# Mobile robotics

Luc Jaulin

July 26, 2017

# Contents

# Introduction

A *mobile robot* can be defined as a mechanical system capable of moving in its environment in an autonomous manner. For that purpose, it must be equipped with:

- *sensors* that will help it gain knowledge of its surroundings (which it is more or less aware of) and determine its location ;
- *actuators* which will allow it to move ;
- an *intelligence* (or algorithm, regulator), which will allow it to compute, based on the data gathered by the sensors, the commands to send to the actuators in order to perform a given task.

Finally, to this we must add the *surroundings* of the robot which correspond to the world in which it evolves and its *mission* which is the task it has to accomplish. Mobile robots are constantly evolving, mainly from the beginning of the 2000s, in military domains (airborne drones [BEA 12], underwater robots [CRE 14], etc.), and even in medical and agricultural fields. They are in particularly high demand for performing tasks considered to be painful or dangerous to humans. This is the case for instance in mine-clearing operations, the search for black boxes of damaged aircraft on the ocean bed and planetary exploration. Artificial satellites, launchers (such as Ariane V), driverless subways and elevators are examples of mobile robots. Airliners, trains and cars evolve in a continuous fashion towards more and more autonomous systems and will very probably become mobile robots in the following decades.

*Mobile robotics* is the discipline which looks at the design of mobile robots [LAU 01]. It is based on other disciplines such as automatic control, signal processing, mechanics, computing and electronics. The aim of this book is to give an overview of the tools and methods of robotics which will aid in the design of mobile robots. The robots will be modeled by *state equations*, in other words first order (mostly non-linear) differential equations. These state equations can be obtained by using the laws of mechanics. It is not in our objectives to teach, in detail, the methods of robot modeling (refer to [JAU 05] and [JAU 13] for more information on the subject), merely to recall its principles. By *modeling*, we mean obtaining the state equations. This step is essential for simulating robots as well as designing controllers. We will however illustrate the principle of modeling in Chapter 1 on deliberately three-dimensional examples. This choice was made in order to introduce important concepts in robotics such as Euler angles and rotation matrices. For instance, we will be looking at the dynamics of a wheel and the kinematics of an underwater robot. Mobile robots are strongly non-linear systems and only a non-linear approach allows the construction of efficient controllers. This construction is the subject of Chapters 2 and 3. Chapter 2 is mainly based on control methods

that rely on the utilization of the robot model. This approach will make use of the concept of *feedback linearization* which will be introduced and illustrated through numerous examples. Chapter 3 presents more pragmatic methods which do not use the state model of the robot and which will be referred to as *without model* or *mimetic*. The approach uses a more intuitive representation of the robot and is adapted to situations in which the robots are relatively simple to remotely control, such as in the case of cars, sailing boats or airplanes. Chapter 4 looks at *guidance*, which is placed at a higher level than control. In other words, it focuses on guiding and supervising the system which is already under control by the tools presented in Chapters 2 and 3. There will therefore be an emphasis on finding the instruction to give to the controller in order for the robot to accomplish its given task. The guidance will then have to take into account the knowledge of the surroundings, the presence of obstacles and the roundness of the Earth. The non-linear control and guidance methods require good knowledge of the state variables of the system, such as those which define the position of the robot. These position variables are the most difficult to find and Chapter 5 focuses on the problem of *positioning*. It introduces the classical non-linear approaches that have been used for a very long time by humans for positioning, such as observing beacons, stars, using the compass or counting steps. Although positing can be viewed as a particular case of state observation, the specific methods derived from it warrant a separate chapter. Chapter 6 on *identification* focuses on finding, with a certain precision, non-measured quantities (parameters, position) from other, measured ones. In order to perform this identification, we will mainly be looking at the so-called *least squares* approach which consists of finding the vector of variables that minimizes the sum of the squares of the errors. Chapter 7 presents the *Kalman filter*. This filter can be seen as a state observer for dynamic linear systems with coefficients that vary in time.

The MATLAB code related to the exercises of this book together with explanatory videos can be found at the following address:

`www.ensta-bretagne.fr/jaulin/isterob.html`

# Chapter 1

# Three-dimensional modeling

This chapter looks at the three-dimensional modeling of a solid (non-articulated) robot. Such modeling is used to represent an airplane, a quadcopter, a submarine, and so forth. Through this modeling we will introduce a number of fundamental concepts in robotics such as state representation, rotation matrices and Euler angles. The robots, whether mobile, manipulator or articulated, can generally be put into a state representation form:

$$\begin{cases} \dot{\mathbf{x}}(t) &= \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \\ \mathbf{y}(t) &= \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t)) \end{cases}$$

where $\mathbf{x}$ is the state vector, $\mathbf{u}$ the input vector and $\mathbf{y}$ the vector of measurements [JAU 05]. We will call *modeling* the step which consists of finding a more or less accurate state representation of the robot in question. In general, constant parameters may appear in the state equations (such as the mass and the moment of inertia of a body, viscosity, etc.). In such cases, an identification step might prove to be necessary. We will assume that all of the parameters are known. Of course, there is no systematic methodology that can be applied for modeling a mobile robot. The aim of this chapter is to present the tools which allow to reach a state representation of three-dimensional solid robots in order for the reader to acquire a certain experience which will be helpful when modeling his/her own robots. This modeling will also allow us to recall a number of important concepts in Euclidean geometry, which are fundamental in mobile robotics. This chapter begins by recalling a number of important concepts in kinematics which will be useful for the modeling.

## 1.1 Rotation matrices

For three-dimensional modeling, it is essential to have a good understanding of the concepts related to rotation matrices, which are recalled in this section. It is by using this tool that we will perform our coordinate system transformations and position our objects in space.

### 1.1.1 Definition

Let us recall that the $j^{\text{th}}$ column of the matrix of a linear application of $\mathbb{R}^n \to \mathbb{R}^n$ represents the image of the $j^{\text{th}}$ vector $\mathbf{e}_j$ of the standard basis (see Figure 1.1). Thus, the expression of a rotation

matrix of angle $\theta$ in the plane $\mathbb{R}^2$ is given by:

$$\mathbf{R} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}.$$

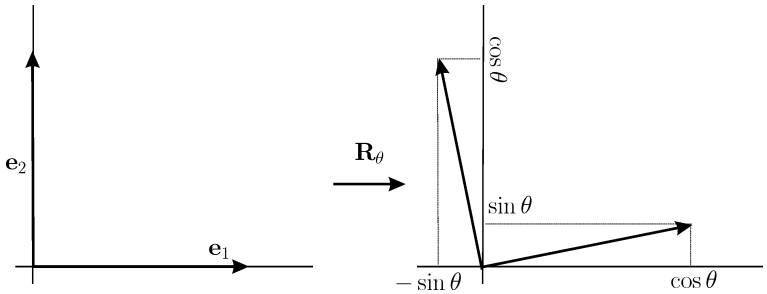


Figure 1.1: Rotation of angle $\theta$ in a plane

Concerning rotations in the space $\mathbb{R}^3$ (see Figure 1.2), it is important to specify the axis of rotation. We distinguish three main rotations: the rotation around the $Ox$ axis, the one around the $Oy$ axis and the one around the $Oz$ axis.

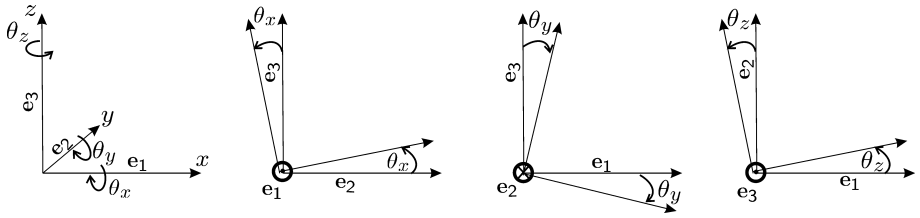


Figure 1.2: Rotations in $\mathbb{R}^3$ following various viewing angles

The associated matrices are respectively given by:

$$\mathbf{R}_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{pmatrix}, \ \mathbf{R}_y = \begin{pmatrix} \cos\theta_y & 0 & \sin\theta_y \\ 0 & 1 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y \end{pmatrix}, \ \mathbf{R}_z = \begin{pmatrix} \cos\theta_z & -\sin\theta_z & 0 \\ \sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Let us recall the formal definition of a rotation. A rotation is a linear application which is an isometry (in other words it preserves the scalar product) which is direct (it does not change the orientation in space).

THEOREM 1.1.– *A matrix* $\mathbf{R}$ *is a rotation matrix if and only if :*

$$\mathbf{R}^T \cdot \mathbf{R} = \mathbf{I} \text{ and } \det \mathbf{R} = 1.$$

PROOF.– The scalar product is preserved by $\mathbf{R}$ if, for any $\mathbf{u}$ and $\mathbf{v}$ in $\mathbb{R}^n$, we have :

$$(\mathbf{Ru})^{\mathrm{T}} \cdot (\mathbf{Rv}) = \mathbf{u}^{\mathrm{T}}\mathbf{R}^{\mathrm{T}}\mathbf{Rv} = \mathbf{u}^{\mathrm{T}}\mathbf{v}.$$

Therefore $\mathbf{R}^{\mathrm{T}}\mathbf{R} = \mathbf{I}$. The symmetries relative to a plane, as well as all the other improper isometries

(isometries that change the orientation of space, such as a mirror), also verify the property $\mathbf{R}^{\mathrm{T}}\cdot\mathbf{R} = \mathbf{I}$. The condition $\det \mathbf{R} = 1$ allows us to be limited to the isometries which are direct. The set of rotation matrices of $\mathbb{R}^n$ forms a group referred to as a *special orthogonal group* (special because $\det \mathbf{R} = 1$, orthogonal because $\mathbf{R}^{\mathrm{T}}\cdot\mathbf{R} = \mathbf{I}$).

### 1.1.2 Rotation vector

If $\mathbf{R}$ is a rotation matrix depending on time $t$, by differentiating the relation $\mathbf{R}\mathbf{R}^{\mathrm{T}} = \mathbf{I}$, one obtains :

$$\dot{\mathbf{R}} \cdot \mathbf{R}^{\mathrm{T}} + \mathbf{R} \cdot \dot{\mathbf{R}}^{\mathrm{T}} = \mathbf{0}.$$

Thus, the matrix $\dot{\mathbf{R}} \cdot \mathbf{R}^{\mathrm{T}}$ is a skew-symmetric matrix (*i.e.*, it satisfies $\mathbf{A}^{\mathrm{T}} = -\mathbf{A}$ and therefore its diagonal contains only zeroes, and for each element of $\mathbf{A}$, we have $a_{ij} = -a_{ji}$). We may therefore write, in the case where $\mathbf{R}$ is of dimension $3 \times 3$:

$$\dot{\mathbf{R}} \cdot \mathbf{R}^{\mathrm{T}} = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix}. \tag{1.1}$$

The vector $\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z)$ is called the *rotation vector* associated with the pair $\left(\mathbf{R}, \dot{\mathbf{R}}\right)$. It must be noted that $\dot{\mathbf{R}}$ is not a matrix with good properties (such as for instance the fact of being skew-symmetric). On the other hand, the matrix $\dot{\mathbf{R}} \cdot \mathbf{R}^{\mathrm{T}}$ has the [1.1] structure since it allows to be positioned within the coordinate system in which the rotation is performed and this, due to the change of basis performed by $\mathbf{R}^{\mathrm{T}}$. We will define the *vector product* between two vectors $\boldsymbol{\omega}$ and $\mathbf{x} \in \mathbb{R}^3$ as follows:

$$\boldsymbol{\omega} \wedge \mathbf{x} = \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} \wedge \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} x_3\omega_y - x_2\omega_z \\ x_1\omega_z - x_3\omega_x \\ x_2\omega_x - x_1\omega_y \end{pmatrix} = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}.$$

### 1.1.3 Adjoint

For each vector $\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z)$, we may adjoin the skew-symmetric matrix:

$$\mathbf{Ad}\left(\boldsymbol{\omega}\right) = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix}$$

which can be interpreted as the matrix associated with a vector product by the vector $\boldsymbol{\omega}$.

PROPOSITION 1.1.– *If $\mathbf{R}(t)$ is a rotation matrix that depends on time, its rotation vector is given by:*

$$\boldsymbol{\omega} = \mathbf{Ad}^{-1}\left(\dot{\mathbf{R}} \cdot \mathbf{R}^{T}\right). \tag{1.2}$$

PROOF.– This relation is a direct consequence of Equation (1.1). ■

PROPOSITION 1.2.– *If* $\mathbf{R}$ *is a rotation matrix in* $\mathbb{R}^3$ *and if* $\mathbf{a}$ *is a vector of* $\mathbb{R}^3$*, we have:*

$$\mathbf{Ad}\left(\mathbf{R}\cdot\mathbf{a}\right)=\mathbf{R}\cdot\mathbf{Ad}\left(\mathbf{a}\right)\cdot\mathbf{R}^{T} \tag{1.3}$$

*which can also be written as:*

$$\left(\mathbf{R}\cdot\mathbf{a}\right)\wedge=\mathbf{R}\cdot\left(\mathbf{a}\wedge\right)\cdot\mathbf{R}^{T}.$$

PROOF.– Let $\mathbf{x}$ be a vector of $\mathbb{R}^3$. We have:

$$\begin{aligned}\mathbf{Ad}\left(\mathbf{R}\cdot\mathbf{a}\right)\cdot\mathbf{x}&=\left(\mathbf{R}\cdot\mathbf{a}\right)\wedge\mathbf{x}=\left(\mathbf{R}\cdot\mathbf{a}\right)\wedge\left(\mathbf{R}\cdot\mathbf{R}^{\mathrm{T}}\mathbf{x}\right)\\&=\mathbf{R}\cdot\left(\mathbf{a}\wedge\mathbf{R}^{\mathrm{T}}\cdot\mathbf{x}\right)=\mathbf{R}\cdot\mathbf{Ad}\left(\mathbf{a}\right)\cdot\mathbf{R}^{\mathrm{T}}\cdot\mathbf{x}.\ \blacksquare\end{aligned}$$

PROPOSITION 1.3.– *(duality). We have:*

$$\mathbf{R}^{T}\dot{\mathbf{R}}=\mathbf{Ad}\left(\mathbf{R}^{T}\boldsymbol{\omega}\right). \tag{1.4}$$

*This relation expresses the fact that the matrix* $\mathbf{R}^{T}\dot{\mathbf{R}}$ *is associated with the rotation vector* $\boldsymbol{\omega}$*, associated with* $\mathbf{R}\left(t\right)$ *but expressed in the coordinate system associated with* $\mathbf{R}$ *whereas* $\dot{\mathbf{R}}\cdot\mathbf{R}^{T}$ *is associated to the same vector, but this time expressed in the coordinate system of the standard basis.*

PROOF.– We have:

$$\mathbf{R}^{\mathrm{T}}\dot{\mathbf{R}}\ =\mathbf{R}^{\mathrm{T}}\left(\dot{\mathbf{R}}\cdot\mathbf{R}^{\mathrm{T}}\right)\mathbf{R}\ \overset{[1.2]}{=}\mathbf{R}^{\mathrm{T}}\cdot\mathbf{Ad}\left(\boldsymbol{\omega}\right)\cdot\mathbf{R}\ \overset{[1.3]}{=}\mathbf{Ad}\left(\mathbf{R}^{\mathrm{T}}\boldsymbol{\omega}\right).\ \blacksquare$$

### 1.1.4 Coordinate system change

Let $\mathcal{R}_0:(\mathbf{o}_0,\mathbf{i}_0,\mathbf{j}_0,\mathbf{k}_0)$ and $\mathcal{R}_1:(\mathbf{o}_1,\mathbf{i}_1,\mathbf{j}_1,\mathbf{k}_1)$ be two coordinate systems and let the point $\mathbf{u}$ be a vector of $\mathbb{R}^3$ (refer to Figure 1.3). We have the following relation:

$$\begin{aligned}\mathbf{u}&=x_0\mathbf{i}_0+y_0\mathbf{j}_0+z_0\mathbf{k}_0\\&=x_1\mathbf{i}_1+y_1\mathbf{j}_1+z_1\mathbf{k}_1\end{aligned}$$

where $(x_0,y_0,z_0)$ and $(x_1,y_1,z_1)$ are the coordinates of $\mathbf{u}$ in $\mathcal{R}_0$ and $\mathcal{R}_1$, respectively.

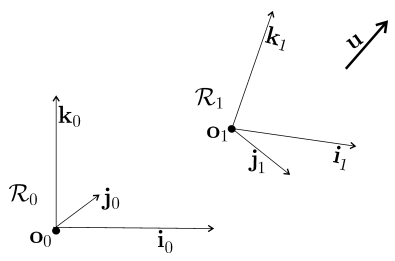


Figure 1.3: Changing the coordinate system $\mathcal{R}_0$ to the system $\mathcal{R}_1$

Thus, for any vector $\mathbf{v}$ we have:

$$\langle x_0\mathbf{i}_0 + y_0\mathbf{j}_0 + z_0\mathbf{k}_0, \mathbf{v}\rangle = \langle x_1\mathbf{i}_1 + y_1\mathbf{j}_1 + z_1\mathbf{k}_1, \mathbf{v}\rangle .$$

By taking respectively $\mathbf{v} = \mathbf{i}_0, \mathbf{j}_0, \mathbf{k}_0$, we obtain the following three relations:

$$\left\{\begin{array}{lcl} \langle x_0\mathbf{i}_0 + y_0\mathbf{j}_0 + z_0\mathbf{k}_0, \mathbf{i}_0\rangle & = & \langle x_1\mathbf{i}_1 + y_1\mathbf{j}_1 + z_1\mathbf{k}_1, \mathbf{i}_0\rangle \\ \langle x_0\mathbf{i}_0 + y_0\mathbf{j}_0 + z_0\mathbf{k}_0, \mathbf{j}_0\rangle & = & \langle x_1\mathbf{i}_1 + y_1\mathbf{j}_1 + z_1\mathbf{k}_1, \mathbf{j}_0\rangle \\ \langle x_0\mathbf{i}_0 + y_0\mathbf{j}_0 + z_0\mathbf{k}_0, \mathbf{k}_0\rangle & = & \langle x_1\mathbf{i}_1 + y_1\mathbf{j}_1 + z_1\mathbf{k}_1, \mathbf{k}_0\rangle . \end{array}\right.$$

However, since the basis $(\mathbf{i}_0, \mathbf{j}_0, \mathbf{k}_0)$ of $\mathcal{R}_0$ is orthonormal, $\langle \mathbf{i}_0, \mathbf{i}_0\rangle = \langle \mathbf{j}_0, \mathbf{j}_0\rangle = \langle \mathbf{k}_0, \mathbf{k}_0\rangle = 1$ and $\langle \mathbf{i}_0, \mathbf{j}_0\rangle = \langle \mathbf{j}_0, \mathbf{k}_0\rangle = \langle \mathbf{i}_0, \mathbf{k}_0\rangle = 0$. Thus, these three relations become:

$$\left\{\begin{array}{lcl} x_0 & = & x_1 \langle \mathbf{i}_1, \mathbf{i}_0\rangle + y_1 \langle \mathbf{j}_1, \mathbf{i}_0\rangle + z_1 \langle \mathbf{k}_1, \mathbf{i}_0\rangle \\ y_0 & = & x_1 \langle \mathbf{i}_1, \mathbf{j}_0\rangle + y_1 \langle \mathbf{j}_1, \mathbf{j}_0\rangle + z_1 \langle \mathbf{k}_1, \mathbf{j}_0\rangle \\ z_0 & = & x_1 \langle \mathbf{i}_1, \mathbf{k}_0\rangle + y_1 \langle \mathbf{j}_1, \mathbf{k}_0\rangle + z_1 \langle \mathbf{k}_1, \mathbf{k}_0\rangle \end{array}\right.$$

Or in matrix form:

$$\underbrace{\begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix}}_{=\mathbf{u}|_{\mathcal{R}_0}} = \underbrace{\begin{pmatrix} \langle \mathbf{i}_1, \mathbf{i}_0\rangle & \langle \mathbf{j}_1, \mathbf{i}_0\rangle & \langle \mathbf{k}_1, \mathbf{i}_0\rangle \\ \langle \mathbf{i}_1, \mathbf{j}_0\rangle & \langle \mathbf{j}_1, \mathbf{j}_0\rangle & \langle \mathbf{k}_1, \mathbf{j}_0\rangle \\ \langle \mathbf{i}_1, \mathbf{k}_0\rangle & \langle \mathbf{j}_1, \mathbf{k}_0\rangle & \langle \mathbf{k}_1, \mathbf{k}_0\rangle \end{pmatrix}}_{=\mathbf{R}_{\mathcal{R}_0}^{\mathcal{R}_1}} \cdot \underbrace{\begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix}}_{\mathbf{u}|_{\mathcal{R}_1}}. \tag{1.5}$$

We can see a rotation matrix $\mathbf{R}_{\mathcal{R}_0}^{\mathcal{R}_1}$ whose columns are the coordinates of $\mathbf{i}_1, \mathbf{j}_1, \mathbf{k}_1$ expressed in the absolute system $\mathcal{R}_0$. In other words:

$$\mathbf{R}_{\mathcal{R}_0}^{\mathcal{R}_1} = \left(\begin{array}{c|c|c} & & \\ \mathbf{i}_1|_{\mathcal{R}_0} & \mathbf{j}_1|_{\mathcal{R}_0} & \mathbf{k}_1|_{\mathcal{R}_0} \\ & & \end{array}\right)$$

This matrix depends on time and links the frame $\mathcal{R}_1$ to $\mathcal{R}_0$. The matrix $\mathbf{R}_{\mathcal{R}_0}^{\mathcal{R}_1}$ is often referred to as a *direction cosine matrix* since its components involve the direction cosines of the basis vectors of the two coordinate systems. Likewise, if we would have several systems $\mathcal{R}_0, \ldots, \mathcal{R}_n$ (see Figure 1.4), we would have:

$$\mathbf{u}|_{\mathcal{R}_0} = \mathbf{R}_{\mathcal{R}_0}^{\mathcal{R}_1} \cdot \mathbf{R}_{\mathcal{R}_1}^{\mathcal{R}_2} \cdot \ \ldots \ \cdot \mathbf{R}_{\mathcal{R}_{n-1}}^{\mathcal{R}_n} \cdot \mathbf{u}|_{\mathcal{R}_n}.$$

**Dead Reckoning**. Let us consider the situation of a robot moving in a three-dimensional environment. Let us call $\mathcal{R}_0 : (\mathbf{o}_0, \mathbf{i}_0, \mathbf{j}_0, \mathbf{k}_0)$ its reference frame (for example, the frame of the robot at an initial time). The position of the robot is represented by the vector $\mathbf{p}(t)$ expressed in $\mathcal{R}_0$ and its attitude (in other words its orientation) by the rotation matrix $\mathbf{R}(t)$ which represents the coordinates of the vectors $\mathbf{i}_1, \mathbf{j}_1, \mathbf{k}_1$ of the coordinate system $\mathcal{R}_1$ of the robot expressed in the coordinate system

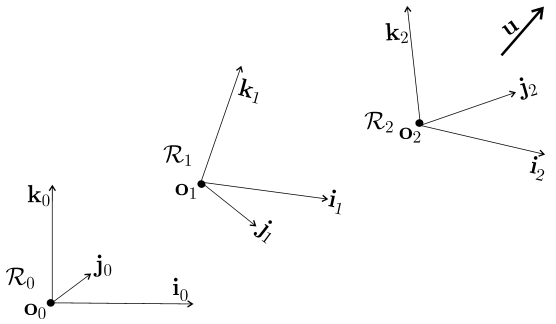


Figure 1.4: Composition of rotations

$\mathcal{R}_0$, at time $t$. It follows that:

$$\mathbf{R}(t) = \left( \left| \begin{array}{c} \\ \mathbf{i}_1|_{\mathcal{R}_0} \\ \\ \end{array} \right| \left| \begin{array}{c} \\ \mathbf{j}_1|_{\mathcal{R}_0} \\ \\ \end{array} \right| \left| \begin{array}{c} \\ \mathbf{k}_1|_{\mathcal{R}_0} \\ \\ \end{array} \right| \right) = \mathbf{R}_{\mathcal{R}_0}^{\mathcal{R}_1}(t)$$

This matrix can be returned by a precise attitude unit positioned on the robot. If the robot is also equipped with a *Doppler Velocity Log* (or DVL) which provides it with its speed vector $\mathbf{v}_r$ relative to the ground or the seabed, expressed in the coordinate system $\mathcal{R}_1$ of the robot, then the speed vector $\mathbf{v}$ of the robot satisfies:

$$\underbrace{\mathbf{v}|_{\mathcal{R}_0}}_{\dot{\mathbf{p}}(t)} \overset{[1.5]}{=} \underbrace{\mathbf{R}_{\mathcal{R}_0}^{\mathcal{R}_1}}_{\mathbf{R}(t)} \cdot \underbrace{\mathbf{v}|_{\mathcal{R}_1}}_{\mathbf{v}_r(t)}$$

Or equivalently:

$$\dot{\mathbf{p}}\,(t) = \mathbf{R}(t) \cdot \mathbf{v}_r(t). \tag{1.6}$$

*Dead reckoning* consists of integrating this state equation from the knowledge of $\mathbf{R}(t)$ and $\mathbf{v}_r(t)$.

## 1.2 Euler angles

### 1.2.1 Definition

In related literature, the angles proposed by Euler in 1770 to represent the orientation of solid bodies in space are not uniquely defined. We mainly distinguish between the roll-yaw-roll, roll-pitch-roll and roll-pitch-yaw formulations. It is the latter that we will choose since it is imposed in the mobile robotics language. Within this roll-pitch-yaw formulation, the Euler angles are sometimes referred to as *Cardan angles*. Any rotation matrix of $\mathbb{R}^3$ can be expressed in the form of the product

of three matrices as follows:

$$\mathbf{R}(\varphi,\theta,\psi) = \underbrace{\begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{\mathbf{R}_\psi} \cdot \underbrace{\begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix}}_{\mathbf{R}_\theta} \cdot \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \sin\varphi & \cos\varphi \end{pmatrix}}_{\mathbf{R}_\varphi}$$

or, in developed form:

$$\begin{pmatrix} \cos\theta\cos\psi & -\cos\varphi\sin\psi+\sin\theta\cos\psi\sin\varphi & \sin\psi\sin\varphi+\sin\theta\cos\psi\cos\varphi \\ \cos\theta\sin\psi & \cos\psi\cos\varphi+\sin\theta\sin\psi\sin\varphi & -\cos\psi\sin\varphi+\sin\theta\cos\varphi\sin\psi \\ -\sin\theta & \cos\theta\sin\varphi & \cos\theta\cos\varphi \end{pmatrix}. \quad (1.7)$$

$$\underbrace{\phantom{\begin{matrix}\cos\theta\cos\psi\\ \cos\theta\sin\psi\\ -\sin\theta\end{matrix}}}_{\mathbf{i}_1|_{\mathcal{R}_0}} \quad \underbrace{\phantom{\begin{matrix}-\cos\varphi\sin\psi+\sin\theta\cos\psi\sin\varphi\\ \cos\psi\cos\varphi+\sin\theta\sin\psi\sin\varphi\\ \cos\theta\sin\varphi\end{matrix}}}_{\mathbf{j}_1|_{\mathcal{R}_0}} \quad \underbrace{\phantom{\begin{matrix}\sin\psi\sin\varphi+\sin\theta\cos\psi\cos\varphi\\ -\cos\psi\sin\varphi+\sin\theta\cos\varphi\sin\psi\\ \cos\theta\cos\varphi\end{matrix}}}_{\mathbf{k}_1|_{\mathcal{R}_0}}$$

The angles $\psi, \theta, \varphi$ are the *Euler angles* and are respectively called the *heading*, the *elevation* and the *bank*. The terms *yaw*, *pitch*, *roll* are often employed, although they correspond, respectively, to variations of heading, elevation and bank.

**Gimbal lock**. When $\theta = \frac{\pi}{2}$ (the same effect happen as soon as $\cos\theta = 0$), we have

$$\begin{aligned} \mathbf{R}(\varphi,\theta,\psi) &= \begin{pmatrix} 0 & -\cos\varphi\sin\psi+\cos\psi\sin\varphi & \sin\psi\sin\varphi+\cos\psi\cos\varphi \\ 0 & \cos\psi\cos\varphi+\sin\psi\sin\varphi & -\cos\psi\sin\varphi+\cos\varphi\sin\psi \\ -1 & 0 & 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 & \sin(\psi-\varphi) & \cos(\psi-\varphi) \\ 0 & \cos(\psi-\varphi) & -\sin(\psi-\varphi) \\ -1 & 0 & 0 \end{pmatrix}. \end{aligned}$$

and thus,

$$\frac{d\mathbf{R}}{d\psi} = -\frac{d\mathbf{R}}{d\varphi}.$$

This corresponds to a singularity which tells us that when $\theta = \frac{\pi}{2}$ , we move on the manifold of rotation matrices SO(3), independently. This means that any trajectory $\mathbf{R}(t)$ cannot be followed by Euler angles.

**Rotation matrix to Euler angles**. Given a rotation matrix $\mathbf{R}$, we can easily find the three Euler angles by solving, following Equation (1.7), the equations:

$$\begin{cases} -\sin\theta = r_{31} & \\ \cos\theta\sin\varphi = r_{32} & \cos\theta\cos\varphi = r_{33} \\ \cos\theta\cos\psi = r_{11} & \cos\theta\sin\psi = r_{21} \end{cases}$$

By imposing $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$, $\varphi \in [-\pi, \pi]$, $\psi \in [-\pi, \pi]$, we find:

$$\theta = -\arcsin r_{31},\ \varphi = \operatorname{atan2}(r_{32}, r_{33}) \text{ et } \psi = \operatorname{atan2}(r_{21}, r_{11}).$$

Here atan2 is the two-argument arctangent function defined by

$$\alpha = \text{atan2}\,(y,x) \Leftrightarrow \alpha \in ]-\pi,\pi] \text{ and } \exists r>0 \mid \left\{ \begin{array}{l} x = r\cos\alpha \\ y = r\cos\alpha \end{array} \right. \tag{1.8}$$

### 1.2.2 Rotation vector of an Euler matrix

Let us consider a solid body moving in a coordinate system $\mathcal{R}_0$ and a coordinate system $\mathcal{R}_1$ attached to this body (refer to Figure 1.5). The conventions chosen here are those of the SNAME (*Society of Naval and Marine Engineers*). The two coordinate systems are assumed to be orthonormal. Let $\mathbf{R}(t) = \mathbf{R}(\psi(t),\theta(t),\varphi(t))$ be the rotation matrix that links the two systems. We need to find the instantaneous rotation vector $\boldsymbol{\omega}$ of the solid body relative to $\mathcal{R}_0$ in function of $\psi,\theta,\varphi,\dot{\psi},\dot{\theta},\dot{\varphi}$. We have:

$$\begin{array}{rcl}
\boldsymbol{\omega}|_{\mathcal{R}_0} & \overset{[1.2]}{=} & \mathbf{Ad}^{-1}\left(\dot{\mathbf{R}}\cdot\mathbf{R}^{\mathrm{T}}\right) \\
& \overset{[1.7]}{=} & \mathbf{Ad}^{-1}\left(\frac{d}{dt}\left(\mathbf{R}_\psi\cdot\mathbf{R}_\theta\cdot\mathbf{R}_\varphi\right)\cdot\mathbf{R}_\varphi^{\mathrm{T}}\cdot\mathbf{R}_\theta^{\mathrm{T}}\cdot\mathbf{R}_\psi^{\mathrm{T}}\right) \\
& = & \mathbf{Ad}^{-1}\left(\left(\dot{\mathbf{R}}_\psi\cdot\mathbf{R}_\theta\cdot\mathbf{R}_\varphi+\mathbf{R}_\psi\cdot\dot{\mathbf{R}}_\theta\cdot\mathbf{R}_\varphi+\mathbf{R}_\psi\cdot\mathbf{R}_\theta\cdot\dot{\mathbf{R}}_\varphi\right)\cdot\mathbf{R}_\varphi^{\mathrm{T}}\cdot\mathbf{R}_\theta^{\mathrm{T}}\cdot\mathbf{R}_\psi^{\mathrm{T}}\right) \\
& = & \mathbf{Ad}^{-1}\left(\dot{\mathbf{R}}_\psi\cdot\mathbf{R}_\psi^{\mathrm{T}}+\mathbf{R}_\psi\cdot\dot{\mathbf{R}}_\theta\cdot\mathbf{R}_\theta^{\mathrm{T}}\cdot\mathbf{R}_\psi^{\mathrm{T}}+\mathbf{R}_\psi\cdot\mathbf{R}_\theta\cdot\dot{\mathbf{R}}_\varphi\cdot\mathbf{R}_\varphi^{\mathrm{T}}\cdot\mathbf{R}_\theta^{\mathrm{T}}\cdot\mathbf{R}_\psi^{\mathrm{T}}\right) \\
& \overset{[1.2]}{=} & \mathbf{Ad}^{-1}\left(\dot{\psi}\mathbf{Ad}(\mathbf{k})+\mathbf{R}_\psi\cdot\left(\dot{\theta}\mathbf{Ad}(\mathbf{j})\right)\cdot\mathbf{R}_\psi^{\mathrm{T}}+\mathbf{R}_\psi\cdot\mathbf{R}_\theta\cdot\left(\dot{\varphi}\mathbf{Ad}(\mathbf{i})\right)\cdot\mathbf{R}_\theta^{\mathrm{T}}\cdot\mathbf{R}_\psi^{\mathrm{T}}\right) \\
& \overset{[1.3]}{=} & \mathbf{Ad}^{-1}\left(\dot{\psi}\cdot\mathbf{Ad}(\mathbf{k})+\dot{\theta}\cdot\mathbf{Ad}(\mathbf{R}_\psi\cdot\mathbf{j})+\dot{\varphi}\cdot\mathbf{Ad}(\mathbf{R}_\psi\cdot\mathbf{R}_\theta\cdot\mathbf{i})\right) \\
& = & \dot{\psi}\cdot\mathbf{k}+\dot{\theta}\cdot\mathbf{R}_\psi\cdot\mathbf{j}+\dot{\varphi}\cdot\mathbf{R}_\psi\cdot\mathbf{R}_\theta\cdot\mathbf{i}.
\end{array}$$

Thus, after having calculated the quantities $\mathbf{k}$, $\mathbf{R}_\psi\mathbf{j}$ and $\mathbf{R}_\psi\cdot\mathbf{R}_\theta\cdot\mathbf{i}$ in the coordinate system $\mathcal{R}_0$, we have:

$$\boldsymbol{\omega}|_{\mathcal{R}_0} = \dot{\psi}\cdot\begin{pmatrix}0\\0\\1\end{pmatrix}+\dot{\theta}\cdot\begin{pmatrix}-\sin\psi\\\cos\psi\\0\end{pmatrix}+\dot{\varphi}\cdot\begin{pmatrix}\cos\theta\cos\psi\\\cos\theta\sin\psi\\-\sin\theta\end{pmatrix}.$$

And from this we get the result:

$$\boldsymbol{\omega}|_{\mathcal{R}_0} = \begin{pmatrix}\cos\theta\cos\psi & -\sin\psi & 0\\ \cos\theta\sin\psi & \cos\psi & 0\\ -\sin\theta & 0 & 1\end{pmatrix}\begin{pmatrix}\dot{\varphi}\\\dot{\theta}\\\psi\end{pmatrix}. \tag{1.9}$$

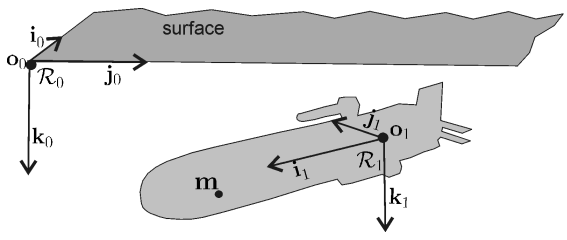


Figure 1.5: The coordinate system $\mathcal{R}_1 : (\mathbf{o}_1,\mathbf{i}_1,\mathbf{j}_1,\mathbf{k}_1)$ attached to the robot

Note that this matrix is singular when $\cos\theta = 0$. We will therefore take care to never have an elevation $\theta$ equal to $\pm\frac{\pi}{2}$, always to avoid the gimbal lock.

## 1.3 Kinematic model of a solid robot

A robot (airplane, submarine, boat) can often be considered a solid whose inputs are the (tangential and angular) accelerations. Indeed, these are analytic functions of the forces that are at the origin of the robot's movement. Here, we will consider that the inputs of the kinematic model are the tangential accelerations and the angular speeds. The reason for this is that these are directly measurable (if expressed in the robot's coordinate system) and that we may consider them to be directly controllable (even if a rotation can take a substantial amount of time for larger structures). The state vector for a kinematic model is composed of the vector $\mathbf{p} = (p_x, p_y, p_z)$ that gives the coordinates of the center of the robot expressed in the absolute inertial coordinate system $\mathcal{R}_0$, the three Euler angles $(\varphi, \theta, \psi)$ and the speed vector $\mathbf{v}_r$ of the robot expressed in its own coordinate system. The inputs of the system are for one the acceleration $\mathbf{a}_r = \mathbf{a}_{\mathcal{R}_1}$ of the center of the robot expressed in its own coordinate system and second, the vector $\boldsymbol{\omega}_r = \boldsymbol{\omega}_{\mathcal{R}_1/\mathcal{R}_0|\mathcal{R}_1} = (\omega_x, \omega_y, \omega_z)$ corresponding to the rotation vector of the robot relative to $\mathcal{R}_0$ expressed in the coordinate system $\mathcal{R}_1$ of the robot. It is indeed conventional to express $\mathbf{a}$, $\boldsymbol{\omega}$ in the coordinate system of the robot since these quantities are generally measured by the robot itself *via* the sensors attached on it. They are therefore naturally expressed in the frame of the robot. The first state equation is:

$$\dot{\mathbf{p}} \overset{[1.6]}{=} \mathbf{R}\left(\varphi, \theta, \psi\right) \cdot \mathbf{v}_r.$$

In order to express $\mathbf{v}_r$, let us differentiate this equation. We obtain:

$$\ddot{\mathbf{p}} = \dot{\mathbf{R}} \cdot \mathbf{v}_r + \mathbf{R} \cdot \dot{\mathbf{v}}_r$$

with $\mathbf{R} = \mathbf{R}\left(\varphi, \theta, \psi\right)$. From this equation, we isolate $\dot{\mathbf{v}}_r$ to get

$$\dot{\mathbf{v}}_r = \underbrace{\mathbf{R}^{\mathrm{T}} \cdot \ddot{\mathbf{p}}}_{\mathbf{a}_r} - \mathbf{R}^{\mathrm{T}}\dot{\mathbf{R}} \cdot \mathbf{v}_r \overset{[1.4]}{=} \mathbf{a}_r - \mathbf{Ad}\left(\boldsymbol{\omega}_{\mathcal{R}_1/\mathcal{R}_0|\mathcal{R}_1}\right) \cdot \mathbf{v}_r.$$

Thus:

$$\dot{\mathbf{v}}_r = \mathbf{a}_r - \boldsymbol{\omega}_r \wedge \mathbf{v}_r$$

constitutes the second state equation. Finally, we also need to express $\dot{\psi}, \dot{\theta}, \dot{\varphi}$ in function of the state variables. The relation :

$$\boldsymbol{\omega}_{|\mathcal{R}_0} = \mathbf{R}\left(\varphi, \theta, \psi\right) \cdot \boldsymbol{\omega}_{|\mathcal{R}_1}$$

becomes, following Equation [1.9]:

$$\begin{pmatrix} \cos\theta\cos\psi & -\sin\psi & 0 \\ \cos\theta\sin\psi & \cos\psi & 0 \\ -\sin\theta & 0 & 1 \end{pmatrix} \begin{pmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \mathbf{R}(\varphi,\theta,\psi)\cdot\boldsymbol{\omega}_r.$$

By isolating in this expression the vector $\left(\dot{\psi},\dot{\theta},\dot{\varphi}\right)$, we obtain the third state equation. By bringing together the three state equations, we obtain the following kinematic model for the robot:

$$\begin{cases} \dot{\mathbf{p}} & = \mathbf{R}(\varphi,\theta,\psi)\ \cdot\ \mathbf{v}_r \\ \dot{\mathbf{v}}_r & = \mathbf{a}_r - \boldsymbol{\omega}_r \wedge \mathbf{v}_r \\ \begin{pmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} & = \begin{pmatrix} 1 & \tan\theta\sin\varphi & \tan\theta\cos\varphi \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \frac{\sin\varphi}{\cos\theta} & \frac{\cos\varphi}{\cos\theta} \end{pmatrix} \cdot \boldsymbol{\omega}_r \end{cases} \tag{1.10}$$

**On an horizontal plane**. For a robot moving on a horizontal plane, we have $\varphi = \theta = 0$. The last equation gives us $\dot{\psi} = \omega_{r3}$, $\dot{\theta} = \omega_{r2}$ and $\dot{\varphi} = \omega_{r1}$. In such a case, there is a perfect correspondence between the components of $\boldsymbol{\omega}_r$ and the differentials of the Euler angles. There are singular cases, for instance when $\theta = \frac{\pi}{2}$ (this is the case when the robot points upwards), in which the differentials of the Euler angles cannot be defined. Using the rotation vector is often preferred since it does not have such singularities.

**Dead reckoning**. For dead reckoning (in other words without external sensors) there are generally extremely precise laser gyrometers (around 0.001 $deg/s.$). These make use of the Sagnac effect (in a circular optical fiber turning around itself, the time taken by light to travel an entire round-trip depends on the path direction). Using three fibers, these gyrometers generate the vector $\boldsymbol{\omega}_r = (\omega_x, \omega_y, \omega_z)$. There are also accelerometers capable of measuring the acceleration $\mathbf{a}_r$ with a very high degree of precision. In pure inertial mode, we determine our position by differentiating Equations [1.10] only using the acceleration $\mathbf{a}_r$ and the rotation speed $\boldsymbol{\omega}_r$, both expressed in the coordinate system of the robot. In the case where we are measuring the quantity $\mathbf{v}_r$ (also expressed in the frame of the robot) with a *Doppler velocity log*, we only need to integrate the first and the last of these three equations. Finally, when the robot is a correctly ballasted submarine or a terrestrial robot moving on a relatively plane ground, we know *a priori* that on average the bank and the elevation are equal to zero. We may therefore incorporate this information through a Kalman filter in order to limit the drift in positioning. An efficient inertial unit integrates an amalgamation of all the available information.

