond \mathbf{T}_\diamond^T = \mathbf{I}$$

orthogonal transformations preserve:

  - the **length** of the vectors;
  - the **angles** between the vectors;

$$\text{length} + \text{angles} \Longrightarrow \textbf{shape preserved}$$

- ✦ Their **eigenvalues** are always $\{1, \pm e^{j\beta}\}$, their **determinant** always equal to 1;

Singularities, Gimbal lock

For certain values of θ some matrices can present some **singularities**, for example the matrix $\mathbf{T}_{123}$. With $\theta = \frac{\pi}{2}$ only the sum $(\psi + \phi)$ can be determined and this results in a **loss of degree of freedom** $\Longrightarrow$ this phenomenon is also known as **gimbal lock** and affect all the *gyroscopes* in which two out of three axes are aligned and this results in a loss of a degree of freedom, in *normal situations* the three axis, instead, can rotate indipendently. In order to overcome this problem, we use non minimal representation in which instead of only three variables, are used **four variables**.
More in general, **there is always a gimbal lock** when:

- ✦ In the **Tait-Bryan rotations**, $\cos\theta = 0$;
- ✦ In the **Proper Euler rotations**, when occurs that $\sin\theta = 0$

This singularities cause problems in the *kinematic equation* like divergent behaviour, this is the reason why alternative method to manage these situations are required.

11.1.5 Euler's rotation theorem

The following theorem is very important because it introduces an alternative way to describe rotations:

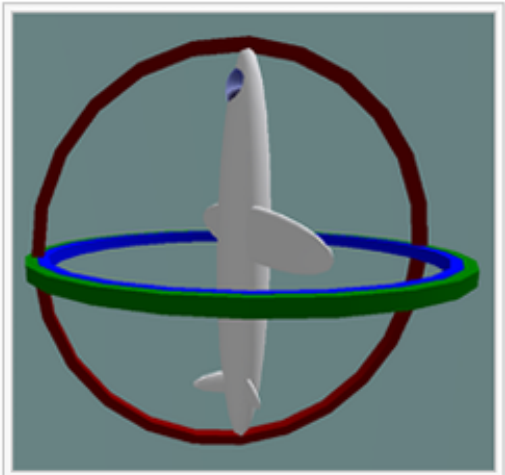
Theorem (angle-axis representation)

(i) Any rotation of a rigid body where a point is fixed is **equivalent** to a rotation in which the rotation axis passes through the fixed point;

(ii) The rotation axis is the **eigenvector u** corresponding to the eigenvalues 1 of the rotation matrix.

Proof
The rotation matrix $\mathbf{T}_\diamond$ has a unitary eigenvalue, it holds (from the definition) that: $\mathbf{T}_\diamond \mathbf{u} = \mathbf{u}$, so the rotation does not change the direction $\mathbf{u}$, which is the rotation axis. □

From this theorem, follows that *any rotation* can be described by using two quantities involving four variables:



Gimbal lock: two out of the three gimbals are in the same plane, one degree of freedom is lost

Figure 11.4: Gimbal lock example

- An angle β (1 variable)
- A rotation axis $\mathbf{u}$ (3 variables)

$$\mathbf{T} \equiv \mathbf{T}(\beta, \mathbf{u})$$

11.1.6 Quaternions

Quaternions are mathematical object introduced by Hamilton. They are: (i) an extension of the concept of *Complex numbers*, (ii) are useful and more robust tool to describe **rotations**. **Complex numbers** are element of a 2D-vector space $\mathbb{C}$ whose basis is $\{1, \mathbf{i}\}$. Each element can be expressed as a linear combination of these two elements.
In an equivalent way we can state that:

DEFINITION (QUATERNIONS)
The quaternions are element of a 4D-dimensional vector space $\mathbb{Q}$ whose basis is $\{1, \mathbf{i}, \mathbf{j}, \mathbf{k}\}$, where is defined a particular **product** indicated with $\otimes$. The following properties hold:

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{i} \otimes \mathbf{j} \otimes \mathbf{k} = -1$$
$$\mathbf{i} \otimes \mathbf{j} = -\mathbf{j} \otimes \mathbf{i} = \mathbf{k}$$
$$\mathbf{j} \otimes \mathbf{k} = -\mathbf{k} \otimes \mathbf{j} = \mathbf{i}$$
$$\mathbf{k} \otimes \mathbf{i} = -\mathbf{i} \otimes \mathbf{k} = \mathbf{j}$$

As we introduced a new mathematical object we have to define a set of properties and some notation.

Notation

One can indicate a quaternion using one of the following notations:

$$\begin{aligned} \mathfrak{q} &= q_0 + q_1\mathbf{i} + q_1\mathbf{j} + q_2\mathbf{k} = \\ &= q_0 + \mathbf{q} = \\ &= (q_0, q_1, q_2, q_3) = \\ &= (q_0, \mathbf{q}) = \begin{bmatrix} q_0 \\ \mathbf{q} \end{bmatrix} \end{aligned}$$

- q_0 is the *real part*
- $\mathbf{q}$ is the so-called *vector part*
- if $q_0 = 0$, the resulting quaternion $\mathfrak{q} = (0, \mathbf{q})$ is said to be *pure*

Null element

The **null element** of the quaternion vector space is

$$\mathfrak{O} = (0, \mathbf{0})$$

Complex conjugate

The **complex conjugate** $\mathfrak{q}^*$ of a quaternion $\mathfrak{q} = q_0 + \mathbf{q}$ is

$$\mathfrak{q}^* = q_0 - \mathbf{q} = \begin{bmatrix} q_0 \\ -\mathbf{q} \end{bmatrix} = (q_0, -\mathbf{q})$$

Norm

We simply define the **quaternion norm** as

$$|\mathfrak{q}| = \|\mathfrak{q}\| = |\mathfrak{q}^*| = \sqrt{\mathfrak{q} \cdot \mathfrak{q}^*} = \sum_{i=0}^{3} {q_i}^2$$

Reciprocal element

Given a quaternion $\mathfrak{q}$, the reciprocal element $\mathfrak{q}^-1$ is

$$\mathfrak{q}^{-1} = \frac{\mathfrak{q}^*}{\|\mathfrak{q}\|}$$

Sum

The **sum** of two quaternions $\mathfrak{q}$ and $\mathfrak{p}$ is

$$\mathfrak{q} + \mathfrak{p} = q_0 + p_0 + \mathbf{q} + \mathbf{p}$$

Dot product

The **dot product** of two quaternions $\mathfrak{q}$ and $\mathfrak{p}$ is

$$\mathfrak{q} \cdot \mathfrak{p} = \sum_{i=0}^{3} q_i p_i$$

Quaternion product (Hamilton product)