**Inertial unit**. A pure inertial unit (without hybridization and without taking into account Earth's gravity) represents the robot by the kinematic model of Figure 1.6, which itself uses the state equations given in [1.10]. This system is written in the form $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{u})$ where $\mathbf{u} = (\mathbf{a}_r, \boldsymbol{\omega}_r)$ is the vector of the measured inertial inputs (accelerations and rotation speeds viewed by an observer on the ground, but expressed in the frame of the robot) and $\mathbf{x} = (\mathbf{p}, \mathbf{v}_r, \psi, \theta, \varphi)$ is the state vector. For the moment, we use a numerical integration method such as the Euler method. This leads to replacing the differential equation $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{u})$ with the recurrence:

$$\mathbf{x}(t+dt) = \mathbf{x}(t) + dt\cdot\mathbf{f}(\mathbf{x}(t),\mathbf{u}(t)).$$

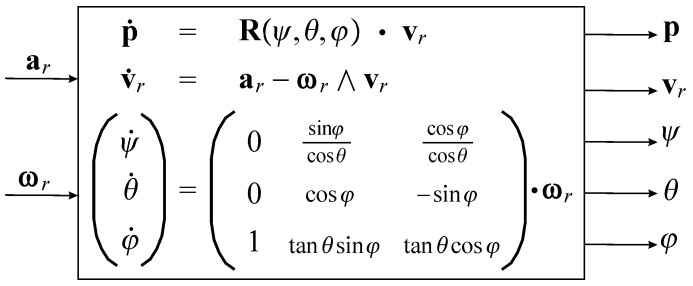


Figure 1.6: Kinematic model used by an inertial unit

## 1.4 Dynamic modeling

### 1.4.1 Principle

For the dynamic modeling of a submarine the reference work is the book of Fossen [FOS 2002], but the related notions can also be used for other types of solid robots such as planes, boats or quadrotors. In order to obtain a dynamic model, it is sufficient to take the kinematic equations and to consider that the angular and tangential accelerations caused by forces and dynamic performance. These quantities become the new inputs of our system. The link between the accelerations and the forces is done by Newton's second law (or the *fundamental principle of dynamics*). Thus, for instance if $\mathbf{f}$ is the net force resulting from the external forces expressed in the inertial frame and $m$ is the mass of the robot, we have :

$$m\ddot{\mathbf{p}} = \mathbf{f}.$$

Since the speed and accelarations are generally measured by sensors embedded in the system, we generally prefer to express the speed and accelerations in the frame of the robot. For instance, since $\mathbf{a}_r = \boldsymbol{\omega}_r \wedge \mathbf{v}_r + \dot{\mathbf{v}}_r$ (see (1.10)), the Newton's second law for translation also be written as

$$m(\boldsymbol{\omega}_r \wedge \mathbf{v}_r + \dot{\mathbf{v}}_r) = \mathbf{f}_r,$$

where $\mathbf{f}_r$ is the applied forces vector expressed in the robot frame. The same relation, known as the *Euler's rotation equation*, exists for rotations. It is given by

$$\mathbf{I}\dot{\boldsymbol{\omega}}_r + \boldsymbol{\omega}_r \wedge (\mathbf{I}\boldsymbol{\omega}_r) = \boldsymbol{\tau}_r \tag{1.11}$$

where $\boldsymbol{\tau}_r$ is the applied torque and $\boldsymbol{\omega}_r$ is the rotation vector both expressed in the robot frame. The inertia matrix $\mathbf{I}$ is attached to the robot (*i.e.*, computed in the robot frame) and we generaly choose the robot frame to make $\mathbf{I}$ diagonal. This relation is a consequence of the *Euler's second law* which states that in an inertial frame, the time derivative of the angular momentum equals the applied

torque. In the inertial frame, this can be expressed as

$$\begin{aligned} & \tfrac{d}{dt}\left(\mathbf{R}\cdot\mathbf{I}\cdot\boldsymbol{\omega}_r\right) = \mathbf{R}\cdot\boldsymbol{\tau}_r \\ \Leftrightarrow\quad & \dot{\mathbf{R}}\mathbf{I}\cdot\boldsymbol{\omega}_r + \mathbf{R}\cdot\mathbf{I}\cdot\dot{\boldsymbol{\omega}}_r = \mathbf{R}\cdot\boldsymbol{\tau}_r \\ \Leftrightarrow\quad & \mathbf{R}^T\dot{\mathbf{R}}\mathbf{I}\cdot\boldsymbol{\omega}_r + \mathbf{I}\cdot\dot{\boldsymbol{\omega}}_r = \boldsymbol{\tau}_r \\ \Leftrightarrow\quad & \boldsymbol{\omega}_r \wedge (\mathbf{I}\boldsymbol{\omega}_r) + \mathbf{I}\dot{\boldsymbol{\omega}}_r = \boldsymbol{\tau}_r. \end{aligned}$$

### 1.4.2 Modeling a quadrotor

As an illustration, we consider a quadrotor (see Figure 1.7) for which we want a dynamic model. The robot has four propellers that can be tuned independently. This will allow us to control the attitude and position of the robot by changing the speeds of the motors.

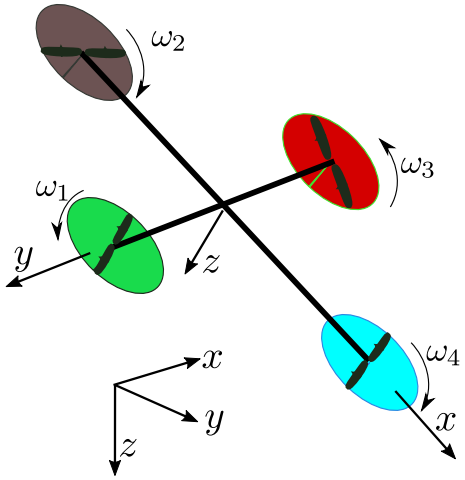


Figure 1.7: Quadrotor

We distinguish the front/rear propellers (blue and black), rotating clockwise and right/left propellers (red and green), rotating counterclockwise. The value of the force generated by the $i$th propeller is proportional to the squared speeds of the rotors, *i.e.*, is equal to $\beta \cdot \omega_i \cdot |\omega_i|$, where $\beta$ is the thrust factor. Denote by $\delta$ the drag factor and $\ell$ the distance between any rotor and the center of the robot. The forces and torques generated on the robot are

$$\begin{pmatrix} \tau_0 \\ \tau_1 \\ \tau_2 \\ \tau_3 \end{pmatrix} = \begin{pmatrix} \beta & \beta & \beta & \beta \\ -\beta\ell & 0 & \beta\ell & 0 \\ 0 & -\beta\ell & 0 & \beta\ell \\ -\delta & \delta & -\delta & \delta \end{pmatrix} \cdot \begin{pmatrix} \omega_1 \cdot |\omega_1| \\ \omega_2 \cdot |\omega_2| \\ \omega_3 \cdot |\omega_3| \\ \omega_4 \cdot |\omega_4| \end{pmatrix}$$

where $\tau_0$ is the total thrust generated by the rotors and $\tau_1, \tau_2, \tau_3$ are the torques generated by differences in the rotor speeds.

We will now show that the state equation for the quadrotor

$$\begin{cases} \dot{\mathbf{p}} = \mathbf{R}(\varphi,\theta,\psi) \cdot \mathbf{v}_r & (i) \\ \begin{pmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} 1 & \tan\theta\sin\varphi & \tan\theta\cos\varphi \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \frac{\sin\varphi}{\cos\theta} & \frac{\cos\varphi}{\cos\theta} \end{pmatrix} \cdot \boldsymbol{\omega}_r & (ii) \\ \dot{\mathbf{v}}_r = \mathbf{R}^T(\varphi,\theta,\psi) \cdot \begin{pmatrix} 0 \\ 0 \\ g \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ -\frac{\tau_0}{m} \end{pmatrix} - \boldsymbol{\omega}_r \wedge \mathbf{v}_r & (iii) \\ \dot{\boldsymbol{\omega}}_r = \mathbf{I}^{-1} \cdot \left( \begin{pmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{pmatrix} - \boldsymbol{\omega}_r \wedge (\mathbf{I} \cdot \boldsymbol{\omega}_r) \right) & (iv) \end{cases}$$

where $\mathbf{p}$ is the position and $(\varphi,\theta,\psi)$, are the Euler angles of the robot. The three first equations (i), (ii), (iii) correspond to the kinematic equations and have already been derived (see Equation (1.10)). Note that in Equation (iii), the acceleration $\mathbf{a}_r$ (expressed in the robot frame) comes from the total force $\tau_0$ and the gravity $g$:

$$\mathbf{a}_r = \mathbf{R}^T(\varphi,\theta,\psi) \cdot \begin{pmatrix} 0 \\ 0 \\ g \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ -\frac{\tau_0}{m} \end{pmatrix}.$$

Since the gravity vector is expressed naturally in the inertial frame (contrary to the global force $\tau_0$), we have to multiply the gravity vector by the Euler rotation matrix $\mathbf{R}^T(\varphi,\theta,\psi)$. Equation (iv) comes from the Euler's rotation equation (1.11)

$$\dot{\boldsymbol{\omega}}_r = \mathbf{I}^{-1} \cdot (\boldsymbol{\tau}_r - \boldsymbol{\omega}_r \wedge (\mathbf{I} \cdot \boldsymbol{\omega}_r))$$

where $\boldsymbol{\tau}_r = (\tau_1, \tau_2, \tau_3)$ is the torque vector.

# Exercises

EXERCISE 1.1.– *Properties of the adjoint matrix*

Let us consider the vector $\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z)$ and its adjoint matrix $\mathbf{Ad}(\boldsymbol{\omega})$.

1) Show that the eigenvalues of $\mathbf{Ad}(\boldsymbol{\omega})$ are $\{0, ||\boldsymbol{\omega}||i, -||\boldsymbol{\omega}||i\}$. Give an eigenvector associated with 0. Discuss.

2) Show that the vector $\mathbf{Ad}(\boldsymbol{\omega})\ \mathbf{x} = \boldsymbol{\omega} \wedge \mathbf{x}$ is a vector perpendicular to $\boldsymbol{\omega}$ and $\mathbf{x}$, such that the trihedron $(\boldsymbol{\omega}, \mathbf{x}, \boldsymbol{\omega} \wedge \mathbf{x})$ is direct.

3) Show that the norm of $\boldsymbol{\omega} \wedge \mathbf{x}$ is surface of the parallelogram $\mathcal{A}$ mediated by $\boldsymbol{\omega}$ and $\mathbf{x}$.

---

EXERCISE 1.2.– *Jacobi identity*

The Jacobi identity is written as:

$$\mathbf{a} \wedge (\mathbf{b} \wedge \mathbf{c}) + \mathbf{c} \wedge (\mathbf{a} \wedge \mathbf{b}) + \mathbf{b} \wedge (\mathbf{c} \wedge \mathbf{a}) = \mathbf{0}.$$

1) Show that this identity is equivalent to:

$$\mathbf{Ad}(\mathbf{a} \wedge \mathbf{b}) = \mathbf{Ad}(\mathbf{a})\,\mathbf{Ad}(\mathbf{b}) - \mathbf{Ad}(\mathbf{b})\,\mathbf{Ad}(\mathbf{a})$$

where $\mathbf{Ad}(\boldsymbol{\omega})$ is the adjoint matrix of the vector $\boldsymbol{\omega} \in \mathbb{R}^3$.

2) In the space of skew-symmetric matrices, the Lie bracket is defined as follows:

$$[\mathbf{A}, \mathbf{B}] = \mathbf{A} \cdot \mathbf{B} - \mathbf{B} \cdot \mathbf{A}.$$

Show that :

$$\mathbf{Ad}(\mathbf{a} \wedge \mathbf{b}) = [\mathbf{Ad}(\mathbf{a}), \mathbf{Ad}(\mathbf{b})].$$

3) An *algebra* is an algebraic structure $(\mathcal{A}, +, \times, \cdot)$ over a body $\mathbb{K}$, if (i) $(\mathcal{A}, +, \cdot)$ is a vector space over $\mathbb{K}$ ; (ii) the multiplication rule $\times$ of $\mathcal{A} \times \mathcal{A} \to \mathcal{A}$ is left- and right-distributive with respect to $+$ and (iii) for all $\alpha, \beta \in \mathbb{K}$, and for all $x, y \in \mathcal{A}$, $\alpha \cdot x \times \beta \cdot y$, $(\alpha\beta) \cdot (x \times y)$. Notice that in general, an algebra is non-commutative ($x \times y \neq y \times x$) and non-associative ($(x \times y) \times z \neq x \times (y \times z)$). A *Lie algebra* $(\mathcal{G}, +, [\,]\,, \cdot)$ is a non-commutative and non-associative algebra in which multiplication, denoted by a so-called Lie bracket, verifies (i) $[,]$ that is bilinear, in other words linear with respect to each variable ; (ii) $[x, y] = -[y, x]$ (antisymmetry) and (iii) $[x, [y, z]] + [y, [z, x]] + [z, [x, y]] = 0$

(Jacobi relation). Verify that the set $(\mathbb{R}^3, +, \wedge, \cdot)$ forms a *Lie algebra*.

---

EXERCISE 1.3.– *Varignon's formula*

Let us consider a solid body whose center of gravity remains at the origin of a Galilean coordinate system and is rotating around an axis $\Delta$ with a rotation vector of $\boldsymbol{\omega}$. Give the equation of the trajectory of a point $\mathbf{x}$ of the body.

---

EXERCISE 1.4.– *Rodrigues' formula*

Let us consider a solid body whose center of gravity remains at the origin of a Galilean coordinate system and is rotating around an axis $\Delta$. The position of a point $\mathbf{x}$ of the body satisfies the state equation (Varignon's formula):

$$\dot{\mathbf{x}} = \boldsymbol{\omega} \wedge \mathbf{x}$$

where $\boldsymbol{\omega}$ is parallel to the axis of rotation $\Delta$ and $||\boldsymbol{\omega}||$ is the rotation speed of the body (in rad.$s^{-1}$).

1) Show that this state equation can be written in the form:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}.$$

Explain why the matrix $\mathbf{A}$ is often denoted by $\boldsymbol{\omega}\wedge$.

2) Give the expression of the solution of the state equation.

3) Deduce from this that the expression of the rotation matrix $\mathbf{R}$ with angle $||\boldsymbol{\omega}||$ around $\boldsymbol{\omega}$ is given by the following formula, referred to as *Rodrigues' formula*:

$$\mathbf{R} = e^{\boldsymbol{\omega}\wedge}.$$

4) Calculate the eigenvalues of $\mathbf{A}$ and show that $\boldsymbol{\omega}$ is the eigenvector associated with the zero eigenvalue. Discuss.

5) What are the eigenvalues of $\mathbf{R}$ ?

6) Using the previous questions, give the expression of a rotation around the vector $\boldsymbol{\omega} = (1,\ 0,\ 0)$ of angle $\alpha$.

7) Write a program in MATLAB, `eulermat(phi,theta,psi)` that uses Rodrigues' formula to return the Euler matrix.

---

EXERCISE 1.5.– *Geometric approach to Rodrigues' formula*

Let us consider the rotation $\mathcal{R}_{\mathbf{n},\varphi}$ of angle $\varphi$ around the unit vector $\mathbf{n}$. Let $\mathbf{u}$ be a vector that we will subject to this rotation. The vector $\mathbf{u}$ can be decomposed as follows:

$$\mathbf{u} = \underbrace{< \mathbf{u}, \mathbf{n} > \cdot \mathbf{n}}_{\mathbf{u}_{||}} + \underbrace{\mathbf{u} - < \mathbf{u}, \mathbf{n} > \cdot \mathbf{n}}_{\mathbf{u}_{\perp}}$$

where $\mathbf{u}_{||}$ is collinear to $\mathbf{n}$ and $\mathbf{u}_{\perp}$ is in the plane $P_{\perp}$ orthogonal to $\mathbf{n}$ (see Figure 1.8).

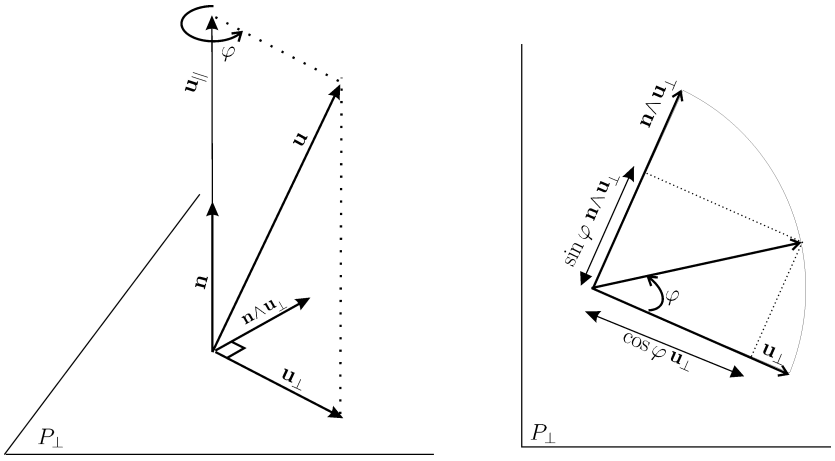


Figure 1.8: Rotation of the vector $\mathbf{u}$ around the vector $\mathbf{n}$ ; left : perspective view ; right : view from above

1) Prove Rodrigues' formula given by:

$$\mathcal{R}_{\mathbf{n},\varphi}(\mathbf{u}) = <\mathbf{u},\mathbf{n}>\cdot\mathbf{n} + (\cos\varphi)(\mathbf{u} - <\mathbf{u},\mathbf{n}>\cdot\mathbf{n}) + (\sin\varphi)(\mathbf{n}\wedge\mathbf{u}).$$

2) Using the double vector product formula $\mathbf{a}\wedge(\mathbf{b}\wedge\mathbf{c}) = (\mathbf{a}^{\mathrm{T}}\mathbf{c})\cdot\mathbf{b} - \left(\mathbf{a}^{\mathrm{T}}\mathbf{b}\right)\cdot\mathbf{c}$, on the element $\mathbf{n}\wedge(\mathbf{n}\wedge\mathbf{u})$, show that Rodrigues' formula can also been written as:

$$\mathcal{R}_{\mathbf{n},\varphi}(\mathbf{u}) = \mathbf{u} + (1-\cos\varphi)(\mathbf{n}\wedge(\mathbf{n}\wedge\mathbf{u})) + (\sin\varphi)(\mathbf{n}\wedge\mathbf{u}).$$

Deduce from this that the matrix associated with the linear operator $\mathcal{R}_{\mathbf{n},\varphi}$ is written as:

$$\mathbf{R}_{\mathbf{n},\varphi} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} + (1-\cos\varphi)\begin{pmatrix} -n_y^2-n_z^2 & n_xn_y & n_xn_z \\ n_xn_y & -n_x^2-n_z^2 & n_yn_z \\ n_xn_z & n_yn_z & -n_x^2-n_y^2 \end{pmatrix} + (\sin\varphi)\begin{pmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{pmatrix}.$$

3) Conversely, we are given a rotation matrix $\mathbf{R}_{\mathbf{n},\varphi}$ for which we wish to find the axis of rotation $\mathbf{n}$ and the angle of rotation $\varphi$. Give an expression for $\mathbf{R}_{\mathbf{n},\varphi} - \mathbf{R}_{\mathbf{n},\varphi}^{\mathrm{T}}$ and use it to obtain $\mathbf{n}$ and $\varphi$ in function of $\mathbf{R}_{\mathbf{n},\varphi}$. For a geometric illustration, Figure 1.9 might prove to be useful.

4) Using a Maclaurin series development of sin$\varphi$ and cos$\varphi$, show that:

$$\mathbf{R}_{\mathbf{n},\varphi} = \exp(\varphi\cdot\mathbf{Ad}(\mathbf{n})),$$

which sometimes written as:

$$\mathbf{R}_{\mathbf{n},\varphi} = \exp(\varphi\cdot\mathbf{n}\wedge).$$

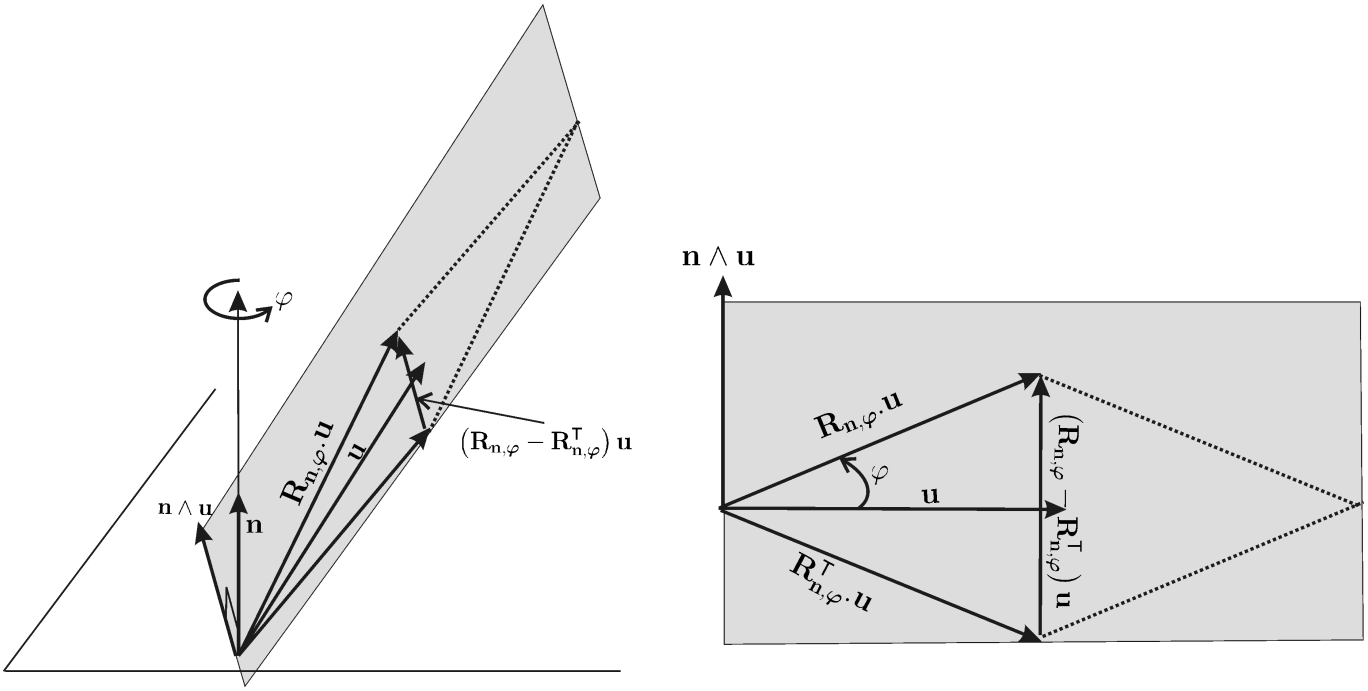


Figure 1.9: On the left, we have a view of the rotation of angle $\varphi$ around $\mathbf{n}$ ; on the right a visualization of the section corresponding to the Rodrigues rhombus

---

Exercise 1.6.– *Quaternions*

Quaternion were discovered by W. R. Hamilton are also used to represent a rotation in a 3-dimensional space (see, *e.g.*, [COR 11]).

A quaternion $\mathring{q}$ is an extension of the complex number. It corresponds to a scalar $s$ plus a vector $\mathbf{v}$ of $\mathbb{R}^3$. We use the equivalent following notations:

$$\mathring{q} = s + v_1 i + v_2 j + v_3 k = s < v_1, v_2, v_3 >= s < \mathbf{v} >,$$

where

$$i^2 = j^2 = k^2 = ijk = -1.$$

1) From these relations, fill the following multiplication table:

| $\cdot$ | 1 | $i$ | $j$ | $k$ |
|---|---|---|---|---|
| 1 | 1 | $i$ | $j$ | $k$ |
| $i$ | $i$ | ? | ? | ? |
| $j$ | $j$ | ? | ? | ? |
| $k$ | $k$ | ? | ? | ? |

2) A unit-quaternion $\mathring{q}$ is a quaternion with a unit magnitude:

$$|\mathring{q}|^2 = s^2 + v_1^2 + v_2^2 + v_3^2 = 1.$$

Consider the rotation obtained by a rotation around the unit vector $\mathbf{n}$ with an angle $\theta$. From the Rodrigues formula, we know that the corresponding rotation matrix is $\exp(\theta \cdot \mathbf{n}\wedge)$. To this rotation, we associate the quaternion:

$$\mathring{q} = \cos(\frac{\theta}{2}) + \sin(\frac{\theta}{2}) \cdot < \mathbf{n} > .$$

Check that this quaternion has a unit magnitude.

3) Show that the quaternion $\mathring{q}$ and its opposite$-\mathring{q}$, both correspond to the same rotation in $\mathbb{R}^3$.

4) Assuming that the composition of rotations corresponds to a multiplication of quaternions, show that

$$(s < \mathbf{v} >)^{-1} \quad = \quad s < -\mathbf{v} > .$$

5) We consider the Euler rotation matrix with $\psi = \theta = \varphi = \frac{\pi}{2}$. It corresponds to a rotation around the unit vector $\boldsymbol{\omega}$ with an angle $\alpha$. Give the values of $\boldsymbol{\omega}$ and $\alpha$ using 3 methods: the geometrical method (using hands only), the matrices (with MATLAB) and the quaternions.

---

EXERCISE 1.7.– *Schuler oscillations*

One of the fundamental components of an inertial unit is the inclinometer. This sensor gives the vertical direction. Traditionally, we use a pendulum (or a plumb line) for this. However, when we are moving, due to the accelerations the pendulum starts to oscillate and it can no longer be used to measure the vertical direction. Here, we are interested in designing a pendulum for which any horizontal acceleration does not lead to oscillations. Let us consider a pendulum with two masses $m$ at each end, situated at a distance $\ell_1$ and $\ell_2$ from the axis of rotation of the rod (see Figure 1.10). The axis moves over the surface of the Earth. We assume that $\ell_1$ and $\ell_2$ are small in comparison to Earth's radius $r$.

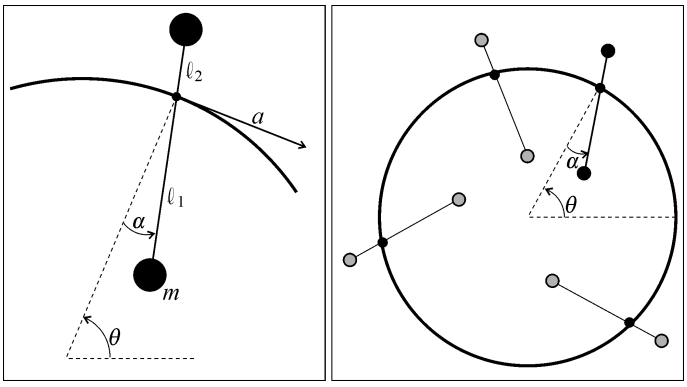


Figure 1.10: Inclinometer pendulum moving over the surface of the Earth

1) Find the state equations of the system.

2) Let us assume that $\alpha = \dot{\alpha} = 0$. For which values of $\ell_1$ and $\ell_2$ does the pendulum remain vertical, for any horizontal movement of the pendulum? What values does $\ell_2$ have to take if we let $\ell_1 = 1$ m and we take $r = 6\ 400$ km for the Earth's radius?

3) Let us assume that, as a result of disturbances, the pendulum starts to oscillate. The period of these oscillations is called the *Schuler period*. Calculate this period.

4) Simulate the system graphically by taking a Gaussian white noise as the acceleration input. Since the system is conservative, an Euler integration method will not perform well (the pendulum would gain energy). A higher-order integration scheme, such as that of *Runge Kutta*, should be used. This is given by:

$$\mathbf{x}(t+dt) \simeq \mathbf{x}(t) + dt \cdot \left( \frac{\mathbf{f}\left(\mathbf{x}(t), \mathbf{u}(t)\right)}{4} + \tfrac{3}{4}\mathbf{f}(\mathbf{x}(t) + \frac{2dt}{3}\ \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \mathbf{u}(t + \tfrac{2}{3}dt)) \right).$$

For an easier graphical representation, we will take $r = 10$ m, $\ell_1 = 1$ m and $g = 10$ ms$^{-2}$. Discuss the results.

---

EXERCISE 1.8.– *Brake detector*

We will now look at a problem that involves basis changes and rotation matrices. A car is preceded by another car $\mathbf{m}$ (which we will assume to be a point). We attach to this car the coordinate system $\mathcal{R}_1 : (\mathbf{o}_1, \mathbf{i}_1, \mathbf{j}_1)$ as represented on Figure 1.11. The coordinate system $\mathcal{R}_0 : (\mathbf{o}_0, \mathbf{i}_0, \mathbf{j}_0)$ is a ground frame assumed to be fixed.

This car is equipped with the following sensors:

- several odometers placed on the rear wheels allowing to measure the speed $v$ of the center of the rear axle ;
- a gyro giving the angular speed of the car $\dot{\theta}$, as well as the angular acceleration $\ddot{\theta}$ ;
- an accelerometer placed at $\mathbf{o}_1$ allowing to measure the acceleration vector $(\alpha, \beta)$ of $\mathbf{o}_1$ expressed in the coordinate system $\mathcal{R}_1$ ;
- using two radars placed at the front, our car is capable of (indirectly) measuring the coordinates $(a_1, b_1)$ of the point $\mathbf{m}$ in the coordinate system $\mathcal{R}_1$ as well as the first two derivatives $\left(\dot{a}_1, \dot{b}_1\right)$ and $\left(\ddot{a}_1, \ddot{b}_1\right)$.

On the other hand, the car is not equipped with a positioning system (such as a GPS, for example) that would allow it to gain knowledge of $x, y, \dot{x}, \dot{y}$. It does not have a compass for measuring the angle $\theta$. The quantities in play are the following:

| | |
|---|---|
| Measured | $v, \dot{\theta}, a_1, b_1, \alpha, \beta.$ |
| Unknown | $x, y, \dot{x}, \dot{y}, \theta, a_0, b_0$ |

When a quantity is measured, we assume that its differentials are also measured, but not their primitive. For example $\dot{a}_1, \dot{b}_1, \ddot{a}_1, \ddot{b}_1$ are considered to be measured since $a_1, b_1$ are measured. However, $\dot{\theta}$ is measured but $\theta$ isn't. We do not know the state equations of our car. The goal of this problem is finding a condition on the measured variables (and their derivatives) that will allow us to tell whether the point $\mathbf{m}$ is braking or not. We understand that such a condition would allow us to build

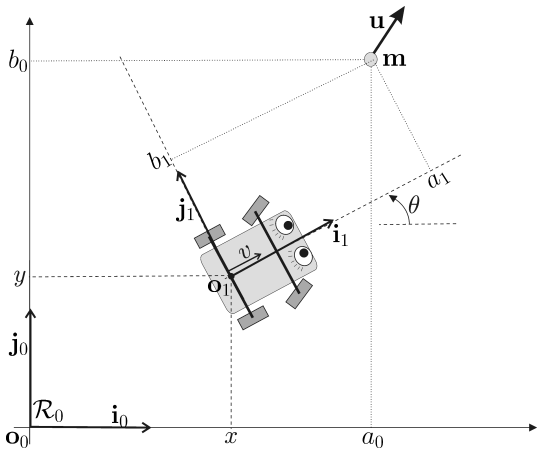


Figure 1.11: Car trying to detect whether the point $\mathbf{m}$ is braking

a warner informing us that the preceding vehicle is braking, even when its rear brake lights are not visible (fog, trailer without brake lights) or defective.

1) By expressing the Chasle relation ($\mathbf{o_0m} = \mathbf{o_0o_1} + \mathbf{o_1m}$) in the coordinate system $\mathcal{R}_0$, show the basis change formula:

$$\begin{pmatrix} a_0 \\ b_0 \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \mathbf{R}_\theta \begin{pmatrix} a_1 \\ b_1 \end{pmatrix}$$

where $\mathbf{R}_\theta$ is a rotation matrix.

2) Show that $\mathbf{R}_\theta^{\mathrm{T}}\ \dot{\mathbf{R}}_\theta$ is a skew-symmetric matrix and find its expression.

3) Let $\mathbf{u}$ be the speed vector of the point $\mathbf{m}$ viewed by a fixed observer. Find an expression $\mathbf{u}_{|\mathcal{R}_1}$ of the speed vector $\mathbf{u}$ expressed in the coordinate system $\mathcal{R}_1$. Express $\mathbf{u}_{|\mathcal{R}_1}$ in function of the measured variables $v, \dot{\theta}, a_1, \dot{a}_1, b_1, \dot{b}_1$.

4) We will now use the accelerometer of our vehicle, which gives us the acceleration vector $(\alpha, \beta)$ of $\mathbf{o}_1$, expressed in $\mathcal{R}_1$. By differentiating two times $\mathbf{m}_{|\mathcal{R}_0}$, give the expression of the acceleration $\mathbf{a}_{|\mathcal{R}_0}$ of $\mathbf{m}$ in the coordinate system $\mathcal{R}_0$. Deduce from this its expression $\mathbf{a}_{|\mathcal{R}_1}$ in the coordinate system $\mathcal{R}_1$. Give the expression of $\mathbf{a}_{|\mathcal{R}_1}$ only in function of the measured variables.

5) Find a condition on the measurements $\left(v, \dot{\theta}, a_1, b_1, \dot{a}_1, \dot{b}_1, \ddot{a}_1, \ddot{b}_1, \alpha, \beta\right)$ which allows to detect whether the vehicle in front is braking.

---

Exercise 1.9.– *Modeling an underwater robot*

The robot we will be modeling is the Redermor (*greyhound of the sea* in the Breton language). It is represented on Figure 1.12. It is an entirely autonomous underwater robot. This robot, developed by GESMA (Groupe d'Etude Sous-Marine de l'Atlantique - *Atlantic underwater research group*), has a length of 6 m, a diameter of 1 m and a weight of 3 800 kg. It has a very efficient propulsion and control system with the aim of finding mines on the seabed.

Let us build a local coordinate system $\mathcal{R}_0 : (\mathbf{o}_0, \mathbf{i}_0, \mathbf{j}_0, \mathbf{k}_0)$ over the area traveled by the robot. The point $\mathbf{o}_0$ is placed on the surface of the ocean. The vector $\mathbf{i}_0$ indicates north, $\mathbf{j}_0$ indicates east and

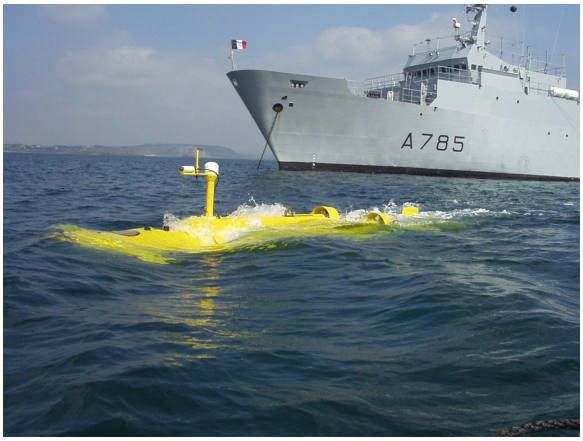


Figure 1.12: *Redermor* built by GESMA (Groupe d'Etude Sous-Marine de l'Atlantique - *Atlantic underwater research group*), on the water surface, still close to the boat it was launched from

$\mathbf{k}_0$ is oriented towards the center of the Earth. Let $\mathbf{p} = (p_x, p_y, p_z)$ be the coordinates of the center of the robot expressed in the coordinate system $\mathcal{R}_0$. The state variables of the underwater robot are its position $\mathbf{p}$ in the coordinate system $\mathcal{R}_0$, its tangential $v$ and its three Euler angles $\psi, \theta, \varphi$. Its inputs are the tangential acceleration $\dot{v}$ as well as three control surfaces which act respectively on $\omega_x, \omega_y, \omega_z$. More formally, we have:

$$\begin{cases} u_1 &= \dot{v} \\ vu_2 &= \omega_y \\ vu_3 &= \omega_z \\ u_4 &= \omega_x \end{cases}$$

where the factor $v$ preceding $u_2, u_3$ indicates that the robot is only able to turn left/right (through $u_3$) or up/down (through $u_2$) when it is advancing. Give the kinematic state model for this system.

---

EXERCISE 1.10.– *3D robot graphics*

Drawing two- or three-dimensional robots or objects on the screen is widely used for simulation in robotics. The classic method (used by OPENGL) relies on modeling the posture of objects using a series of affine transformations (rotations, translations, homotheties) of the form:

$$\mathbf{f}_i : \begin{array}{ccc} \mathbb{R}^n & \rightarrow & \mathbb{R}^n \\ \mathbf{x} & \mapsto & \mathbf{A}_i\mathbf{x} + \mathbf{b}_i \end{array}$$

with $n = 2$ or 3. However, the manipulation of compositions of affine functions is less simple than that of linear applications. The idea of the transformation in *homogeneous coordinates* is to transform a system of affine equations into a system of linear equations. Notice first of all that an affine equation

of the type $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{b}$ can be written as:

$$\begin{pmatrix} \mathbf{y} \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{A} & \mathbf{b} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix}.$$

We will thus define the *homogeneous transformation* of a vector as follows:

$$\mathbf{x} \mapsto \mathbf{x}_{\mathrm{h}} = \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix}.$$

Thus, an equation of the type:

$$\mathbf{y} = \mathbf{A}_3 \left( \mathbf{A}_2 \left( \mathbf{A}_1 \mathbf{x} + \mathbf{b}_1 \right) + \mathbf{b}_2 \right) + \mathbf{b}_3$$

involving the composition of three affine transformations, can be written as:

$$\mathbf{y}_{\mathrm{h}} = \begin{pmatrix} \mathbf{A}_3 & \mathbf{b}_3 \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{A}_2 & \mathbf{b}_2 \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{A}_1 & \mathbf{b}_1 \\ \mathbf{0} & 1 \end{pmatrix} \mathbf{x}_{\mathrm{h}}.$$

A *pattern* is a matrix with two or three rows (depending on the object being in the plane or in space) and $n$ columns representing the $n$ vertices of a rigid polygon embodying the object. It is important that the union of all the segments formed by two consecutive points of the pattern forms all the vertices of the polygon that we wish to represent.

1) Let us consider the underwater robot (or AUV for *Autonomous Underwater Vehicle*) whose pattern in homogeneous coordinates is the following:

$$\begin{pmatrix} 0 & 0 & 10 & 0 & 0 & 10 & 0 & 0 \\ -1 & 1 & 0 & -1 & -0.2 & 0 & 0.2 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Draw this pattern in perspective view on a piece of paper.

2) The state equations of the robot are the following:

$$\begin{cases} \dot{p}_x &= v \cos\theta \cos\psi \\ \dot{p}_y &= v \cos\theta \sin\psi \\ \dot{p}_z &= -v \sin\theta \\ \dot{v} &= u_1 \\ \dot{\psi} &= \frac{\sin\varphi}{\cos\theta} \cdot v \cdot u_2 + \frac{\cos\varphi}{\cos\theta} \cdot v \cdot u_3 \\ \dot{\theta} &= \cos\varphi \cdot v \cdot u_2 - \sin\varphi \cdot v \cdot u_3 \\ \dot{\varphi} &= -0 \cdot 1 \ \sin\varphi \cdot \cos\theta + \tan\theta \cdot v \cdot (\sin\varphi \cdot u_2 + \cos\varphi \cdot u_3) \end{cases}$$

where $(\varphi, \theta, \psi)$ are the three Euler angles. The inputs of the system are the tangential acceleration $u_1$, the pitch $u_2$ and the yaw $u_3$. The state vector is therefore equal to $\mathbf{x} = (p_x, p_y, p_z, v, \psi, \theta, \varphi)$. Give a MATLAB function capable of drawing the robot in 3D together with its shadow, in the plane $x$-$y$. Verify that the drawing is correct by moving the six degrees of freedom of the robot one by

one. Use the `plot3` function of MATLAB to obtain a three-dimensional representation such as the one illustrated in Figure 1.13.

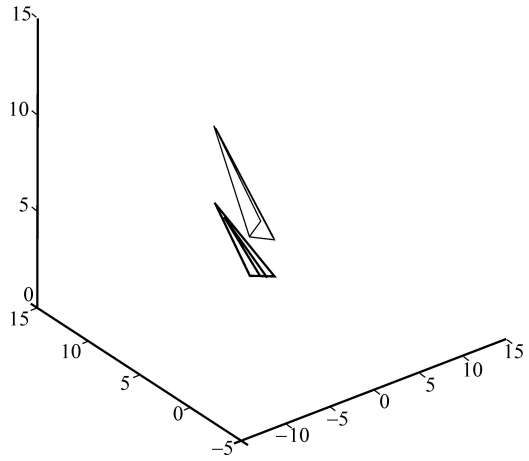


Figure 1.13: 3D representation of the robot together with its shadow in the horizontal plane

3) By using the relation:

$$\omega|_{\mathcal{R}_0} = \begin{pmatrix} 0 & -\sin\psi & \cos\theta\cos\psi \\ 0 & \cos\psi & \cos\theta\sin\psi \\ 1 & 0 & -\sin\theta \end{pmatrix} \begin{pmatrix} \dot{\psi} \\ \dot{\theta} \\ \dot{\varphi} \end{pmatrix}$$

draw the instantaneous rotation vector of the robot.

4) Simulate the robot in MATLAB in various conditions using a Euler method.

---

EXERCISE 1.11.– *Manipulator robot*

A manipulator robot, such as *Staubli* represented on Figure 1.14, is composed of several rigid arms. We retrieve the coordinates of the end effector, at the extremity of the robot, using a series of geometric transformations. We can show that a parametrization with four degrees of freedom allows to represent these transformations. There are several possible parametrizations, each with its own advantages and disadvantages. The most widely used one is probably the *Denavit-Hartenberg* parametrization. In the case where the articulations are rotational joints (as is the case of the Staubli

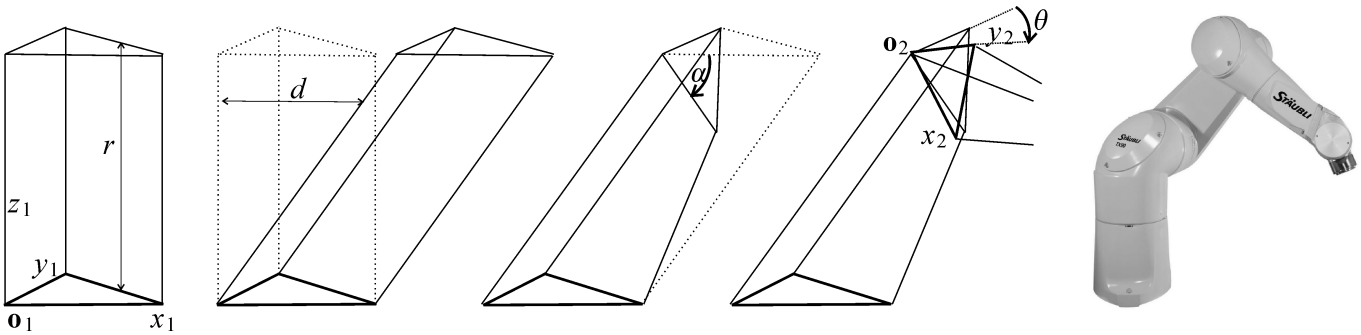


Figure 1.14: Parametrization for the direct geometric model of a manipulator robot

robot where the joins can turn), the parametrization represented by the figure might prove to be practical since it makes drawing the robot easier. This transformation is the composition of four elementary transformations: (i) a translation of length $r$ following $z$ ; (ii) a translation of length $d$ following $x$ ; (iii) a rotation of $\alpha$ around $y$ and (iv) a rotation of $\theta$ (the variable activated around $z$. Using the figure for drawing the arms and the photo for the robot, perform a realistic simulation of the robot's movement in MATLAB.

---

EXERCISE 1.12.– *Three-dimensional modeling of a wheel*

Let us consider the wheel rolling in a plane as shown on Figure 1.15 and for which we seek to obtain the state equations. In this figure **u** is the unit vector indicating the movement direction of the point of contact **p**. The wheel is assumed to roll without friction and therefore the ground reaction force **r** is orthogonal to **n**.

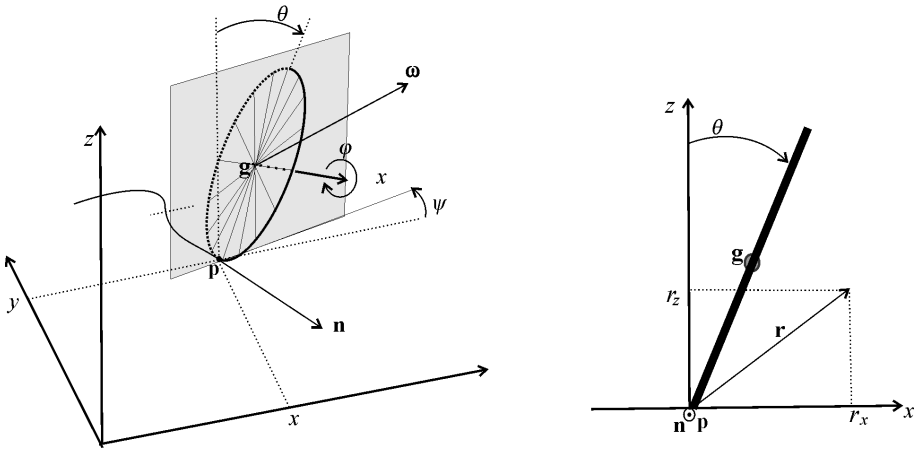


Figure 1.15: Wheel rolling in a plane. On the left, three-dimensional view ; on the right, section in a plane perpendicular to **u** following plane in grey

First of all, we need to define the Euler angles in the context of the wheel, where the concepts of elevation and bank are meaningless. Let us choose for $\psi$ the angle of the horizontal projection of the wheel axis (indicating the horizontal direction to the left of **n**). For $\theta$, we will take the wheel dishing and for $\varphi$, the angle of the wheel made on itself. The reason for this choice is that the angle $\theta$ will be within the interval $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ in accordance with what happens with Euler angles. Thus, the singularities of the equations will correspond to physical singularities. Indeed, the matrix involved in [1.9] is singular if $\cos\theta = 0$ and in such a case, the wheel can no longer roll. We will assume that the wheel is solid, similar to a homogeneous disk of mass $m$ and radius $\rho$. Its inertia matrix is given by:

$$\mathbf{I} = \begin{pmatrix} \frac{m\rho^2}{2} & 0 & 0 \\ 0 & \frac{m\rho^2}{4} & 0 \\ 0 & 0 & \frac{m\rho^2}{4} \end{pmatrix}$$

Give the state equations of this wheel.

2) Let us assume that we are only able to move masses in the plane of the wheel, while conserving the cylindrical symmetry of the wheel. The center of gravity is therefore always in the center of the wheel and we are only able to influence the rotation vector $\boldsymbol{\omega}$ following the wheel axis. Is it possible to control the trajectory of the wheel as well as its speed ?

---

Exercise 1.13.– *Mechanical stability of an underwater robot*

Let us consider a homogeneous body in water with the same density as water. Its center of gravity is denoted by **g**. We place on this body, at a given point denoted by **a**, a point mass (and therefore of infinite density) exerting a force **f**, as illustrated by Figure 1.16, on the left.

1) What is the condition for rotational equilibrium (in other words one which does not lead to a rotation of the body) ?

2) Let us assume that we have rotational equilibrium. Under what condition do we have a stable equilibrium ?

3) Consider an underwater robot that we wish to evolve in a horizontal plane. At equilibrium under water, we notice that the robot is slightly leaning forwards as shown in Figure 1.16, on the right. What needs to be done in order to counter this ?

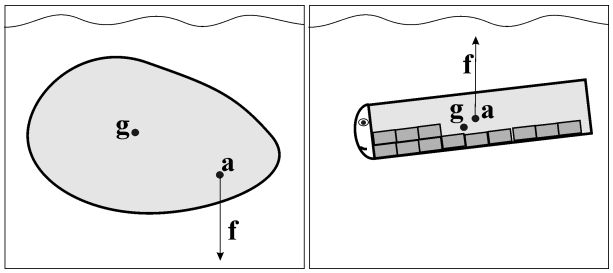


Figure 1.16: Left : submerged homogeneous body ; right : unbalanced robot

4) The robot is now in horizontal equilibrium, which is what we wanted. However, this equilibrium is unstable, in other words the robot tends to turn around as soon as it departs from its position of equilibrium. What needs to be done in order to have stability ?

5) The robot is now correctly in equilibrium with respect to Archimedes' force. We give it an initial horizontal speed and we let go of it. The robot submerges. Why ? What needs to be done to keep it from submerging ?

6) The robot is now in stable equilibrium relative to Archimedes' principle and the drag. We now act on the propeller, but the robot submerges once again. Why ? What needs to be done to avoid this ?

7) The heading of the robot is now controlled using its compass, and it is trying to move closer to a concrete wall parallel to the compass. It then starts to strongly oscillate while remaining in a horizontal plane. Why ? How do we avoid this ?

8) Figure 1.17 represents an robot entirely in equilibrium with various configurations for the position of its propellers. Is configuration (a) more stable or more maneuverable than configuration (b) ? Is configuration (c) more stable or more maneuverable than configuration (d) ?

---

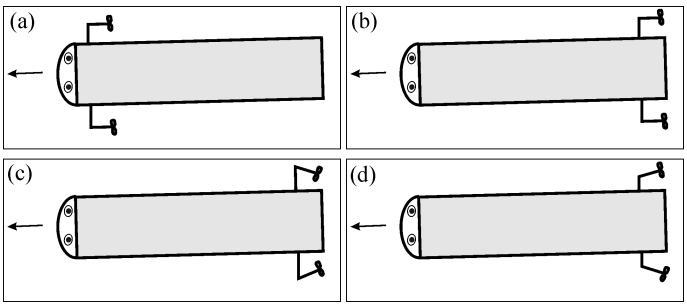


Figure 1.17: An underwater robot viewed from above with various configurations for its propellers

# Chapter 2

# Feedback linearization

Due to their multiple rotation capabilities, robots are considered to be strongly nonlinear systems. In this chapter, we will look at designing nonlinear controllers in order to constrain the state vector of the robot to follow a fixed forward path or to remain within a determined area of its workspace. In contrast to the linear approach, which offers a general methodology but is limited to the neighborhood of a point of the state space [KAI 80] [JAU 13], nonlinear approaches only apply to limited classes of systems, but they allow to extend the effective operating range of the system. Indeed, there is no general method of globally stabilizing nonlinear systems [KHA 02]. However, there is a multitude of methods that apply to particular cases [SLO 91] [FAN 01]. The aim of this chapter is to present one of the more representative theoretical methods (whereas in the following chapter, we will be looking at more pragmatic approaches). This method is called *feedback linearization* and it requires knowledge of an accurate and reliable state machine for our robot. The robots considered here are mechanical systems whose modeling can be found in [JAU 05]. We will assume in this chapter that the state vector is entirely known. In practice, it has to be approximated from sensor measurements. We will see in Chapter 7 how such an approximation is performed.

## 2.1 Controlling an integrator chain

As we will show further on in this chapter, feedback linearization leads to the problem of controlling a system which is composed of several integrator chains decoupled from one another. In this paragraph we will therefore consider an integrator chain whose input $u$ and output $y$ are linked together by the differential equation:

$$y^{(n)} = u.$$

### 2.1.1 Proportional-derivative controller

Let us first of all stabilize this system using a *proportional-derivative* controller of the type:

$$u = \alpha_0 (w - y) + \alpha_1 (\dot{w} - \dot{y}) + \cdots + \alpha_{n-1} \left( w^{(n-1)} - y^{(n-1)} \right) + w^{(n)}$$

where $w$ is the wanted setpoint for $y$. Let us note that $w$ may depend on time. The fact that this controller requires the differentials of $y$ is not a problem within the frame defined by the feedback linearization. Indeed, all of these derivatives can be described as analytic functions of the state $\mathbf{x}$ of the system and the input $\mathbf{u}$. Concerning the setpoint $w(t)$, it is chosen by the user and an analytic expression of $w(t)$ may be assumed to be known (for instance $w(t) = \sin(t)$). Thus, calculating the differentials of $w$ is done in a formal manner and no sensitivity of the differential operator with respect to the noise has to be feared.

The feedback system is described by the differential equation:

$$y^{(n)} = u = \alpha_0 (w - y) + \alpha_1 (\dot{w} - \dot{y}) + \cdots + \alpha_{n-1} \left( w^{(n-1)} - y^{(n-1)} \right) + w^{(n)}.$$

If we define the error $e$ between the setpoint $w$ and the output $y$ as $e = w - y$, this equation becomes:

$$e^{(n)} + \alpha_{n-1} e^{(n-1)} + \cdots + \alpha_1 \dot{e} + \alpha_0 e = 0.$$

This differential equation is called the *error dynamics equation*. Its characteristic polynomial, given by:

$$P(s) = s^n + \alpha_{n-1} s^{n-1} + \cdots + \alpha_1 s + \alpha_0, \tag{2.1}$$

can thus be chosen arbitrarily among the polynomials of degree $n$. Of course, we will choose all roots with a negative real part, in order to ensure the stability of the system. For instance, if $n = 3$ and if we want all the poles to be equal to $-1$, we will take:

$$s^3 + \alpha_2 s^2 + \alpha_1 s + \alpha_0 = (s + 1)^3 = s^3 + 3s^2 + 3s + 1.$$

Whence:

$$\alpha_2 = 3, \alpha_1 = 3, \alpha_0 = 1.$$

The controller obtained is then given by:

$$u = (w - y) + 3 (\dot{w} - \dot{y}) + 3 (\ddot{w} - \ddot{y}) + \dddot{w}.$$

REMARK 2.1.– *In this book, we will choose, for reasons of simplicity, to position all our poles at -1. The previous reasoning, applied for various degrees n, leads us to the following controls:*

$$\begin{array}{ll} n = 1 & u = (w - y) + \dot{w} \\ n = 2 & u = (w - y) + 2 (\dot{w} - \dot{y}) + \ddot{w} \\ n = 3 & u = (w - y) + 3 (\dot{w} - \dot{y}) + 3 (\ddot{w} - \ddot{y}) + \dddot{w} \\ n = 4 & u = (w - y) + 4 (\dot{w} - \dot{y}) + 6 (\ddot{w} - \ddot{y}) + 4 (\dddot{w} - \dddot{y}) + \ddddot{w}. \end{array} \tag{2.2}$$

*Notice that the coefficients correspond to those of Pascal's triangle:*

$$\begin{array}{ccccc} 1 & & & & \\ 1 & 1 & & & \\ 1 & 2 & 1 & & \\ 1 & 3 & 3 & 1 & \\ 1 & 4 & 6 & 4 & 1 \end{array}$$

### 2.1.2 Proportional-integral-derivative controller

In order to compensate for the constant disturbances, we may decide to add an integral term. We then obtain a PID (proportional-integral-derivative) controller, which is of the form:

$$\begin{aligned} u = \alpha_{-1} \int_{\tau=0}^{t} (w(\tau) - y(\tau))\, d\tau \\ + \alpha_0 (w - y) + \alpha_1 (\dot{w} - \dot{y}) + \cdots + \alpha_{n-1} \left( w^{(n-1)} - y^{(n-1)} \right) + w^{(n)}. \end{aligned} \tag{2.3}$$

The feedback system is described by the differential equation:

$$\begin{aligned} y^{(n)} = \alpha_{-1} \int_{\tau=0}^{t} (w(\tau) - y(\tau))\, d\tau \\ + \alpha_0 (w - y) + \alpha_1 (\dot{w} - \dot{y}) + \cdots + \alpha_{n-1} \left( w^{(n-1)} - y^{(n-1)} \right) + w^{(n)}. \end{aligned}$$

Hence, by differentiating once:

$$e^{(n+1)} + \alpha_{n-1} e^{(n)} + \cdots + \alpha_1 \ddot{e} + \alpha_0 \dot{e} + \alpha_{-1} e = 0.$$

The characteristic polynomial:

$$P(s) = s^{n+1} + \alpha_{n-1} s^n + \cdots + \alpha_1 s^2 + \alpha_0 s + \alpha_{-1}$$

can be chosen arbitrarily, as with the proportional-derivative controller.

## 2.2 Introductory example

Before giving the principles of feedback linearization, we will consider an introductory example. Let us take the pendulum of Figure 2.1. The input of this system is the torque $u$ exerted on the pendulum.

Its state representation is assumed to be:

$$\left\{ \begin{array}{ccc} \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} & = & \begin{pmatrix} x_2 \\ -\sin x_1 + u \end{pmatrix} \\ y & = & x_1 \end{array} \right.$$

This is of course a normalized model in which the coefficients (mass, gravity, length) have all been

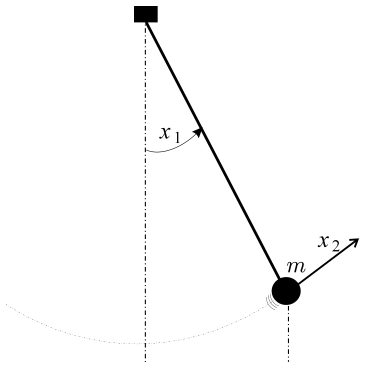


Figure 2.1: Simple pendulum with state vector $\mathbf{x} = (x_1, x_2)$

set to 1. We would like the position $x_1(t)$ of the pendulum to be equal to some setpoint $w(t)$ which may vary over time. By using a feedback linearization method, explained later, we would like to have a state feedback controller such that the error $e = w - x_1$ converges towards 0 at $\exp(-t)$ (which means that we place the poles at $-1$). Let us differentiate $y$ until the input $u$ appears. We have:

$$\begin{aligned} \dot{y} &= x_2 \\ \ddot{y} &= -\sin x_1 + u. \end{aligned}$$

Let us choose:

$$u = \sin x_1 + v, \tag{2.4}$$

where $v$ corresponds to the new, so-called intermediate input. We obtain

$$\ddot{y} = v. \tag{2.5}$$

Such a feedback is called *linearizing feedback* because it transforms the nonlinear system into a linear system. The system obtained in this way can be stabilized by standard linear techniques. Let us take as an example a proportional-derivative controller:

$$\begin{aligned} v &= (w - y) + 2(\dot{w} - \dot{y}) + \ddot{w} \\ &= (w - x_1) + 2(\dot{w} - x_2) + \ddot{w}. \end{aligned}$$

By injecting this expression of $v$ into [2.5], we obtain:

$$\ddot{y} = (w - x_1) + 2(\dot{w} - x_2) + \ddot{w}.$$

Which yields:

$$e + 2\dot{e} + \ddot{e} = 0$$

where $e = w - x_1$ is the error between the position of the pendulum and its setpoint. The complete

controller is expressed by:

$$u \overset{[2.4]}{=} \sin x_1 + (w - x_1) + 2(\dot{w} - x_2) + \ddot{w}.$$

If we now want the angle $x_1$ of the pendulum to be equal to $\sin t$ once the transient regime has passed, we simply need to take $w(t) = \sin t$. Thus, $\dot{w}(t) = \cos t$ and $\ddot{w} = -\sin t$. Consequently, the controller is given by:

$$u = \sin x_1 + (\sin t - x_1) + 2(\cos t - x_2) - \sin t.$$

In this very simple example, we can see that the proposed controller is nonlinear and depends on time. No approximation arising from linearization has been performed. Of course, a linearization has been done by a first feedback in order to make the system linear, but this linearization did not introduce any approximation.

## 2.3 Principle of the method

### 2.3.1 Principle

Here we will look at generalizing the method described in the previous section. Let us consider the nonlinear system described by:

$$\begin{cases} \dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\mathbf{u} \\ \mathbf{y} &= \mathbf{h}(\mathbf{x}) \end{cases} \tag{2.6}$$

where the number of inputs and the number of outputs are both equal to $m$. The idea of the method of feedback linearization is to transform the system using a controller of the type $\mathbf{u} = \mathbf{r}(\mathbf{x}, \mathbf{v})$, where $\mathbf{v}$ is the new input, also of dimension $m$. This operation requires that the state is completely accessible. If this is not the case, we need build an observer in a nonlinear context, which is a very difficult operation. Since the state is assumed to be accessible, the vector $\mathbf{y}$ must not really be considered an output, but rather as the vector of the setpoint variables.

In order to perform this transformation, we need to express the successive derivatives of each of the $y_i$ in function of the state and of the input. We stop differentiating $y_i$ as soon as the inputs begin to be involved in the expression of the derivative. We thus have an equation of the type:

$$\begin{pmatrix} y_1^{(k_1)} \\ \vdots \\ y_m^{(k_m)} \end{pmatrix} = \mathbf{A}(\mathbf{x})\,\mathbf{u} + \mathbf{b}(\mathbf{x}) \tag{2.7}$$

where $k_i$ denotes the number of times we need to differentiate $y_i$ in order to make an input appear (refer to the examples given in the following paragraphs for a better understanding). Under the hypothesis that the matrix $\mathbf{A}(\mathbf{x})$ is invertible, the transformation:

$$\mathbf{u} = \mathbf{A}^{-1}(\mathbf{x})\,(\mathbf{v} - \mathbf{b}(\mathbf{x})) \tag{2.8}$$

where $\mathbf{v}$ is our new input (see Figure 2.2), forms a linear system $\mathcal{S}_{\mathrm{L}}$ of $m$ inputs to $m$ outputs described by the differential equations:

$$\mathcal{S}_{\mathrm{L}} : \begin{cases} y_1^{(k_1)} &= v_1 \\ \vdots &= \vdots \\ y_m^{(k_m)} &= v_m \end{cases}$$

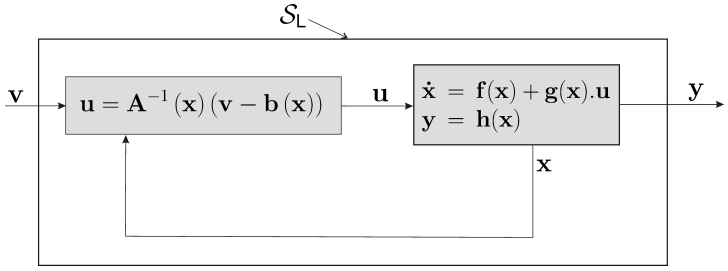


Figure 2.2: The nonlinear system, once transformed, becomes linear and decoupled ; and therefore becomes easy to control

This system is linear and completely decoupled (in other words each input $v_i$ acts on one and only one output $y_i$). It is therefore very simple to control using standard linear techniques. Here, the system to control is composed of decoupled integrator chains, we will use $m$ PID (proportional-integral-derivative) controllers whose principles we have already recalled in Section 2.1. Let us note that in order to use such controller, it is necessary to have the derivatives of the outputs. Since we assumed to have access to all the state variables $x_i$ of the system, a formal expression of these derivatives in function of $x_i$ is easily obtained by using the state equations.

Remark 2.2.– *Robots are called* redundant *if they have more inputs than necessary, in other words if* $\dim \mathbf{u} > \dim \mathbf{y}$. *In this case, the matrix* $\mathbf{A}(\mathbf{x})$ *is rectangular. In order to apply the transformation in [2.8], we may use a* Moore-Penrose *pseudoinverse. If* $\mathbf{A}$ *is of full rank (in other words equal to* $\dim \mathbf{y}$*), this pseudoinverse is given by:*

$$\mathbf{A}^{\dagger} = \mathbf{A}^{T} \cdot \left(\mathbf{A} \cdot \mathbf{A}^{T}\right)^{-1}.$$

*We will therefore have:*

$$\dim \mathbf{v} = \dim \mathbf{y} < \dim \mathbf{u}$$

*and we are in a situation identical to that if the square robot (i.e. non-redundant).*

### 2.3.2 Relative degree

By properly analyzing the path used to obtain Equation [2.7], we realize that the $k^{\text{th}}$ derivative of the $i^{\text{th}}$ output $y_i^{(k)}$ is expressed in the form:

$$
\begin{aligned}
y_i^{(k)} &= \hat{b}_{ik}(\mathbf{x}) \text{ if } j < k_i \\
y_i^{(k)} &= \hat{\mathbf{a}}_{ik}^{\mathrm{T}}(\mathbf{x}) \cdot \mathbf{u} + \hat{b}_{ik}(\mathbf{x}) \text{ if } k = k_i \\
y_i^{(k)} &= \hat{\mathbf{a}}_{ik}(\mathbf{x}, \mathbf{u}, \dot{\mathbf{u}}, \ddot{\mathbf{u}}, \dots) \text{ if } k > k_i
\end{aligned}
$$

The coefficient $k_i$ is called the *relative degree* of the $i^{\text{th}}$ output. By measuring the state of the system $\mathbf{x}$ and its input $\mathbf{u}$, we can thus have all the successive derivatives of the outputs $y_i^{(k)}$, as long as $k$ remains smaller than or equal to $k_i$. Indeed, given the high-frequency noise that appears in the signals, we cannot reliably obtain the derivative of the signals by using derivators. We therefore have an analytic function:

$$
\Delta : \begin{array}{ccl} \mathbb{R}^m & \rightarrow & \mathbb{R}^{(k_1+1)\cdot\cdots\cdot(k_m+1)} \\ (\mathbf{x}, \mathbf{u}) & \mapsto & \Delta(\mathbf{x}, \mathbf{u}) = (y_1, \dot{y}_1, \dots, y_1^{(k_1)}, y_2, \dot{y}_2, \dots, y_m^{(k_m)}) \end{array}
$$

that allows us to have all the derivatives of the outputs (until their relative degree), and this without using digital derivators.

Example 2.1.– *Let us consider the system described by:*

$$
\left\{ \begin{array}{rl} \dot{x} = & xu + x^3 \\ y = & 2x. \end{array} \right.
$$

*We have:*

$$
\begin{array}{rcl} y & = & 2x \\ \dot{y} & = & 2\dot{x} = 2xu + 2x^3 \\ \ddot{y} & = & 2\dot{x}u + 2x\dot{u} + 6\dot{x}x^2 = 2(xu + x^3)u + 2x\dot{u} + 6(xu + x^3)x^2. \end{array}
$$

*We therefore have a relative degree $k = 1$ for the output $y$. We can therefore have $\dot{y}$ without using digital derivators. This is not the case for $\ddot{y}$ because having $u$ with a high level of precision does not mean that we have $\dot{u}$. Here, we have $\Delta(x, u) = (2x, 2xu + 2x^3)$.*

### 2.3.3 Differential delay matrix

We call *differential delay* $r_{ij}$ separating the input $u_j$ from the output $y_i$ the number of times we need to differentiate $y_i$ in order to make $u_j$ appear. The matrix $\mathbf{R}$ of the $r_{ij}$ is called the *differential delay matrix*. When plainly reading the state equations, this matrix can be obtained without calculation, simply by counting the number of integrators each input $u_j$ must be subjected to in order to algebraically affect the output $y_i$ (some examples are discussed in more detail in the exercises). The relative degree for each output can be obtained by taking the minimum of each row.

Let us take for example:

$$\mathbf{R} = \begin{pmatrix} \mathbf{1} & 2 & 2 \\ \mathbf{3} & 4 & \mathbf{3} \\ 4 & \infty & \mathbf{2} \end{pmatrix}.$$

The associated system is composed of three inputs, three outputs and the relative degrees (components in bold in the preceding formula) are $k_1 = 1, k_2 = 3, k_3 = 2$. If there is a $j$ such that $\forall i, r_{ij} > k_i$ (or equivalently if a column has no element in bold), the matrix $\mathbf{R}$ is called *unbalanced*. In our example, it is unbalanced since there is a $j$ (here $j = 2$) such that $\forall i, r_{ij} > k_i$. If the matrix is unbalanced, then for all $i$, $y_i^{(k_i)}$ does not depend on $u_j$. In this case, the $j^{\text{th}}$ column of $\mathbf{A}(\mathbf{x})$ will be zero and $\mathbf{A}(\mathbf{x})$ will always be singular. Thus, the transformation [2.8] will have no meaning. One method to overcome this is to delay some of the inputs $u_j$ by adding one or more integrators in front of the system. Adding an integrator in front of the $j^{\text{th}}$ input amounts to adding 1 to the $j^{\text{th}}$ column of $\mathbf{R}$. In our example, if we add an integrator in front of $u_1$, we obtain:

$$\mathbf{R} = \begin{pmatrix} \mathbf{2} & \mathbf{2} & \mathbf{2} \\ 4 & 4 & \mathbf{3} \\ 5 & \infty & \mathbf{2} \end{pmatrix}.$$

The relative degrees become $k_1 = 2, k_2 = 3, k_3 = 2$ and the matrix $\mathbf{R}$ becomes balanced.

### 2.3.4 Singularities

The matrix $\mathbf{A}(\mathbf{x})$ involved in feedback [2.8] may not be invertible. The values for $\mathbf{x}$ such that $\det(\mathbf{A}(\mathbf{x})) = 0$ are called *singularities*. Although they generally form a set of zero measures in the state space, studying singularities is fundamental since they are sometimes impossible to avoid. We call *set of acceptable outputs* of the system [2.6] the quantity:

$$\mathbb{S}_y = \{\mathbf{y} \in \mathbb{R}^m \mid \exists \mathbf{x} \in \mathbb{R}^n, \exists \mathbf{u} \in \mathbb{R}^m, \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x}) \cdot \mathbf{u} = \mathbf{0}, \mathbf{y} = \mathbf{h}(\mathbf{x})\}.$$

The set $\mathbb{S}_y$ is therefore composed, by the projection on $\mathbb{R}^m$, of a differentiable manifold (or surface) of dimension $m$ (since we have $m + n$ equations for $2m + n$ variables). Thus, except for the degenerate case, $\mathbb{S}_y$ is a subset of $\mathbb{R}^m$ with a non-empty interior and an exterior.

In order to properly understand this, consider the example of a cart on rails as represented on Figure 2.3. This cart can be propelled by a horizontal ventilator whose angle of thrust can be controlled. But be careful, the rotation speed of the ventilator is fixed.

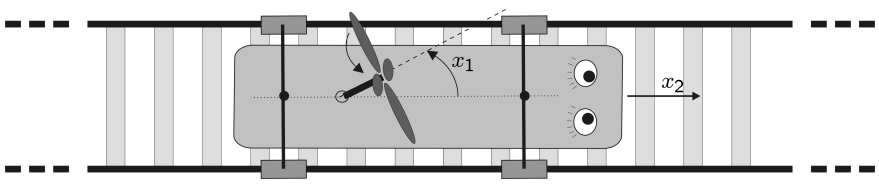


Figure 2.3: Robot cart propelled by a ventilator

The state equations that model this system are given by:

$$\begin{cases} \dot{x}_1 &= u \\ \dot{x}_2 &= \cos x_1 - x_2 \\ y &= x_2 \end{cases}$$

where $x_1$ is the ventilator's angle of thrust, $x_2$ is the speed of the cart. Let us note that we have taken into account a viscous friction force. The set of acceptable outputs is:

$$\begin{aligned} \mathbb{S}_y &= \{y \mid \exists x_1, \exists x_2, \exists u, u = 0, \cos x_1 - x_2 = 0, y = x_2\} \\ &= \{y \mid \exists x_1, \cos x_1 = y\} = [-1, 1]. \end{aligned}$$

This means that we will not be able to stabilize the cart at a speed that is strictly superior to 1, in absolute value. Let us apply a feedback linearizing method. We have:

$$\begin{aligned} \dot{y} &= \cos x_1 - x_2 \\ \ddot{y} &= -\left(\sin x_1\right) u - \cos x_1 + x_2 \end{aligned}$$

and therefore, the linearizing controller is given by:

$$u = \frac{-1}{\sin x_1}\left(v + \cos x_1 - x_2\right).$$

The feedback system therefore has the following equation:

$$\ddot{y} = v.$$

It may appear that any value for $y$ can be reached, since $v$ can be chosen arbitrarily. This would be correct, if we would not have the singularity that appears when $\sin x_1 = 0$. Let us take, for example, $v\left(t\right) = 1$ and $\mathbf{x}(0) = \left(\frac{\pi}{3}, 0\right)$. We should have:

$$\begin{aligned} \ddot{y}(t) &= v\left(t\right) = 1 \\ \dot{y}(t) &= \dot{y}\left(0\right) + \int_0^t \ddot{y}(\tau)d\tau = \cos x_1(0) - x_2(0) + t = \frac{1}{2} + t \\ y(t) &= y\left(0\right) + \int_0^t \dot{y}(\tau)d\tau = x_2\left(0\right) + t^2 + \frac{1}{2}t = t^2 + \frac{1}{2}t \end{aligned}$$

which is physically impossible. This is what happens when we apply such a controller: the input $u$ directs the angle of the ventilator towards the correct direction, and the equation $\ddot{y} = v$ is then satisfied, at least in the very beginning. Then $x_1$ is canceled out and the singularity is reached. The equation $\ddot{y} = v$ can no longer be satisfied. For some systems, it can happen that such a singularity can be crossed. This is not the case here.

## 2.4 Cart

### 2.4.1 First model

Consider a cart described by the following state equations:

$$\begin{cases} \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= u_1 \\ \dot{v} &= u_2 \end{cases}$$

where $v$ is the speed of the cart, $\theta$ its orientation and $(x, y)$ the coordinates of its center (see Figure 2.4).

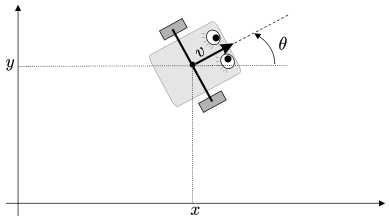


Figure 2.4: Cart (also called Dubin's car)

The state vector is given by $\mathbf{x} = (x, y, \theta, v)$. We would like to calculate a controller that would allow us to describe a cycloid with the equation:

$$\begin{cases} x_d(t) &= R\sin(f_1 t) + R\sin(f_2 t) \\ y_d(t) &= R\cos(f_1 t) + R\cos(f_2 t) \end{cases}$$

with $R = 15$, $f_1 = 0.02$ and $f_2 = 0.12$. For this, we use a feedback linearizing method. We have:

$$\begin{pmatrix} \ddot{x} \\ \ddot{y} \end{pmatrix} = \begin{pmatrix} u_2\cos\theta - u_1 v\sin\theta \\ u_2\sin\theta + u_1 v\cos\theta \end{pmatrix} = \begin{pmatrix} -v\sin\theta & \cos\theta \\ v\cos\theta & \sin\theta \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}.$$

If we take as input:

$$\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} -v\sin\theta & \cos\theta \\ v\cos\theta & \sin\theta \end{pmatrix}^{-1} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$$

where $(v_1, v_2)$ is the new input vector, we obtain the linear system:

$$\begin{pmatrix} \ddot{x} \\ \ddot{y} \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}.$$

Let us transform this system so that all our poles are at $-1$. Following [2.2], we obtain:

$$\begin{cases} v_1 = & (x_d - x) + 2\,(\dot{x}_d - \dot{x}) + \ddot{x}_d &= (x_d - x) + 2\,(\dot{x}_d - v\cos\theta) + \ddot{x}_d \\ v_2 = & (y_d - y) + 2\,(\dot{y}_d - \dot{y}) + \ddot{y}_d &= (y_d - y) + 2\,(\dot{y}_d - v\sin\theta) + \ddot{y}_d \end{cases}$$

The transformed system then obeys the following differential equations:

$$\begin{pmatrix} (x_d - x) + 2(\dot{x}_d - \dot{x}) + (\ddot{x}_d - \ddot{x}) \\ (y_d - y) + 2(\dot{y}_d - \dot{y}) + (\ddot{y}_d - \ddot{y}) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

If we define the error vector $\mathbf{e} = (e_x, e_y) = (x_d - x, y_d - y)$, the error dynamics are written as:

$$\begin{pmatrix} e_x + 2\dot{e}_x + \ddot{e}_x \\ e_y + 2\dot{e}_y + \ddot{e}_y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix},$$

which is stable and quickly converges towards 0. The controller will therefore be:

$$\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} -v\sin\theta & \cos\theta \\ v\cos\theta & \sin\theta \end{pmatrix}^{-1} \begin{pmatrix} (x_d - x) + 2(\dot{x}_d - v\cos\theta) + \ddot{x}_d \\ (y_d - y) + 2(\dot{y}_d - v\sin\theta) + \ddot{y}_d \end{pmatrix} \tag{2.9}$$

with:

$$\begin{aligned} \dot{x}_d(t) &= Rf_1\cos(f_1 t) + Rf_2\cos(f_2 t) \\ \dot{y}_d(t) &= -Rf_1\sin(f_1 t) - Rf_2\sin(f_2 t) \\ \ddot{x}_d(t) &= -Rf_1^2\sin(f_1 t) - Rf_2^2\sin(f_2 t) \\ \ddot{y}_d(t) &= -Rf_1^2\cos(f_1 t) - Rf_2^2\cos(f_2 t). \end{aligned}$$

The following MATLAB program, found in the file `cycloide.m`, simulates the behavior of the controlled system:

```
x=[10;10;0;2]; %x=[x,y,theta,v)
dt=0.1; R=15; f1=0.02;f2=0.12;
for t=0:dt:10,
   w = R*[sin(f1*t)+sin(f2*t);cos(f1*t)+cos(f2*t)];
   dw = R*[f1*cos(f1*t)+f2*cos(f2*t);-f1*sin(f1*t)-f2*sin(f2*t)];
   ddw=R*[-f1*f1*sin(f1*t)-f2*f2*sin(f2*t);-f1*f1*cos(f1*t)-f2*f2*cos(f2*t)];
   u=control(x,w,dw,ddw);
   x=x+f(x,u)*dt;
end;
```

For the evolution of the robot, the program uses the following evolution function:

```
function xdot=f(x,u)
   theta=x(3);v=x(4);
   xdot=[v*cos(theta); v*sin(theta); u(1); u(2)];
end
```

As for the control, it is performed by the function:

```
function u=control(x,w,dw,ddw)
   v=0.25 *( w - [x(1);x(2)])+1*(dw - [x(4)*cos(x(3));x(4)*sin(x(3))])+ddw;
   A=[-x(4)*sin(x(3)), cos(x(3)); x(4)*cos(x(3)), sin(x(3))];
```

```
   u=inv(A)*v;
end
```

### 2.4.2 Second model

Let us now assume that the cart is described by the state equations:

$$\left\{ \begin{array}{rcl} \dot{x} & = & u_1 \cos\theta \\ \dot{y} & = & u_1 \sin\theta \\ \dot{\theta} & = & u_2. \end{array} \right.$$

Let us choose as output the vector $\mathbf{y} = (x, y)$. The method of feedback linearization leads to a matrix $\mathbf{A}(\mathbf{x})$ which is still singular. As it was explained in Section 2.3.3, this can be predicted without any calculation simply by observing the differential delay matrix:

$$\mathbf{R} = \begin{pmatrix} \mathbf{1} & 2 \\ \mathbf{1} & 2 \end{pmatrix}.$$

This matrix contains a column whose elements never correspond to the minimum of the related row (in other words a column without elements in bold). We will illustrate how to get out of such a situation by adding integrators in front of certain inputs. Let us for instance add an integrator, whose state variable will be denoted by $z$, in front of the first input. Recall that adding an integrator in front of the $j^{\text{th}}$ input of the system means delaying this input and therefore adding 1 to all the elements of column $j$ of $\mathbf{R}$. The matrix $\mathbf{R}$ is then in equilibrium. We obtain a new system described by:

$$\left\{ \begin{array}{rcl} \dot{x} & = & z \cos\theta \\ \dot{y} & = & z \sin\theta \\ \dot{\theta} & = & u_2 \\ \dot{z} & = & c_1. \end{array} \right.$$

We have:

$$\left\{ \begin{array}{lll} \ddot{x} & = \dot{z}\cos\theta - z\dot{\theta}\sin\theta & = c_1\cos\theta - zu_2\sin\theta \\ \ddot{y} & = \dot{z}\sin\theta + z\dot{\theta}\cos\theta & = c_1\sin\theta + zu_2\cos\theta \end{array} \right.$$

in other words:

$$\begin{pmatrix} \ddot{x} \\ \ddot{y} \end{pmatrix} = \begin{pmatrix} \cos\theta & -z\sin\theta \\ \sin\theta & z\cos\theta \end{pmatrix} \begin{pmatrix} c_1 \\ u_2 \end{pmatrix}.$$

The matrix is not singular, except in the unlikely case where the variable $z$ is zero (here, $z$ can be understood as the speed of the vehicle). The method of feedback linearization can therefore work.

Let us take:

$$\begin{pmatrix} c_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} \cos\theta & -z\sin\theta \\ \sin\theta & z\cos\theta \end{pmatrix}^{-1} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\frac{\sin\theta}{z} & \frac{\cos\theta}{z} \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$$

in order to have a feedback system of the form:

$$\begin{pmatrix} \ddot{x} \\ \ddot{y} \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}.$$

Figure 2.5 illustrates the feedback linearization that we have just performed.