Given two quaternions $\mathfrak{q}$ and $\mathfrak{p}$ we define the **Hamilton product** (which is ASSOCIATIVE and NON-COMMUTATIVE) as:

$$\begin{aligned} \mathfrak{q} \otimes \mathfrak{p} &= (q_0 + \mathbf{q}) \otimes (p_0 + \mathbf{p}) = (\ldots) \\ &= (q_0 p_0 - \mathbf{q} \cdot \mathbf{p}) + (q_0 \mathbf{p} + p_0 \mathbf{q} + \mathbf{q} \times \mathbf{p}) \end{aligned}$$

Where $\mathbf{q} \times \mathbf{p}$ is the **cross product** (which is NON-ASSOCIATIVE and ANTI-COMMUTATIVE)

$$\mathbf{q} \times \mathbf{p} = \begin{bmatrix} q_2 p_3 - q_3 p_2 \\ q_3 p_1 - q_1 p_3 \\ q_1 p_2 - q_2 p_1 \end{bmatrix}$$

Also the quaternion vector space has an **identity element** equal to

$$\mathfrak{I} = (1, \mathbf{0})$$

11.1.7 Describe rotations through quaternions

Through this subsection we will entry into details of why quaternions can be useful to describe rotations. At first, we have to say that given a vector $\mathbf{r}$ with component x, y, z, its 4D extension is $(0, \mathbf{r})$. We have seen thanks to the **angle-axis** representation that a rotation of the vector $\mathbf{r}$ of an angle β and rotation axis $\mathbf{u}$ is $\mathbf{p} = \mathbf{T}(\beta, \mathbf{u})\mathbf{r}$.
For given β and $\mathbf{u}$, the associated unit quaternion is defined as

$$\mathfrak{q} \doteq \left(\cos\frac{\beta}{2},\ \mathbf{u} \sin\frac{\beta}{2} \right) = \left(\cos\frac{\beta}{2},\ u_1 \sin\frac{\beta}{2},\ u_2 \sin\frac{\beta}{2},\ u_3 \sin\frac{\beta}{2} \right)$$

The *components* of this vector are called **Euler parameters**.

Theorem The *rotated vector* $\mathbf{p}$ can be defined as

$$(0, \mathbf{p}) = \mathfrak{q} \otimes (0, \mathbf{r}) \otimes \mathfrak{q}^*$$

From the moment that we can associate each *angle-axis* representation to a specific quaternion, we can retrieve the quaternions of the *elementary rotation matrices* as follows:

$$\mathbf{T}_1(\phi) \longleftrightarrow \mathfrak{q}_1 = \left(\cos\frac{\phi}{2}, \sin\frac{\phi}{2}, 0, 0 \right)$$
$$\mathbf{T}_2(\theta) \longleftrightarrow \mathfrak{q}_2 = \left(\cos\frac{\theta}{2}, 0, \sin\frac{\theta}{2}, 0 \right)$$
$$\mathbf{T}_3(\psi) \longleftrightarrow \mathfrak{q}_3 = \left(\cos\frac{\psi}{2}, 0, 0, \sin\frac{\psi}{2} \right)$$

At this point we can note that any rotation can be expressed in the form an Hamilton product between the **quaternions**, that is

$$\mathfrak{q} = \mathfrak{q}_1 \otimes \mathfrak{q}_2 \otimes \cdots \otimes \mathfrak{q}_n$$

The **inverse rotation** is

$$\mathfrak{q}^{-1} = \mathfrak{q}^*$$

11.2 Attitude kinematics

The objective of this paragraph is to derive the **kinematic equations** for a generic *rigid body* in rotational motion, these equations are crucially important for the **spacecraft attitude control**.
In general, the dynamic and kinematic equations can be seen as a **series of two nonlinear systems**.

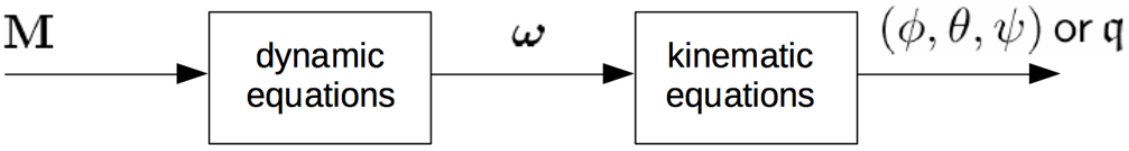


Figure 11.5: Dynamic-Kinematic system

In this figure:

- The **dynamic equations** block has got as input an external torque $\mathbf{M}$ and as output the angular velocity vector $\boldsymbol{\omega}$.
- The **kinematic equations** block has got as input the output of dynamic block, then the angular velocity $\boldsymbol{\omega}$, as output one between the **Euler angles** (ϕ, θ, ψ) and the rotation **quaternion** $\mathfrak{q}$.

Then, the overall system has: input $\mathbf{M}(t)$ and ouput (ϕ, θ, ψ) or $\mathfrak{q}$ which represents the output to control. For instance the rotation is crucial in order to obtain a certain orientation of the spacecraft toward a certain direction.

The classical setting in this case is considering that the rigid body is rotating with respect to some **observer reference frame** OF, with an *angular velocity* $\boldsymbol{\omega} = \omega_1 \mathbf{b}_1 + \omega_2 \mathbf{b}_2 + \omega_3 \mathbf{b}_3$ in which:

- $\omega = |\boldsymbol{\omega}|$ is the speed of rotation;
- The unit vector $\frac{\boldsymbol{\omega}}{|\boldsymbol{\omega}|}$ is the *rotation axis*.

We can individuate two *reference frames*: the **observer frame** (OF) and the **body frame** (BF), when we introduce some reference frames we must give some information like: where is the origin? Which are the unit vectors? And the axis? In our case we have:

1. **Observer frame**(OF) $\begin{cases} \text{origin: somewhere} \\ \text{unit vectors} : \{\mathbf{i}_1 \mathbf{i}_2, \mathbf{i}_3\} \\ \text{axis:} X, Y, Z \end{cases}$
2. **Body frame**(BF) $\begin{cases} \text{origin: CoM of Rigid body } \rightarrow \text{rotates with the body} \\ \text{unit vectors: } \{\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3\} \\ \text{axis} : x, y, z \end{cases}$

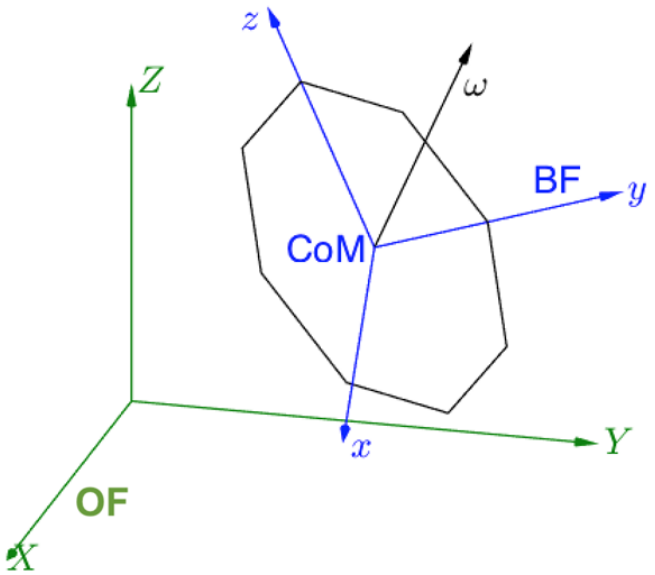


Figure 11.6: Observer and Body frames for a generic rigid body

From the moment that the rotation from OF to BF can be described by using either the Euler angles or the quaternions, we can derive some kinematic equations for both type of representation. Before doing this, it is important to give the definition of what is the **derivative of a vector**.