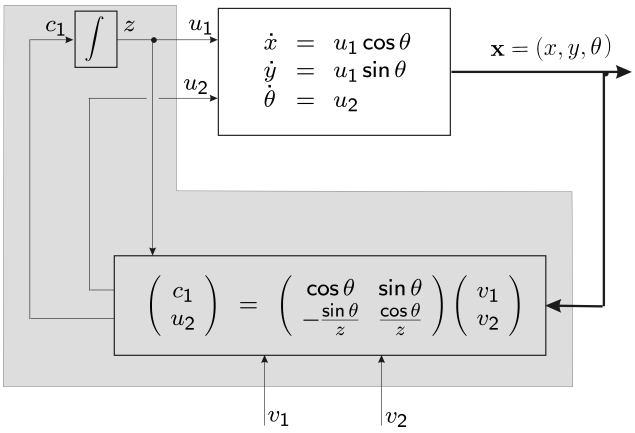


Figure 2.5: Dynamic feedback linearization

In order to have all the poles at $-1$, we need to take (see Equation 2.2 page 38):

$$\begin{pmatrix} c_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\frac{\sin\theta}{z} & \frac{\cos\theta}{z} \end{pmatrix} \begin{pmatrix} (x_d - x) + 2(\dot{x}_d - z\cos\theta) + \ddot{x}_d \\ (y_d - y) + 2(\dot{y}_d - z\sin\theta) + \ddot{y}_d \end{pmatrix}$$

The state equations of the controller are therefore:

$$\begin{cases} \dot{z} &= (\cos\theta)(x_d - x + 2(\dot{x}_d - z\cos\theta) + \ddot{x}_d) + (\sin\theta)(y_d - y + 2(\dot{y}_d - z\sin\theta) + \ddot{y}_d) \\ u_1 &= z \\ u_2 &= -\frac{\sin\theta}{z}(x_d - x + 2(\dot{x}_d - z\cos\theta) + \ddot{x}_d) + \frac{\cos\theta}{z} \cdot (y_d - y + 2(\dot{y}_d - z\sin\theta) + \ddot{y}_d) \end{cases}$$

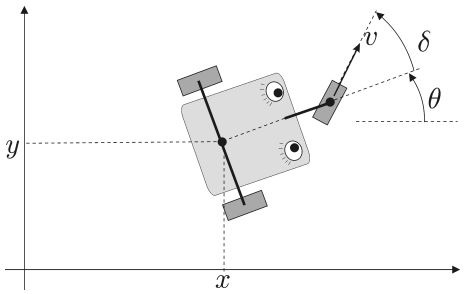


Figure 2.6: Tricycle robot to be controlled

## 2.5 Controlling a tricycle

### 2.5.1 Speed and heading control

Let us consider the tricycle represented in Figure 2.6. Its evolution equation is given by:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \\ \dot{\delta} \end{pmatrix} = \begin{pmatrix} v\cos\delta\cos\theta \\ v\cos\delta\sin\theta \\ v\sin\delta \\ u_1 \\ u_2 \end{pmatrix}$$

We have assumed here that the distance between the center of the rear axle and the axis of the front wheel was equal to 1 m. Let us choose as output the vector $\mathbf{y} = (v, \theta)$. The first-order derivatives of the outputs $y_1$ and $y_2$ are expressed by:

$$\dot{y}_1 = \dot{v} = u_1,$$
$$\dot{y}_2 = \dot{\theta} = v\sin\delta.$$

Since the derivative $\dot{y}_2$ of $y_2$ does not involve the input, we may differentiate it once more:

$$\ddot{y}_2 = \dot{v}\sin\delta + v\dot{\delta}\cos\delta = u_1\sin\delta + u_2 v\cos\delta.$$

The expressions for $\dot{y}_1$ and $\ddot{y}_2$ can be rewritten in matrix form:

$$\begin{pmatrix} \dot{y}_1 \\ \ddot{y}_2 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 \\ \sin\delta & v\cos\delta \end{pmatrix}}_{\mathbf{A}(\mathbf{x})} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}.$$

By setting the feedback $\mathbf{u} = \mathbf{A}^{-1}(\mathbf{x})\ \mathbf{v}$, where $\mathbf{v}$ is the new input, our feedback system is rewritten as:

$$\mathcal{S}_{\mathrm{L}} : \begin{pmatrix} \dot{y}_1 \\ \ddot{y}_2 \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$$

and therefore becomes linear and decoupled. We now have two decoupled monovariate systems. The first, of order 1, can be stabilized by a proportional controller. As for the second, second-order system, a proportional-derivative controller is best adapted. If $\mathbf{w} = (w_1,\ w_2)$ represents the setpoint for $\mathbf{y}$, this controller is expressed by:

$$\left\{ \begin{array}{rcl} v_1 & = & (w_1 - y_1) + \dot{w}_1 \\ v_2 & = & (w_2 - y_2) + 2\,(\dot{w}_2 - \dot{y}_2) + \ddot{w}_2 \end{array} \right.$$

if we want all our poles to be equal to $-1$ (refer to Equation [2.2]). Therefore the equations of a state feedback controller for our nonlinear system are given by:

$$\mathbf{u} = \begin{pmatrix} 1 & 0 \\ \sin\delta & v\cos\delta \end{pmatrix}^{-1} \begin{pmatrix} (w_1 - v) + \dot{w}_1 \\ w_2 - \theta + 2\left(\dot{w}_2 - \frac{v\sin\delta}{L}\right) + \ddot{w}_2 \end{pmatrix} \tag{2.10}$$

Let us note that this controller does not have a state variable. It is therefore a *static* controller.

Remark 2.3.– *Since:*

$$\det\left(\mathbf{A}\left(\mathbf{x}\right)\right) = \frac{v\cos\delta}{L}$$

*can be zero, there are singularities for which the control* $\mathbf{u}$ *is not defined. Appropriate processing has to be provided when such singularities are encountered by the system.*

### 2.5.2 Position control

Let us now try to make our tricycle follow a desired trajectory $(x_d, y_d)$. For this, let us choose as output the vector $\mathbf{y} = (x, y)$. We have:

$$\left\{ \begin{array}{rcl} \dot{x} & = & v\cos\delta\cos\theta \\ \ddot{x} & = & \dot{v}\cos\delta\cos\theta - v\dot{\delta}\sin\delta\cos\theta - v\dot{\theta}\cos\delta\sin\theta \\ & = & u_1\cos\delta\cos\theta - vu_2\sin\delta\cos\theta - v^2\sin\delta\cos\delta\sin\theta \\ \dot{y} & = & v\cos\delta\sin\theta \\ \ddot{y} & = & \dot{v}\cos\delta\sin\theta - v\dot{\delta}\sin\delta\sin\theta + v\dot{\theta}\cos\delta\cos\theta \\ & & u_1\cos\delta\sin\theta - vu_2\sin\delta\sin\theta + v^2\sin\delta\cos\delta\cos\theta \end{array} \right.$$

Thus:

$$\begin{pmatrix} \ddot{x} \\ \ddot{y} \end{pmatrix} = \underbrace{\begin{pmatrix} \cos\delta\cos\theta & -v\sin\delta\cos\theta \\ \cos\delta\sin\theta & -v\sin\delta\sin\theta \end{pmatrix}}_{\mathbf{A}(\mathbf{x})} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} + \underbrace{\begin{pmatrix} -v^2\sin\delta\cos\delta\sin\theta \\ v^2\sin\delta\cos\delta\cos\theta \end{pmatrix}}_{\mathbf{b}(\mathbf{x})}.$$

However, the determinant of $\mathbf{A}(\mathbf{x})$ is zero since the two columns of the matrix $\mathbf{A}(\mathbf{x})$ are collinear to the vector $(\cos\theta,\ \sin\theta)$. This means that the controllable part of the acceleration is forcibly in the vehicle heading direction. Thus, $\ddot{x}$ and $\ddot{y}$ will not be independently controllable. The method of feedback linearization can therefore not be applied.

### 2.5.3 Choosing another output

In order to avoid having a singular matrix $\mathbf{A}(\mathbf{x})$, let us now choose the center of the front wheel as output. We have:

$$\mathbf{y} = \begin{pmatrix} x + \cos\theta \\ y + \sin\theta \end{pmatrix}$$

By differentiating once, we have:

$$\begin{pmatrix} \dot{y}_1 \\ \dot{y}_2 \end{pmatrix} = \begin{pmatrix} \dot{x} - \dot{\theta}\sin\theta \\ \dot{y} + \dot{\theta}\cos\theta \end{pmatrix} = v \begin{pmatrix} \cos\delta\cos\theta - \sin\delta\sin\theta \\ \cos\delta\sin\theta + \sin\delta\cos\theta \end{pmatrix} = v \begin{pmatrix} \cos(\delta+\theta) \\ \sin(\delta+\theta) \end{pmatrix}$$

Differentiating again, we obtain:

$$\begin{aligned} \begin{pmatrix} \ddot{y}_1 \\ \ddot{y}_2 \end{pmatrix} &= \begin{pmatrix} \dot{v}\cos(\delta+\theta) - v\left(\dot{\delta}+\dot{\theta}\right)\sin(\delta+\theta) \\ \dot{v}\sin(\delta+\theta) + v\cos(\delta+\theta) \end{pmatrix} \\ &= \begin{pmatrix} u_1\cos(\delta+\theta) - v\,(u_2 + v\sin\delta)\sin(\delta+\theta) \\ u_1\sin(\delta+\theta) + v\,(u_2 + v\sin\delta)\cos(\delta+\theta) \end{pmatrix} \end{aligned}$$

And therefore:

$$\begin{pmatrix} \ddot{y}_1 \\ \ddot{y}_2 \end{pmatrix} = \underbrace{\begin{pmatrix} \cos(\delta+\theta) & -v\sin(\delta+\theta) \\ \sin(\delta+\theta) & v\cos(\delta+\theta) \end{pmatrix}}_{\mathbf{A}(\mathbf{x})} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} + \underbrace{v^2\sin\delta \begin{pmatrix} -\sin(\delta+\theta) \\ \cos(\delta+\theta) \end{pmatrix}}_{\mathbf{b}(\mathbf{x})}$$

The determinant of $\mathbf{A}(\mathbf{x})$ is never equal to zero, except when $v = 0$. The linearizing control is therefore $\mathbf{u} = \mathbf{A}^{-1}(\mathbf{x}) \cdot (\mathbf{v} - \mathbf{b}(\mathbf{x}))$. And thus the tricycle control (the one that places all the poles at $-1$) is expressed by:

$$\begin{aligned} \mathbf{u} = \mathbf{A}^{-1}(\mathbf{x}) \Bigg( \left( \begin{pmatrix} x_d \\ y_d \end{pmatrix} - \begin{pmatrix} x + \cos\theta \\ y + \sin\theta \end{pmatrix} \right) + 2 \left( \begin{pmatrix} \dot{x}_d \\ \dot{y}_d \end{pmatrix} - \begin{pmatrix} v\cos(\delta+\theta) \\ v\sin(\delta+\theta) \end{pmatrix} \right) \\ + \begin{pmatrix} \ddot{x}_d \\ \ddot{y}_d \end{pmatrix} - \mathbf{b}(\mathbf{x}) \Bigg) \end{aligned}$$

where $\mathbf{w} = (x_d, y_d)$ is the desired trajectory for the output $\mathbf{y}$.

## 2.6 Sailboat

Automatic control for sailing robots [ROM 12] is a complex problem given the strong nonlinearities implied in the evolution of the system. Here we will consider the sailboat of Figure 2.7 whose

state equations [JAU 05] are given by:

$$\begin{cases} \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta - 1 \\ \dot{\theta} &= \omega \\ \dot{\delta}_s &= u_1 \\ \dot{\delta}_r &= u_2 \\ \dot{v} &= f_s\sin\delta_s - f_r\sin\delta_r - v \\ \dot{\omega} &= (1-\cos\delta_s)\, f_s - \cos\delta_r \cdot f_r - \omega \\ f_s &= \cos(\theta+\delta_s) - v\sin\delta_s \\ f_r &= v\sin\delta_r \end{cases} \tag{2.11}$$

This is of course a normalized model in which many coefficients (masses, lengths, etc.) have been set to 1 in order to simplify the following developments. The state vector $\mathbf{x} = (x, y, \theta, \delta_s, \delta_r, v, \omega)$, of dimension 7, is composed of:

- position coordinates, in other words the $x, y$ coordinates of the sailboat's center of gravity, the orientation $\theta$, and the angles $\delta_s$ and $\delta_r$ of the sail and the rudder ;
- kinematic coordinates $v$ and $\omega$ representing respectively the speed of the center of gravity and the angular speed of the boat.

The inputs $u_1$ and $u_2$ of the system are the differentials of the angles $\delta_s$ and $\delta_r$. The indices «$s$ »and «$r$ »refer respectively to the sail and the rudder.

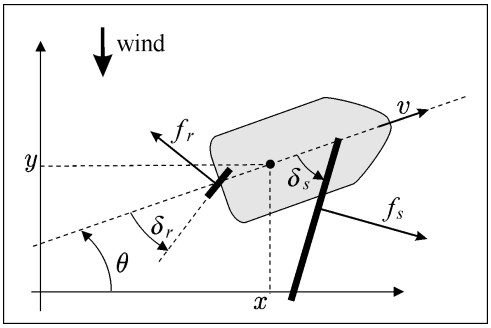


Figure 2.7: Sailing robot to be controlled

### 2.6.1 Polar curve

Let us take as outputs $\mathbf{y} = (\theta, v)$. The polar curve is the set of acceptable outputs (refer to paragraph 2.3.4), in other words the set $\mathbb{S}_y$ of all pairs $(\theta, v)$ over which we are able to stabilize. In stationary regime, we have:

$$\dot{\theta} = 0, \dot{\delta}_s = 0, \dot{\delta}_r = 0, \dot{v} = 0, \dot{\omega} = 0$$

Thus, following the state equations in [2.11], we obtain:

$$\begin{aligned}\mathbb{S}_y = \{(\theta, v) \quad | \quad & f_s \sin\delta_s - f_r \sin\delta_r - v = 0 \\ & (1-\cos\delta_s) f_s - \cos\delta_r f_r = 0 \\ & f_s = \cos(\theta+\delta_s) - v\sin\delta_s \\ & f_r = v\sin\delta_r \ \}\end{aligned}$$

An interval calculation method [HER 10] allows us to obtain the estimation of Figure 2.8.

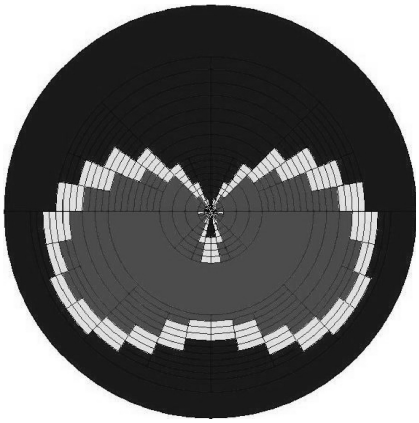

Figure 2.8: Internal frame (in light grey) and external frame (in dark grey) of the polar curve

### 2.6.2 Differential delay

We may associate to the state equations of our sailboat a *differential delay graph* between the variables (see Figure 2.9). Within this graph, a solid arrow can be interpreted, depending on the reader, either as a cause and effect relationship, a differential delay, or a state equation. A dotted arrow represents an algebraic (and not a differential) dependency. On the graph, we can distinguish two types of variables: the state variables, pointed at by solid arrows and link variables (in grey), pointed at by dotted arrows. The derivative of a state variable is expressed as an algebraic function of all the variables which are directly before it. Likewise, a link variable is an algebraic function of the variables which are directly before it.

The differential delay between a variable and an input $u_j$ is thus the minimum number of solid arrows to traverse in order to reach this variable from $u_j$. Just as in [JAU 05], let us take as output the vector $\mathbf{y} = (\delta_s, \theta)$. The differential delay matrix is:

$$\mathbf{R} = \begin{pmatrix} 1 & \infty \\ 3 & 3 \end{pmatrix}.$$

The infinity here can be interpreted as the fact that there is no causal connection that links $u_2$ to $\delta_s$. The relative degrees are therefore $k_1 = 1$ and $k_2 = 3$.

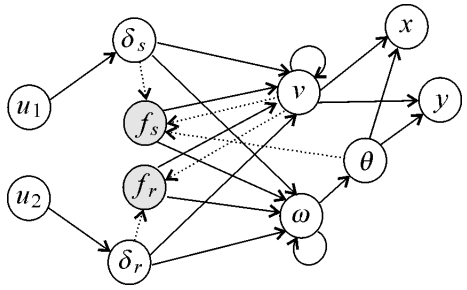


Figure 2.9: Graph of the differential delays for the sailing robot

### 2.6.3 The method of feedback linearization

Let us recall that the outputs (these are actually setpoint variables) chosen are the sail opening $y_1 = \delta_s$ and the heading $y_2 = \theta$. In order to apply a feedback linearization method, we first of all need to differentiate the outputs as many times as the relative degree requires it, in other words three times for $\theta$ and once for $\delta_s$. By looking at the differential dependency graph, we can observe that in order to express $\dddot{\theta}$ in function of $\mathbf{x}$ and $\mathbf{u}$, we need to do the same with $\ddot{\omega}, \dot{\omega}, \dot{\delta}_r, \dot{\delta}_s, \dot{f}_r, \dot{f}_s, \dot{v}$. This yields:

$$\left\{\begin{array}{rcl} \dot{v} & = & f_s \sin\delta_s - f_r \sin\delta_r - v \\ \dot{f}_s & = & -(\omega + u_1)\sin(\theta + \delta_s) - \dot{v}\sin\delta_s - v u_1 \cos\delta_s \\ \dot{f}_r & = & \dot{v}\sin\delta_r + v u_2 \cos\delta_r \\ \dot{\omega} & = & (1-\cos\delta_s)\cdot f_s - \cos\delta_r \cdot f_r - \omega \\ \ddot{\omega} & = & u_1 \sin\delta_s \cdot f_s + (1-\cos\delta_s)\cdot \dot{f}_s + u_2 \sin\delta_r \cdot f_r - \cos\delta_r \cdot \dot{f}_r - \dot{\omega} \\ \dddot{\theta} & = & \ddot{\omega} \end{array}\right.$$

We have:

$$\begin{pmatrix} \dot{y}_1 \\ \ddot{y}_2 \end{pmatrix} = \begin{pmatrix} \dot{\delta}_s \\ \ddot{\theta} \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 \\ f_s \sin\delta_s & f_r \sin\delta_r \end{pmatrix}}_{\mathbf{A}_1(\mathbf{x})} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} + \underbrace{\begin{pmatrix} 0 & 0 \\ 1-\cos\delta_s & -\cos\delta_r \end{pmatrix}}_{\mathbf{A}_2(\mathbf{x})} \begin{pmatrix} \dot{f}_s \\ \dot{f}_r \end{pmatrix} + \underbrace{\begin{pmatrix} 0 \\ -\dot{\omega} \end{pmatrix}}_{\mathbf{b}_1(\mathbf{x})}$$

However:

$$\begin{pmatrix} \dot{f}_s \\ \dot{f}_r \end{pmatrix} = \underbrace{\begin{pmatrix} -(\sin(\theta+\delta_s) + v\cos\delta_s) & 0 \\ 0 & v\cos\delta_r \end{pmatrix}}_{\mathbf{A}_3(\mathbf{x})} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} + \underbrace{\begin{pmatrix} -\omega\sin(\theta+\delta_s) + \dot{v}\sin\delta_s \\ \dot{v}\sin\delta_r \end{pmatrix}}_{\mathbf{b}_2(\mathbf{x})}$$

And thus we have a relation of the form:

$$\begin{pmatrix} \dot{y}_1 \\ \ddot{y}_2 \end{pmatrix} = \mathbf{A}_1\mathbf{u} + \mathbf{A}_2\left(\mathbf{A}_3\mathbf{u} + \mathbf{b}_2\right) + \mathbf{b}_1$$
$$= \left(\mathbf{A}_1 + \mathbf{A}_2\mathbf{A}_3\right)\mathbf{u} + \mathbf{A}_2\mathbf{b}_2 + \mathbf{b}_1 = \mathbf{A}\mathbf{u} + \mathbf{b}$$

In order to set $(\dot{y}_1, \ddot{y}_2)$ to a certain setpoint $\mathbf{v} = (v_1, v_2)$, we need to take:

$$\mathbf{u} = \mathbf{A}^{-1}(\mathbf{x})\left(\mathbf{v} - \mathbf{b}\left(\mathbf{x}\right)\right)$$

The system looped in this manner is governed by the differential equations:

$$\mathcal{S}_{\mathrm{L}} : \begin{cases} \dot{y}_1 &= v_1 \\ \ddot{y}_2 &= v_2 \end{cases} \tag{2.12}$$

which are linear and decoupled. The linearized system is of order 4 instead of 7. We have thus lost control over three variables which happen to be $x, y$ and $v$. The loss of control over $x$ and $y$ was predictable (we want the boat to advance and therefore it is only natural that this corresponds to an instability for these two variables $x$ and $y$). As for the loss of control over $v$, this is without consequence since the associated dynamics are stable. How indeed would it be possible to design a boat that would be able to keep a fixed heading and sail opening, without its speed converging towards a finite value ?

Let us now determine the singularities of our linearizing feedback loop. By calculating the expression of $\mathbf{A}(\mathbf{x})$, we can show that:

$$\det\left(\mathbf{A}(\mathbf{x})\right) = f_r \sin\delta_r - v\cos^2\delta_r \overset{[2.11]}{=} v\left(2\sin^2\delta_r - 1\right).$$

We have a singularity when this quantity is equal to zero, in other words if:

$$v = 0 \text{ or } \delta_r = \frac{\pi}{4} + k\frac{\pi}{2}. \tag{2.13}$$

The singularity corresponding to $v = 0$ is relatively simple to understand: when the boat is not advancing, we can no longer control it. The condition on the rudder angle $\delta_r$ is more delicate to interpret. Indeed, the condition $\delta_r = \pm\frac{\pi}{4}$ translates to a maximal rotation. Any action on the rudder when $\delta_r = \pm\frac{\pi}{4}$ translates to a slower rotation. This is what this singularity means.

We are dealing with two decoupled monovariate systems here. Let us denote by $\mathbf{w} = (w_1,\ w_2)$ the setpoint for $\mathbf{y}$. We will sometimes write $\mathbf{w} = (\hat{\delta}_s, \hat{\theta})$ in order to recall that $w_1$ and $w_2$ are the setpoints corresponding to the sail opening angle and the heading. Let us choose the proportional and derivative controller given by:

$$\begin{cases} v_1 &= (w_1 - y_1) + \dot{w}_1 \\ v_2 &= (w_2 - y_2) + 3\left(\dot{w}_2 - \dot{y}_2\right) + 3\left(\ddot{w}_2 - \ddot{y}_2\right) + \dddot{w}_2 \end{cases}$$

which allows all poles of the feedback system to be equal to $-1$ (refer to Equation [2.2]). By assuming that the setpoint $\mathbf{w}$ is constant, the state equations of the state feedback controller for our nonlinear

system are given by:

$$\mathbf{u} = \mathbf{A}^{-1}(\mathbf{x}) \left( \begin{pmatrix} w_1 - \delta_s \\ w_2 - \theta - 3\dot{\theta} - 3\ddot{\theta} \end{pmatrix} - \mathbf{b}(\mathbf{x}) \right). \tag{2.14}$$

But $\dot{\theta}$ and $\ddot{\theta}$ are analytic functions of the state $\mathbf{x}$. Indeed, we have:

$$\begin{array}{rcl} \dot{\theta} & = & \omega \\ \ddot{\theta} & = & (1 - \cos\delta_s) f_s - \cos\delta_r f_r - \omega \end{array}$$

Equation [2.14] can therefore be written in the form:

$$\mathbf{u} = \mathbf{r}(\mathbf{x}, \mathbf{w}) = \mathbf{r}\left(\mathbf{x}, \hat{\delta}_s, \hat{\theta}\right) \tag{2.15}$$

This controller is *static* since it does not have a state variable.

### 2.6.4 Polar curve control

In some situations, the boater does not want complete autonomy of his boat, only steering assistance. He does not wish to decide the angle of the sails, but simply its speed and heading. In summary, he would like to choose a point on the polar curve and it is up to the controller to perform low-level control. In cruising regime, we have:

$$\left\{ \begin{array}{rcl} 0 & = & \bar{f}_s \sin\bar{\delta}_s - \bar{f}_r \sin\bar{\delta}_r - \bar{v} \\ 0 & = & \left(1 - \cos\bar{\delta}_s\right) \cdot \bar{f}_s - \cos\bar{\delta}_r \cdot \bar{f}_r \\ \bar{f}_s & = & \cos\left(\bar{\theta} + \bar{\delta}_s\right) - \bar{v}\sin\bar{\delta}_s \\ \bar{f}_r & = & \bar{v}\sin\bar{\delta}_r \end{array} \right.$$

If $\left(\bar{\theta}, \bar{v}\right)$ is in the polar curve, we can calculate $\left(\bar{f}_r, \bar{\delta}_r, \bar{f}_s, \bar{\delta}_s\right)$ (there is at least one solution, by definition of the polar curve). Thus, it is sufficient to inject $\left(\bar{\theta}, \bar{\delta}_s\right)$ in controller [2.15] in order to preform our control. Figure 2.10 illustrates the docking of a sailboat in a harbor using this approach [HER 10].

## 2.7 Sliding mode

*Sliding mode* is a set of control methods that belong to the family of feedback linearization techniques. Assume that the system to be controlled has the form

$$\left\{ \begin{array}{rcl} \dot{\mathbf{x}} & = & \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x}) \cdot \mathbf{u} \\ \mathbf{y} & = & \mathbf{h}(\mathbf{x}) \end{array} \right.$$

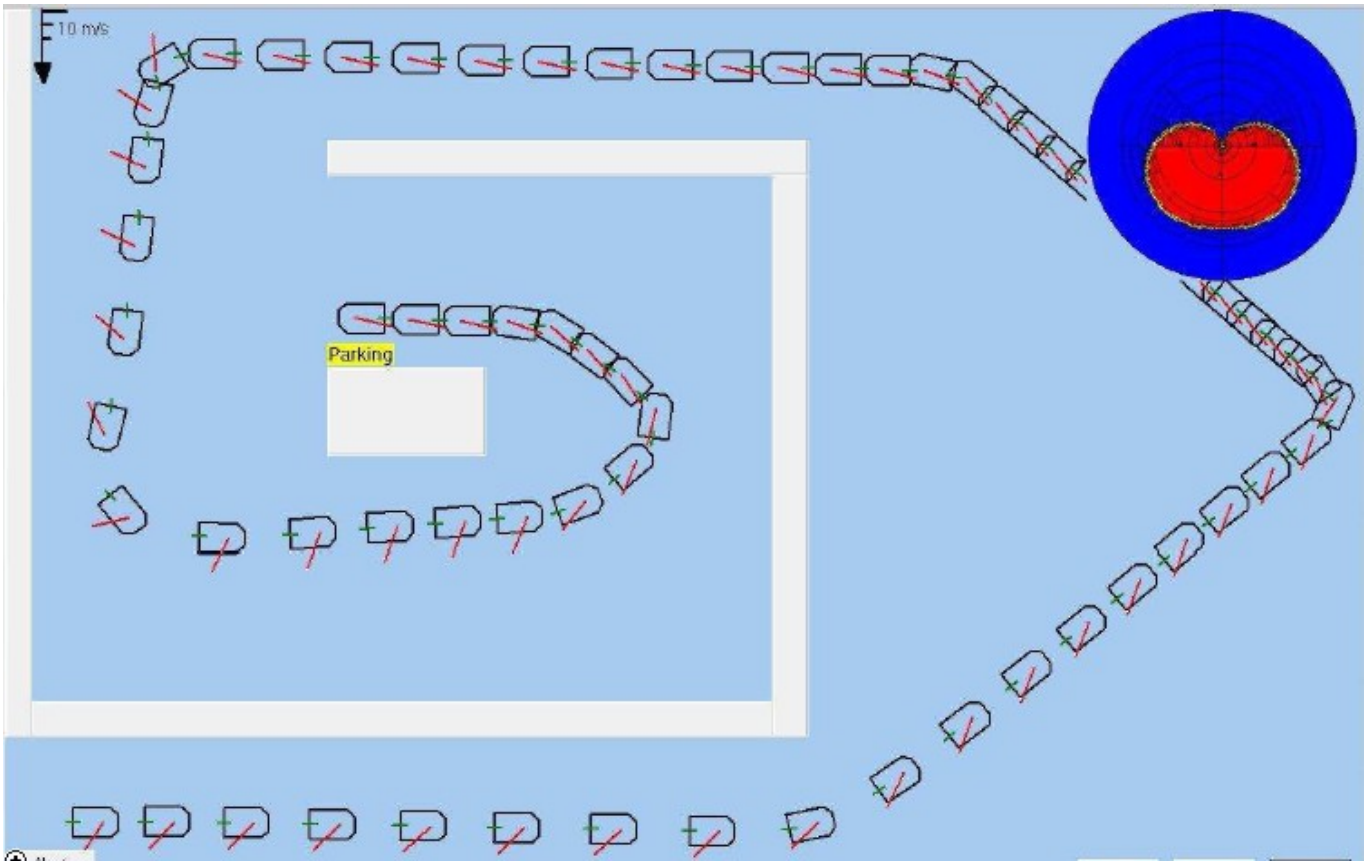


Figure 2.10: By using a linearizing controller, the robot docks in its place in the harbor ; the polar curve is represented on the top right corner

where dim $\mathbf{u} = \dim \mathbf{y}$. Recall that, after computing some derivatives (see Formula (2.7)), we get

$$\begin{pmatrix} y_1^{(k_1)} \\ \vdots \\ y_m^{(k_m)} \end{pmatrix} = \mathbf{A}(\mathbf{x}) \cdot \mathbf{u} + \mathbf{b}(\mathbf{x}).$$

This means that, if $\mathbf{A}(\mathbf{x})$ is invertible, we can chose $\mathbf{y}^{(k)} = \left(y_1^{(k_1)}, \ldots, y_m^{(k_m)}\right)$ independently. For this, it suffices to take

$$\mathbf{u} = \mathbf{A}^{-1}(\mathbf{x}) \left( \begin{pmatrix} y_1^{(k_1)} \\ \vdots \\ y_m^{(k_m)} \end{pmatrix} - \mathbf{b}(\mathbf{x}) \right) = \boldsymbol{\phi}\left(\mathbf{x}, y_1^{(k_1)}, \ldots, y_m^{(k_m)}\right).$$

Now, which vector $\left(y_1^{(k_1)}, \ldots, y_m^{(k_m)}\right)$ do we have to choose to achieve our control goal. Two different types of approach can be considered for this purpose. The first one is the *proportional and derivative* control approach seen previously in this chapter. The second one in the *sliding surface method* that will be explained in this section.

For simplicity and without loss of generality, let us assume that we have a single input and a

single output, *i.e.*, dim $u = \dim y = 1$. We thus have

$$u = \frac{y^{(k)} - b(\mathbf{x})}{a(\mathbf{x})} = \phi\left(\mathbf{x}, y^{(k)}\right). \tag{2.16}$$

**Proportional and derivative control method**. We choose $y^{(k)}$ as follows:

$$y^{(k)} = \alpha_0 (y_d - y) + \alpha_1 (\dot{y}_d - \dot{y}) + \cdots + \alpha_{k-1}\left(y_d^{(k-1)} - y^{(k-1)}\right) + y_d^{(k)}.$$

If the $\alpha_i$ are well chosen, then, $y(t)$ will quickly be equal to the desired output $y(t)$, or equivalently, the error $e = y_d - y$ converges toward 0 with the dynamic

$$e^{(k)} + \alpha_{k-1} e^{(k-1)} + \cdots + \alpha_1 \dot{e} + \alpha_0 e = 0.$$

The corresponding feedback is

$$u = \phi\left(\mathbf{x}, \alpha_0 (y_d - y) + \cdots + \alpha_{k-1}\left(y_d^{(k-1)} - y^{(k-1)}\right) + y_d^{(k)}\right).$$

**Sliding surface** method. Instead of choosing a dynamic for $y$ of order $k$, we choose a dynamic of order $n-1$:

$$y^{(k-1)} = \alpha_0 (y_d - y) + \alpha_1 (\dot{y}_d - \dot{y}) + \cdots + \alpha_{k-2}\left(y_d^{(k-2)} - y^{(k-2)}\right) + y_d^{(k-1)},$$

or equivalently

$$s(\mathbf{x}, t) = \underbrace{y_d^{(k-1)} - y^{(k-1)}}_{e^{(k-1)}} + \alpha_{k-2}\underbrace{(y_d^{(k-2)} - y^{(k-2)})}_{e^{(k-2)}} + \cdots + \alpha_1\underbrace{(\dot{y}_d - \dot{y})}_{\dot{e}} + \alpha_0\underbrace{(y_d - y)}_{e} = 0.$$

This equation corresponds to the dynamics of the error, but can also be interpreted as a *surface* of dimension $k-1$ of the $k$-dimensional space $(y, \dot{y}, \ddot{y}, \ldots, y^{(k-1)})$. This surface is moving when $y_d$ is time dependant. We were able to write this surface under the form $s(\mathbf{x}, t) = 0$ because the elements $(y, \dot{y}, \ddot{y}, \ldots, y^{(k-1)})$ are all functions of $\mathbf{x}$ which is not the case for $y^{(k)}$ which depends on both $\mathbf{x}$ and $u$. Now, it remains to choose what we have to put for $y^{(k)}$ in (2.16). The principle of sliding mode is to bring the system to this surface $s(\mathbf{x}, t)$ on to stay on it. If we achieve this goal, then the error $y_d - y$ will have the required dynamics. To converge to the surface, we may take

$$y^{(k)} = K \cdot \operatorname{sign}(s(\mathbf{x}, t)),$$

where $K > 0$ is large. If $s > 0$, it means that $y^{(k-1)}$ is too small and we have to increase it. This is why $K$ should be positive. From (2.16) the sliding mode control will be

$$u = \phi(\mathbf{x}, K \cdot \operatorname{sign}(s(\mathbf{x}, t))).$$

One important feature of sliding mode is that we do not need to insert an integral effect to compensate constant perturbations and the controller is robust with respect to many types of perturbation. Indeed, any perturbation that would make escape the system from the sliding surface yields an

immediate and strong reaction from the controller that bring it back to the surface. On the other hand, a sliding mode control generates a behavior with a high-frequency and non-deterministic switching control signal that causes the system to *chatter* in a small neighborhood of the sliding surface.

## 2.8 Kinematic model and dynamic model

### 2.8.1 Principle

The dynamic models for the robots are of the form:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$$

where $\mathbf{u}$ is the vector of the external forces (that are under our control). The function $\mathbf{f}$ involves dynamic coefficients (such as masses, inertial moments, coefficients of friction, etc.) as well as geometric coefficients (such as lengths). The dynamic coefficients are generally not well known and can change in time with wear or usage. If we now take as new input the vector $\mathbf{a}$ of the desired accelerations at the application points of the forces (in the direction of the forces), we obtain a new model, referred to as *kinematic*, of the form:

$$\dot{\mathbf{x}} = \boldsymbol{\varphi}(\mathbf{x}, \mathbf{a})$$

but in this new model most of the dynamic coefficients have disappeared. It is possible to switch from a dynamic model to a kinematic model using a so-called *high-gain* controller, of the form:

$$\mathbf{u} = K(\mathbf{a} - \mathbf{a}(\mathbf{x}, \mathbf{u}))$$

where $K$ is a very large real number. The function $\mathbf{a}(\mathbf{x}, \mathbf{u})$ is a function that allows to obtain the acceleration associated with the forces, in function of the forces $\mathbf{u}$ and of the state $\mathbf{x}$. In practice, we are not looking to express $\mathbf{a}(\mathbf{x}, \mathbf{u})$ within the controller, but rather to measure $\mathbf{a}(\mathbf{x}, \mathbf{u})$ using accelerometers. The controller that will actually be implemented is:

$$\mathbf{u} = K(\mathbf{a} - \widetilde{\mathbf{a}})$$

where $\widetilde{\mathbf{a}}$ corresponds to the vector of the measured accelerations. Thus, we have:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, K(\mathbf{a} - \mathbf{a}(\mathbf{x}, \mathbf{u}))) \Leftrightarrow \dot{\mathbf{x}} = \boldsymbol{\varphi}(\mathbf{x}, \mathbf{a}).$$

The simple high-gain feedback has allowed us to get rid of numerous dynamic parameters and switch from an uncertain system to a reliable one, with well-known geometric coefficients. This high-gain feedback is known in electronics as an operational amplifier, where it is used with the same idea of robustness.

Switching from a dynamic model to a kinematic one has the following advantages:

- the linearizing controller developed in this chapter requires using a reliable model such as a

kinematic model. If the coefficients (which are not measured) are not well known (such as in the case of dynamic systems), the linearizing controller will not work in practice ;

- the kinematic model is easier to put into equations. It is not necessary to have a dynamic model to obtain the latter ;
- the servo-motors (see paragraph 2.8.3) are cheap and easy to find. They incorporate this high-gain controller. We may see them as mechanical operational amplifiers.

We will illustrate the concept in the following paragraph, through the example of the inverted rod pendulum.

## 2.8.2 Example of the inverted rod pendulum

Let us consider the inverted rod pendulum, composed of a pendulum in an unstable equilibrium on top of a moving cart, as represented on Figure 2.11.

### 2.8.2.1 Dynamic model

The quantity $u$ is the force exerted on the cart of mass $M$, $x$ indicates the position of the cart, $\theta$ is the angle between the pendulum and the vertical direction. The state equations are written in the form:

$$\frac{d}{dt}\begin{pmatrix} x \\ \theta \\ \dot{x} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \dot{x} \\ \dot{\theta} \\ \frac{-m\sin\theta(\ell\dot{\theta}^2 - g\cos\theta)}{M+m\sin^2\theta} \\ \frac{\sin\theta((M+m)g - m\ell\dot{\theta}^2\cos\theta)}{\ell(M+m\sin^2\theta)} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \frac{1}{M+m\sin^2\theta} \\ \frac{\cos\theta}{\ell(M+m\sin^2\theta)} \end{pmatrix} u \tag{2.17}$$

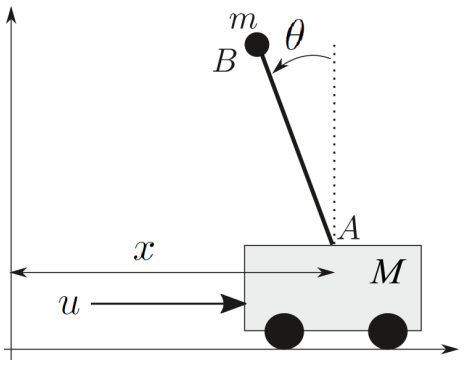


Figure 2.11: Inverted rod pendulum to be modeled and controlled

### 2.8.2.2 Kinematic model

Let us recall the the state equations of the inverted rod pendulum, but instead of taking the force as input, we take the acceleration $a = \ddot{x}$. We obtain, following [2.17]:

$$a = \frac{1}{M + m\sin^2\theta}\left(-m\sin\theta(\ell\dot{\theta}^2 - g\cos\theta) + u\right) \tag{2.18}$$

Therefore:

$$\begin{aligned}
\dot{\theta} &\overset{[2.17]}{=} \frac{\sin\theta((M+m)g - m\ell\dot{\theta}^2\cos\theta)}{\ell(M+m\sin^2\theta)} + \frac{\cos\theta}{\ell(M+m\sin^2\theta)}u \\
&\overset{[2.18]}{=} \frac{\sin\theta((M+m)g - m\ell\dot{\theta}^2\cos\theta)}{\ell(M+m\sin^2\theta)} \\
&\quad + \frac{\cos\theta}{\ell(M+m\sin^2\theta)}\left(m\sin\theta\left(\ell\dot{\theta}^2 - g\cos\theta\right)) + (M+m\sin^2\theta)\,a\right) \\
&= \frac{(M+m)g\sin\theta - gm\sin\theta\cos^2\theta + \left(M+m\sin^2\theta\right)\cos\theta\cdot a}{\ell(M+m\sin^2\theta)} \\
&= \frac{g\sin\theta}{\ell} + \frac{\cos\theta}{\ell}a.
\end{aligned}$$

Let us note that this relation could have been obtained directly by noticing that:

$$\ell\ddot{\theta} = a\cdot\cos\theta + g\cdot\sin\theta.$$

REMARK 2.4.– *In order to obtain this relation in a more rigorous manner, we need to write the temporal derivative of the speed composition formula, in other words:*

$$\dot{\mathbf{v}}_A = \dot{\mathbf{v}}_B + \overrightarrow{AB}\wedge\overrightarrow{\dot{\omega}}$$

*and write this formula in the coordinate system related to the pendulum. We obtain:*

$$\begin{pmatrix} a\cos\theta \\ -a\sin\theta \\ 0 \end{pmatrix} = \begin{pmatrix} -g\sin\theta \\ n \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ \ell \\ 0 \end{pmatrix} \wedge \begin{pmatrix} 0 \\ 0 \\ \dot{\omega} \end{pmatrix}$$

*where $n$ corresponds to the normal acceleration of the mass $m$. We thus obtain the desired relation as well as the normal acceleration $n = -a\sin\theta$ which will not be used.*

Finally, the kinematic model is written as:

$$\frac{d}{dt}\begin{pmatrix} x \\ \theta \\ \dot{x} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \dot{x} \\ \dot{\theta} \\ 0 \\ \frac{g\sin\theta}{\ell} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \\ \frac{\cos\theta}{\ell} \end{pmatrix} a \tag{2.19}$$

This model, referred to as the *kinematic model*, only involves positions, speeds and accelerations. It is a lot more simple than the dynamic model and involves less coefficients. However, it corresponds less to reality since the correct input is a force and not an acceleration. In practice, we may switch from the dynamic model [2.17] with input $u$ to the kinematic model [2.19] with input $a$ by calculating $u$ using a *high-gain proportional controller* of the form:

$$u = K\,(a - \ddot{x}) \tag{2.20}$$

with $K$ very large and where $a$ is a new input.

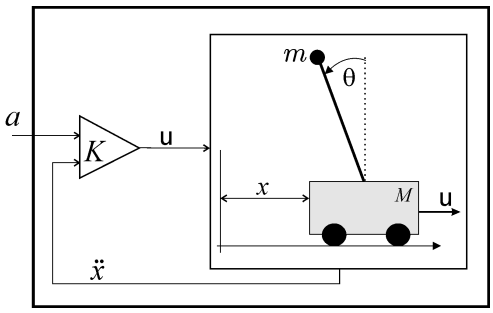


Figure 2.12: The inverted rod pendulum, looped by a high gain $K$, behaves like a kinematic model

The acceleration $\ddot{x}$ can be measured using an accelerometer. If $K$ is sufficiently large, we will of course have the controller $u$ that will give us the desired acceleration $a$, in other words we will have $\ddot{x} = a$. Thus, System [2.17] can be described by the state equations in (2.19) which do not involve any of the inertial parameters of the system. A controller designed over the kinematic model will therefore be more robust than one designed over the dynamic system since the controller will function for any values of $m, M$, the inertial momentums, friction, etc. Let us recall that this high-gain controller is very close to the principle of the operational amplifier. In addition to being more robust, such an approach allows to have a simpler model that is easier to obtain. For the implementation of the controller in [2.20], we of course do not need to use the state equations [2.17] in order to express $\ddot{x}$, but measure $\ddot{x}$ instead. It is this measurement that allows us to have a controller that is independent of the dynamic parameters.

Let us recall our inverted rod pendulum and try to make the pendulum oscillate from left to right with a desired angle of the form $\theta_d = \sin t$. Let us apply a linearizing controller for this. We have:

$$\ddot{\theta} = \frac{g \sin\theta}{\ell} + \frac{\cos\theta}{\ell} a.$$

We will therefore take:

$$a = \frac{\ell}{\cos\theta}\left(v - \frac{g\sin\theta}{\ell}\right)$$

where $v$ is the new input. We will then choose:

$$v = (\theta_d - \theta) + 2\left(\dot{\theta}_d - \dot{\theta}\right) + \ddot{\theta}_d = \sin t - \theta + 2\cos t - 2\dot{\theta} - \sin t$$

And finally:

$$\begin{aligned} u &= K\left(a - \ddot{x}\right) \\ &= K\left(\frac{\ell}{\cos\theta}\left(\sin t - \theta + 2\cos t - 2\dot{\theta} - \sin t - \frac{g\sin\theta}{\ell}\right) - \ddot{x}\right). \end{aligned}$$

Note that the inertial parameters are not taken into account in this controller. This controller ensures that the system will respect its setpoint angle. On the other hand, the position of the cart can diverge, since $u$ does not depend on $x$. The dynamics of $x$ are hidden and moreover unstable

here. These hidden dynamics are conventionally referred to as *zero dynamics*.

### 2.8.3 Servo-motors

A mechanical system is controlled by forces or torques and obeys a dynamic system that depends on numerous little-known coefficients. This same mechanical system represented by a kinematic model is controlled by positions, speeds or accelerations. The kinematic model depends on well-known geometric coefficients and is much simpler to put into equations. In practice, one switches from a dynamic model to its kinematic equivalent by adding servo-motors. In summary, a servo-motor is a DC motor with an electrical control circuit and a sensor (of position, speed or acceleration). The control circuit calculates the voltage $u$ to give the motor in order for the quantity measured by the sensor to correspond to the setpoint $w$. There are three types of servo-motors:

- the *position servo*. The sensor measures the position (or the angle) $x$ of the motor and the control law is expressed by $u = K(x - w)$. If $K$ is large, we may conclude that $x \simeq w$ ;
- the *speed servo*. The sensor measures the speed (or the angular speed) $\dot{x}$ of the motor and the control law is expressed by $u = K(\dot{x} - w)$. If $K$ is large, we have $\dot{x} \simeq w$ ;
- the *acceleration servo*. The sensor measures the acceleration (tangential or angular) $\ddot{x}$ of the motor and the control law is expressed by $u = K(\ddot{x} - w)$. If $K$ is large, we have $\ddot{x} \simeq w$. It is this type of servo-motor that we have chosen for the inverted rod pendulum.

Thus, when we wish to control a mechanical system, the use of servo-motors allows us (i) to have a model that is easier to obtain, (ii) to have a model with fewer coefficients that is closer to reality and (iii) to have a more robust controller with respect to any modification of the dynamic coefficients of the system.

# Exercises

Exercise 2.1.– *Crank*

Let us consider the manipulator robot, or *crank* of Figure 2.13 (on the left). This robot is composed of two arms of length $\ell_1$ and $\ell_2$. Its two degrees of freedom denoted by $x_1$ and $x_2$ are represented on the figure. The inputs $u_1, u_2$ of the system are the angular speeds of the arms (in other words $u_1 = \dot{x}_1, u_2 = \dot{x}_2$). We will take as output the vector $\mathbf{y} = (y_1, y_2)$ corresponding to the end of the second arm.

1) Give the state equations of the robot. We will take the state vector $\mathbf{x} = (x_1, x_2)$.

2) We would like $\mathbf{y}$ to follow a setpoint $\mathbf{w}$ describing a target circle (on the right of Figure 2.13). This setpoint satisfies:

$$\mathbf{w} = \mathbf{c} + r \cdot \begin{pmatrix} \cos t \\ \sin t \end{pmatrix}.$$

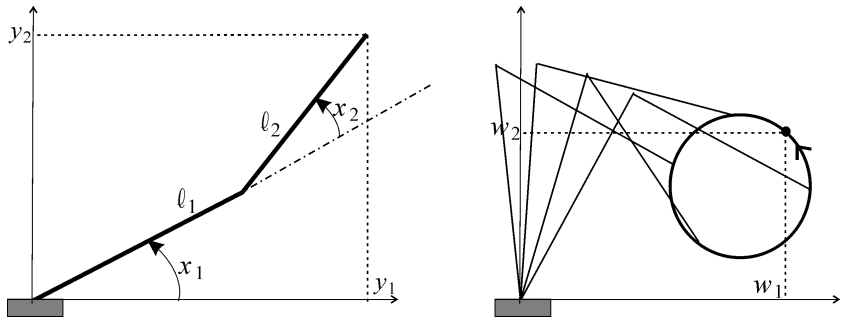


Figure 2.13: Manipulator robot whose end effector must follow a circle

Give the expression of a control law that allows to perform this task. We will use a feedback linearization method and we will place the poles at $-1$.

3) Study the singularities of the control.

4) Let us consider the case $\ell_1 = \ell_2$, $\mathbf{c} = (3, 4)$ and $r = 1$. For which values of $\ell_1$ are we certain to be able to move freely on the target circle, without encountering singularities ?

5) Write a MATLAB program illustrating this control law. Start with the file `ex_crank.m`.

---

Exercise 2.2.– *The three pools*

Let us consider a flow system with three pools as represented on Figure 2.14.

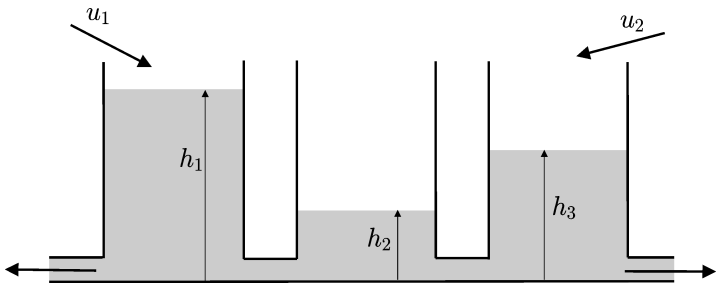


Figure 2.14: System composed of three pools containing water and connected by two channels

This system is described by the following state equations:

$$\begin{cases} \dot{h}_1 &= -\alpha(h_1) - \alpha(h_1 - h_2) + u_1 \\ \dot{h}_2 &= \alpha(h_1 - h_2) - \alpha(h_2 - h_3) \\ \dot{h}_3 &= -\alpha(h_3) + \alpha(h_2 - h_3) + u_2 \\ y_1 &= h_1 \\ y_2 &= h_3 \end{cases}$$

where $\alpha(h) = a \cdot \text{sign}(h)\sqrt{2g|h|}$. We have chosen our outputs here to be the water levels in the first and third pools.

1) Propose a feedback that would make the system linear and decoupled.

2) Propose a proportional-integral controller for the linearized system.

3) Give the state equations of the obtained controller.

4) Write a MATLAB program that simulates the system and its control law. Start with the file `ex_pools.m`.

---

EXERCISE 2.3.– *Train robot*

Let us consider a robot A (on the left of Figure 2.15) described by the following state equations (tank model):

$$\begin{cases} \dot{x}_a &= v_a \cos\theta_a \\ \dot{y}_a &= v_a \sin\theta_a \\ \dot{\theta}_a &= u_{a1} \\ \dot{v}_a &= u_{a2} \end{cases}$$

where $v_a$ is the speed of the robot, $\theta_a$ its orientation and $(x_a, y_a)$ the coordinates of its center. We assume to be able to measure the state variables of our robot with very high precision.

1) Calculate $\ddot{x}_a, \ddot{y}_a$ in function of $x_a, y_a, v_a, \theta_a, u_{a1}, u_{a2}$.

2) Propose a controller that allows to follow the trajectory:

$$\begin{cases} \hat{x}_a(t) &= L_x \sin(\omega t) \\ \hat{y}_a(t) &= L_y \cos(\omega t) \end{cases}$$

with $\omega = 0.1$, $L_x = 15$ and $L_y = 7$. A feedback linearization method must be used for this. Illustrate the behavior of your controller in MATLAB. Start with the file `ex_train.m`.

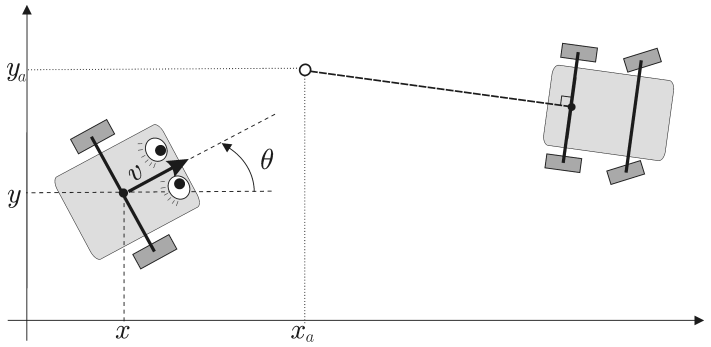


Figure 2.15: Our robot (with eyes) following a vehicle (here a car) whose state equations are unknown. This car has an imaginary attachment point (small white circle) that we must attach to

3) A second robot B of the same type as A wishes to follow robot A (see Figure 2.16). We define a virtual attachment point with coordinates $(\hat{x}_b, \hat{y}_b)$ in order for vehicle B (on the left of the figure) to be able to attach itself to our robot A. We can send it the information associated with the attachment point wirelessly.

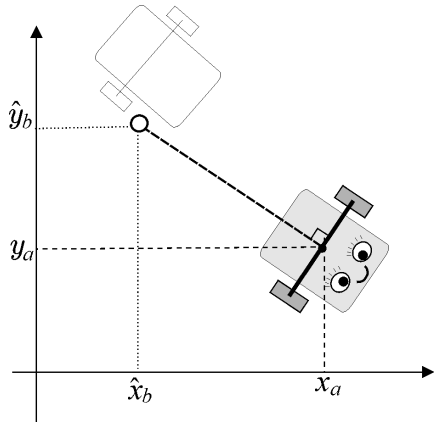


Figure 2.16: Robot B (dotted) has to follow robot A

This point will be positioned at the rear of robot A at a distance $\ell$ of our reference point $(x_a, y_a)$. Give the expression of these quantities in function of the state of our vehicle A.

4) Simulate this second vehicle B in MATLAB together with its controller following robot A.

5) Add a third robot C that follows B with the same principle. Simulate the entire system in MATLAB.

6) Given that in this exercise the reference path is precisely known, propose a controller that will allow robots B and C to precisely follow robot A.

---

EXERCISE 2.4.– *Controlling a 3D underwater robot*

Let us consider the underwater robot already discussed in Exercise 1.10. This robot is described by the following state equations:

$$\begin{cases} \dot{p}_x &= v\cos\theta\cos\psi \\ \dot{p}_y &= v\cos\theta\sin\psi \\ \dot{p}_z &= -v\sin\theta \\ \dot{v} &= u_1 \\ \dot{\varphi} &= -0.1\ \sin\varphi\cdot\cos\theta + \tan\theta\cdot v\cdot(\sin\varphi\cdot u_2 + \cos\varphi\cdot u_3) \\ \dot{\theta} &= \cos\varphi\cdot v\cdot u_2 - \sin\varphi\cdot v\cdot u_3 \\ \dot{\psi} &= \frac{\sin\varphi}{\cos\theta}\cdot v\cdot u_2 + \frac{\cos\varphi}{\cos\theta}\cdot v\cdot u_3 \end{cases}$$

where $(p_x, p_y, p_z)$ is the position of its center and $(\varphi, \theta, \psi)$ are the three Euler angles. We took here $u_4 = -0.1\ \sin\varphi\cdot\cos\theta$ in order to simulate a static ballast which tends to make the roll equal to zero via a pendulum effect. Its inputs are the tangential acceleration $u_1$, the pitch $u_2$ and the yaw $u_3$. Suggest a controller capable of controlling the robot around the cycloid of equation:

$$\begin{pmatrix} x_d \\ y_d \\ z_d \end{pmatrix} = \begin{pmatrix} R\cdot\sin(f_1 t) + R\cdot\sin(f_2 t) \\ R\cdot\cos(f_1 t) + R\cdot\cos(f_2 t) \\ R\cdot\sin(f_3 t) \end{pmatrix}$$

with $f_1 = 0.01$, $f_2 = 6f_1$, $f_3 = 3f_1$ and $R = 20$. For the control, we will choose a time constant of 5 seconds. Simulate the behavior of the controller in MATLAB. Start with the file `ex_auv3d.m`.

---

EXERCISE 2.5.– *Flat system*

Consider the system described by the state equations:

$$\begin{cases} \dot{x}_1 &= x_1 + x_2 \\ \dot{x}_2 &= x_2^2 + u \\ y &= x_1. \end{cases}$$

1) A system is called *flat* if there are two functions $\phi, \psi$ such that:

$$\begin{cases} \mathbf{x} &= \phi\left(y, \dot{y}, \ldots, y^{(r-1)}\right) \\ u &= \psi\left(y, \dot{y}, \ldots, y^{(r-1)}, y^{(r)}\right) \end{cases}$$

Show that our system is flat. Give the expressions of $\phi$ and $\psi$.

2) Give the expression of a linearizing feedback for the system of the form $u = \gamma(v, y, \dot{y})$, where $v$ is a new input. This feedback transforms our system into a system described by $\ddot{y} = v$.

3) We would like to control the system $\ddot{y} = v$ using a proportional-derivative controller. Give the expression of the controller $v = \eta(w, \dot{w}, \ddot{w}, y, \dot{y})$, where $w$ is a setpoint that varies with time and that allows us to have an error $e = w - y$ that converges towards zero. All the poles are equal to $-1$.

4) Deduce from the above equations an output feedback controller for the initial system of the form $u = \rho(w, \dot{w}, \ddot{w}, y, \dot{y})$ which is such that the error $e = w - y$ converges towards 0.

---

EXERCISE 2.6.– *Pursuit*

Let us consider two robots described by the following state equations:

$$\left\{\begin{array}{lcl}\dot{x}_1 &=& u_1\cos\theta_1\\ \dot{y}_1 &=& u_1\sin\theta_1\\ \dot{\theta}_1 &=& u_2\end{array}\right. \quad \text{and} \quad \left\{\begin{array}{lcl}\dot{x}_2 &=& v_1\cos\theta_2\\ \dot{y}_2 &=& v_1\sin\theta_2\\ \dot{\theta}_2 &=& v_2\end{array}\right.$$

In this exercise, robot 1 tries to follow robot 2 (see Figure 2.17).

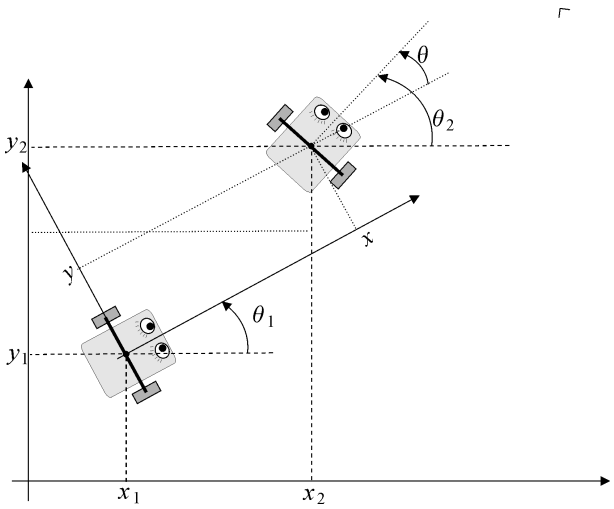


Figure 2.17: Robot 1 is in pursuit of robot 2

1) Let $\mathbf{x} = (x, y, \theta)$ be the position vector of robot 2 in the coordinate system of robot 1. Show that $\mathbf{x}$ satisfies a state equation of the form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{v}, \mathbf{u})$$

2) We assume that the control variables $v_1$ and $v_2$ of robot 2 are known (a polynomial in $t$, for example). Suggest a controller that generates us $\mathbf{u}$ in order to have $x = w_1$ and $y = w_2$, where $\mathbf{w} = (w_1, w_2)$ corresponds to a setpoint in relative position. The poles for the error are fixed at $-1$.

3) Study the singularities of this controller.

4) Illustrate this control law with MATLAB in the situation where robot 1 would like to point towards robot 2 while keeping a distance of 10 m. Start with the file `ex_pursuit.m`.

---

EXERCISE 2.7.– *Controlling the* SAUCISSE *robot*

Consider the underwater robot represented on Figure 2.18.

This is the SAUCISSE robot, built by students of the ENSTA Bretagne for the SAUC'E competition (*Student Autonomous Underwater Challenge Europe*). It includes three propellers. Propellers 1 and 2 on the left and the right are able to act on the speed of the robot and its angular speed. Propeller

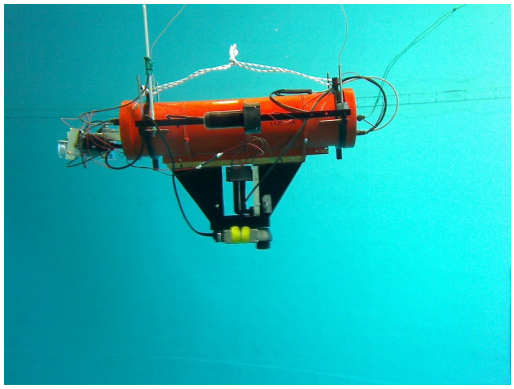

Figure 2.18: The SAUCISSE robot in a pool

3 acts on the depth of the robot. This robot is stable in roll and pitch and we will assume that its angles of bank $\varphi$ and elevation $\theta$ are always zero. The state equations of the robot are the following:

$$\begin{cases} \dot{x} &= v_x \\ \dot{y} &= v_y \\ \dot{z} &= v_z \\ \dot{\psi} &= \omega \\ \dot{v}_x &= u_1 \cos\psi \\ \dot{v}_y &= u_1 \sin\psi \\ \dot{v}_z &= u_3 \\ \dot{\omega} &= u_2 \end{cases}$$

Let us note that no nonholonomic constraint has been assumed in this model. The speed vector of the robot $(v_x, v_y)$ is not necessarily in its axis, in contrast to the case of the cart model. The robot can therefore operate in crab steering mode. On the other hand, the propulsion is necessarily in the direction of the robot axis. If we are limited to the horizontal plane, this model is known as a *hovercraft*.

1) Give the differential dependency graph associated with this system.

2) Let us choose as output the vector $\mathbf{y} = (x, y, z)$. Give the differential delay matrix and deduce the relative degrees from it. What can we conclude ?

3) In order to balance the differential delays by delaying $u_1$, we add two integrators in front of $u_1$. Our new system will admit as new inputs $\mathbf{a} = (a_1, a_2, a_3)$ with:

$$\begin{cases} \ddot{u}_1 &= a_1 \\ u_2 &= a_2 \\ u_3 &= a_3 \end{cases} \tag{2.21}$$

What are the new state equations of the delayed system ? Give the differential dependency graph as well as the associated differential delay matrix.

4) Perform a feedback linearization of the delayed system.

5) Deduce from the above the controller corresponding to our robot. We will place all the poles at $-1$.

---

EXERCISE 2.8.– *Sliding mode control of a cart*

Consider the cart described by

$$\begin{cases} \dot{x}_1 &= x_4 \cos x_3 \\ \dot{x}_2 &= x_4 \sin x_3 \\ \dot{x}_3 &= u_1 \\ \dot{x}_4 &= u_2 \end{cases}$$

where $(x_1, x_2)$ corresponds to the position of the cart, $x_3$ to its heading and $x_4$ to its speed.

1) Provide a controller based on a feedback linearization to make the cart follows the Lissajou trajectory:

$$\mathbf{y}_d(t) = 10 \cdot \begin{pmatrix} \cos t \\ \sin 3t \end{pmatrix}.$$

Illustrate the behavior of your controller in MATLAB. Start with the file `ex_sliding.m`.

2) Implement now a sliding mode controller which makes the cart following to desired Lissajou trajectory. Compare with Question 1.

---

EXERCISE 2.9.– *Group of robots*

Consider a group of $m = 20$ carts the motion of which is described by the state equation

$$\begin{cases} \dot{x}_1 &= x_4 \cos x_3 \\ \dot{x}_2 &= x_4 \sin x_3 \\ \dot{x}_3 &= u_1 \\ \dot{x}_4 &= u_2 \end{cases}$$

where $(x_1, x_2)$ corresponds to the position of the cart, $x_3$ to its heading and $x_4$ to its speed.

1) Provide a controller for each of these robots so that the $i$th robot follows the trajectory

$$\begin{pmatrix} \cos(at + \frac{2i\pi}{m}) \\ \sin(at + \frac{2i\pi}{m}) \end{pmatrix}.$$

where $a = 0.1$. As a consequence, after the initialization step, all robots are uniformly distributed on the unit circle, turning around the origin.

2) By using a linear transformation of the unit circle, change the controllers for the robots so that all robots stay on a moving ellipse with the first axis of length $20 + 15 \cdot \sin(at)$ and the second axis of length 20. Moreover, we make the ellipse rotating by choosing an angle for the first axis of $\theta = at$. Illustrate the behavior of the controlled group using MATLAB. Start with the file `ex_group.m`.

---

EXERCISE 2.10.– *Convoy*

Let us consider one robot $\mathcal{R}_A$ described by the following state equations:

$$\begin{cases} \dot{x}_a &= v_a \cos\theta_a \\ \dot{y}_a &= v_a \sin\theta_a \\ \dot{\theta}_a &= u_{a1} \\ \dot{v}_a &= u_{a2} \end{cases}$$

where $v_a$ is the speed of $\mathcal{R}_A$ the robot, $\theta_a$ its orientation and $(x_a, y_a)$ the coordinates of its center.

1) As for Exercise 2.3, propose a controller for $\mathcal{R}_A$ that allows to follow the trajectory:

$$\begin{cases} \hat{x}_a(t) &= L_x \sin(\omega t) \\ \hat{y}_a(t) &= L_y \cos(\omega t) \end{cases}$$

with $\omega = 0.1$, $L_x = 20$ and $L_y = 5$. Illustrate the behavior of the control in MATLAB with a sampling time $dt = 0.03\,\text{sec}$. Start with the file `ex_convoy.m`.

2) We want that $m = 6$ other robots with the same state equations follow this robot taking exactly the same path. The distance between two robots should be $d = 5m$. To achieve this goal, we propose to save every $ds = 0.1m$ the value of the state of $\mathcal{R}_A$ and to communicate this information to the $m$ followers. To synchronize the time with the traveled distance. For this, we propose to add a new state variable $s$ to $\mathcal{R}_A$ which corresponds to the curvilinear value that could have been measured by a virtual odometer. Each time the distance $ds$ has been measured by the virtual odometer, $s$ is initialized to zero and the value for the state of $\mathcal{R}_A$ is broadcast. Simulate the principle in MATLAB.

---

EXERCISE 2.11.– *Reaction wheel pendulum*

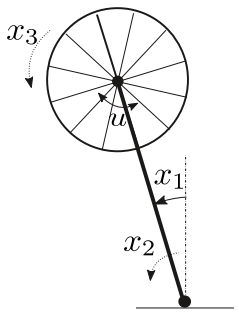


Figure 2.19: Reaction wheel pendulum

The Reaction Wheel Pendulum [SPO 01], as shown in Figure 2.19, is a physical pendulum with a disk attached to the end. The disk can rotate and is actuated by a motor. The coupling torque generated by the angular acceleration of the disk can be used to actively control the pendulum. We

assume that the system can be described by the following equations

$$\begin{cases} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= a \cdot \sin(x_1) - b \cdot u \\ \dot{x}_3 &= -a \cdot \sin(x_1) + c \cdot u \end{cases}$$

where $x_1$ is the pendulum angle, $x_2$ is its angular velocity, $x_3$ is the disk angular velocity and $u$ is the motor torque input. The parameters are taken as

$$a = 10, b = 1, c = 2$$

and depend on the mass, inertia, and the dimensions of the system.

1) Simulate the system with an initial condition of $\mathbf{x} = (1\,0\,0)^{\mathrm{T}}$. Start with the file `ex_reaction.m`.

2) Taking as an output $y = x_1$, stabilize the pendulum at the top (*i.e.*, $x_1 = 0$). Explain why the wheel never stops.

3) Taking as an output of the form

$$y = \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3$$

where $\alpha_1, \alpha_2, \alpha_3$ should be choosen accordingly, stabilize the pendulum at the top (*i.e.*, $x_1 = 0$) with a motionless wheel.

---

# Chapter 3

# Model-free control

When we implement a controller for a robot and perform the initial tests we rarely succeed on the first try, which leads us to the problem of debugging. It might be that the compass is subject to electromagnetic disturbances, that it is placed upside-down, that there is a unit conversion problem in the sensors, that the motors are saturated or that there is a sign problem in the equations of the controller. The problem of debugging is a complex one and it is wise to respect the *continuity principle*: each step in the construction of the robot must be of reasonable size and has to be validated before pursuing construction. Thus, for a robot, it is desirable to implement a simple intuitive controller that is easy to debug before setting up a more advanced one. This principle can not always be applied. However, if we have a good *a priori* understanding of the control law to apply, then such a continuity principle can be followed. Among mobile robots for which a pragmatic controller can be imagined, we can distinguish at least two sub-classes:

- *vehicle-robots*. These are systems built by man to be controlled by man such as the bicycle, the sailboat, the car, etc. We will try to copy the control law used by humans and transform it into an algorithm ;
- *biomimetic robots*. These robots are inspired by the movement of human beings. We have been able to observe them for long periods of time and deduce the strategy developed by nature to design its control law. This is the *biomimetic* approach (see for example [BOY 06]). We do not include walking robots in this category because, even though we all know how to walk, it is near to impossible to know which control law we use for it. Thus, designing a control law for walking robots [CHE 07] can not be done without a complete mechanical modeling of walking and without using any theoretical automatic control methods such as those evoked in the previous chapter.

For these two classes, we often do not have simple and reliable models available (this is the case for example of the sailboat or the bicycle). However, the strong understanding we have of them will allow us to build a robust control law.

The aim of this chapter is to show, using several examples, how to design such control laws. These will be referred to as *mimetic control* (we are trying to imitate humans or animals) or *model-free control* (we do not use the state equations of the robot to design the controller). Although model-free approaches have been largely explored in theory (see for instance [FLI 13]), here we will use the intuition we have of the functioning of our robot as much as possible.

## 3.1 Model-free control of a robot cart

In order to illustrate the principle of model-free control, let us consider the case of a robot cart described by the equations:

$$\begin{cases} \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= u_2 \\ \dot{v} &= u_1 - v \end{cases}$$

This model can be used for simulation, but not for obtaining the controller.

### 3.1.1 Proportional heading and speed controller

We will now propose a simple controller for this system by using our intuition about the system. Let us take $\widetilde{\theta} = \theta_d - \theta$ where $\theta_d$ is the desired heading and $\widetilde{v} = v_d - v$ where $v_d$ is the desired speed.

- For speed control, we take:

  $$u_1 = a_1 \tanh \widetilde{v}$$

  where $a_1$ is a constant representing the maximum acceleration (in absolute value) that the motor is able to deliver. The hyperbolic tangent *tanh* (see Figure 3.1) is used as saturation function. Let us recall that :

  $$\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3.1}$$

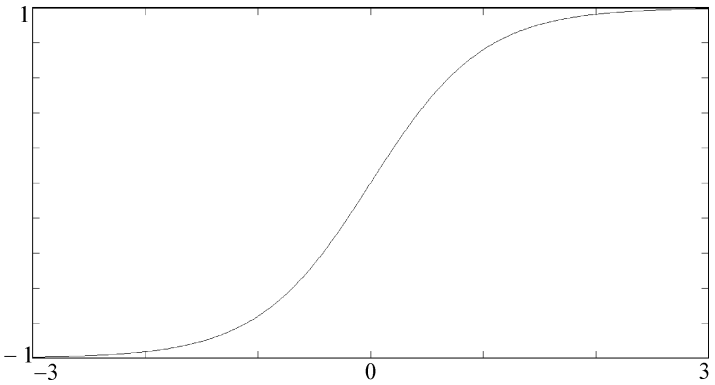


Figure 3.1: Hyperbolic tangent function used as saturation function

- For the heading control, we take:

  $$u_2 = a_2 \cdot \text{sawtooth}(\widetilde{\theta})$$

In this last formula, *sawtooth* corresponds to the sawtooth function defined by:

$$\text{sawtooth}(\widetilde{\theta}) = 2\text{atan}\left(\tan\frac{\widetilde{\theta}}{2}\right) = \text{mod}(\widetilde{\theta}+\pi, 2\pi) - \pi \tag{3.2}$$

Let us note that for numerical reasons, it is preferable to use the expression containing the *modulus* function (`mod` in MATLAB). As illustrated in Figure 3.2, the function corresponds to an error in heading. The interest in taking an error $\widetilde{\theta}$ filtered by the *sawtooth* function is to avoid the problem of the $2k\pi$ modulus: we would like a $2k\pi$ to be considered non-zero.

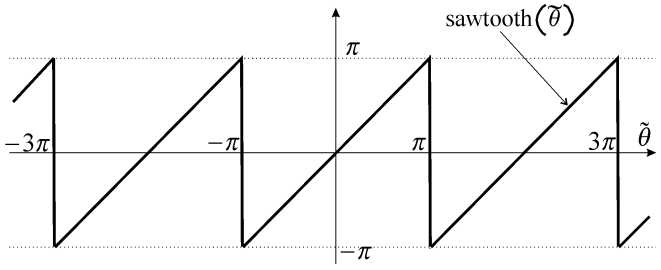


Figure 3.2: *Sawtooth* function used to avoid the jumps in the heading control

We may summarize this controller by:

$$\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} a_1 \cdot \tanh(v_d - v) \\ a_2 \cdot \text{sawtooth}(\theta_d - \theta) \end{pmatrix}$$

We will thus perform heading control. This model-free control, which works very well in practice, does not need to use the state equations of the robot. It is based on the understanding that we have of the dynamics of the system and recalls our wireless operation method of the cart robot. It has two parameters $a_1$ and $a_2$ that are easy to set ($a_1$ represents the propelling power and $a_2$ the directional disturbances). Finally, this controller is easy to implement and to debug.

### 3.1.2 Proportional-derivative heading controller

For many robots, a proportional controller creates oscillations and it might prove to be necessary to add an damping or derivative term. This is the case for underwater exploration robots (of type ROV, *Remotely Operated Vehicle*) which are meant to stabilize above the zone of interest. Underwater torpedo robots do not have this oscillation problem given their control surfaces that stabilize the heading while in movement. If the heading if constant, such a proportional-derivative controller is given by:

$$u_2 = a_2 \cdot \text{sawtooth}(\theta_d - \theta) + b_2\dot{\theta}$$

The quantity $\theta$ may be obtained by a compass, for example. As for $\dot{\theta}$, it is generally obtained by a gyro. Low-cost robots do not always have a gyro available and we must try to approximate $\dot{\theta}$ from measurements of $\theta$. However, a compass might jump by $2\pi$ for small variations in heading. This is the case for instance when a compass returns an angle within the interval $[-\pi, \pi]$ and the

heading varies around $(2k+1)\pi$. In this case, an approximation of $\dot{\theta}$ must be obtained and this approximation has to be insensitive to these jumps. Let us denote by :

$$\mathbf{R}_t = \begin{pmatrix} \cos\theta(t) & -\sin\theta(t) \\ \sin\theta(t) & \cos\theta(t) \end{pmatrix}$$

the rotation matrix corresponding to the heading $\theta(t)$ of the robot (let us note that this matrix is insensitive to jumps of $2k\pi$). Notice that :

$$\mathbf{R}_t^{\mathrm{T}}\dot{\mathbf{R}}_t = \begin{pmatrix} 0 & -\dot{\theta}(t) \\ \dot{\theta}(t) & 0 \end{pmatrix}.$$

This relation can be seen as a two-dimensional version of Relation (1.1) on page 11, but can also be directly obtained by using the expression of $\mathbf{R}_t$. Thus, an Euler integration of the rotation matrix:

$$\mathbf{R}_{t+dt} = \mathbf{R}_t + dt\dot{\mathbf{R}}_t$$

translates to:

$$\mathbf{R}_{t+dt} = \mathbf{R}_t + dt \cdot \mathbf{R}_t \begin{pmatrix} 0 & -\dot{\theta}(t) \\ \dot{\theta}(t) & 0 \end{pmatrix} = \mathbf{R}_t \left( \mathbf{I} + dt \begin{pmatrix} 0 & -\dot{\theta}(t) \\ \dot{\theta}(t) & 0 \end{pmatrix} \right).$$

Therefore:

$$\begin{aligned} dt \begin{pmatrix} 0 & -\dot{\theta}(t) \\ \dot{\theta}(t) & 0 \end{pmatrix} &= \mathbf{R}_t^{\mathrm{T}}\mathbf{R}_{t+dt} - \mathbf{I} \\ &= \begin{pmatrix} \cos(\theta(t+dt) - \theta(t)) & -\sin(\theta(t+dt) - \theta(t)) \\ \sin(\theta(t+dt) - \theta(t)) & \cos(\theta(t+dt) - \theta(t)) \end{pmatrix} \\ &\quad - \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \end{aligned}$$

Let us take in this matrix equation the scalar equation corresponding to the second row and first column. We obtain:

$$\dot{\theta}(t) = \frac{\sin(\theta(t+dt) - \theta(t))}{dt}.$$

The proportional-derivative heading controller can therefore be written as:

$$u_2(t) = a_2 \cdot \mathrm{sawtooth}(\theta_d - \theta(t)) + b_2 \frac{\sin(\theta(t) - \theta(t-dt))}{dt}$$

which will be insensitive to jumps of $2\pi$.

## 3.2 Skate car

Let us consider the skating vehicle [JAU 10] represented on Figure 3.3.

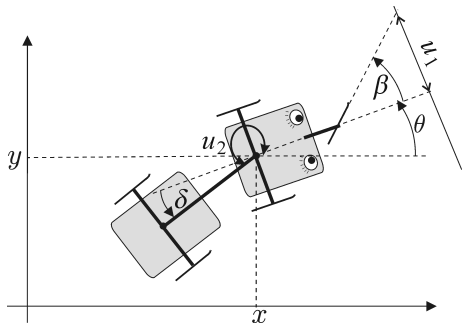


Figure 3.3: Skating robot moving like a snake

This vehicle that we will refer to as *skate car* is purely imaginary. It is designed move on a frozen lake and stands on five ice skates. This system has two inputs: the tangent $u_1$ of the angle $\beta$ of the front skate (we have chosen the tangent as input in order to avoid the singularities) and $u_2$ the torque exerted at the articulation between the two carts and corresponding to the angle $\delta$. The thrust therefore only comes from the torque $u_2$ and recalls the propulsion mode of a snake or an eel [BOY 06]. Any control over $u_1$ will therefore not bring any energy to the system, but indirectly participates in the propulsion by generating waves. In this paragraph, we will propose model in the form of a state for simulating the system. Concerning the control law, the existing general methods cannot deal with this kind of system and it is necessary to take into account the physics of the problem. We will therefore propose a mimetic control law that allows to obtain an efficient controller.

### 3.2.1 Model

Let us try to obtain state equations capable of representing the dynamics of the system in order to simulate our system. The state variables are chosen to be $\mathbf{x} = (x, y, \theta, v, \delta)$, where $x, y, \theta$ correspond to the position of the front cart, $v$ represents the speed of the center of the front sled axle and $\delta$ is the angle between the two carts. The angular speed of the front sled is given by:

$$\dot{\theta} = \frac{v_1 \sin\beta}{L_1} \tag{3.3}$$

where $v_1$ is the speed of the front skate and $L_1$ is the distance between the front skate and the center of the front sled axle. However:

$$v = v_1 \cos\beta$$

and therefore:

$$\dot{\theta} = \frac{v \tan\beta}{L_1} = \frac{v u_1}{L_1}. \tag{3.4}$$

Viewed from the rear sled, everything is as if there was a virtual skate in the middle of the front sled axle, moving together with it. Thus, by recalling Formula [3.3], the angular speed of the rear

sled is:

$$\dot{\theta} + \dot{\delta} = -\frac{v \sin \delta}{L_2}$$

where $L_2$ is the distance between the centers of the axles. And therefore:

$$\dot{\delta} = -\frac{v \sin \delta}{L_2} - \dot{\theta} \overset{[3.4]}{=} -\frac{v \sin \delta}{L_2} - \frac{v u_1}{L_1}. \tag{3.5}$$

Following the theorem of kinetic energy, the temporal derivative of kinetic energy is equal to the sum of the powers supplied to the system, in other words:

$$\frac{d}{dt}\left(\frac{1}{2} m v^2\right) = \underbrace{u_2 \cdot \dot{\delta}}_{\text{engine power}} - \underbrace{(\alpha v) \cdot v}_{\text{dissipated power}} \tag{3.6}$$

where $\alpha$ is the coefficient of viscous friction. For reasons of simplicity, we will assume here that the force of friction of equal to $\alpha v$, which is the same as assuming that only the front sled is braking. We therefore have:

$$m v \dot{v} \overset{[3.6]}{=} u_2 \cdot \dot{\delta} - \alpha v^2 \overset{[3.5]}{=} u_2 \cdot \left(-\frac{v \sin \delta}{L_2} - \frac{v u_1}{L_1}\right) - \alpha v^2$$

and

$$m \dot{v} = u_2 \cdot \left(-\frac{\sin \delta}{L_2} - \frac{u_1}{L_1}\right) - \alpha v. \tag{3.7}$$

The system can be described by the following state equations:

$$\begin{cases} \dot{x} & = & v \cos \theta \\ \dot{y} & = & v \sin \theta \\ \dot{\theta} & \overset{[3.4]}{=} & v u_1 \\ \dot{v} & \overset{[3.7]}{=} & -\left(u_1 + \sin \delta\right) u_2 - v \\ \dot{\delta} & \overset{[3.5]}{=} & -v \left(u_1 + \sin \delta\right) \end{cases} \tag{3.8}$$

where, for reasons of simplicity, the coefficients (mass $m$, coefficient of viscous friction $\alpha$, inter-axle distances $L_1, L_2$, etc.) have been given unit values. This system could be made control-affine (refer to Equation [2.6]) by adding an integrator in front of $u_1$, however the feedback linearization method cannot be applied due to the numerous singularities. Indeed, it can be easily shown that when the speed $v$ is zero (easy to avoid) or when $\dot{\delta} = 0$ (which necessarily happens regularly), we have a singularity. A *biomimetic* controller that imitates the propulsion of the snake or the eel might be feasible.

### 3.2.2 Sinusoidal control

By trying to imitate the control strategy of an undulating snake's movement, we choose $u_1$ of the form:

$$u_1 = p_1 \cos(p_2 t) + p_3$$

where $p_1$ is the amplitude, $p_2$ the pulse and $p_3$ the bias. We choose $u_2$ such that the propelling torque is a motor torque, in other words $\dot{\delta} u_2 \geq 0$. Indeed, $\dot{\delta} u_2$ corresponds to the power supplied to the robot that is transformed into kinetic energy. If $u_2$ is bounded by the interval $[-p_4, p_4]$, we choose a bang-bang type controller for $u_2$ of the form:

$$u_2 = p_4 \cdot \text{sign}(\dot{\delta})$$

which is equivalent to exerting maximum propulsion. The chosen state feedback controller is therefore:

$$\mathbf{u} = \begin{pmatrix} p_1 \cos(p_2 t) + p_3 \\ p_4 \ \text{sign}\left(-v\left(u_1 + \sin\delta\right)\right) \end{pmatrix}.$$

The parameters of the controller remain to be determined. The bias parameter $p_3$ allows it to direct its heading. The power of the motor torque gives us $p_4$. The parameter $p_1$ is directly linked to the amplitude of the oscillation created during movement. Finally, the parameter $p_2$ gives the frequency of the oscillations. The simulations can help us to set the parameters $p_1$ and $p_2$ correctly. Figure 3.4 illustrates two simulations in which the robot begins with an almost zero speed. In the simulation on top, the bias $p_3$ is equal to zero. In the bottom simulation, $p_3 > 0$.

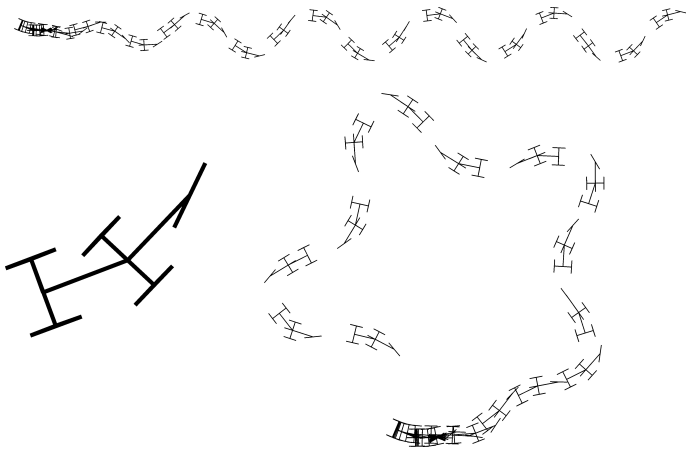

Figure 3.4: Various simulations illustrate the control law for the skating robot

Figure 3.5 represents the advance as a function of time. It is clear that the power supplied by the engine is very strong at startup whereas in cruising regime, it is under-utilized. Such a controller forces us to oversize our engine. It would be to our advantage to have a thrust as constant as possible. The script in `snake.m` contains an implementation of this control law.

### 3.2.3 Maximum thrust control

The propulsion of the robot is done by the thrust $u_2 \cdot \dot{\delta} = -v\,(u_1 + \sin\delta) \cdot u_2$ and therefore by the engine that generates the torque $u_2$. In order to move as fast as possible, for a given motor, the engine should supply a maximum amount of power denoted by $\bar{p}$ which will be transformed into kinetic energy. Thus:

$$\underbrace{-v\,(u_1 + \sin\delta)}_{\dot{\delta}} \cdot u_2 = \bar{p}.$$

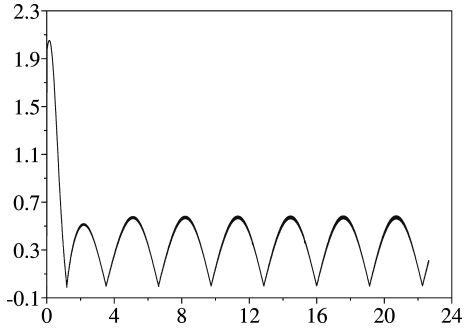


Figure 3.5: Thrust supplied by the engine $u_2$

There are therefore several torques $(u_1, u_2)$ capable of supplying the desired power $\bar{p}$. We will therefore choose for $u_2$ the form:

$$u_2 = \varepsilon \cdot \bar{u}_2 \text{ with } \varepsilon = \pm 1$$

where $\varepsilon(t)$ is a square wave and $\bar{u}_2$ is a constant. This choice for $u_2$ may be to bound the engine torque and thereby limit the mechanical load. If we choose the frequency of $\varepsilon$ too low, the power supplied will be respected, but the front cart will collide with the rear cart. In the borderline case where $\varepsilon$ is constant, we can observe, through the simulation, the first cart roll up to the second (which means that $\delta$ increases to infinity). We obtain, by isolating the orientation $u_1$ of the front skate:

$$u_1 = -\left(\frac{\bar{p}}{v\varepsilon\bar{u}_2} + \sin\delta\right)$$

The maximum thrust controller is therefore given by:

$$\mathbf{u} = \begin{pmatrix} -\left(\frac{\bar{p}}{v\varepsilon\bar{u}_2} + \sin\delta\right) \\ \varepsilon\bar{u}_2 \end{pmatrix}. \tag{3.9}$$

Thus, with this controller, not only do we always thrust in the correct direction through $u_2$ but we can adjust the direction $u_1$ in order for the torque supplied by $u_2$ to translate into a maximum thrust $\bar{p}$. Now we only need to act on $\varepsilon$ (which, as we recall, is a square wave equal to $\pm 1$) and on the power $\bar{p}$. The duty cycle of the signal $\varepsilon(t)$ will allow us to direct our orientation and its frequency will give us the amplitude of the oscillations for the robot's path. As for $\bar{u}_2$, it allows us to control the

average speed of the robot. In the simulation, this controller turns out indeed to be more efficient than the sinusoidal controller. Figure 3.6 shows the angle of the skate $\beta$ in function of time, once the cruising regime has been reached. Let us note that the angle of the front skate $\beta = \text{atan}(u_1)$ makes discontinuities appear.