11.2.1 Vector derivative

Any physical vector can be expressed in the form

$$\mathbf{r} = x\mathbf{b}_1 + y\mathbf{b}_2 + z\mathbf{b}_3 \tag{11.2}$$

If one makes the **derivative** of such vector applying the basic rules from the differential calcus, it will be obtained

$$\dot{\mathbf{r}} = \dot{x}\mathbf{b}_1 + \dot{y}\mathbf{b}_2 + \dot{z}\mathbf{b}_3 + x\dot{\mathbf{b}}_1 + y\dot{\mathbf{b}}_2 + z\dot{\mathbf{b}}_3 \tag{11.3}$$

Between two different time instants, the difference vector is

$$\delta\mathbf{b}_i = \delta\boldsymbol{\theta} \times \mathbf{b}_i \tag{11.4}$$

but if the vector $\mathbf{r}$ is rotating with an angular velocity $\boldsymbol{\omega}$, then the angle $\delta\boldsymbol{\theta}$ is equal to $\delta\theta \doteq \boldsymbol{\omega}\delta t$. When $\delta t \to 0$, the vectors before and after the infinitesimal rotation are very similar, then the vector $\delta\mathbf{b}_i$ becomes $\delta\mathbf{b}_i = \boldsymbol{\omega} \times \mathbf{b}_i$. Putting all together we have:

$$\dot{\mathbf{r}} = \dot{\mathbf{r}}_B + \boldsymbol{\omega} \times \mathbf{r} \tag{11.5}$$

Where $\dot{\mathbf{r}}_B = \dot{x}\mathbf{b}_1 + \dot{y}\mathbf{b}_2 + \dot{z}\mathbf{b}_3$ is the derivative of $\mathbf{r}$ in the **body frame**, while $\dot{\mathbf{r}}$ is the derivative of $\mathbf{r}$ in the **observer frame**.

11.2.2 Euler angles kinematic

We want to retrieve a relation between the input $\omega(t)$ and the **Euler angles** (ϕ, θ, ψ). Unfortunately, for the rotations it is not so easy as in the case of *traslational motion*, moreover the relation between the Euler angles derivatives and the components of the angular velocity is NOT the identity matrix. We will provide the kinematic equations for the most used *Tait-Bryan* rotations and for the *Proper Euler angle* rotation. [The derivation of such formulas is not trivial at all].
It is interesting to note that, in the case we use this type of representation we have some **singularities** in the equations due to the *gimbal-lock* phenomenon. On the other hand, by using a non-minimal representation, rotations described through the quaternions is free from this problem.

Tait-Bryan 321

It can be demonstrated that

$$\begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\end{bmatrix} = \begin{bmatrix}1 & 0 & -\sin\theta\\ 0 & \cos\phi & \sin\phi\cos\theta\\ 0 & -\sin\phi & \cos\phi\cos\theta\end{bmatrix}\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} \tag{11.6}$$

We can invert the matrix and we can obtain the *Tait-Bryan 321 kinematic equation*:

$$\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = \frac{1}{\cos\theta}\begin{bmatrix}\cos\theta & \sin\phi\sin\theta & \cos\phi\sin\theta\\ 0 & \cos\phi\cos\theta & -\sin\phi\cos\theta\\ 0 & \sin\phi & \cos\phi\end{bmatrix}\begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\end{bmatrix} \tag{11.7}$$

When $\cos\theta = 0$ the gimbal-lock occurs, then we have a singularity in the kinematic equations.

Tait-Bryan 123

It can be demonstrated that

$$\begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\end{bmatrix} = \begin{bmatrix}\cos\theta\cos\psi & \sin\psi & 0\\ -\cos\theta\sin\psi & \cos\psi & 0\\ \sin\theta & 0 & 1\end{bmatrix}\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} \tag{11.8}$$

We can invert the matrix and we can obtain the *Tait-Bryan 123 kinematic equation*:

$$\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = \frac{1}{\cos\theta}\begin{bmatrix}\cos\psi & -\sin\psi & 0\\ \cos\theta\sin\psi & \cos\theta\cos\psi & 0\\ -\sin\theta\cos\psi & \sin\theta\sin\psi & \cos\theta\end{bmatrix}\begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\end{bmatrix} \tag{11.9}$$

When $\cos\theta = 0$ the gimbal-lock occurs, then we have a singularity in the kinematic equations.

Proper-Euler 313

It can be demonstrated that

$$\begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\end{bmatrix} = \begin{bmatrix}\sin\theta\sin\psi & \cos\psi & 0\\ \sin\theta\cos\psi & -\sin\psi & 0\\ \cos\theta & 0 & 1\end{bmatrix}\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} \tag{11.10}$$

We can invert the matrix and we can obtain the *Tait-Bryan 313 kinematic equation*:

$$\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = \frac{1}{\sin\theta}\begin{bmatrix}\sin\psi & \cos\psi & 0\\ \sin\theta\cos\psi & -\sin\theta\sin\psi & 0\\ -\cos\theta\sin\psi & -\cos\theta\cos\psi & \sin\theta\end{bmatrix}\begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\end{bmatrix} \tag{11.11}$$

When $\cos\theta = 0$ the gimbal-lock occurs, then we have a singularity in the kinematic equations.