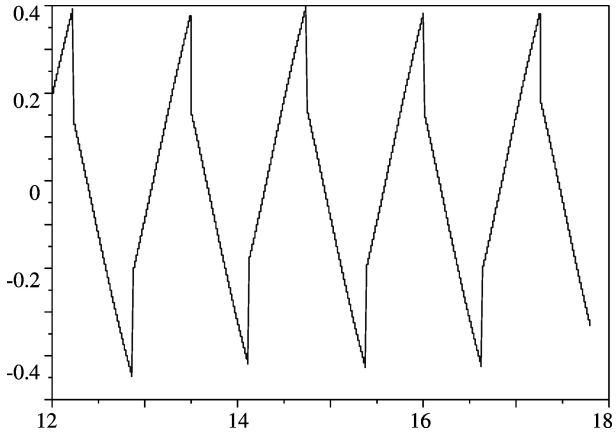


Figure 3.6: Evolution of the front skate angle $\beta$ in cruising regime

### 3.2.4 Simplification of the fast dynamics

The state equations for the *snakeboard* contain numerous singularities and we would like to simplify them here. However, in our system, we have two interfering dynamics: one that is slow (representing the smooth evolution of the state variables) and one that is fast (rated by $\varepsilon$) which creates the undulation. The idea, relatively standard in automatic control, is to average these values in a way to make the fast dynamics disappear. We can find this idea in PWM-controlled (*Pulse Width Modulation*) DC engines.

Let us consider a high-frequency square wave signal $\varepsilon(t)$. Its temporal average $\bar{\varepsilon}$ is called the *duty cycle*. This duty cycle is set to vary very slowly in time. The *temporal average* operator is linear (just like the mathematical expectation). For example:

$$\overline{2\varepsilon_1(t) - 3\varepsilon_2(t)} = 2\bar{\varepsilon}_1(t) - 3\bar{\varepsilon}_2(t).$$

On the other hand, for a nonlinear function $f$, we cannot write $\overline{f(\varepsilon)} = f(\bar{\varepsilon})$. For instance, $\overline{\varepsilon^{-1}} \neq \bar{\varepsilon}^{-1}$. However, we will have $\overline{\varepsilon^{-1}} = \bar{\varepsilon}$ if $\varepsilon(t) \in \{-1, 1\}$. This comes from the fact that the signals $\varepsilon$ and $\varepsilon^{-1}$ are equal in such a case. If $a(t)$ and $b(t)$ are slowly varying signals in time, we will also have:

$$\overline{a(t)\ \varepsilon_1(t) + b(t)\ \varepsilon_2(t)} = a(t)\ \bar{\varepsilon}_1(t) + b(t)\ \bar{\varepsilon}_2(t).$$

We will try to apply these approximations to the case of the *snakeboard* in order to eliminate the fast dynamics. By recalling the state equations in [3.8] and by injecting control law [3.9], we obtain

a feedback system described by the following state equations:

$$\begin{cases} \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= -\frac{\bar{p}}{\varepsilon\bar{u}_2} - v\sin\delta \\ \dot{v} &= \frac{\bar{p}}{v} - v \\ \dot{\delta} &= \frac{\bar{p}}{\varepsilon\bar{u}_2} \end{cases}$$

Recall that we can act on the constants $\bar{p}, \bar{u}_2$ and on the square wave signal $\varepsilon = \pm 1$ that we will here consider to be high-frequency and with a duty cycle of $\bar{\varepsilon}$. We can approximate:

$$\frac{\bar{p}}{\varepsilon\bar{u}_2} = \frac{\varepsilon\bar{p}}{\bar{u}_2} \overset{\text{(by linearization)}}{\simeq} \bar{\varepsilon}\frac{\bar{p}}{\bar{u}_2}.$$

The system thus becomes:

$$\begin{cases} \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= -\bar{p}\bar{q} - v\sin\delta \\ \dot{v} &= \frac{\bar{p}}{v} - v \\ \dot{\delta} &= \bar{p}\bar{q} \end{cases}$$

which now only has two inputs: $\bar{p}$ and $\bar{q} = \frac{\bar{\varepsilon}}{\bar{u}_2}$. Let us try to control the inputs $\theta$ and $v$ by using feedback linearization method. Note that although this system is not affine in its inputs ($\bar{p}$ and $\bar{q}$), the method can be applied because, as we will see below, the necessary inversion is possible here. For this, let us define two new inputs $v_1, v_2$ such that:

$$\begin{cases} v_1 &= -\bar{p}\bar{q} - v\sin\delta \\ v_2 &= \frac{\bar{p}}{v} - v \end{cases}$$

By inverting this system relative to the inputs, we obtain:

$$\begin{array}{rcl} \bar{p} &=& v\,(v_2 + v) \\ \bar{q} &=& -\frac{v_1 + v\sin\delta}{v(v_2+v)} \end{array}$$

Thus, the feedback linearized system is:

$$\begin{cases} \dot{\theta} &= v_1 \\ \dot{v} &= v_2 \end{cases}$$

A proportional controller is therefore sufficient. We will take one that places the poles at $-1$, in other words:

$$\begin{cases} v_1 &= w_1 - \theta + \dot{w}_1 \\ v_2 &= w_2 - v + \dot{w}_2 \end{cases}$$

Let us summarize the control law in its entirety. Its setpoints are $w_1, w_2$ which correspond to the

desired heading and speed. It is given by the following table:

$$
\begin{aligned}
&\bar{p} = v\,(w_2 + \dot{w}_2)\\
&\bar{q} = -\frac{w_1 - \theta + \dot{w}_1 + v\sin\delta}{v\,(w_2 + \dot{w}_2)}\\
&\bar{\varepsilon} = \bar{q}\cdot\bar{u}_2 \text{ (choose } \bar{\varepsilon} \in [-1,1]\,)\\
&\varepsilon: \text{ duty cycle } \bar{\varepsilon} \text{ and frequency slot } \infty\\
&\mathbf{u} = \begin{pmatrix} -\left(\frac{\bar{p}}{v\varepsilon\bar{u}_2} + \sin\delta\right)\\ \varepsilon\bar{u}_2 \end{pmatrix}
\end{aligned}
$$

The adjustment parameter $\bar{u}_2$ involved in this controller is quite delicate to set. We need to choose $\bar{u}_2$ small enough to have $\bar{\varepsilon} \in [-1,1]$. But it must not be too close to zero in order for $u_1$ to not be too large (which would cause too significant front skate movements). This variable $\bar{u}_2$ influences the necessary distribution between the torque (through $u_2$) and the movement (through $u_1$) for generating power. Let us finally note that this latter control law, supposed to be more efficient, uses state equations of the system in its design and it is therefore difficult to call it *model-free*.

## 3.3 Sailboat

### 3.3.1 Problem

Let us recall the principles of model-free control and try to adapt it to a line-tracking controller for our sailboat. Here we will consider a sailboat whose sheet length is variable, but not directly the angle of the sail as it was the case until now (see [2.11] on page 53). This robot has two inputs which are the angle of the rudder $u_1 = \delta_r$ and the maximum angle of the sail $u_2 = \delta_s^{\max}$ (equivalently $u_2$ corresponds to the length of the sheet). We will try to make the robot follow a line which passes through points **a** and **b** (see Figure 3.7).

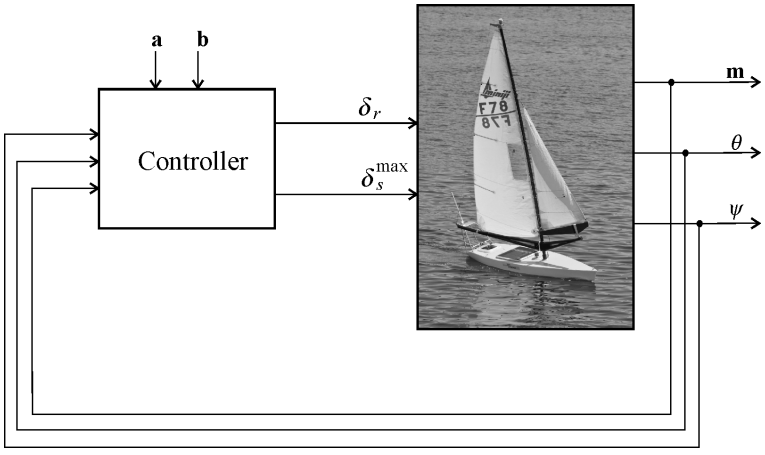


Figure 3.7: Feedback control of the sailing robot

This problem is influenced by the control strategies of the VAIMOS robot [GOR 11] of IFREMER,

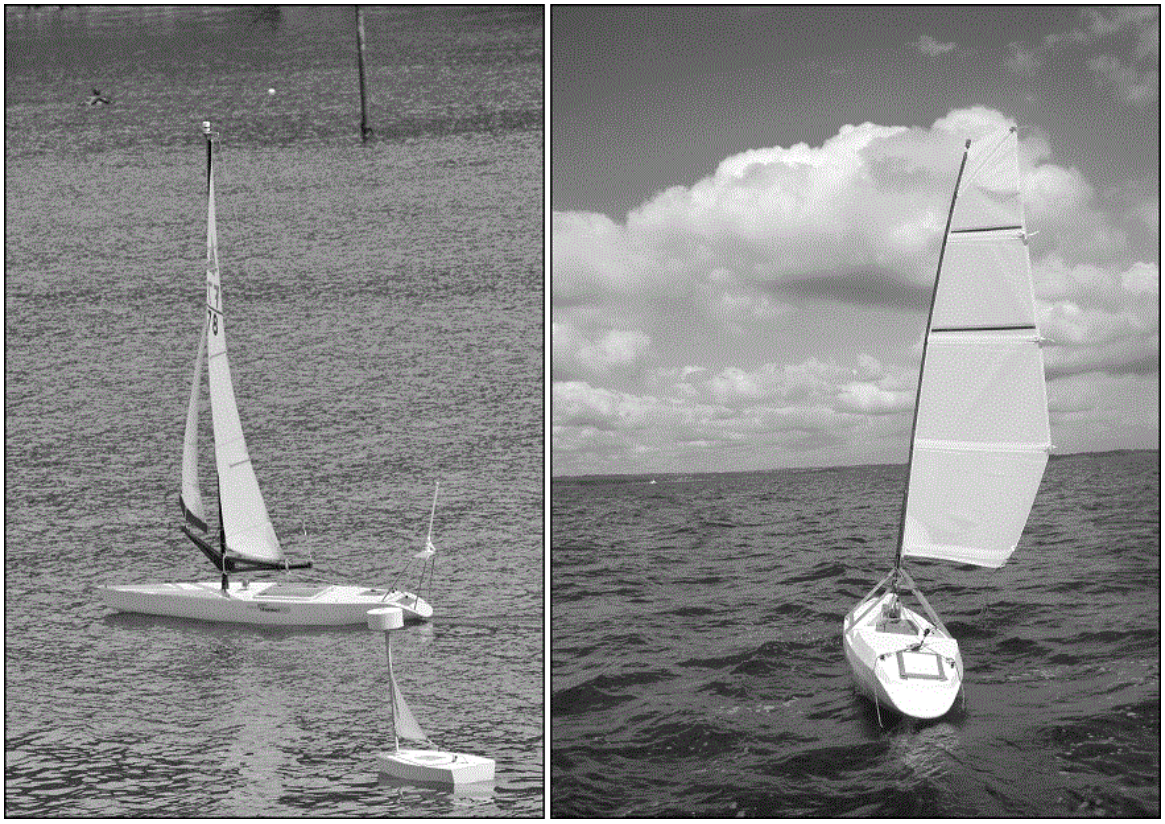

Figure 3.8: Left : the VAIMOS sailing robot of IFREMER (in the background) and the Optimousse robot from the ENSTA Bretagne ; right : robot of the Ecole Navale (Naval School) following a line

the sailing boat of the ERWAN naval school and the Optimousse robot from the ENSTA-Bretagne (see Figure 3.8).

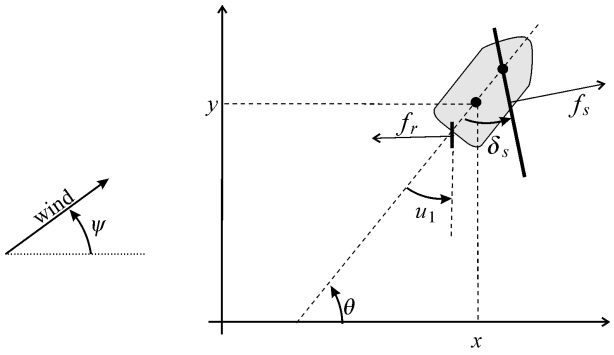


Figure 3.9: Variables used in the state equations of the robot

As illustrated by Figure 3.9, we will denote by $(x, y, \theta)$ the posture of the boat, by $v$ its advancing speed, by $\omega$ its angular speed, by $f_s$ the force of the wind on the sail, by $f_r$ the force of the water on the rudder, by $\delta_s$ the angle of the sail and by $\psi$ the angle of the wind.

### 3.3.2 Controller

We will now try to find a controller that will enable the robot to track a line. The robot will be equipped with three sensors: a compass that gives us the heading $\theta$, a weathervane that measures the angle of the wind $\psi$ and a GPS that returns the position $\mathbf{m}$ of the boat. The robot will also be equipped with two actuators: a servo-motor that controls the angle of the rudder $\delta_r$ and a stepper motor that sets the length of the sheet and therefore the maximum angle $\delta_s^{\max}$ of the sail (i.e. $|\delta_s| \leq \delta_s^{\max}$). As for the controller, its setpoint is the line $\mathbf{ab}$ to track and it has a binary variable $q \in \{-1, 1\}$ called the *hysteresis* which will be used for close hauled sailing. This controller will have few parameters which will also be easy to control. Among these parameters, we find the maximum rudder angle $\delta_r^{\max}$ (typically $\delta_r^{\max} = \frac{\pi}{4}$), the cutting distance $r$ (i.e. we would like the distance to the line to be always smaller than $r$), the close haul angle $\zeta$ (typically $\zeta = \frac{\pi}{4}$), and the angle of the sail in crosswind $\beta$ (typically $\beta = 0.3$ rad). We propose the following controller, taken from the article [JAU 12], that we will explain later:

| | **Controller** in:$\mathbf{m}, \theta, \psi, \mathbf{a}, \mathbf{b}$; out:$\delta_r, \delta_s^{\max}$; inout:$q$ |
|---|---|
| 1 | $e = \det\left(\frac{\mathbf{b}-\mathbf{a}}{\|\mathbf{b}-\mathbf{a}\|}, \mathbf{m}-\mathbf{a}\right)$ |
| 2 | if $\|e\| > r$then $q = \text{ sign}\,(e)$ |
| 3 | $\varphi = \text{ angle}\,(\mathbf{b}-\mathbf{a})$ |
| 4 | $\bar{\theta} = \varphi - \text{ atan}\left(\frac{e}{r}\right)$ |
| 5 | if $\cos\left(\psi - \bar{\theta}\right) + \cos\zeta < 0$ |
| 6 | or $(\|e\| < r$ and $(\cos(\psi-\varphi) + \cos\zeta < 0))$ |
| 7 | *then* $\bar{\theta} = \pi + \psi - q\zeta$. |
| 8 | $\delta_r = \frac{\delta_r^{\max}}{\pi}$ sawtooth$(\theta - \bar{\theta})$ |
| 9 | $\delta_s^{\max} = \frac{\pi}{2}\left(\frac{\cos\left(\psi-\bar{\theta}\right)+1}{2}\right)^{\frac{\log\left(\frac{\pi}{2\beta}\right)}{\log(2)}}$ |

The controller has a single state variable which is the binary variable $q \in \{-1, 1\}$. It is for this reason that it appears at the same time as input and output of the algorithm. Let us comment on this algorithm.

**Line 1 (calculation of the algebraic distance)**. We calculate the algebraic distance between the robot and its line. If $e > 0$ the robot is on the left of its line and if $e < 0$, it is on the right. In the formula, the determinant is to be understood in the following way:

$$\det(\mathbf{u}, \mathbf{v}) = u_1 v_2 - v_1 u_2.$$

**Line 2 (update of the hysteresis variable)**. When $|e| > r$, the robot is far from its line and the hysteresis variable $q$ (that memorizes the starboard tack) is allowed to change value. If for instance $e > r$, then $q$ will take the value 1 and will keep it until $e < -r$.

**Line 3 (calculation of the line angle)**. We calculate the angle $\varphi$ of the line to track (see Figure 3.10). In the instruction, angle($\mathbf{u}$) represents the angle made by the vector $\mathbf{u} \in \mathbb{R}^2$ relative to the $Ox$ axis (towards the East). The corresponding function can be found in `angle.m`.

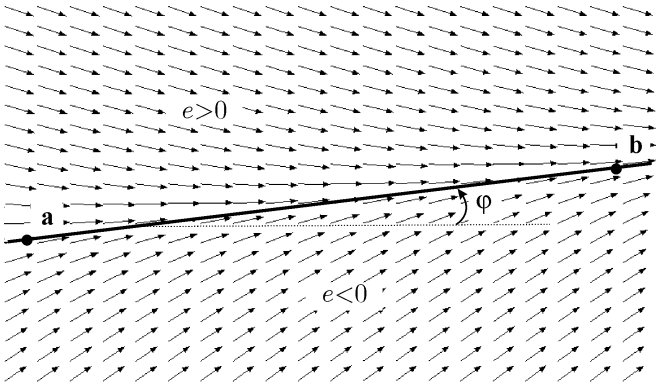


Figure 3.10: Nominal vector field that the robot tries to track, when possible

**Line 4 (calculation of the nominal heading)**. We calculate the nominal angle $\bar{\theta}$ (see Figure 3.10), in other words the one that we would like to have without worrying about the wind. We take:

$$\bar{\theta} = \varphi - \text{atan}\left(\frac{e}{r}\right).$$

This expression for $\bar{\theta}$ translates to an attractive line. When $e = \pm\infty$, we have $\bar{\theta} = \varphi \pm \frac{\pi}{2}$, which means that the robot has a heading that forms an angle of $\frac{\pi}{2}$ with the line. For a distance $e$ corresponding to the cutting distance $r$, i.e. $e = \pm r$, we have $\bar{\theta} = \varphi \pm \frac{\pi}{4}$. Finally, on the line we have $e = 0$ and therefore $\bar{\theta} = \varphi$, which corresponds to a heading with the direction of the line. As illustrated on Figure 3.11a, some directions $\bar{\theta}$ may be incompatible with that of the wind.

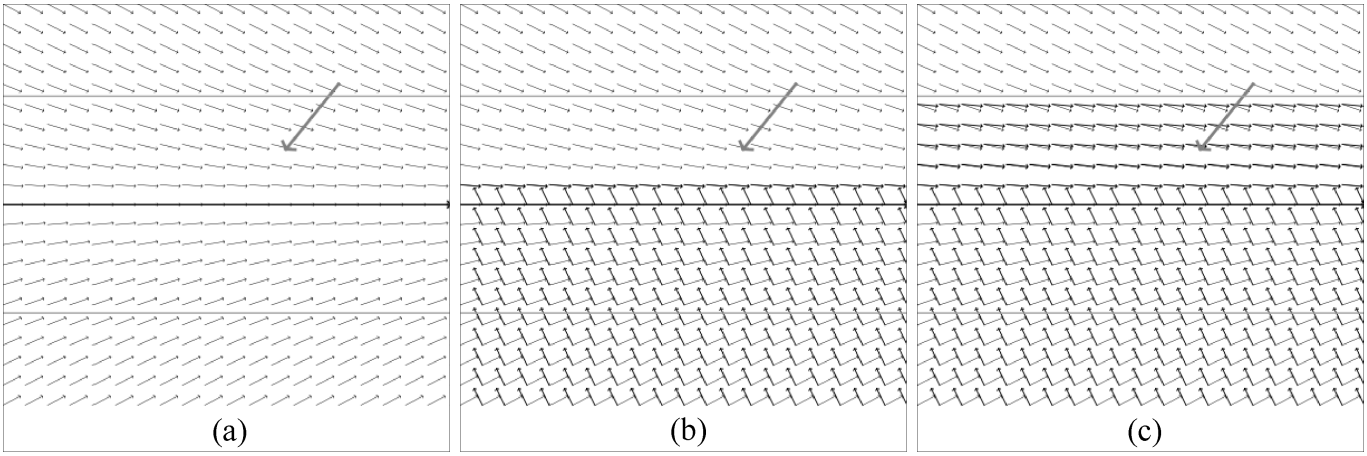


Figure 3.11: (a) The nominal vector field may be incompatible with the wind (here represented by the large arrow) ; (b) Vector field generated by the controller if we remove line 6. The thin arrows correspond to the nominal paths and the bold arrows correspond to the corrected paths ; (c) Vector field generated by the controller with line 6 included.

**Line 5**. When $\cos\left(\psi - \bar{\theta}\right) + \cos\zeta < 0$, the path $\bar{\theta}$ corresponds to a direction that is too close to the wind that the robot is incapable of following (see Figure 3.12).

The heading $\bar{\theta}$ is then impossible to keep. In this case, we need to switch to *close haul* mode, which means that the robot will do everything it can to face the wind, or more formally, the new

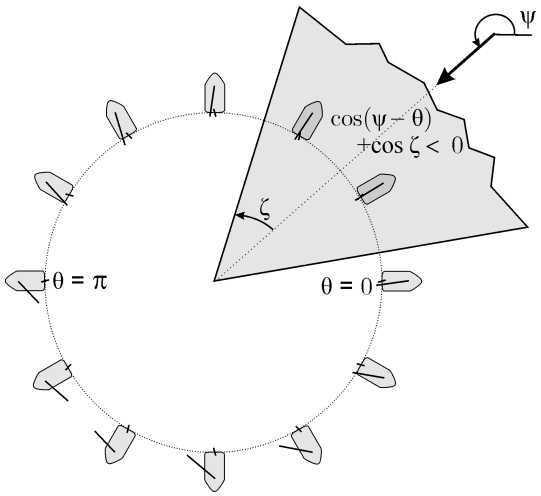


Figure 3.12: Some directions are not possible for the sailboat. These unfeasible directions form the *no-go* zone, represented in grey

direction becomes $\bar{\theta} = \pi + \psi \pm \zeta$ (see line 7). Figure 3.11b represents the corresponding vector field. The thin arrows correspond to the nominal field and the bold arrows represent the corrected field when necessary. In this representation, we have removed the hysteresis effect induced by the variable $q$ (which is means that we always have $q = \text{sign}(e)$).

**Line 6 (keep close hauled strategy).** This instruction implements the so-called *keep close hauled strategy*. If $|e| < r$ or if $\cos(\psi - \varphi) + \cos\zeta < 0$, then the boat is forced to move upwind, even when the heading $\bar{\theta}$ is admissible and this, for reasons of efficiency. This strategy is illustrated on Figure 3.11c. On this figure, we have chosen a close haul angle of $\zeta = \frac{\pi}{3}$ (which corresponds to difficulties moving upwind) and given this, the line is considered to be against the wind.

**Line 7 (close hauled heading)**. The boat is in close haul and we choose $\bar{\theta} = \pi + \psi - q\zeta$ (the wind direction plus or minus the close haul angle $\zeta$). The hysteresis variable $q$ is forced to keep the same point of sail as long as the distance of $r$ to the line is not reached. An illustration of the resulting behavior is represented on Figure 3.13. If the nominal heading can be kept, then it is followed.

**Line 8 (rudder control)**. At this level, the heading to maintain $\bar{\theta}$ has already been chosen and we are trying to follow it using the rudder. We perform a proportional control relative to the error $\theta - \bar{\theta}$. In order to filter out the modulus-$2\pi$ problem, we use the *sawtooth* function (see Formula [3.2]). We thus obtain:

$$\delta_r = \frac{\delta_r^{\max}}{\pi} \cdot \text{sawtooth}(\theta - \bar{\theta})$$

where $\delta_r^{\max}$ is the maximum angle of the rudder (for example $\delta_r^{\max} = 0.5$ rad). The resulting controller is illustrated on Figure 3.14.

**Line 9 (sail control)**. We choose a sail angle $\beta$ (half-open sail) that the sail needs to have in crosswind. This parameter is determined experimentally depending on the sailboat and the steering mode we would like to use. The maximum angle of the sail $\delta_s^{\max}$ is a function of $\psi - \theta$ which is

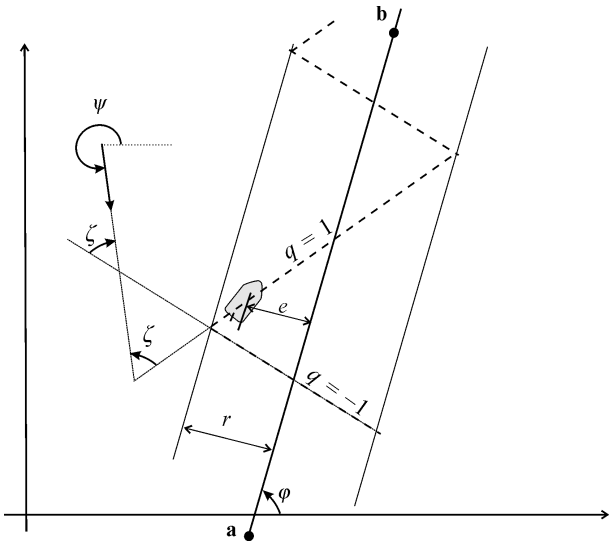


Figure 3.13: Keep close hauled strategy by remaining within the strip centered on the line **ab** with diameter $r$

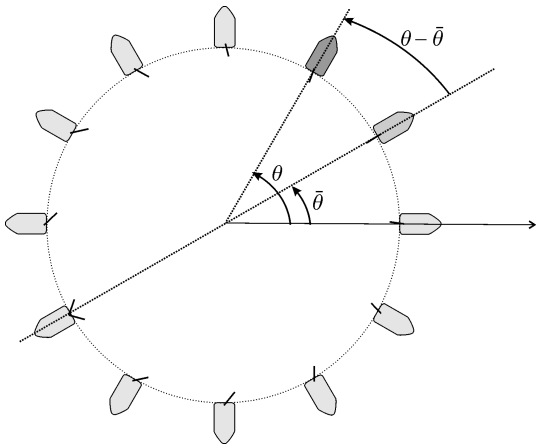


Figure 3.14: Rudder control for the sailing robot

periodic with period $2\pi$. One possible model [JAU 12] is that of the cardioid:

$$\delta_s^{\max} = \frac{\pi}{2} \cdot \left( \frac{\cos(\psi - \theta) + 1}{2} \right)^{\eta}$$

where the parameter $\eta$ is positive. When $\psi = \theta + \pi$, the boat is facing the wind and the model gives us $\delta_s^{\max} = 0$. When $\psi = \theta$, we have $\delta_s^{\max} = \frac{\pi}{2}$, which means that the sail is wide open when the robot is with the wind. The choice of the parameter $\eta$ will be based on the angle of the sail in crosswind, in other words for $\psi = \theta \pm \frac{\pi}{2}$. The equation $\delta_s^{\max} = \beta$ for $\psi = \theta \pm \frac{\pi}{2}$ is translated by:

$$\frac{\pi}{2} \cdot \left( \frac{1}{2} \right)^{\eta} = \beta$$

i.e.:

$$\eta = \frac{\log\left(\frac{\pi}{2\beta}\right)}{\log(2)}$$

The function $\delta_s^{\max}$ is represented on Figure 3.15.

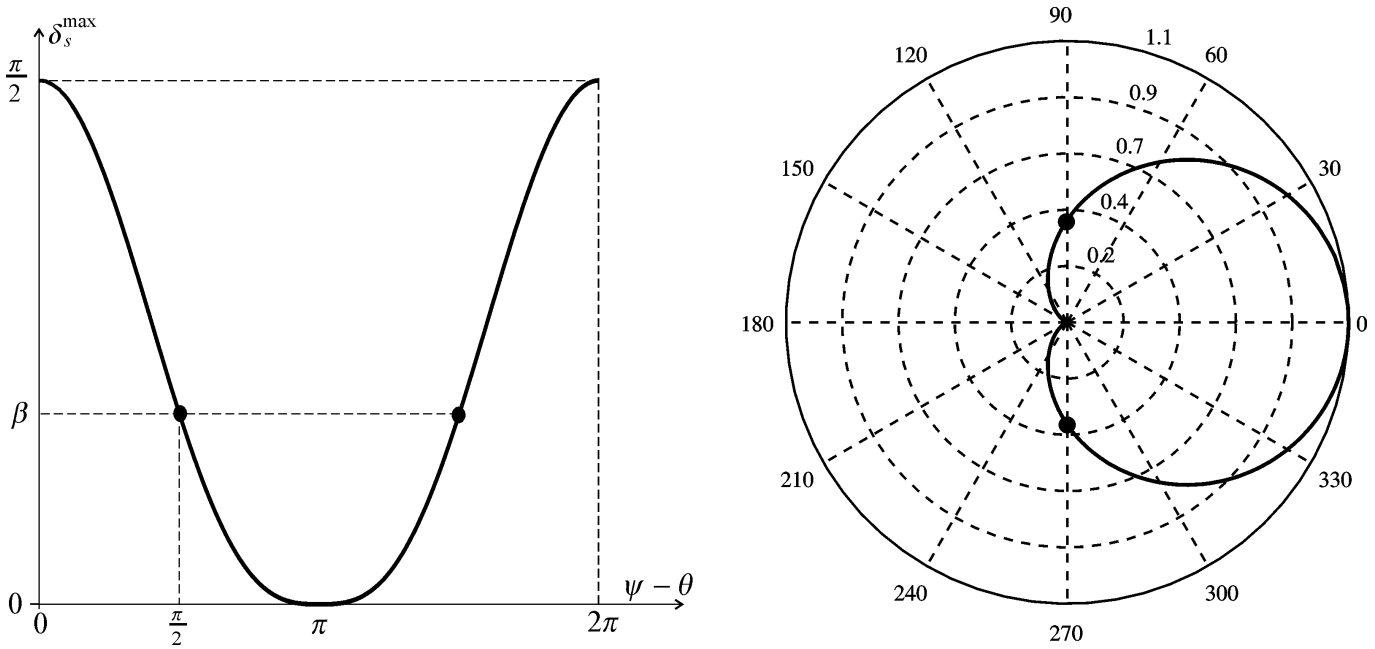


Figure 3.15: Adjusting the maximum sail angle (or the sheet length) ; left : Cartesian representation ; right : polar representation

On the tests that have been carried out, this adjustment of the sail was shown to be efficient and easy to control, given the few number of parameters.

### 3.3.3 Navigation

Once the line tracking has been correctly implemented and validated, a number of lines should be chained together with the aim of performing complex missions (such as for instance connecting two points of the globe). In such a context, a Petri net strategy is well-adapted for representing the discrete state changes [MUR 89]. Figure 3.16 illustrates a Petri net allowing to manage the mission. Before the robot is launched, it is in an initial state represented by the place $p_0$. The transition $t_1$ is crossed at the start of the mission. If everything goes well, the robot is in state $p_1$ and is tracking its first line $\mathbf{a}_1\mathbf{b}_1$. The line $\mathbf{a}_j\mathbf{b}_j$ is validated as soon as the point $\mathbf{b}_j$ is surpassed, in other words if $\langle \mathbf{b}_j - \mathbf{a}_j, \mathbf{m} - \mathbf{b}_j \rangle > 0$. This stopping criterion coupled with the path can be interpreted as a sort of validation. Once this is validated, we proceed to the next line. When the list of lines to track is empty, the mission ends (place $p_3$).

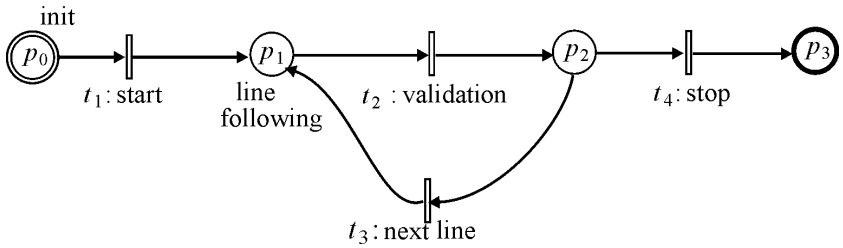


Figure 3.16: Petri net supervising the navigation of the robot

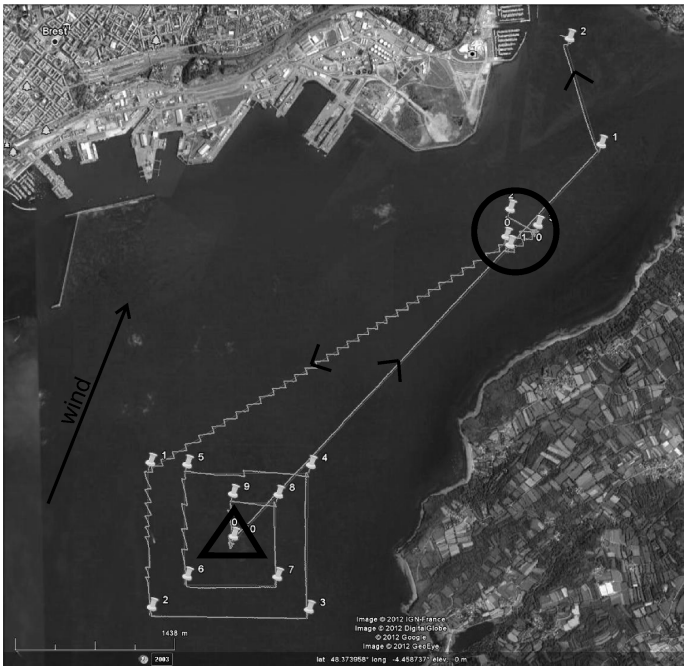


Figure 3.17: Experiment of the spiral composed of five stages : (a) the robot begins with a triangle (in the circle) ; (b) it goes upwind following a line ; (c) it described a spiral ; (d) it anchors virtually at the center of the spiral for several minutes ; (e) and goes back to the harbor.

### 3.3.4 Experiment

Beginning in September 2011, we carried out a series of experiments with the VAIMOS sailboat in autonomous mode. We will describe one of these experiments which is simultaneously simple and representative, which took place on Thursday 28 June 2012 in the Brest bay, close to the Moulin Blanc harbor. The trajectory performed by the robot is represented on Figure 3.17. The wind comes from South-South-East, as indicated by the arrow that represents the average wind on the robot during the entire mission, deduced from the sensor. In this zone of heavy maritime traffic, interruptions in the mission can be anticipated and a permanent wifi link is necessary between the robot and the tracking boat in order to be able to cancel to mission at any time and avoid collision with other boats. Apart from these security interruptions (which by the way were not necessary during the mission), the robot is entirely autonomous. The mission is broken down into five sub-missions. First of all, the robot begins with a triangle (in the circle) in order to check whether everything is working properly. Then, it proceeds South-East against the wind by tracking the required path. It then describes a spiral. Once it is in the center of the spiral, the robot anchors virtually, in other words it maneuvers

in order to remain around its attachment point. Finally, the robot returns to the harbor with the wind.

Other larger-scale experiments were also carried out, such as the journey from Brest to Douarnenez (see Figure 3.18) undertaken on the 17th-18th January 2012, thus completing a path of more than 100 km. From very high up (as in the figure), the lines seem to be tracked perfectly. Upon closer inspection, things are revealed to be less idealistic: the sailboat tacks in order to go upwind, re-calibrates itself or is subjected to large waves otherwise. However, in both previously described experiments, the robot is never more than 50 meters away from its track (except of course in the situations of avoidance in which it is being hauled).

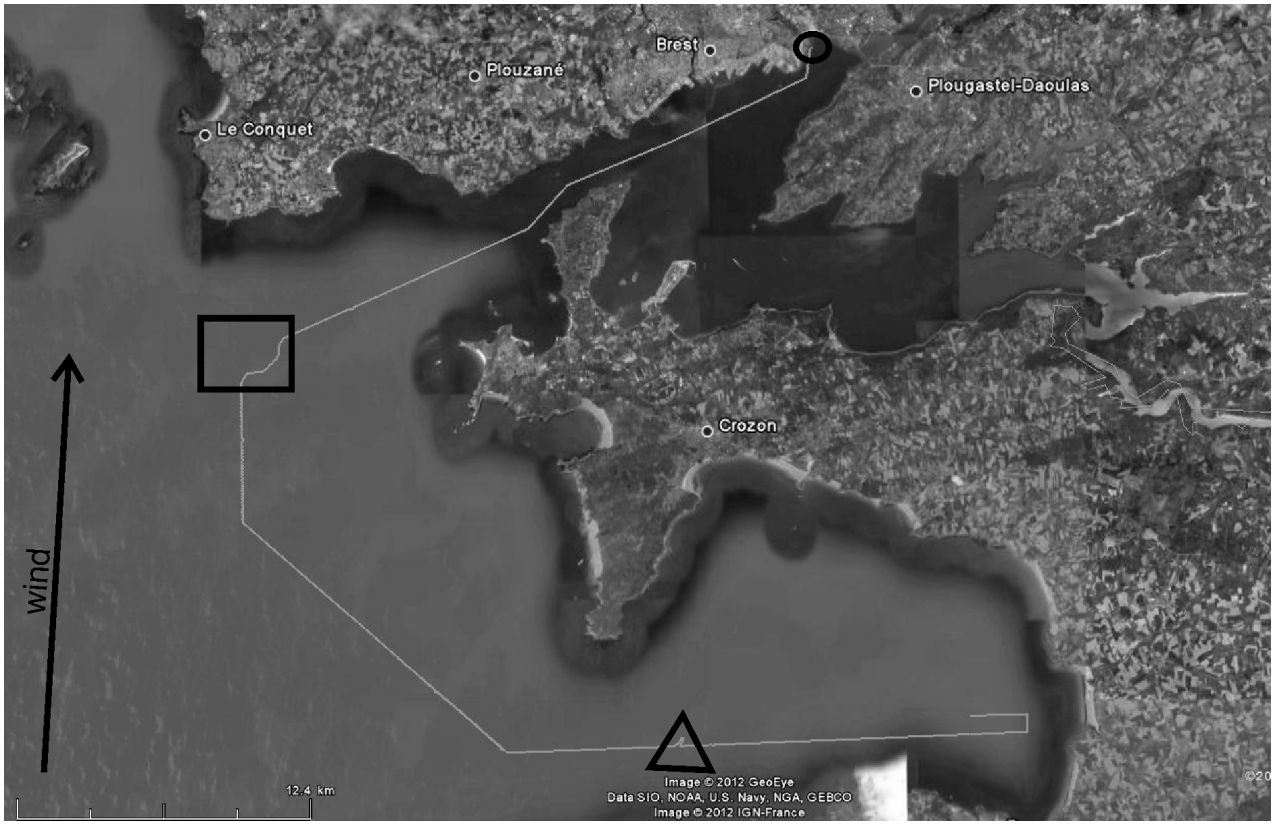


Figure 3.18: Journey from Brest to Douarnenez made by VAIMOS : (a) the robot leaves the Moulin-Blanc harbour (in the circle) ; (b) it avoids a submarine (in the square) ; (c) it avoids a cargo ship (triangle)

REMARK 3.1.– *Even though the robot never surpasses its line by more than 50 m, we could do better and improve this precision when the robot follows the nominal heading (i.e. the angle $\varphi$ of the line corresponds to a sustainable heading). Indeed, in our experiments, a 10 m bias can be observed in nominal mode, which means that the distance to the line does not converge towards zero (with GPS precision). The role of the integrator is to remove such a bias. In order to implement such an integrator, we simply replace line 4 of the controller with the following two instructions:*

$$\begin{cases} z &= z + \alpha\ dt\ e \\ \theta &= \varphi - atan\left(\frac{e+z}{r}\right) \end{cases}$$

*where dt is the sampling period. The variable z corresponds to the value of the integrator and naturally converges towards the constant bias that we had without the integrator and which we would like to remove. The coefficient $\alpha$ has to be sufficiently small to avoid a change in the behavior of our robot (which could appear in transient regime). For instance, if $e = 10$ m for 100 seconds, we may want a correction of 1 m of the bias. for this, we need to take $\alpha = 0.001$. Let us note that as soon as the distance on the line is greater than r (this is the case for instance during initialization), when the robot validates a line and continues on to the next, or when the robot is in large mode, then the integrator has to be forced to zero. Indeed, an integrator must not take up its function unless the permanent regime has been established.*

# Exercises

EXERCISE 3.1.– *Robot tank on a line*

Let us consider a robot moving on a plane and described by the following state equations:

$$\begin{cases} \dot{x} &= \cos\theta \\ \dot{y} &= \sin\theta \\ \dot{\theta} &= u \end{cases}$$

where $\theta$ is the heading of the robot and $(x, y)$ are the coordinates of its center. This model corresponds to the Dubins car[DUB 57]. The state vector is given by $\mathbf{x} = (x, y, \theta)$.

1) Simulate this system graphically with MATLAB in various situations. You may start with the program `ex_tank_line.m`.

2) Propose a heading controller for the robot.

3) Propose a controller tracking a line **ab**. This line must be attractive. Stop the program when the point **b** is overtaken, in other words when $(\mathbf{b} - \mathbf{a})^{\mathrm{T}} (\mathbf{b} - \mathbf{m}) < 0$.

4) Make the robot track a closed path composed of a sequence of lines $\mathbf{a}_j\mathbf{b}_j$, $j \in \{1, \ldots, j_{\max}\}$.

5) Make several identical robots track the same circuit, but with different speeds. Modify the control laws in order to avoid the collisions.

---

EXERCISE 3.2.– *Van der Pol car*

Consider the car represented on Figure 3.19.

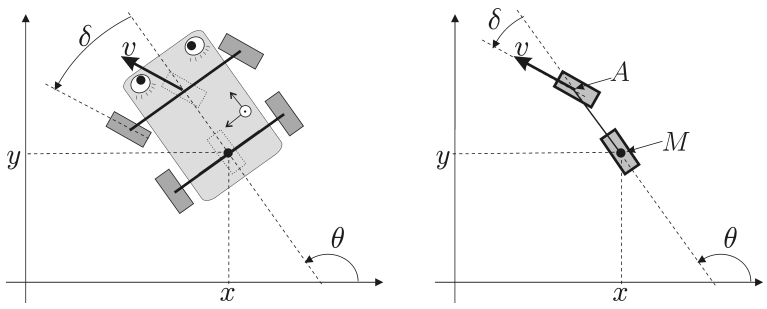


Figure 3.19: Car moving on a plane (view from above)

This car has two controls: the acceleration of the front wheels and the rotation speed of the steering wheel. The state variables of our system are composed of the position coordinates (the

coordinates $x, y$ of the center of the rear axle, the heading $\theta$ of the car and the angle $\delta$ of the front wheels) and the speed $v$ of the center of the front axle. The evolution equation of the car is assumed to be identical to that of a tricycle (see page 50). It is written as:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \\ \dot{\delta} \end{pmatrix} = \begin{pmatrix} v\cos\delta\cos\theta \\ v\cos\delta\sin\theta \\ \frac{v\sin\delta}{L} \\ u_1 \\ u_2 \end{pmatrix}.$$

1) Simulate this system in MATLAB by using Euler's method. You can start with the program `ex_vanderpol.m`.

2) Perform a first high-gain proportional feedback $\mathbf{u} = \boldsymbol{\rho}(\mathbf{x}, \bar{\mathbf{u}})$ that would allow switching to a cart model of the form:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \bar{u}_1\cos\theta \\ \bar{u}_1\sin\theta \\ \bar{u}_2 \end{pmatrix}.$$

The new inputs of the feedback system $\bar{u}_1$ and $\bar{u}_2$ correspond respectively to the speed $v$ and to the angular speed $\dot{\theta}$.

3) Perform a second feedback $\bar{\mathbf{u}} = \boldsymbol{\sigma}(\mathbf{x}, \mathbf{w})$ that would allow us to control this car's heading and speed. The new input will be $\mathbf{w} = (w_1, w_2)$ where $w_1, w_2$ correspond respectively to the desired speed and heading.

4) We would like the car to follow a path that obeys the Van der Pol equation:

$$\begin{cases} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -(0.01\, x_1^2 - 1)\, x_2 - x_1 \end{cases}$$

Propose a third controller $\mathbf{w} = \boldsymbol{\tau}(\mathbf{x})$ that would allow us to perform this. Validate it using a simulation by superimposing the vector field by using the `quiver` instruction.

---

EXERCISE 3.3.– *Anchoring*

Let us consider the robot described by the following state equations:

$$\begin{cases} \dot{x} &= \cos\theta \\ \dot{y} &= \sin\theta \\ \dot{\theta} &= u \end{cases}$$

The aim of *anchoring* the robot is to remain within the neighborhood of zero.

1) Does this system have a point of equilibrium ?

2) Given the fact that the problem of remaining around zero admits a rotational symmetry, we suggest switching from a Cartesian representation $(x, y, \theta)$ towards a polar representation $(\alpha, d, \varphi)$,

as shown on Figure 3.20. Give the state equations in the polar representation.

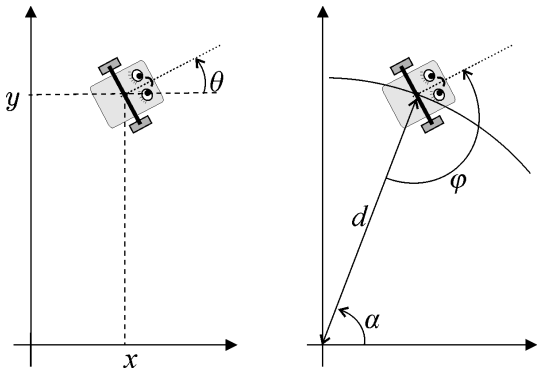


Figure 3.20: Coordinate system change allowing to take advantage of the rotational symmetry

3) How is this new representation of interest for the graphical representation of the system dynamics ?

4) In order to solve the anchoring problem we propose the control law:

$$u = \begin{cases} +1 & \text{if } \cos\varphi \leq \frac{1}{\sqrt{2}} \quad \text{(the robot turns to the left)} \\ -\sin\varphi & \text{otherwise} \qquad\quad \text{(proportional control)} \end{cases}$$

An illustration of this control law is given in Figure 3.21. Explain how this control solves the anchoring problem. Is there a configuration that allows the robot to move arbitrarily far away from 0 ?

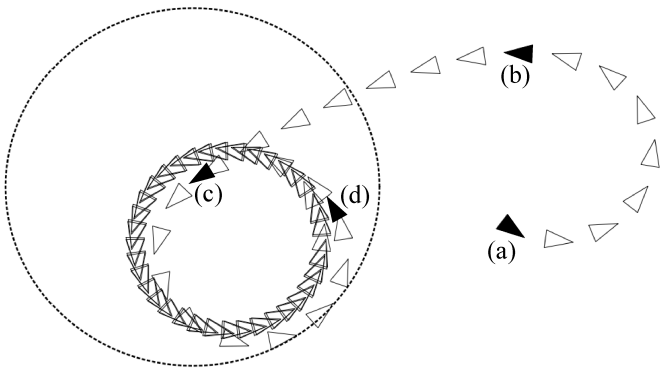


Figure 3.21: Path of the controlled system in order to remain inside the disk

5) Simulate this control law in MATLAB with various initial conditions. You can start with the program `ex_anchor.m`.

---

EXERCISE 3.4.– *Discretization of the state space*

Let us consider the following system (which stems from the previous exercise):

$$\begin{cases} \text{(i)} \quad \dot{\varphi} &= \begin{cases} \frac{\sin\varphi}{d} + 1 & \text{if } \cos\varphi \leq \frac{1}{\sqrt{2}} \\ \left(\frac{1}{d} - 1\right)\sin\varphi & \text{otherwise} \end{cases} \\ \text{(ii)} \quad \dot{d} &= -\cos\varphi \end{cases}$$

The associated vector field is represented on Figure 3.22, where $\varphi \in [\pi, \pi]$ and $d \in [0, 10]$. A path $\phi(t, \mathbf{x}_0)$ which corresponds to the simulation visualized in the previous exercise is also drawn.

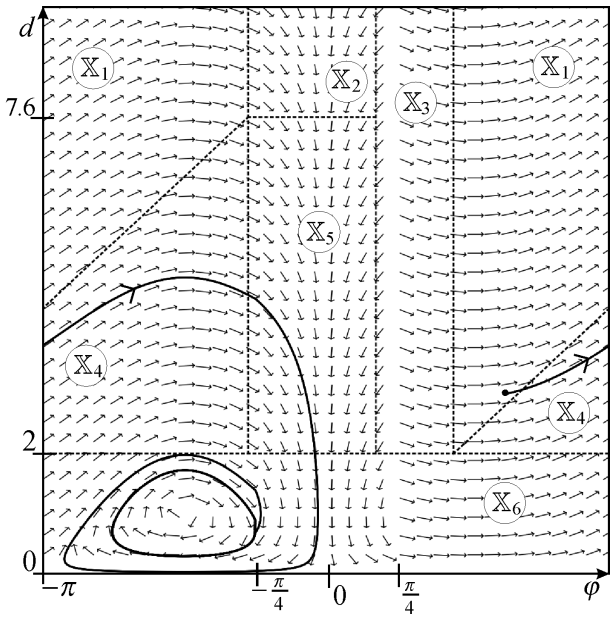


Figure 3.22: Vector field associated with our system

On this same figure, the state space is cut into six areas. Indeed, given the adjunction of the line $\varphi = \pi$ with the line $\varphi = -\pi$, the state space has a cylindrical nature and we have six areas, whereas we can see eight areas on the figure.

**Succession relation**. We define the rotation, denoted by $\hookrightarrow$ between zones $\mathbb{A}, \mathbb{B}$ of the state space as follows:

$$(\mathbb{A} \hookrightarrow \mathbb{B}) \Leftrightarrow \exists \mathbf{x}_0 \in \mathbb{A} \in \phi(\eta(\mathbf{x}_0), \mathbf{x}_0) \in \mathbb{B}$$

where $\eta(\mathbf{x}_0)$ is the time the system exits $\mathbb{A}$.

**Convention** If there is $\mathbf{x}_0 \in \mathbb{A}$, such that $\forall t > 0, \phi(t, \mathbf{x}_0) \subset \mathbb{A}$ then $\eta(\mathbf{x}_0) = \infty$. Thus, $\phi(\eta(\mathbf{x}_0), \mathbf{x}_0) \in \mathbb{A}$ and therefore will we have $(\mathbb{A} \hookrightarrow \mathbb{A})$.

1) Draw the graph associated with this relation.

2) From this, deduce a superset of the state space in which the system will remain trapped.

---

EXERCISE 3.5.– *Sailing robot*

Consider the sailboat described by the following state equations:

$$\left\{\begin{array}{lcl}
\dot{x} & = & v\cos\theta + p_1 a\cos\psi \\
\dot{y} & = & v\sin\theta + p_1 a\sin\psi \\
\dot{\theta} & = & \omega \\
\dot{v} & = & \frac{f_s\sin\delta_s - f_r\sin u_1 - p_2 v^2}{p_9} \\
\dot{\omega} & = & \frac{f_s(p_6 - p_7\cos\delta_s) - p_8 f_r\cos u_1 - p_3\omega v}{p_{10}} \\
f_s & = & p_4\,\|\mathbf{w}_{\rm ap}\|\sin(\delta_s - \psi_{\rm ap}) \\
f_r & = & p_5 v\sin u_1 \\
\sigma & = & \cos\psi_{\rm ap} + \cos u_2 \\
\delta_s & = & \left\{\begin{array}{ll} \pi + \psi_{\rm ap} & \text{if } \sigma \le 0 \\ -\text{sign}(\sin\psi_{\rm ap})\cdot u_2 & \text{otherwise}\end{array}\right. \\
\mathbf{w}_{\rm ap} & = & \begin{pmatrix} a\ \cos(\psi-\theta) - v \\ a\ \sin(\psi-\theta)\end{pmatrix} \\
\psi_{\rm ap} & = & \text{angle } \mathbf{w}_{\rm ap}
\end{array}\right.$$

where $(x, y, \theta)$ corresponds to the posture of the boat, $v$ is its forward speed, $\omega$ is its angular speed, $f_s$ ( $s$ for *sail*) is the force of the wind on the sail, $f_r$ ($r$ for *rudder*) is the force of the water on the rudder, $\delta_s$ is the angle of the sail, $a$ is the true wind speed, $\psi$ is the true wind angle (see Figure 3.9) and $\mathbf{w}_{\rm ap}$ is the apparent wind vector. The quantity $\sigma$ is an indicator of the sheet tension. Thus, if $\sigma \le 0$, the sheet is released and the sail is flapping. If $\sigma \ge 0$, the sheet is stretched and inflated by the wind. In these equations, the $p_i$ are design parameters of the sailboat. We will take the following values, given in international units: $p_1 = 0.1$ (drift coefficient), $p_2 = 1$ (drag coefficient), $p_3 = 6\ 000$ (angular friction of the hull against the water), $p_4 = 1\ 000$ (sail lift), $p_5 = 2\ 000$ (rudder lift), $p_6 = 1$ (position of the wind's center of thrust on the sail), $p_7 = 1$ (position of the mast), $p_8 = 2$ (position of the rudder), $p_9 = 300$ (mass of the sailboat) and $p_{10} = 10\ 000$ (inertial momentum of the sailboat).

1) Simulate the boat in MATLAB. You can start with the program `ex_sailboat.m`.

2) Implement the controller proposed in Section 3.3. We will use the following parameters $\zeta = \frac{\pi}{4}$ for the large-angle, $r = 10$ m for the radius of the air corridor, $\delta_r^{\max} = 1$ rad for the maximum angle of the rudder and $\beta = \frac{\pi}{4}$ for the angle of the sail in crosswind.

---

EXERCISE 3.6.– *Flying drone*

Consider a flying drone [BEA 12] such as the one represented on Figure 3.23a. This is a 1 kg fully autonomous plane. One possible model to describe its dynamics, very strongly inspired by those of

*Faser Ultra Stick* [KLE 06] in Figure 3.23b, is given by:

$$\begin{pmatrix} \dot{\mathbf{p}} \\ \begin{pmatrix} \dot{\psi} \\ \dot{\theta} \\ \dot{\varphi} \end{pmatrix} \\ \dot{\mathbf{v}} \\ \dot{\boldsymbol{\omega}} \end{pmatrix} = \begin{pmatrix} \mathbf{R}_{\text{euler}}(\varphi,\theta,\psi)\cdot\mathbf{v} \\ \begin{pmatrix} 0 & \frac{\sin\varphi}{\cos\theta} & \frac{\cos\varphi}{\cos\theta} \\ 0 & \cos\varphi & -\sin\varphi \\ 1 & \tan\theta\sin\varphi & \tan\theta\cos\varphi \end{pmatrix}\cdot\boldsymbol{\omega} \\ 9.81\cdot\begin{pmatrix} -\sin\theta \\ \cos\theta\sin\varphi \\ \cos\theta\cos\varphi \end{pmatrix} + \mathbf{f}_a + \begin{pmatrix} u_1 \\ 0 \\ 0 \end{pmatrix} - \boldsymbol{\omega}\wedge\mathbf{v} \\ \begin{pmatrix} -\omega_3\omega_2 - \frac{\|\mathbf{v}\|^2}{10}(\beta + 2u_3 + \frac{5\omega_1-\omega_3}{\|\mathbf{v}\|}) \\ \omega_3\omega_1 - \frac{\|\mathbf{v}\|^2}{100}(1 + 20\alpha - 2u_3 + 30u_2 + \frac{300\omega_2}{\|\mathbf{v}\|}) \\ \frac{\omega_1\omega_2}{10} + \frac{\|\mathbf{v}\|^2}{10}(\beta + \frac{u_3}{2} + \frac{\omega_1-2\omega_3}{2\|\mathbf{v}\|}) \end{pmatrix} \end{pmatrix}$$

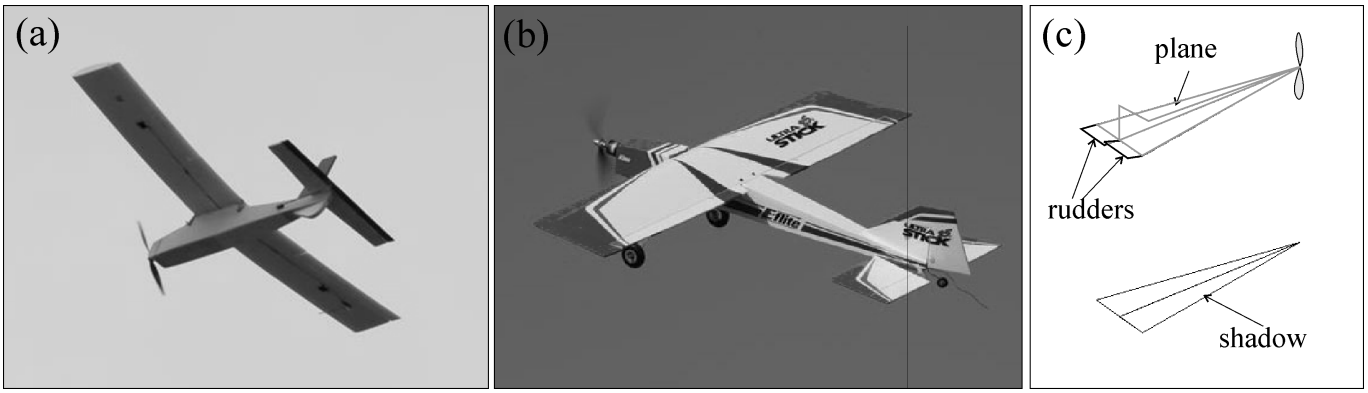


Figure 3.23: (a) $\mu$-STIC plane made by the ENSTA Bretagne ; (b) Faser Ultra Stick plane from the University of Minnesota ; (c) graphical representation used for the simulation

with:

$$\alpha = \text{atan}\left(\frac{v_3}{v_1}\right),\ \beta = \text{asin}\left(\frac{v_2}{\|\mathbf{v}\|}\right)$$

$$\mathbf{f}_a = \frac{\|\mathbf{v}\|^2}{500}\begin{pmatrix} -\cos\alpha\cos\beta & \cos\alpha\sin\beta & \sin\alpha \\ \sin\beta & \cos\beta & 0 \\ -\sin\alpha\cos\beta & \sin\alpha\sin\beta & -\cos\alpha \end{pmatrix} \cdot \begin{pmatrix} 4 + (-0.3 + 10\alpha + \frac{10\omega_2}{\|\mathbf{v}\|} + 2u_3 + 0.3u_2)^2 + |u_2| + 3|u_3| \\ -50\beta + \frac{10\omega_3 - 3\omega_1}{\|\mathbf{v}\|} \\ 10 + 500\alpha + \frac{400\omega_2}{\|\mathbf{v}\|} + 50u_3 + 10u_2 \end{pmatrix}$$

In this model, all the quantities are given in international units. The vector $\mathbf{p} = (x, y, z)$ represents the position of the drone, with the $z$ axis oriented towards the center of the Earth. The orientation

of the drone is represented by the Euler angles $(\varphi, \theta, \psi)$ and the Euler matrix $\mathbf{R}_{\text{euler}}(\varphi, \theta, \psi)$ is given by Formula [1.7] on page 15. The vector $\mathbf{v}$ represents the speed of the drone expressed in its own coordinate system. The rotation vector of the plane is denoted here by $\boldsymbol{\omega}$. It is linked to the derivatives of the Euler angles by Formula [1.10] on page 18. Let us note that Formula [1.10] gives the first three equations of our state model. The angles $\alpha$ and $\beta$ correspond to the angle of attack and the sideslip angle. The vector $\mathbf{f}_a$ corresponds to the acceleration caused by the forces created by air. A simplified geometric view of the drone, given in Figure 3.23c, shows a helix for propulsion and two ailerons for direction. The input vector $\mathbf{u} = (u_1, u_2, u_3)$ involved in our state model contained the propulsive acceleration $u_1 \in [0, 10]$ (in $\text{ms}^{-2}$), the sum $u_2 \in [-0.6, 0.6]$ (in radians) of the two aileron angles and $u_3 \in [-0.3, 0.3]$ (in radians) the differential between these two ailerons.

1) Starting with the program `ex_plane.m`, simulate this drone in MATLAB. For the graphics, see Figure 3.23c.

2) Propose a heading, elevation and speed control law.

3) We would like the robot to be positioned on a circle of radius $\bar{r} = 100$ m, centered around 0 at an altitude of 50 m and a speed of $\bar{v} = 15$ $\text{ms}^{-1}$. Give the control law and illustrate the associated behavior of the robot in MATLAB.

---

EXERCISE 3.7.– *Quadrotor*

We consider the quadrotor represented on Figure 3.24.

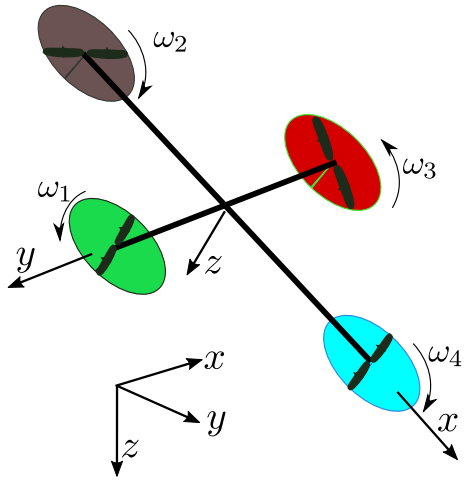


Figure 3.24: Quadrotor to be controlled

Its dynamic model (see Subsection (1.4.2) page 20) is described by the equations of Figure 3.25. Note that the graph is represented in a causal manner from the speed $\omega_i$ of each motor which can be tuned independently to the position $\mathbf{p}$ of the robot. This causal sequence will be used for the synthesis of the controller.

The state variables are the position $\mathbf{p}$ of the robot, $(\varphi, \theta, \psi)$ the Euler angles, the speed $\mathbf{v}_r$ expressed in the robot frame and the rotation vector $\boldsymbol{\omega}_r$ also expressed in the robot frame. The inertia matrix will be taken as

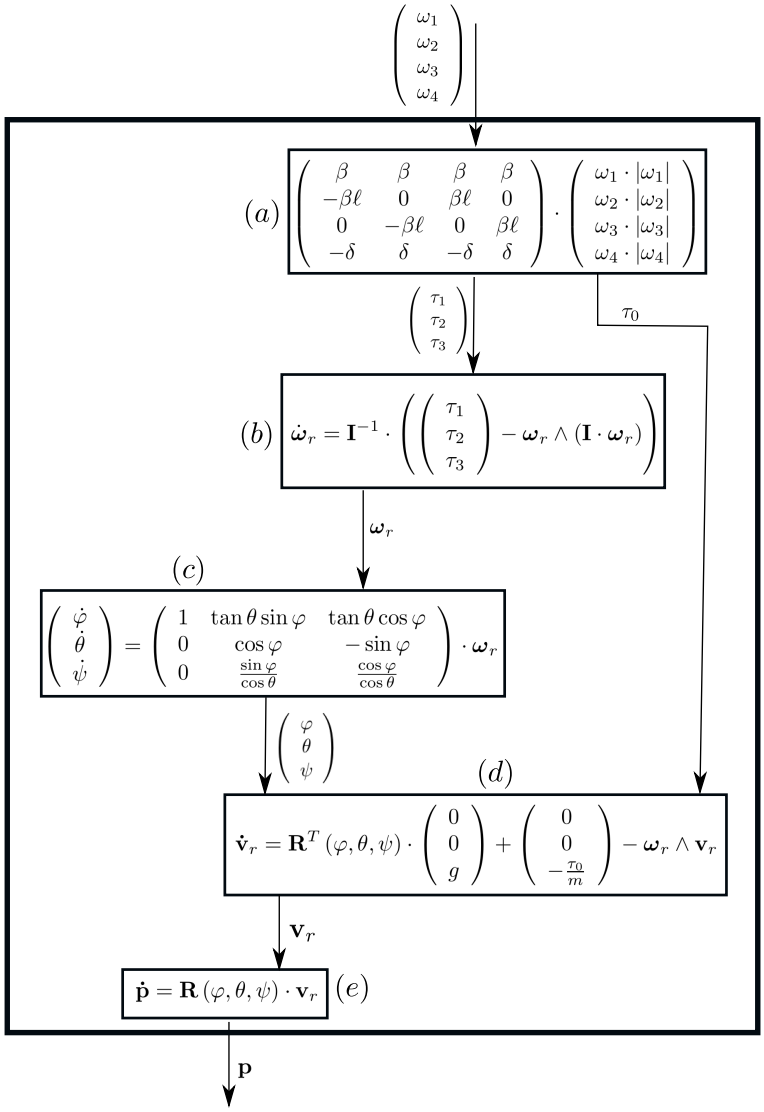


Figure 3.25: Dynamic of the quadrotor represented by a causal chain which links 5 blocks *(a)*, *(b)*, *(c)*, *(d)*, *(e)*

$$\mathbf{I} = \begin{pmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 20 \end{pmatrix}.$$

Moreover, we will take $m = 10$, $g = 9.81$, $\beta = 2$, $d = 1$, and $\ell = 1$, all expressed in the international metric system.

To control this quadrotor, a feedback linearization method could be done, but the computation of the controller is not easy and assumes that the model corresponds to the true system. Here, we want to use a more pragmatic method, based on a backstepping technique [KHA 02]. It corresponds to a sequence of loops, each of which inverting one block in the causal chain. This will make the controller much easy to develop and to debug.

1) Propose a feed forward controller with the new input $\left(\tau_0^d, \tau_1^d, \tau_2^d, \tau_2^d\right)$ with eliminates the effect of Block *(a)*. Here, the subscript $d$ means 'desired' since it is what we want for the $\tau_i$.

2) Using a feedback linearization approach, build a controller with a desired input $\boldsymbol{\omega}_r^d$ and an output $\boldsymbol{\tau}_{1:3}^d = \left(\tau_1^d, \tau_2^d, \tau_2^d\right)$, such that the vector $\boldsymbol{\omega}_r$ converges to $\boldsymbol{\omega}_r^d$. This loop will make us possible to eliminate Block *(b)*.

3) Build a controller with a desired input $\left(\varphi^d, \theta^d, \psi^d\right)$ which and an output $\boldsymbol{\omega}_r^d$, such that the vector $(\varphi, \theta, \psi)$ converges to $\left(\varphi^d, \theta^d, \psi^d\right)$.

4) Add another control loop with an output $\left(\varphi^d, \theta^d, \psi^d, \tau_0^d\right)$ , based on a vector field approach, so that the quadrotor follows a path that obeys the Van der Pol equation:

$$\begin{cases} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\left(0.001\ x_1^2 - 1\right) x_2 - x_1 \end{cases}$$

The quadrotor should also stay at an altitude of 15m and a speed of $v_d = 10\text{ms}^{-1}$. Moreover, we want the front of the robot points forward. An illustration of what should be obtained is depicted on Figure 3.26. The Van der Pol cycle is painted green.

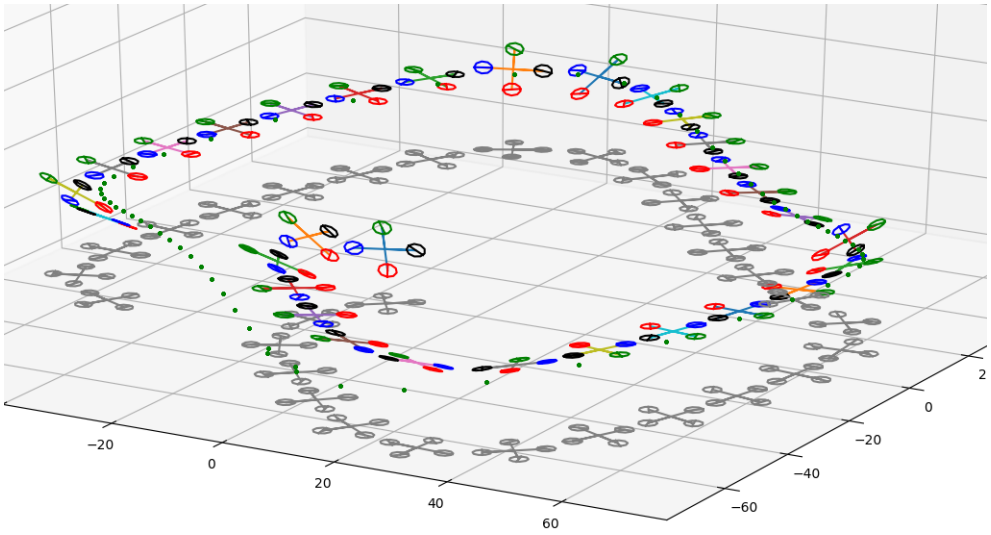


Figure 3.26: Simulation of the quadrotor which follows the Van der Pol cycle

# Chapter 4

# Guidance

In previous chapters, we have studied how to build a control law for a robot described by nonlinear state equations (see Chapter 2) or when the robot's behavior is known (see Chapter 3). *Guidance* is performed on a higher level and focuses on the setpoint to give the controller in order for the robot to be able to accomplish its assigned mission. It will therefore have to take into account the knowledge of its surroundings, the presence of obstacles, the roundness of the Earth, and so forth. Conventionally, guidance is applied in four different environments: terrestrial, marine, aerial and spatial. Given the fields of application covered in this book, we will not study the spatial environment.

## 4.1 Guidance on a sphere

For longer paths over the surface of the Earth, the Cartesian coordinate system, which assumes a flat Earth, can no longer be considered. We then have to rethink our control laws by navigating relative to a spherical coordinate system (also referred to as *geographical coordinates*), which rotates together with the Earth. Let us denote by $\ell_x$ the longitude and by $\ell_y$ the latitude of the point being considered. The transformation in the geographical coordinate system is written as:

$$\mathcal{T} : \begin{pmatrix} \ell_x \\ \ell_y \\ \rho \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \rho \cos \ell_y \cos \ell_x \\ \rho \cos \ell_y \sin \ell_x \\ \rho \sin \ell_y \end{pmatrix} \tag{4.1}$$

When $\rho = 6\ 370$ km, we are on the surface of the Earth, which we will assume to be spherical (see Figure 4.1a).

Let us consider two points $\mathbf{a}$, $\mathbf{m}$ on the surface of the Earth, as illustrated on Figure 4.1b, located by their geographical coordinates. Here, $\mathbf{a}$ for instance represents a reference point to reach and $\mathbf{m}$ is the center of our robot. By assuming that the two points $\mathbf{a}$, $\mathbf{m}$ are not too far apart (by no more than 100 km), we may consider being in the plane and use a local map, as shown on Figure 4.2a.

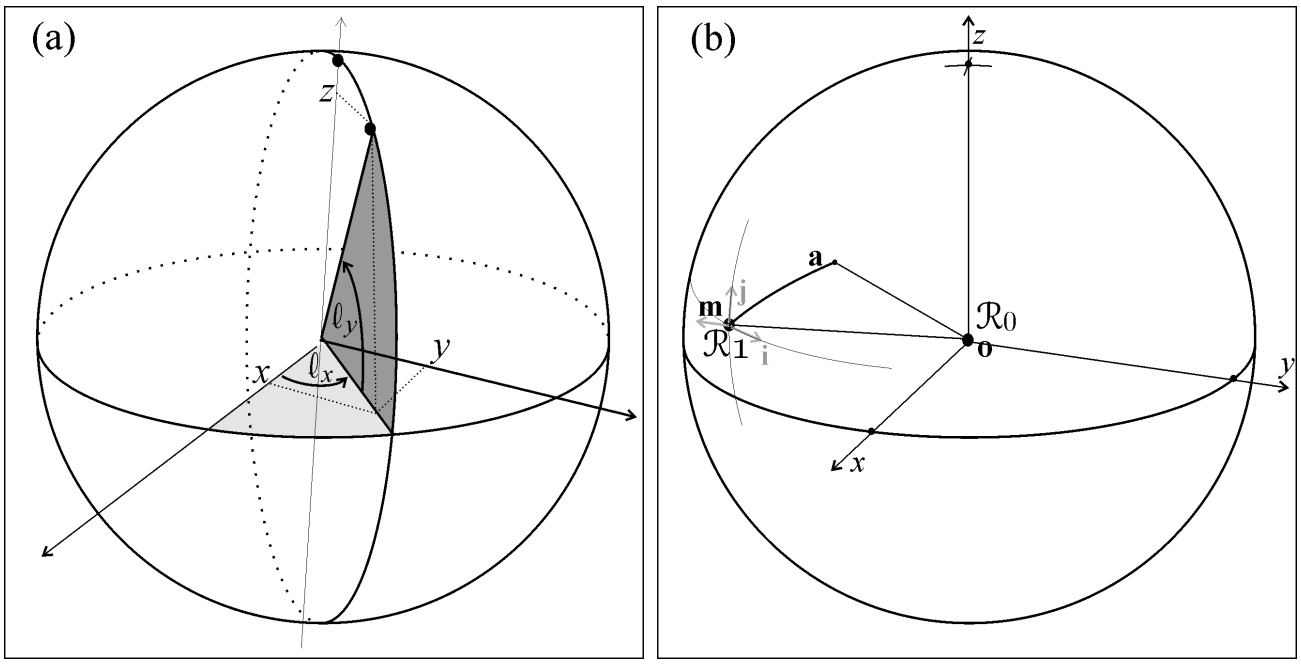


Figure 4.1: (a) Geographical coordinate system ; (b) we would like to express $\mathbf{a}$ in the local map $\mathcal{R}_1$

Let us differentiate Relation (4.1). We obtain:

$$\begin{pmatrix} dx \\ dy \\ dz \end{pmatrix} = \underbrace{\begin{pmatrix} -\rho\cos\ell_y\sin\ell_x & -\rho\sin\ell_y\cos\ell_x & \cos\ell_y\cos\ell_x \\ \rho\cos\ell_y\cos\ell_x & -\rho\sin\ell_y\sin\ell_x & \cos\ell_y\sin\ell_x \\ 0 & \rho\cos\ell_y & \sin\ell_y \end{pmatrix}}_{=\mathbf{J}} \cdot \begin{pmatrix} d\ell_x \\ d\ell_y \\ d\rho \end{pmatrix}$$

This formula can be used to find the geographical coordinates of the cardinal directions, which change depending on the location of the robot $\mathbf{m}$. For instance, the vector corresponding to *East* is found in the first column of the matrix $\mathbf{J}$, North in the second column and the altitude in the third column. We will therefore be able to build a coordinate system (East-North-Altitude) $\mathcal{R}_1$ centered on the robot (in grey in Figure 4.2a) that corresponds to the local map. The corresponding rotation matrix is obtained by normalizing each column of the Jacobian matrix $\mathbf{J}$. We obtain:

$$\mathbf{R} = \begin{pmatrix} -\sin\ell_x & -\sin\ell_y\cos\ell_x & \cos\ell_y\cos\ell_x \\ \cos\ell_x & -\sin\ell_y\sin\ell_x & \cos\ell_y\sin\ell_x \\ 0 & \cos\ell_y & \sin\ell_y \end{pmatrix} \tag{4.2}$$

The transformation that allows switching from the geographic coordinate system $\mathcal{R}_0$ to the local map $\mathcal{R}_1$ is:

$$\mathbf{v}_{|\mathcal{R}_1} = \mathbf{R}^{\mathrm{T}} \cdot \mathbf{v}_{|\mathcal{R}_0} \tag{4.3}$$

This coordinate system change relation can be applied in different contexts such as for instance when establishing coherence in the data collected by two different robots.

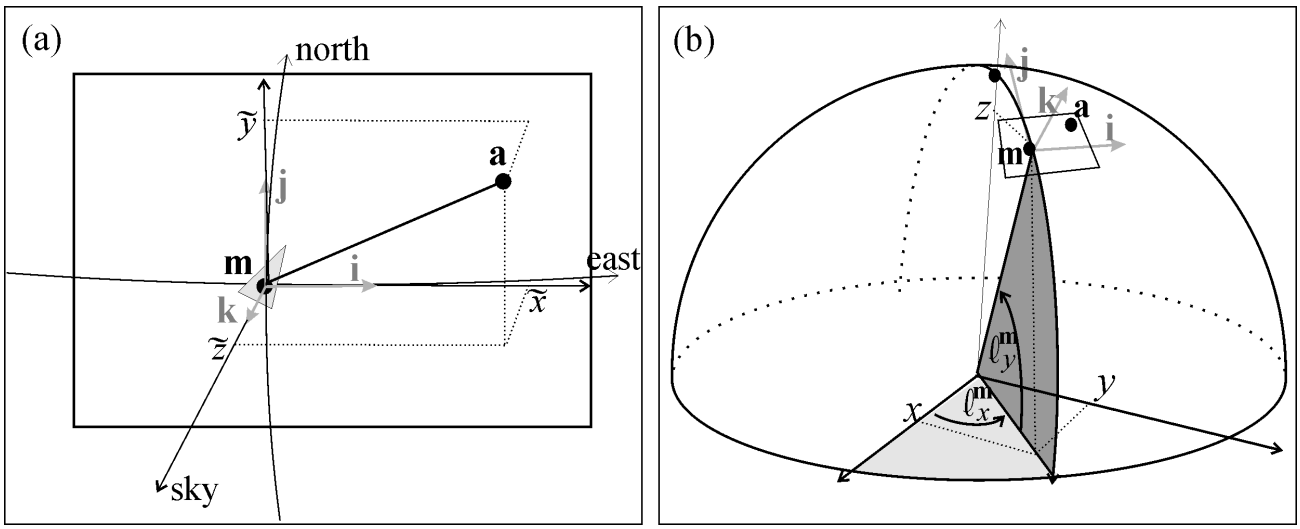


Figure 4.2: (a) The map gives a local Cartesian viewpoint around the robot ; (b) a slight shift $d\ell_x, d\ell_y, d\rho$ creates a displacement $dx, dy, dz$ in the local map

EXAMPLE 4.1.– *A robot situated at* $\mathbf{m} : \left(\ell_x^{\mathbf{m}}, \ell_y^{\mathbf{m}}\right)$ *is moving with a speed vector* $\mathbf{v}^{\mathbf{m}}$*, relative to a fixed ground. This vector is expressed in the local map of the robot. We want to find the speed with which this vector* $\mathbf{v}^{\mathbf{m}}$ *is perceived in the local map of an observer situated at* $\mathbf{a} : \left(\ell_x^{\mathbf{a}}, \ell_y^{\mathbf{a}}\right)$*. Following [4.3], we have:*

$$\begin{cases} \mathbf{v}^{\mathbf{m}} = \mathbf{R}^T\left(\ell_x^{\mathbf{m}}, \ell_y^{\mathbf{m}}\right) \cdot \mathbf{v}_{|\mathcal{R}_0} \\ \mathbf{v}^{\mathbf{a}} = \mathbf{R}^T\left(\ell_x^{\mathbf{a}}, \ell_y^{\mathbf{a}}\right) \cdot \mathbf{v}_{|\mathcal{R}_0} \end{cases}$$

*Therefore:*

$$\mathbf{v}^{\mathbf{a}} = \mathbf{R}^T\left(\ell_x^{\mathbf{a}}, \ell_y^{\mathbf{a}}\right) \cdot \mathbf{R}\left(\ell_x^{\mathbf{m}}, \ell_y^{\mathbf{m}}\right) \cdot \mathbf{v}^{\mathbf{m}}.$$

*This kind of calculation is useful when, for instance, two robots are trying to meet.*

**Switching to a local map**. Let us take a local map $\mathcal{R}_{\mathbf{m}}$ centered on a point $\mathbf{m}$, (in other words a coordinate system located on the surface of the Earth whose origin is $\mathbf{m}$ and whose directions are East-North-upwards. Let us now try to express in $\mathcal{R}_{\mathbf{m}}$ the coordinates $(\widetilde{x}, \widetilde{y}, \widetilde{z})$ of a point $\mathbf{a}$ located by its GPS coordinates $(\ell_x, \ell_y, \rho)$. We can switch from the geographical coordinates to the local coordinates with the following relation:

$$\underbrace{\begin{pmatrix} \widetilde{x} \\ \widetilde{y} \\ \widetilde{z} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \rho^{\mathbf{m}} \end{pmatrix}}_{\mathbf{oa}_{|\mathcal{R}_{\mathbf{m}}}} \overset{[4.3]}{=} \underbrace{\begin{pmatrix} -\sin\ell_x^{\mathbf{m}} & \cos\ell_x^{\mathbf{m}} & 0 \\ -\cos\ell_x^{\mathbf{m}}\sin\ell_y^{\mathbf{m}} & -\sin\ell_x^{\mathbf{m}}\sin\ell_y^{\mathbf{m}} & \cos\ell_y^{\mathbf{m}} \\ \cos\ell_x^{\mathbf{m}}\cos\ell_y^{\mathbf{m}} & \cos\ell_y^{\mathbf{m}}\sin\ell_x^{\mathbf{m}} & \sin\ell_y^{\mathbf{m}} \end{pmatrix}}_{\mathbf{R}^{\mathrm{T}}\left(\ell_x^{\mathbf{m}}, \ell_y^{\mathbf{m}}\right)} \cdot \underbrace{\begin{pmatrix} \rho\cos\ell_y\cos\ell_x \\ \rho\cos\ell_y\sin\ell_x \\ \rho\sin\ell_y \end{pmatrix}}_{\mathbf{oa}_{|\mathcal{R}_0}}$$

where $\mathbf{o}$ is the origin corresponding to the center of the Earth. When $\ell_x^{\mathbf{m}} \simeq \ell_x$ and $\ell_y^{\mathbf{m}} \simeq \ell_y$, a

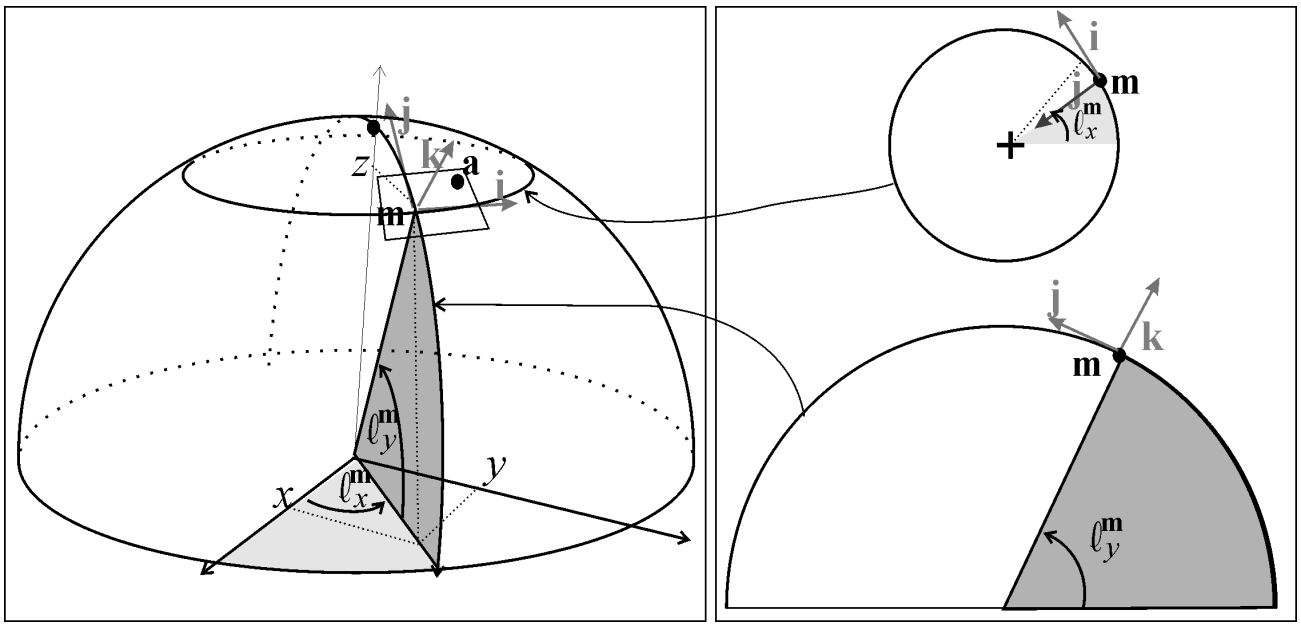


Figure 4.3: Getting Cartesian coordinates in a local frame centred in $\mathbf{m}$

first-order approximation is directly obtained with the aid of Figure 4.3.

$$\begin{pmatrix} \widetilde{x} \\ \widetilde{y} \\ \widetilde{z} \end{pmatrix} \simeq \begin{pmatrix} \rho \cdot \cos \ell_y \cdot (\ell_x - \ell_x^{\mathbf{m}}) \\ \rho \cdot \left(\ell_y - \ell_y^{\mathbf{m}}\right) \\ \rho - \rho^{\mathbf{m}} \end{pmatrix}. \tag{4.4}$$

Note that the circle of latitude has a radius of $\rho \cos \ell_y$.