11.2.3 Quaternion kinematic

The **quaternion kinematic** can be written in different equivalent forms:

$$\dot{\mathfrak{q}} = \frac{1}{2}\mathfrak{q}\otimes(0,\boldsymbol{\omega}) \tag{11.12}$$

$$\dot{\mathfrak{q}} = \frac{1}{2}\boldsymbol{\Omega}\mathfrak{q} \tag{11.13}$$

$$\dot{\mathfrak{q}} = \frac{1}{2}\mathbf{Q}\boldsymbol{\omega} \tag{11.14}$$

where the matrices $\boldsymbol{\Omega}$ and $\mathbf{Q}$ are defined as follows

$$\boldsymbol{\Omega} \doteq \begin{bmatrix}0 & -\omega_1 & -\omega_2 & -\omega_3\\ \omega_1 & 0 & \omega_3 & -\omega_2\\ \omega_2 & -\omega_3 & 0 & \omega_1\\ \omega_3 & \omega_2 & -\omega_1 & 0\end{bmatrix}, \quad \mathbf{Q} \doteq \begin{bmatrix}-q_1 & -q_2 & -q3\\ q_0 & -q_3 & q_2\\ q_3 & q_0 & -q_1\\ -q_2 & q_1 & q_0\end{bmatrix}$$

11.2.4 Discussion

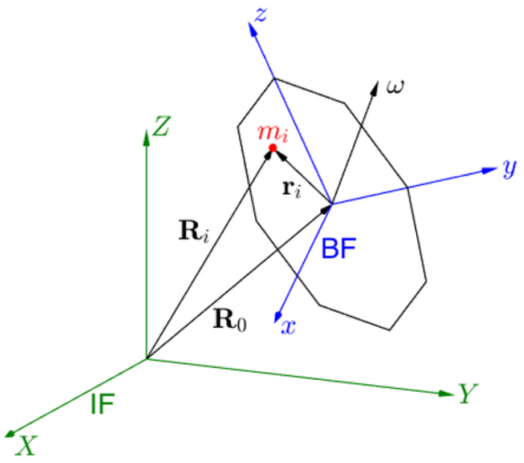
Through these paragraphs we have found the **kinematic equations** in the general form

$$\dot{\mathbf{x}} = A(\mathbf{x})\boldsymbol{\omega} \tag{11.15}$$

where $\mathbf{x} = (\phi,\theta,\psi)$ or $\mathbf{x} = \mathfrak{q}$. This is a **state equation**, with state $\mathbf{x}$ and input $\boldsymbol{\omega}$. State equations allow us to predict the time evolution of a system given a certain initial condition x_0 and a given input $\boldsymbol{\omega}(t)$. The solution can be found sometimes analitically, otherwise numerical methods have to be exploited, but in this case a solution - even if approximately - can be always provided.

11.3 Attitude dynamics

In the previous chapter we analysed the equations for the kinematic part of rotations. In this chapter, instead we want to describe the **attitude dynamic equations** how mentioned they play a fundamental role for the atttitude control of spacecrafts.
We use the same setting with two reference frames, the inertial frame (constant v_O) and the body frame, a generic particle m_i of the rigid body has got the following properties:



$$\mathbf{R}_i = X\mathbf{i}_1 + Y\mathbf{i}_2 + Z\mathbf{i}_3 = \mathbf{R}_O + \mathbf{r}_i$$
$$R_O = X_O\mathbf{i}_1 + Y_O\mathbf{i}_2 + Z_O\mathbf{i}_3$$
$$\dot{\mathbf{R}}_i = \dot{\mathbf{R}}_O + \dot{\mathbf{r}}_{iB} + \boldsymbol{\omega} \times \mathbf{r}_i$$
$$\mathbf{r}_i = x\mathbf{b}_1 + y\mathbf{b}_2 + z\mathbf{b}_3$$
$$\mathbf{r}_{iB} = \dot{x}\mathbf{b}_1 + \dot{y}\mathbf{b}_2 + \dot{z}\mathbf{b}_3$$
$$\boldsymbol{\omega} = \omega_1\mathbf{b}_1 + \omega_2\mathbf{b}_2 + \omega_3\mathbf{b}_3$$

11.3.1 Angular momentum

Let us start introducing what is the **angular momentum** of a particle m_i. It is defined as

$$\mathbf{H}_i = r_i \times m\dot{\mathbf{R}}_i = \mathbf{r}_i \times m_i\big(\dot{\mathbf{R}}_O + \dot{\mathbf{r}}_{iB} + \boldsymbol{\omega} \times \mathbf{r}_i\big) \tag{11.16}$$

Being a rigid body the position of the single particle m_i does not change in time, so the term $\dot{\mathbf{r}}_{iB} = 0$, then it will be obtained

$$\begin{aligned}\mathbf{H}_i &= \mathbf{r}_i \times m_i\left(\dot{\mathbf{R}}_O + \boldsymbol{\omega} \times \mathbf{r}_i\right)\\ &= -\dot{\mathbf{R}}_O \times m_i\mathbf{r}_i + \mathbf{r}_i \times m_i(\boldsymbol{\omega} \times \mathbf{r}_i)\end{aligned}$$

If we extend the reasing for all the particles of the body we will obtain:

$$\begin{aligned}\mathbf{H} &= -\sum_i \dot{\mathbf{R}}_O \times m_i\mathbf{r}_i + \sum_i \mathbf{r}_i \times m_i(\boldsymbol{\omega} \times \mathbf{r}_i)\\ &= -\dot{\mathbf{R}}_O \times \sum_i m_i\mathbf{r}_i + \sum_i \mathbf{r}_i \times m_i(\boldsymbol{\omega} \times \mathbf{r}_i)\\ &= \sum_i \mathbf{r}_i \times m_i(\boldsymbol{\omega} \times \mathbf{r}_i)\end{aligned}$$

in which the term $\sum_i m_i \mathbf{r}_i = \mathbf{0}$ by definition of Center of Mass (CoM). In a rigid body the particle are not discrete, the particles for which we can compute the angular momentum are *infinitesimal*: $m_i \to dm$. We have substancially replace the summation with the integral obtaining at the end:

$$\mathbf{H} = \int_B \mathbf{r} \times (\boldsymbol{\omega} \times \mathbf{r}) \, dm \tag{11.17}$$

11.3.2 Inertia matrix (o tensor)

If we perform the products, we will obtain in body coordinates:

$$\mathbf{H} = \begin{bmatrix} J_{11} & J_{12} & J_{13} \\ J_{21} & J_{22} & J_{23} \\ J_{31} & J_{32} & J_{33} \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} = \boldsymbol{J\omega} \tag{11.18}$$

The matrix $\mathbf{J}$ is the **inertia matrix** or **inertia tensor** and it depends only on the properties of the body. The elements J_{ij} are defined as follows:

$$\text{moments of inertia} \begin{cases} J_{11} = \int_B (z^2 + y^2) \, dm \\ J_{22} = \int_B (x^2 + z^2) \, dm \\ J_{33} = \int_B (x^2 + y^2) \, dm \end{cases}$$

$$\text{products of inertia} \begin{cases} J_{12} = J_{21} = -\int_B xy \, dm \\ J_{13} = J_{31} = -\int_B xz \, dm \\ J_{23} = J_{32} = -\int_B yx \, dm \end{cases}$$

The inertia matrix $\mathbf{J}_o$ has real entries and it is symmetric, for the spectral theorem, a **diagonal matrix J** such that

$$\mathbf{J} = \mathbf{T}^T \mathbf{J}_o \mathbf{T}$$

$J = \text{diag}(J_1, J_2, J_3)$ is a diagonal matrix whose entries are the eigenvalues of the non-symmetric matrix $\mathbf{J}_o$. The matrix $\mathbf{T} = [\mathbf{e}_1 \ \mathbf{e}_2 \ \mathbf{e}_3]$ is a rotation matrix and its columns are the eigenvector of $\mathbf{J}_o$. At this point we call:

- **Principal moments of inertia** the eigenvalues of $\mathbf{J}_o$
- **Principal axes of inertia** the eigenvectors $\mathbf{e}_i$ of $\mathbf{J}_o$

11.3.3 Euler momentum equations

Let $\mathbf{M} = M_1\mathbf{b}_1 + M_2\mathbf{b}_2 + M_3\mathbf{b}_3$ an **input** momentum acting on the system. From the second law of dynamics for rigid body we have that

$$\dot{\mathbf{H}} = \mathbf{M} \tag{11.19}$$

The derivative $\dot{\mathbf{H}}$ is equal to

$$\dot{\mathbf{H}} = \dot{\mathbf{H}}_B + \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} = \mathbf{J}\dot{\boldsymbol{\omega}} + \omega \times \mathbf{J}\boldsymbol{\omega} = \mathbf{M}$$

Putting all together we obtain the **Euler moment equations**:

$$\mathbf{J}\dot{\boldsymbol{\omega}} = \mathbf{M} - \boldsymbol{\omega} \times \boldsymbol{J\omega} \tag{11.20}$$

In the case that the inertia matrix **J** is diagonal, the equation (11.20) and performing the product, the equation becomes:

$$\begin{aligned} J_1\dot{\omega_1} + (J_3 - J_2)\omega_2\omega_3 &= M_1 \\ J_2\dot{\omega_2} + (J_1 - J_3)\omega_1\omega_3 &= M_2 \\ J_3\dot{\omega_3} + (J_2 - J_1)\omega_1\omega_2 &= M_3 \end{aligned}$$

In matrix form:

$$\begin{bmatrix} \dot{\omega_1} \\ \dot{\omega_2} \\ \dot{\omega_3} \end{bmatrix} = \begin{bmatrix} 0 & \sigma_1\omega_3 & 0 \\ \sigma_2\omega_3 & 0 & 0 \\ \sigma_3\omega_2 & 0 & 0 \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} + \begin{bmatrix} M_1/J_1 \\ M_2/J_2 \\ M_3/J_3 \end{bmatrix} \tag{11.21}$$

$$\sigma_1 = \frac{J_2 - J_3}{J_1},\ \sigma_2 = \frac{J_3 - J_1}{J_2},\ \sigma_3 = \frac{J_1 - J_2}{J_3}$$

11.4 Free rotational motion

When we want to control any dynamic system we have to analyse its features. For instance it is useful to study its behaviour when no input is applied → We want to analyse the free motion or free response of the system.

The aim here is to describe the free rotational motion of a generic spacecraft, starting from the simpler case of a **symmetric body**, going to the more general case where no symmetry appears.

$$\text{free motion} \iff \mathbf{M} = \begin{bmatrix} M_1 \\ M_2 \\ M_3 \end{bmatrix} = 0$$

11.4.1 Free motion of a symmetric body

In this case we will consider that $J_1 = J_2 \neq J_3$, in this case the Euler equations becomes:

$$\begin{aligned} J_1\dot{\omega_1} + (J_3 - J_2)\omega_2\omega_3 &= 0 \\ J_2\dot{\omega_2} + (J_1 - J_3)\omega_1\omega_3 &= 0 \\ J_3\dot{\omega_3} &= 0 \end{aligned}$$

From this we can observe a lot of things:

- $\dot{\omega_3} = 0 \iff \omega =$constant;
- The angular velocity according to the z axis is constant
- The equations are linear this allows a complete analysis using the well known theoretical results.

We can algebraically manipulate the equations in order to analyse the system as follows:

$$\dot{\omega_1} + \eta\omega_2 = 0 \tag{11.22}$$

$$\dot{\omega_2} - \eta\omega_1 = 0 \tag{11.23}$$

Multiplying by ω_1 the two equations we obtain

$$\omega_1\dot{\omega_1} + \omega_2\dot{\omega_2} = 0 \iff \omega_1\frac{d\omega_1}{dt} + \omega_2\frac{d\omega_2}{dt} = 0 \iff$$
$$\omega_1\ d\omega_1 + \omega_2\ d\omega_2 = 0 \overset{\int}{\iff} \int \omega_1\ d\omega + \int \omega_2\ d\omega_2 = const. \iff$$
$$\omega_1^2 + \omega_2^2 = const.$$

Since also ω_3 is constant, we can state that

$$|\boldsymbol{\omega}| = \sqrt{\omega_1^2 + \omega_2^2 + \omega_3^2} = \text{const.}$$

We know from this property that also the norm of the angular velocity is constant.
Computing the derivatives of (11.22) and using the (11.23), the equation of an *harmonic oscillator is obtained*:

$$\ddot{\omega_1} + \eta^2\omega_1 = 0 \tag{11.24}$$

It is well-known that this system is marginally stable system (the natural modes associated with two complex conjugate poles with null real part) $\rightarrow$ ω_1 remains bounded. An analogue procedure can be done for ω_2, differentiating the (11.23) and using the (11.22).

11.4.2 Free motion of an asymmetric body

Also here we assume that no torques are applied to the body, then $\mathbf{M} = 0$, but all the principal moments of inertia are different each other, then $J_1 \neq J_2 \neq J_3 \neq J_1$. We also suppose that ω_3 is a constant ω_0 to which is added a small *perturbation* ϵ, then $\omega_3 = \omega_0 + \epsilon$.
When $\epsilon \rightarrow 0$, the moment equations become:

$$J_1\dot{\omega_1} + (J_3 - J_2)\omega_2\omega_0 = 0$$
$$J_2\dot{\omega_2} + (J_1 - J_3)\omega_1\omega_0 = 0$$
$$J_3\dot{\epsilon} + (J_2 - J_1)\omega_1\omega_2 = 0$$

If we compute the derivative of the first and we use the second and defining a constant

$$\gamma \doteq \omega_0\sqrt{\left(1 - \frac{J_3}{J_1}\right)\left(1 - \frac{J_3}{J_2}\right)}$$

, we obtain a similar equation than the previous case of simmetry:

$$\ddot{\omega_1} + \gamma^2\omega_1 = 0$$

Two situations can be arised:

1. The constant γ is a real number, then we have either $J_3 > J_1, J_2$ (J_3 in this case is the **major axis** since has a moment of inertia which is greater than the others) or $J_3 < J_1, J_2$ (in the same way J_3 in this case is the **minor axis**); the square of any real number is a positive number $\rightarrow$ in this case we will obtain an armonic oscillator which is **marginally stable**;

2. The constant γ is an imaginary number, this can occur when either $J_1 > J_3 > J_2$ or $J_2 > J_3 > J_1$; this indicates that J_3 is the **intermediate axis**. The squared imaginary number produces a negative number implying that the system is **unstable**.

Note that... When we say either **stability** or **unstability**, in the first case we mean that if we start with ω_1 near an axis, we will remain always in the neighbourhood of such axis, in the second case even if one remains close the axis, the angular velocity will diverge. An analogue reasoning can be done for ω_2.

In the reality a body is never perfectly rigid due to, for example, *structural deflection*. The assumption that a body is **semirigid** is more realistic. One refers to a **semirigid body** when: (i) there are no moving parts, (ii) there is *energy dissipation*.
In this case the results about the stability change, and in particular:

1. The motion about the **major principal axis** is *asymptotically stable*;
2. The motion about the minor and intermediate axis is *unstable*.

11.5 Dynamic-Kinematic equations

The **kinematic** and **dynamic** equations can be put together in order to obtain the equations for the overall system composed by the cascade of the two analysed in the previous chapters.

In general we have that the **Dynamic-Kinematic state equation** is:

$$\dot{x} = \mathbf{A}(\mathbf{x})\mathbf{x} + \mathbf{B}\mathbf{u} \tag{11.25}$$

In the next paragraphs the matrices for the equations of the most common rotation are presented:

11.5.1 Dynamic-Kinematic equation for Tait-Bryan 321

$$\begin{aligned}
\mathbf{x} &= \begin{bmatrix} \phi \\ \theta \\ \psi \\ \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}, \quad \mathbf{A}(\mathbf{x}) = \left[\begin{array}{ccc|ccc} 0 & 0 & 0 & 1 & \sin\theta\tan\theta & \cos\phi\tan\theta \\ 0 & 0 & 0 & 0 & \cos\theta & -\sin\theta \\ 0 & 0 & 0 & 0 & \sin\phi/\cos\theta & \cos\phi/\cos\theta \\ \hline 0 & 0 & 0 & 0 & \sigma_1\omega_3 & 0 \\ 0 & 0 & 0 & \sigma_2\omega_3 & 0 & 0 \\ 0 & 0 & 0 & \sigma_3\omega_2 & 0 & 0 \end{array}\right] \\
\mathbf{u} &= \begin{bmatrix} M_1 \\ M_2 \\ M_3 \end{bmatrix}, \quad \mathbf{B} = \left[\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \hline 1/J_1 & 0 & 0 \\ 0 & 1/J_2 & 0 \\ 0 & 0 & 1/J_3 \end{array}\right]
\end{aligned} \tag{11.26}$$

11.5.2 Dynamic-Kinematic equations for Tait-Bryan 123

$$\mathbf{x} = \begin{bmatrix} \phi \\ \theta \\ \psi \\ \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}, \quad \mathbf{A}(\mathbf{x}) = \left[\begin{array}{ccc|cc} 0 & 0 & 0 & \cos\psi/\cos\theta & -\sin\psi\cos\theta & 0 \\ 0 & 0 & 0 & \sin\psi & \cos\psi & 0 \\ 0 & 0 & 0 & -\tan\theta\cos\psi & \tan\theta\sin\psi & 1 \\ \hline 0 & 0 & 0 & 0 & \sigma_1\omega_3 & 0 \\ 0 & 0 & 0 & \sigma_2\omega_3 & 0 & 0 \\ 0 & 0 & 0 & \sigma_3\omega_2 & 0 & 0 \end{array}\right]$$

$$\mathbf{u} = \begin{bmatrix} M_1 \\ M_2 \\ M_3 \end{bmatrix}, \quad \mathbf{B} = \left[\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \hline 1/J_1 & 0 & 0 \\ 0 & 1/J_2 & 0 \\ 0 & 0 & 1/J_3 \end{array}\right] \tag{11.27}$$

11.5.3 Dynamic-Kinematic equations for Proper-Euler 313

$$\mathbf{x} = \begin{bmatrix} \phi \\ \theta \\ \psi \\ \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}, \quad \mathbf{A}(\mathbf{x}) = \left[\begin{array}{ccc|cc} 0 & 0 & 0 & \sin\psi/\sin\theta & \cos\psi/\sin\theta & 0 \\ 0 & 0 & 0 & \cos\psi & -\sin\psi & 0 \\ 0 & 0 & 0 & -\cot\theta\sin\psi & -\cot\theta\cos\psi & 1 \\ \hline 0 & 0 & 0 & 0 & \sigma_1\omega_3 & 0 \\ 0 & 0 & 0 & \sigma_2\omega_3 & 0 & 0 \\ 0 & 0 & 0 & \sigma_3\omega_2 & 0 & 0 \end{array}\right]$$

$$\mathbf{u} = \begin{bmatrix} M_1 \\ M_2 \\ M_3 \end{bmatrix}, \quad \mathbf{B} = \left[\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \hline 1/J_1 & 0 & 0 \\ 0 & 1/J_2 & 0 \\ 0 & 0 & 1/J_3 \end{array}\right] \tag{11.28}$$

11.5.4 Dynamic-Kinematic equations for Quaternions

$$\mathbf{x} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \\ \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}, \quad \mathbf{A}(\mathbf{x}) = \frac{1}{2}\left[\begin{array}{cccc|ccc} 0 & 0 & 0 & 0 & -q_1 & -q_2 & -q3 \\ 0 & 0 & 0 & 0 & q_0 & -q_3 & q_2 \\ 0 & 0 & 0 & 0 & q_3 & q_0 & -q_1 \\ 0 & 0 & 0 & 0 & -q_2 & q_1 & q_0 \\ \hline 0 & 0 & 0 & 0 & 0 & 2\sigma_1\omega_3 & 0 \\ 0 & 0 & 0 & 0 & 2\sigma_2\omega_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2\sigma_3\omega_2 & 0 & 0 \end{array}\right]$$

$$\mathbf{u} = \begin{bmatrix} M_1 \\ M_2 \\ M_3 \end{bmatrix}, \quad \mathbf{B} = \left[\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \hline 1/J_1 & 0 & 0 \\ 0 & 1/J_2 & 0 \\ 0 & 0 & 1/J_3 \end{array}\right] \tag{11.29}$$

Chapter 12

Attitude control

The aim of this chapter is to introduce *control methods* for **spacecraft**, then we will include the spacecraft (described by Dynamic-Kinematic equations) in a **feedback loop**. The *Attitude Control System* is in charge to:

1. Give a proper orientation to the body
2. Stabilize the spacecraft about a reference in the presence of disturbance torques.

Important: The Attitude orientation is fundamental in any space mission.

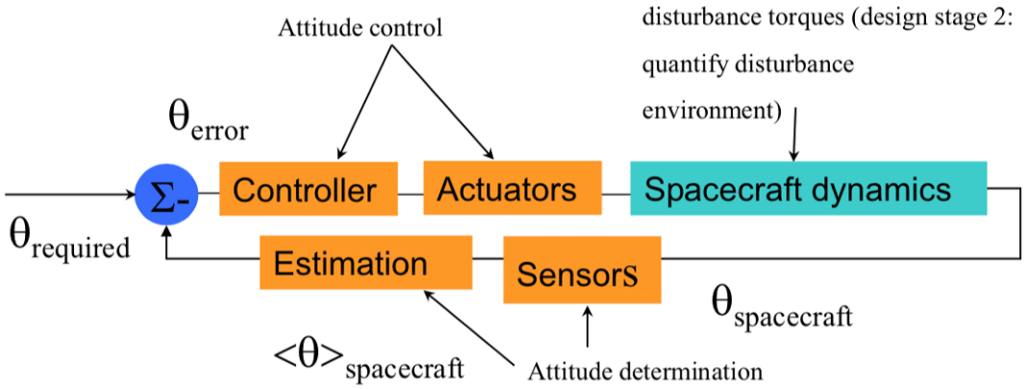


Figure 12.1: Attitude Control System

12.1 Sensors

A **sensor** employed in this field must be able to provide **in real time** the measurement of the attitude. This is fundamental to build any effective control system. The available sensors can be broken into two families:

- **Absolute attitude sensors** which determine the attitude with respect to some celestial body like the Sun or other stars (an example can be a **star tracker**).
- **Relative attitude sensors** they are capable to measure the *angular rate*, then the attitude is obtained through integration (they are usually **gyroscopes**).

12.2 Actuators

An **actuator** is fundamental in order to give to the system to control (the spacecraft) the **control input**. Even here three groups of actuators can be distinguished:

1. **Dampers** which are used to reduce the effect of nutation and external disturbances torques;
2. **Actuators which provides external torques**, they include *thrusters, reaction wheels, particular types of Gyros*;
3. **Actuators which use external forces** like magnetic fields, gravity gradient and so on.

The control laws we will present can be applied by using thrusters, reaction wheels or combination of them.

12.3 Control system approaches

How it has been mentioned in the introduction, the main objective of an attitude control system can be the **stabilization** about a reference attitude, or the tracking in the case that the orientation changes in time. We can split the *attitude control systems* in three categories:

1. PASSIVE: they are based on enviromental forces and on the properties of the body. They exist because environment perturbations can lead to stable behaviours. Passive control systems are seldom used.
2. SEMI-ACTIVE, in this case some *reaction wheels* are used that exploit the angular momentum consevation. Alternatively can be used *magnetic systems* which interacting with the magnetic field tend to produce useful torques.
3. ACTIVE: is the most used in the field of ACS (Attitude Control Systems), the technique is based on the use of *thrusters*.

A lot of classifications are available, another possible classification is that beetween: *Spin stabilization* and *3-axis stabilization*. The presented approach is based on **Three-axis control**.

12.4 Attitude control: a general setting

In the previous chapter we have seen the equations for dynamics and kinematics of a spacecraft, moreover we have understood that a useful and effective description of the orientation can be provided by *quaternions*, as they are powerful and *gimbal lock free*.
The **general goal** of *control* is to make the state vector $(\mathfrak{q}, \boldsymbol{\omega})$ track a **reference vector** $(\mathfrak{q}_r, \boldsymbol{\omega_r})$ which can be eventually time-varying. In order to determine what is the 'distance' from the desired behaviour we have to compute the tracking error:

- The **angular velocity tracking error** is defined simply as

$$\tilde{\boldsymbol{\omega}} \doteq \boldsymbol{\omega}_r - \boldsymbol{\omega}$$

- The **quaternion tracking error** is defined as a product beetween quaternions

$$\tilde{\mathfrak{q}} \equiv \begin{bmatrix} \tilde{q}_0 \\ \tilde{\mathbf{q}} \end{bmatrix} \doteq \mathfrak{q}^{-1} \otimes \mathfrak{q}_r = \mathfrak{q}^* \otimes \mathfrak{q}_r$$

  The quaternion $\tilde{\mathfrak{q}}$ is nothing but the quaternion which starting from $\mathfrak{q}$ goes to $\mathfrak{q}_r$.

12.5 First problem: Regulation (PD control law)

The *goal of regulation* is obtaining

$$\mathfrak{q} \to \mathfrak{q}_r = const \quad \text{and} \quad \boldsymbol{\omega} \to 0$$
$$\tilde{\mathfrak{q}} \to \mathfrak{I} \doteq (1, \mathbf{0}) \quad \text{and} \quad \tilde{\boldsymbol{\omega}} \to 0$$

This is the simplest problem of control with respect to the other, because the quaternions to reach is constant and then does not change in time.

A **simple linear control law** is

$$\mathbf{u} = k_p \tilde{\mathbf{q}} - k_d \boldsymbol{\omega} \tag{12.1}$$

where $k_p > 0$ and $k_d > 0$ are parameters to tune. Note that in this law enters only the vector part of the *quaternion tracking error*, and the law it is substantially a Proporzional-Derivative (PD) one, where you have a term **proportional** to $\tilde{\mathbf{q}}$ and a term which depends by a constant k_d by the derivative.

In general the Kinematic-Dynamic equations are:

$$\dot{\mathfrak{q}} = \frac{1}{2}\mathbf{Q}\boldsymbol{\omega}$$
$$\dot{\boldsymbol{\omega}} = -\mathbf{J}^{-1}\boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} + \mathbf{J}^{-1}\mathbf{u}$$

By applying the control law we have just introduced, we will obtain:

$$\dot{\tilde{\mathfrak{q}}} = -\frac{1}{2}\boldsymbol{\omega}^q \otimes \tilde{\mathfrak{q}}$$
$$\dot{\boldsymbol{\omega}} = -\mathbf{J}^{-1}(-\boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} + k_p\tilde{\mathbf{q}} - k_d\boldsymbol{\omega})$$

The defined system has **two equilibria** associated with the same attitude, specifically they are $(\tilde{q_0}, \tilde{\mathbf{q}}, \boldsymbol{\omega}) = (\pm 1, \mathbf{0}, \mathbf{0})$, this is why it is the same cosine function whose arguments are different each for an angle equal to π, by using the trigonometry it is easy to show it.
Reguarding such equilibria, an important theorem holds:

Theorem
The equilibrium points $(\pm 1, \mathbf{0}, \mathbf{0})$ of the *closed-loop system* are **locally asympotically stable**, moreover for any initial condition $(q_0(0), \tilde{\mathbf{q}}(0), \boldsymbol{\omega}(0))$,

$$\lim_{t\to\infty}(q_0(t), \tilde{\mathbf{q}}(t), \boldsymbol{\omega}(t)) = (\pm 1, \mathbf{0}, \mathbf{0})$$

The proof is based on the *Lyapunov function*

$$V = \frac{1}{4}\boldsymbol{\omega}^T J\boldsymbol{\omega} + \frac{1}{2}k_p\tilde{\mathbf{q}}^T\tilde{\mathbf{q}} + \frac{1}{2}k_p(1 \mp \tilde{q_0})^2$$

For the problem of regulation, there are other effective control laws which provides a **shortest path to tracking** (when $\text{sign}(\tilde{q_0})$ appears), or **better performances** in terms of command activity and time response (when the non linear term $(1 \pm \tilde{\mathbf{q}}^T\tilde{\mathbf{q}})$ appears). They are:

$$\mathbf{u} = k_p \text{sign}(\tilde{q_0})\tilde{\mathbf{q}} - k_d\boldsymbol{\omega} \tag{12.2}$$
$$\mathbf{u} = k_p\tilde{\mathbf{q}} - k_d(1 \pm \tilde{\mathbf{q}}^T\tilde{\mathbf{q}})\boldsymbol{\omega} \tag{12.3}$$
$$\mathbf{u} = k_p \text{sign}(\tilde{q_0})\tilde{\mathbf{q}} - k_d(1 \pm \tilde{\mathbf{q}}^T\tilde{\mathbf{q}})\boldsymbol{\omega} \tag{12.4}$$

12.6 Second problem: Tracking (Sliding mode control)

The *goal of tracking* problems is:

$$\mathfrak{q}(t) \to \mathfrak{q}_r(t) \quad \text{and} \quad \boldsymbol{\omega} \to \boldsymbol{\omega}_r(t)$$
$$\tilde{\mathfrak{q}} \to \mathfrak{I} \doteq (1, \mathbf{0}) \quad \text{and} \quad \tilde{\boldsymbol{\omega}}(t) \to 0$$

The laws which has been just presented can be used, but you will have poor performances and robustness. On the other hand, an **effective way** to face the tracking problem is using the **sliding mode control** that provides, differently than the simpler linear control law, **high performances** and **robustness**. The system to control is:

$$\dot{\mathfrak{q}} = \frac{1}{2}\mathbf{Q}\boldsymbol{\omega}$$
$$\dot{\boldsymbol{\omega}} = -\mathbf{J}^{-1}\boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} + \mathbf{J}^{-1}\mathbf{u}$$

which is in a sort of *generalized normal form*, and how happens in the great majority of mechanical systems the **relative degree** is $\gamma = 2$. Remind that when you want to exploit the **sliding mode control** the steps to follow are:

1. The definition of a *sliding surface*;
2. The definition of a *feedback law.*

12.6.1 Sliding mode control for attitude tracking

In the expression of the sliding surface, usually appears the tracking error and its derivative in a way that is consistent with the *relative degree* of the system. In our case we have to use the **(quaternion) tracking error** (more specifically for the purposes we use its vector part), and its derivative that is the **angular velocity**. The resulting sliding surface is

$$\mathbf{s}(\mathbf{q}, \boldsymbol{\omega}, t) = k_2\tilde{\mathbf{q}} + \tilde{\boldsymbol{\omega}} \tag{12.5}$$

It is known that on such sliding surface the tracking error is null. At this point, having defined the first ingredient for SM control, we have to define a **control law** $\mathbf{u}_s$ such that the surface is:

- Invariant, if the state is on the surface, it must remain on it;
- Attractive, if the stato is **not** on the surface it will be led on it, in finite time.

Invariance of the surface

Practically speaking, we have to impose that the derivative of the expression defining the sliding surface is 0, that is $\dot{\mathbf{s}} = 0$. The derivative is:

$$\dot{\mathbf{s}} = \dot{\boldsymbol{\omega}}_r - \dot{\boldsymbol{\omega}} + k_2\dot{\tilde{\mathbf{q}}} =$$
$$\dot{\boldsymbol{\omega}}_r + \mathbf{J}^{-1}\boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} - \boldsymbol{J}^{-1}\mathbf{u} + \frac{k_2}{2}(\tilde{q}_0\tilde{\boldsymbol{\omega}} + \tilde{\mathbf{q}} \times (\boldsymbol{\omega}_r + \boldsymbol{\omega}))$$

If we invert this expression with respect to $\mathbf{u}$ we can find (a part of) the control law $\mathbf{u}_s$ that is:

$$\mathbf{u}_s = \mathbf{J}\left(\dot{\boldsymbol{\omega}}_r + \frac{k_2}{2}(\tilde{q}_0\tilde{\boldsymbol{\omega}} + \tilde{\mathbf{q}} \times (\boldsymbol{\omega}_r + \boldsymbol{\omega}))\right) + \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} \tag{12.6}$$

Attractiveness of the surface

We have to make the surface *attractive* an additional term, which has a sigmoid shape, has to be used. In this way the final **feedback control law** is obtained:

$$\mathbf{u} = \mathbf{u}_s + k_1 \mathbf{J} \tanh(\eta \mathbf{s}) \tag{12.7}$$

12.6.2 Another sliding mode control law

In an analogue way than the regulation case, also here an alternative control law can be defined, guaranteeing the **shortest reorientation**. Also here a sign function is used in the *sliding function*.

$$\begin{aligned}
&\mathbf{s}(\mathbf{q}, \boldsymbol{w}) = \tilde{\boldsymbol{\omega}} + k_2 \text{sign}(q_0)\tilde{\mathbf{q}} \\
&\mathbf{u}_s = \mathbf{J}\left(\dot{\boldsymbol{\omega}}_r + \frac{k_2}{2}(|\tilde{q_0}|\tilde{\boldsymbol{\omega}} + \text{sign}(q_0)\tilde{\mathbf{q}} \times (\boldsymbol{\omega}_r + \boldsymbol{\omega}))\right) + \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} \\
&\mathbf{u} = \mathbf{u}_s + k_1 \mathbf{J} \tanh(\eta \mathbf{s})
\end{aligned}$$

In (12.7) appear some additional terms, in particular $\boldsymbol{\omega}_r$ e $\dot{\boldsymbol{\omega}}$, for this reason a **reference generator** (like in SMC) has to be employed. An example may be the following:

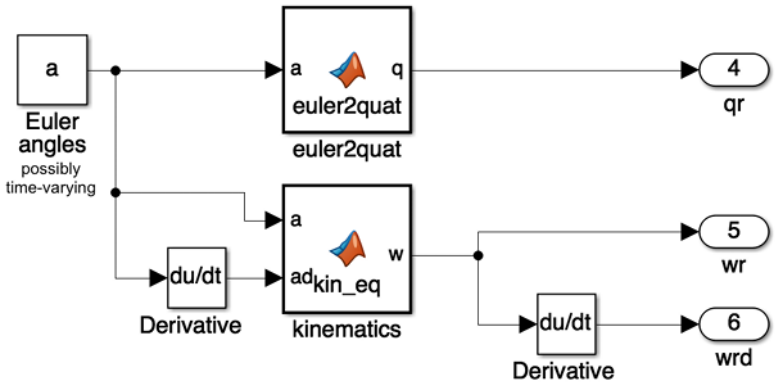


Figure 12.2: Simulink scheme for a reference generator

Chapter 13

Orbital dynamics

Chapter 14

Orbital control