We could also have formally obtained these results using trigonometric relations:

$$\begin{aligned}
\begin{pmatrix} \widetilde{x} \\ \widetilde{y} \\ \widetilde{z} \end{pmatrix} &= \begin{pmatrix} 0 \\ 0 \\ -\rho^{\mathbf{m}} \end{pmatrix} + \rho \begin{pmatrix} -\sin \ell_x^{\mathbf{m}} \cos \ell_y \cos \ell_x & + & \cos \ell_x^{\mathbf{m}} \cos \ell_y \sin \ell_x \\ -\cos \ell_x^{\mathbf{m}} \sin \ell_y^{\mathbf{m}} \cos \ell_y \cos \ell_x & - & \sin \ell_x^{\mathbf{m}} \sin \ell_y^{\mathbf{m}} \cos \ell_y \sin \ell_x & + & \cos \ell_y^{\mathbf{m}} \sin \ell_y \\ \cos \ell_x^{\mathbf{m}} \cos \ell_y^{\mathbf{m}} \cos \ell_y \cos \ell_x & + & \cos \ell_y^{\mathbf{m}} \sin \ell_x^{\mathbf{m}} \cos \ell_y \sin \ell_x & + & \sin \ell_y^{\mathbf{m}} \sin \ell_y \end{pmatrix} \\
&= \begin{pmatrix} 0 \\ 0 \\ -\rho^{\mathbf{m}} \end{pmatrix} + \rho \begin{pmatrix} \cos \ell_y \left(\cos \ell_x^{\mathbf{m}} \sin \ell_x - \sin \ell_x^{\mathbf{m}} \cos \ell_x\right) \\ -\sin \ell_y^{\mathbf{m}} \cos \ell_y \left(\cos \ell_x^{\mathbf{m}} \cos \ell_x + \sin \ell_x^{\mathbf{m}} \sin \ell_x\right) + \cos \ell_y^{\mathbf{m}} \sin \ell_y \\ \cos \ell_y^{\mathbf{m}} \cos \ell_y \left(\cos \ell_x^{\mathbf{m}} \cos \ell_x + \sin \ell_x^{\mathbf{m}} \sin \ell_x\right) + \sin \ell_y^{\mathbf{m}} \sin \ell_y \end{pmatrix} \\
&= \begin{pmatrix} 0 \\ 0 \\ -\rho^{\mathbf{m}} \end{pmatrix} + \rho \begin{pmatrix} \cos \ell_y \sin \left(\ell_x - \ell_x^{\mathbf{m}}\right) \\ -\sin \ell_y^{\mathbf{m}} \cos \ell_y \cos \left(\ell_x^{\mathbf{m}} - \ell_x\right) + \cos \ell_y^{\mathbf{m}} \sin \ell_y \\ \cos \ell_y^{\mathbf{m}} \cos \ell_y \cos \left(\ell_x^{\mathbf{m}} - \ell_x\right) + \sin \ell_y^{\mathbf{m}} \sin \ell_y \end{pmatrix} \\
&\simeq \begin{pmatrix} 0 \\ 0 \\ -\rho^{\mathbf{m}} \end{pmatrix} + \rho \begin{pmatrix} \cos \ell_y \cdot \left(\ell_x - \ell_x^{\mathbf{m}}\right) \\ -\sin \ell_y^{\mathbf{m}} \cos \ell_y + \cos \ell_y^{\mathbf{m}} \sin \ell_y \\ \cos \ell_y^{\mathbf{m}} \cos \ell_y + \sin \ell_y^{\mathbf{m}} \sin \ell_y \end{pmatrix} \\
&\simeq \begin{pmatrix} 0 \\ 0 \\ -\rho^{\mathbf{m}} \end{pmatrix} + \rho \begin{pmatrix} \cos \ell_y \cdot \left(\ell_x - \ell_x^{\mathbf{m}}\right) \\ \sin \left(\ell_y - \ell_y^{\mathbf{m}}\right) \\ \cos \left(\ell_y - \ell_y^{\mathbf{m}}\right) \end{pmatrix} \simeq \begin{pmatrix} \rho \cdot \cos \ell_y \cdot \left(\ell_x - \ell_x^{\mathbf{m}}\right) \\ \rho \cdot \left(\ell_y - \ell_y^{\mathbf{m}}\right) \\ \rho - \rho^{\mathbf{m}} \end{pmatrix}.
\end{aligned}$$

Let us note that when the robot is moving in a small-diameter area, we sometimes choose a reference

point other than the center **m** of the robot, such as for instance its launch position.

## 4.2 Path planning

When the robot is completely autonomous, the desired path must be planned out [LaV 06]. Very often, these paths are polynomials, for two reasons. First of all, the space of polynomials has a vector space structure and can therefore utilize the power of linear algebra. Second, they are easier to differentiate, which is useful for feedback linearization, since it requires the successive derivatives of the setpoints.

### 4.2.1 Simple example

Let us illustrate how such a planning is performed on the example of a robot tank. Assume that at the initial moment $t = 0$, the robot is located at the point $(x_0, y_0)$ and that we would like to reach the point $(x_1, y_1)$ at time $t_1$ with a speed equal to $(v_x^1, v_y^1)$. We suggest a polynomial path of the form:

$$\begin{aligned} x_d &= a_x t^2 + b_x t + c_x \\ y_d &= a_y t^2 + b_y t + c_y \end{aligned}$$

We need to solve the system of equations:

$$\begin{array}{ll} c_x = x_0, & c_y = y_0 \\ a_x t_1^2 + b_x t_1 + c_x = x_1 & a_y t_1^2 + b_y t_1 + c_y = y_1, \\ 2a_x t_1 + b_x = v_x^1, & 2a_y t_1 + b_y = v_y^1 \end{array}$$

which is linear. We easily obtain:

$$\begin{pmatrix} a_x \\ a_y \\ b_x \\ b_y \\ c_x \\ c_y \end{pmatrix} = \begin{pmatrix} \frac{1}{t_1^2} x_0 - \frac{1}{t_1^2} x_1 + \frac{1}{t_1} v_x^1 \\ \frac{1}{t_1^2} y_0 - \frac{1}{t_1^2} y_1 + \frac{1}{t_1} v_y^1 \\ -v_x^1 - \frac{2}{t_1} x_0 + \frac{2}{t_1} x_1 \\ -v_y^1 - \frac{2}{t_1} y_0 + \frac{2}{t_1} y_1 \\ x_0 \\ y_0 \end{pmatrix}$$

We therefore have:

$$\begin{array}{ll} \dot{x}_d = 2a_x t + b_x & \dot{y}_d = 2a_y t + b_y \\ \ddot{x}_d = 2a_x, & \ddot{y}_d = 2a_y \end{array}$$

By inserting these quantities into a control law obtained using linearizing feedback (as is the case for instance with Equation [2.9]), we obtain a controller that meets our objectives.

### 4.2.2 Bézier polynomials

Here we will look at generalizing the approach presented in the previous section. Given the control points $\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n$, we can generate a polynomial $\mathbf{f}(t)$ such that $\mathbf{f}(0) = \mathbf{p}_0$, $\mathbf{f}(1) = \mathbf{p}_n$ and such that for $t \in [0, 1]$, the polynomial $\mathbf{f}(t)$ is successively attracted by the $\mathbf{p}_i$ with $i \in \{0, \ldots, n\}$. In order to correctly understand the method of building Bézier polynomials, let us examine various cases:

- case $n = 1$. We take the standard linear interpolation:

  $$\mathbf{f}(t) = (1-t)\,\mathbf{p}_0 + t\mathbf{p}_1.$$

  The point $\mathbf{f}(t)$ corresponds to a barycenter between the control points $\mathbf{p}_0$ and $\mathbf{p}_1$, and the weights assigned to these two points change with time:

- case $n = 2$. We now have three control points $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2$. We create an auxiliary control point $\mathbf{p}_{01}$ which moves on the segment $[\mathbf{p}_0, \mathbf{p}_1]$ and another $\mathbf{p}_{12}$ which is associated with the segment $[\mathbf{p}_1, \mathbf{p}_2]$. We take:

  $$\begin{aligned}\mathbf{f}(t) &= (1-t)\,\mathbf{p}_{01} + t\mathbf{p}_{12} \\ &= (1-t)\,\underbrace{((1-t)\,\mathbf{p}_0 + t\mathbf{p}_1)}_{\mathbf{p}_{01}} + t\underbrace{((1-t)\,\mathbf{p}_1 + t\mathbf{p}_2)}_{\mathbf{p}_{12}} \\ &= (1-t)^2\,\mathbf{p}_0 + 2\,(1-t)\,t\mathbf{p}_1 + t^2\mathbf{p}_2.\end{aligned}$$

  We thus obtain a second-order polynomial ;

- case $n = 3$. We apply the previous method for four control points. We obtain:

  $$\begin{aligned}\mathbf{f}(t) &= (1-t)\,\mathbf{p}_{012} + t\mathbf{p}_{123} \\ &= (1-t)\,\underbrace{((1-t)\,\mathbf{p}_{01} + t\mathbf{p}_{12})}_{\mathbf{p}_{012}} + t\underbrace{((1-t)\,\mathbf{p}_{12} + t\mathbf{p}_{23})}_{\mathbf{p}_{123}} \\ &= (1-t)^3\,\mathbf{p}_0 + 3\,(1-t)^2\,t\mathbf{p}_1 + 3\,(1-t)\,t^2\mathbf{p}_2 + t^3\mathbf{p}_3\end{aligned}$$

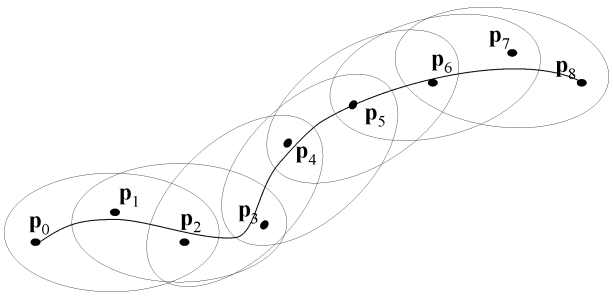


Figure 4.4: Illustration of second-order B-splines

- for a given $n$, we obtain:

$$\mathbf{f}(t) = \sum_{i=0}^{n} \underbrace{\frac{n!}{i!\,(n-i)!}\,(1-t)^{n-i}\,t^i}_{b_{i,n}(t)}\,\mathbf{p}_i.$$

The polynomials $b_{i,n}(t)$, called *Bernstein polynomials*, form a basis of the space of $n$-degree polynomials. When we increase the degree (in other words the number of control points), numerical instability and oscillations appear. This is called *Runge's phenomenon*. For complex curves with hundreds of control points, it is preferable to use B-splines corresponding to a concatenation of Bézier curves of limited order. Figure 4.4 illustrates such a concatenation in which, for each group of three points, we can calculate a second-order Bézier polynomial.

## 4.3 Voronoi diagram

Let us consider $n$ points $\mathbf{p}_1, \ldots, \mathbf{p}_n$. Contrarily to the previous sections, the $\mathbf{p}_i$ here do not correspond to control points, but to point obstacles that we try to avoid. To each of these points, we associate the set:

$$\mathbb{P}_i = \left\{\mathbf{x} \in \mathbb{R}^d, \forall j, \|\mathbf{x} - \mathbf{p}_i\| \leq \|\mathbf{x} - \mathbf{p}_j\|\right\}.$$

For all $i$, this set is a polygon. The collection of these $\mathbb{P}_i$ is called a *Voronoi diagram*. Figure 4.5 represents a set of points with the associated Voronoi diagram. If an environment contains obstacles, the robot will have to plan a path that remains on the borders of the $\mathbb{P}_i$.

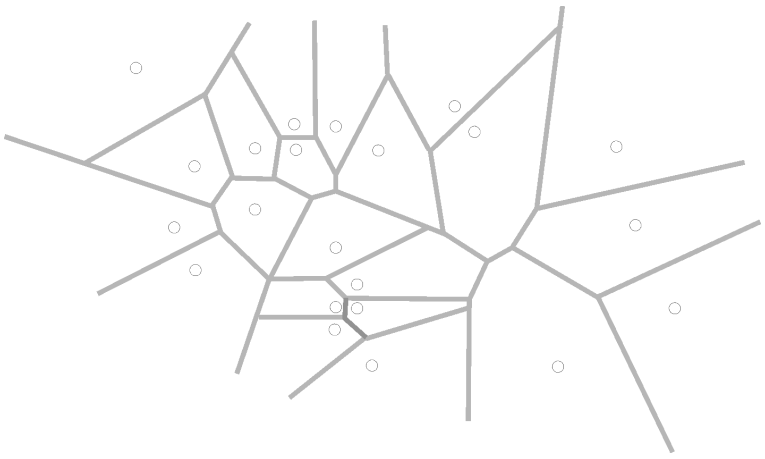

Figure 4.5: Voronoi diagram

**Delaunay triangulation.** Given $n$ points in space, we can use the Voronoi diagrams to perform a triangulation of the space. This corresponding triangulation, referred to as *Delaunay triangulation*, allows maximizing the quantity of acute angles and thus avoid elongated triangles. It is obtained by connecting the neighboring points of the corresponding regions with an edge in the Voronoi diagram. In a Delaunay triangulation none of the triangles contains another point within its circumscribed

circle. Figure 4.6 represents the Delaunay triangulation associated with the Voronoi diagram in Figure 4.5. A Delaunay triangulation is often used in robotics to represent space such as for example the area already explored, restricted areas, lakes, etc. We often associate a color with each triangle following the characteristics of the space the triangle belongs to (water, land, road, etc.).

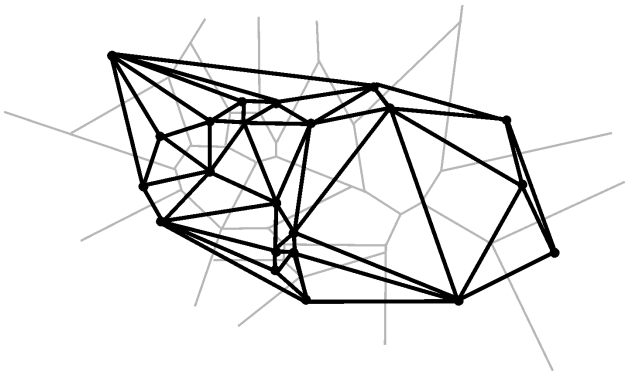

Figure 4.6: Delaunay triangulation

## 4.4 Artificial potential field method

A mobile robot has to move in a congested environment that contains mobile and stationary obstacles. The artificial potential field method [LAT 91] consists of imagining that the robot can behave like an electric particle that can be attracted or repelled by other objects following the sign of their electric charge. This is a reactive approach to guidance in which the path is not planned in advance. In physics, we have the following relation:

$$\mathbf{f} = -\text{grad}V\left(\mathbf{p}\right),$$

where $\mathbf{p}$ is the position of point particle in space, $V$ is the potential and $\mathbf{f}$ the force applied on the particle. We will have the same relation in mobile robotics, but with $\mathbf{p}$ the position of the center of the robot, $V$ a potential imagined by the robot and $\mathbf{f}$ the speed vector to follow. The potential fields will help us express a desired behavior for a robot. The obstacles will be represented by potentials exerting a repulsive force on the robot while the objective to follow will exert an attractive force. In a situation where several robots need to remain grouped while avoiding collisions, we can use a near-field repulsive potential and a far-field attractive potential. More generally, the vector fields used might not derive from a potential, as is the case if we would like the robot to have a cyclic behavior. The following table gives several types of potential that can be used:

| **Potential** | $V\left(\mathbf{p}\right)$ | $-\text{grad}(V\left(\mathbf{p}\right))$ |
|---|---|---|
| attractive conical | $\left\|\mathbf{p}-\hat{\mathbf{p}}\right\|$ | $-\frac{\mathbf{p}-\hat{\mathbf{p}}}{\left\|\mathbf{p}-\hat{\mathbf{p}}\right\|}$ |
| attractive quadratic | $\left\|\mathbf{p}-\hat{\mathbf{p}}\right\|^2$ | $-2\left(\mathbf{p}-\hat{\mathbf{p}}\right)$ |
| attractive plane or line | $\left(\mathbf{p}-\hat{\mathbf{p}}\right)^{\mathrm{T}}\cdot\hat{\mathbf{n}}\,\hat{\mathbf{n}}^{\mathrm{T}}\cdot\left(\mathbf{p}-\hat{\mathbf{p}}\right)$ | $-2\,\hat{\mathbf{n}}\,\hat{\mathbf{n}}^{\mathrm{T}}\left(\mathbf{p}-\hat{\mathbf{p}}\right)$ |
| repulsive | $\frac{1}{\left\|\mathbf{p}-\hat{\mathbf{q}}\right\|}$ | $\frac{\left(\mathbf{p}-\hat{\mathbf{q}}\right)}{\left\|\mathbf{p}-\hat{\mathbf{q}}\right\|^3}$ |
| uniform | $-\hat{\mathbf{v}}^{\mathrm{T}}\cdot\mathbf{p}$ | $\hat{\mathbf{v}}$ |

In this table, $\hat{\mathbf{p}}$ represents an attractive point, $\hat{\mathbf{q}}$ a repulsive point and $\hat{\mathbf{v}}$ a desired speed for the robot. In the case of the attractive plane, $\hat{\mathbf{p}}$ is a point of the plane and $\hat{\mathbf{n}}$ is a vector orthonormal to the plane. By adding several potentials, we can ask the robot (which is supposed to follow the direction that tends to decrease the potential) to accomplish its objectives while moving away from the obstacles. Figure 4.7 represents three vector fields derived from artificial potentials. The one on the left corresponds to a uniform field, the one in the middle to a repulsive potential that is added to a uniform field and the one on the right to the sum of an attractive potential and a repulsive potential.

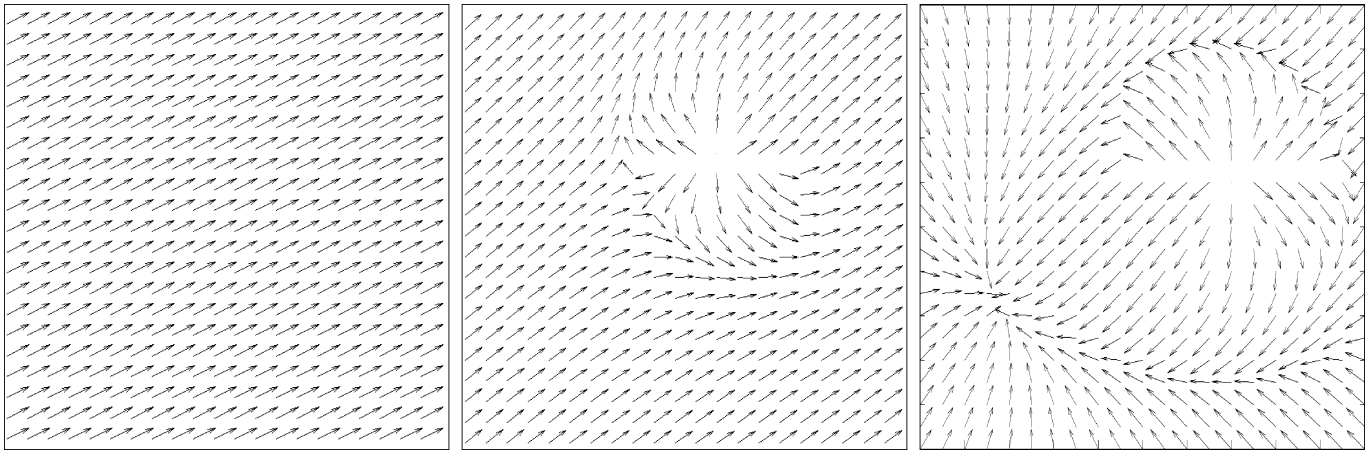

Figure 4.7: Artificial potential fields

# Exercises

EXERCISE 4.1.– *Pursuit on a sphere*

Consider a robot $\mathcal{R}$ moving on the surface of a sphere similar to that of the Earth, with a radius of $\rho = 30$ m. This robot is located by its longitude $\ell_x$, its latitude $\ell_y$ and its heading $\psi$, relative to the East. In a local coordinate system, the state equations of the robot are of the type:

$$\left\{ \begin{array}{rcl} \dot{x} & = & \cos\psi \\ \dot{y} & = & \sin\psi \\ \dot{\psi} & = & u \end{array} \right.$$

1) Give the state equations in the case in which the state vector is $(\ell_x, \ell_y, \psi)$.

2) Simulate this evolving system graphically in 3D, in MATLAB.

3) A second robot $\mathcal{R}_a$, described by the same equations, is moving randomly on the sphere (see Figure 4.8). Suggest a control law that allows the robot $\mathcal{R}$ to meet the robot $\mathcal{R}_a$. You can start with the program `ex_earth.m`.

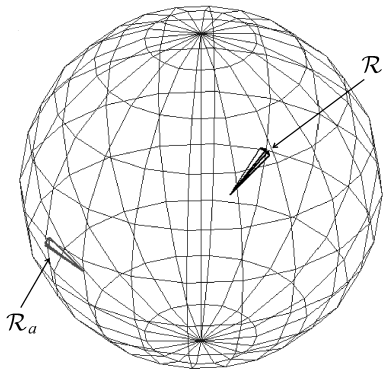


Figure 4.8: On the sphere, the robot $\mathcal{R}$ follows the robot $\mathcal{R}_a$

---

EXERCISE 4.2.– *Planning a path*

Let us consider a scene with two triangles as shown on Figure 4.9, and a robot described by the

state equations:

$$\begin{cases} \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= u_1 \\ \dot{v} &= u_2 \end{cases}$$

with initial state $(x, y, \theta, v) = (0, 0, 0, 1)$. This robot has to reach the point with coordinates $(8, 8)$.

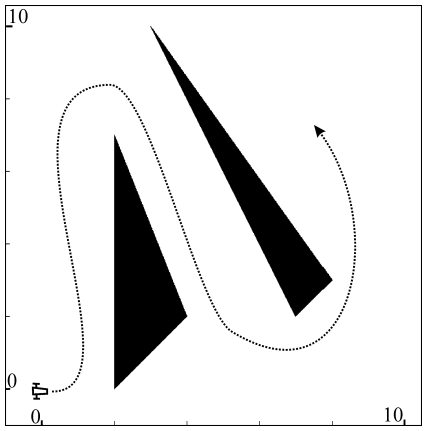


Figure 4.9: The robot has to follow a path without hitting the obstacles

1) In MATLAB, find the control points for a Bézier polynomial that connects the initial position to the desired position as shown on the Figure. You can start with the program `ex_bezier.m`.

2) Using feedback linearization, deduce the control law that allows to reach the objective in 50 sec.

---

EXERCISE 4.3.– *Drawing a Voronoi diagram*

Let us consider the ten points in Figure 4.10. Draw the associated Voronoi diagram on a piece of paper as well as the corresponding Delaunay triangulation.

---

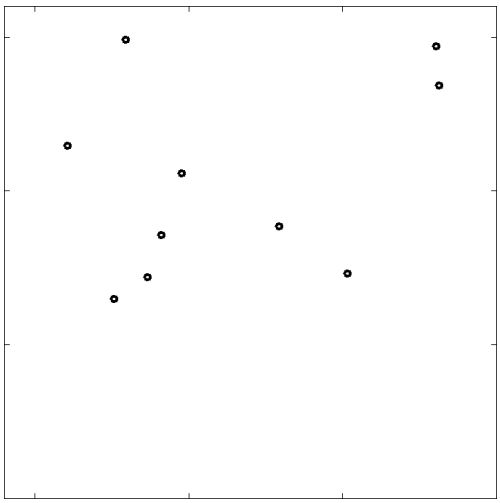

Figure 4.10: Ten points for which we want to build a Voronoi diagram

EXERCISE 4.4.– *Calculating a Voronoi diagram*

1) Show that if $\mathbf{x}$ and $\mathbf{y}$ are two vectors of $\mathbb{R}^n$, we have the so-called *polarization* equations:

$$\left\{ \|\mathbf{x}-\mathbf{y}\|^2 = \|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 - 2\langle \mathbf{x}, \mathbf{y}\rangle \right.$$

For this, develop the expression of the scalar product $\langle \mathbf{x}-\mathbf{y}, \mathbf{x}-\mathbf{y}\rangle$.

2) Let us consider $n+1$ points $\mathbf{a}^1, \ldots, \mathbf{a}^{n+1}$ and their circumscribed sphere denoted by $\mathcal{S}$. With the aid of the previous question, give an expression, in function of the $\mathbf{a}^i$, of the center $\mathbf{c}$ of $\mathcal{S}$ and of its radius $r$.

3) Let us now consider three points in the plane $\mathbf{a}^1, \mathbf{a}^2, \mathbf{a}^3$. Which conditions must be verified for $\mathbf{m}$ to be within the circle circumscribed to the triangle $(\mathbf{a}^1, \mathbf{a}^2, \mathbf{a}^3)$ ?

4) Consider $m$ points in the plane $\mathbf{p}^1, \ldots, \mathbf{p}^m$, a Delaunay triangulation is a partition of the space into triangles $\mathcal{T}(k) = (\mathbf{a}^1(k), \mathbf{a}^2(k), \mathbf{a}^3(k))$, whose vertices are taken from the $\mathbf{p}^i$ such that each circle $\mathcal{C}(k)$ circumscribed to this triangle $\mathcal{T}(k)$ contains none of the $\mathbf{p}^i$. Starting with the program `ex_voronoi.m`, write a MATLAB program that takes $m = 10$ random points of the plane and draws a Delaunay triangulation. What is the complexity of the algorithm ?

5) Given the triangulation established in the previous question, build a Voronoi diagram associated with the points $\mathbf{a}^1, \ldots, \mathbf{a}^{n+1}$.

---

EXERCISE 4.5.– *Heading control of a Dubins car*

The results of this exercise will be used in Exercise 4.6 for calculating Dubins paths. A Dubins car is described by the state equations:

$$\left\{ \begin{array}{rcl} \dot{x} &=& \cos\theta \\ \dot{y} &=& \sin\theta \\ \dot{\theta} &=& u \end{array} \right.$$

$\theta$ is the robot's heading and $(x, y)$ the coordinates of its center. This robot has to be aligned with a heading setpoint $\bar{\theta}$.

1) We assume that the input $u \in [-1, 1]$. Give an analytic expression of the error angle $\delta \in [-\pi, \pi]$ in function of $\theta$ and $\bar{\theta}$ that indicates the angle the robot has to turn by in order to reach its setpoint as fast as possible. Taken into account that the expression of $\delta$ has to be periodic relative to $\theta$ and $\bar{\theta}$. Indeed, the angles of $-\pi, \pi$ or $3\pi$ have to be considered as equivalent. Give the associated control law and simulate it in MATLAB. You can start with the program `ex_dubins.m`.

2) Same as above, with the exception that the robot can only turn left (in the direct trigonometric sense), in other words $u \in [0, 1]$. Let us note that now, $\delta \in [0, 2\pi]$.

3) Same as above, but the robot can now only turn right, i.e. $u \in [-1, 0]$.

---

EXERCISE 4.6.– *Dubins paths*

As in the previous exercise, let us consider a robot moving on a plane, described by:

$$\begin{cases} \dot{x} &= \cos\theta \\ \dot{y} &= \sin\theta \\ \dot{\theta} &= u \end{cases}$$

where $\theta$ is the robot's heading and $(x, y)$ the coordinates of its center. Its state vector is given by $\mathbf{x} = (x, y, \theta)$ and its input $u$ must remain within the interval $[-u_{\max}, u_{\max}]$. This is the Dubins car [DUB 57] corresponding to the simplest possible nonholonomic mobile vehicle. Despite its simplicity, it illustrates many difficulties that may appear within the context of nonholonomic robots.

1) Calculate the maximum radius of curvature $r$ that can be executed by the path of the robot.

2) Dubins showed that in order to switch from a configuration $\mathbf{a} = (x_a, y_a, \theta_a)$ to a configuration $\mathbf{b} = (x_b, y_b, \theta_b)$ that aren't too close together (in other words separated by a distance superior to $4r$), the minimum time strategy always consists of (1) turning to the maximum in one direction (in other words $u = \pm u_{\max}$) ; (2) moving straight ahead ; (3) then turning to the maximum again. The path corresponding to such a maneuver is called a *Dubins path*, and is thus composed of a starting arc, a segment and a termination arc. There are four ways to construct a Dubins path: LSL, LSR, RSL, RSR, where L means left, R means right and S stands for straight ahead. Give a configuration for $\mathbf{a}$ and $\mathbf{b}$ such that none of the four paths corresponds to an optimal strategy (and therefore $\mathbf{a}$ and $\mathbf{b}$ are quite close together). A situation in which the optimal strategy is RLR will be chosen.

3) In the case of an RSL strategy as illustrated on Figure 4.11, calculate the length $L$ of the Dubins path in function of $\mathbf{a}$ and $\mathbf{b}$.

4) In the case of an LSL strategy, calculate the length $L$ of the Dubins path in function of $\mathbf{a}$ and $\mathbf{b}$.

5) By using the reflection symmetry of the problem, deduce $L$ in the case of RSR and LSR strategies, then write a MATLAB function that calculates $L$ in all situations. For this, the two Booleans $\varepsilon_a$ ,$\varepsilon_b$ will be used, which are equal to 1 if the corresponding arc is in the forward direction (i.e. to the left) and $-1$ otherwise. You can start with the program `ex_dubinspath.m`.

6) Use the previous questions to write a MATLAB program that calculates the minimum length path for a Dubins car.

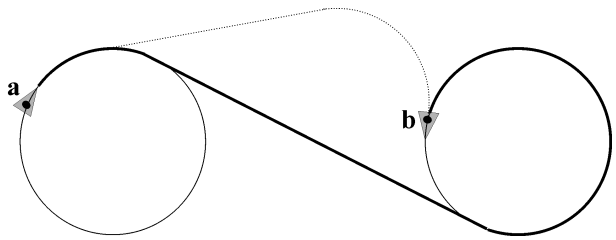


Figure 4.11: In bold : an RSL-type (*Right-Straight-Left*) Dubins path leading from **a** to **b** ; in dotted : an RLR-type Dubins path

---

EXERCISE 4.7.– *Artificial potentials*

A robot situated at $\mathbf{p} = (x, y)$ must reach a target of unknown movement whose position $\hat{\mathbf{p}}$ and speed $\hat{\mathbf{v}}$ are known at the present time. This pair $(\hat{\mathbf{p}}, \hat{\mathbf{v}})$ might for instance correspond to a setpoint given by a human operator. A fixed obstacle located at position $\hat{\mathbf{q}}$ must be avoided. We model the desired behavior of our robot by the potential:

$$V(\mathbf{p}) = -\hat{\mathbf{v}}^{\mathrm{T}} \cdot \mathbf{p} + \|\mathbf{p} - \hat{\mathbf{p}}\|^2 + \frac{1}{\|\mathbf{p} - \hat{\mathbf{q}}\|}$$

where the potential $-\hat{\mathbf{v}}^{\mathrm{T}} \cdot \mathbf{p}$ represents the speed setpoint, the potential $\|\mathbf{p} - \hat{\mathbf{p}}\|^2$ makes the target position $\hat{\mathbf{p}}$ attractive and the potential $\frac{1}{\|\mathbf{p} - \hat{\mathbf{q}}\|}$ makes the obstacle $\hat{\mathbf{q}}$ repulsive.

1) Calculate the gradient of the potential $V(\mathbf{p})$ and deduce the speed vector setpoint $\mathbf{w}(\mathbf{p}, t)$ to apply to our robot so that it responds correctly to this potential.

2) We assume that our robot obeys the following state equations:

$$\begin{cases} \dot{x} &= v \cos\theta \\ \dot{y} &= v \sin\theta \\ \dot{v} &= u_1 \\ \dot{\theta} &= u_2 \end{cases}$$

Give the control law that corresponds to the desired potential field. We will use the same principle as the one shown in Figure 4.12. First of all, disassemble the robot (in grey on the figure) into a chain made of two blocks. The first block forms the speeds from the actuators and the second builds the position vector $\mathbf{p} = (x, y)$. Then calculate the left inverse of the first block in order to end up with a system of the type:

$$\begin{cases} \dot{x} &= \bar{v} \cos\bar{\theta} \\ \dot{y} &= \bar{v} \sin\bar{\theta} \end{cases}$$

Use a simple proportional control to perform this approximate inversion. Then, generate the new input $\left(\bar{v}, \bar{\theta}\right)$ using the potential to be satisfied. Illustrate the behavior of the robot in MATLAB with a target $\hat{\mathbf{p}} = (t, t)$ and a fixed obstacle placed at $\hat{\mathbf{q}} = (4, 5)$. You can start with the program `ex_potential.m`.

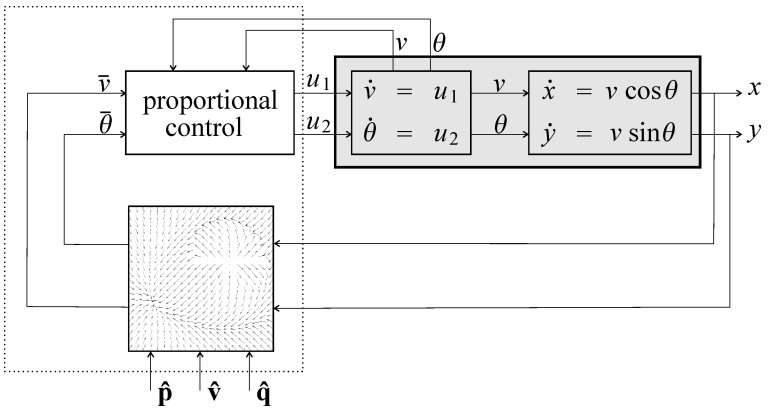


Figure 4.12: Controller (dotted) obtained by the potential method

3) We would now like to follow the target $\hat{\mathbf{p}}$ with a mobile obstacle at $\hat{\mathbf{q}}$ with:

$$\hat{\mathbf{p}} = \begin{pmatrix} \cos\frac{t}{10} \\ 2\sin\frac{t}{10} \end{pmatrix} \text{ and } \hat{\mathbf{q}} = \begin{pmatrix} 2\cos\frac{t}{5} \\ 2\sin\frac{t}{5} \end{pmatrix}.$$

Adjust the parameters of the potential in order to follow the target without hitting the obstacle. Illustrate the behavior of the controlled robot in MATLAB.

---

EXERCISE 4.8.– *Flocking*

We consider $m = 20$ robots described by the following state equations:

$$\begin{cases} \dot{x}_i &= \cos\theta_i \\ \dot{y}_i &= \sin\theta_i \\ \dot{\theta}_i &= u_i \end{cases}$$

The state vector is $\mathbf{x}(i) = (x_i, y_i, \theta_i)$. These robots can see all other robots, but are not able to communicate with them. We want that there robots behave as a flock as illustrated by Figure 4.13. Basic models of flocking behavior are controlled by three rules of Reynolds: the *separation* (short range repulsion) the *alignment* and the *cohesion* (long range attraction). Using a potential based method, find a controller for each robot to obtain a flock. You can start with the program `ex_flocking.m`.

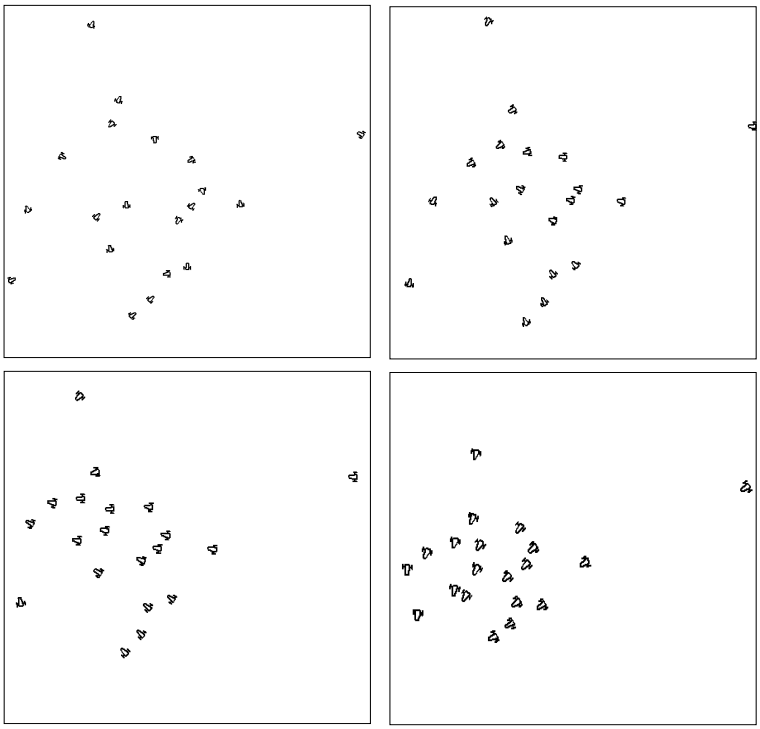

Figure 4.13: Illustration of the flocking behavior from a random initialization

# Chapter 5

# Instantaneous localization

Localization consists of finding the position of the robot (in other words the coordinates of its center as well as its orientation), or more generally all its degrees of freedom. This problem is encountered in navigation, where we need to approximate the position, orientation and speed of the robot. The problem of localization is often considered to be a particular case of state estimation, which will be presented in the following chapters. However, in the case where an accurate state model is not available for our robot, an instantaneous localization often remains possible and may be sufficient for making a decision. Let us take for instance the situation in which we are aboard a ship and have just detected a lighthouse whose absolute position and height are known. By measuring the perceived height of the lighthouse and its angle relative to the ship, we may deduce the position of the ship using a compass and this, without using a state model for the ship. Instantaneous, or *model-free* localization is an approach to localization that does not utilize the evolution equation of the robot, in other words it does not seek to make the measures coherent through time. This localization mainly consists of solving equations of geometric nature which are often nonlinear. The variables involved may be position variables or kinematic variables such as speed or accelerations. Since these localization methods are specific and quite far from state estimation methods, we will devote an entire chapter to them. After introducing the main sensors used for localization, we will present goniometric localization (in which the robot uses the angles of perception of landmarks) followed by multilateration which uses distances between the robot and the landmarks.

## 5.1 Sensors

The robots are equipped with numerous sensors that are used for their localization. We will now present some of these.

**Odometers**. Robots with wheels are generally equipped with odometers that measure the angular movements of the wheels. Given only the odometers, it is possible to calculate an estimation of the position of the robot. The precision of such a localization is very low given the systematic integration of the estimation error. We say that the estimation is drifting.

**Doppler log**. This type of sensor, mainly used in underwater robotics, allows to calculate the speed of the robot. A Doppler log emits ultrasounds that are reflected by the ocean bed. Since the ocean bed is immobile, the sensor is able to estimate the speed of the robot by using the Doppler

effect with a very high precision (around 0.1 $m/s$).

**Accelerometer**s. These sensors provide measurements of the instantaneous forward acceleration. The principle of the axis-based accelerometer is illustrated on Figure 5.1. Generally, three accelerometers are used by the robot. Due to gravity, the value measured according to the vertical axis must be compensated.

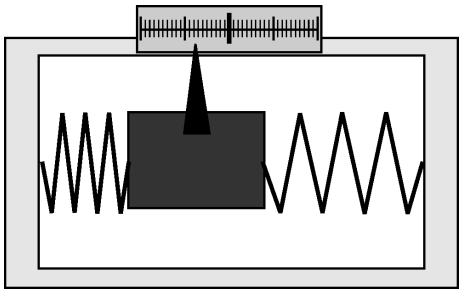

Figure 5.1: Operating principle of an accelerometer

**Gyro or gyroscope**. These sensors provide measurements of the instantaneous rotation speed. There are three types of gyros: the Coriolis vibratory gyro, the mechanical gyro and the optical gyro. The principle of the Coriolis vibratory gyro is illustrated on the left of Figure 5.2. A vertical rod placed on a horizontal disk vibrates from left to right. As a result of the Coriolis force, if the disk rotates there is an angular vibration following the axis of the rod whose amplitude allows to get the rotation speed of the disk. If the disk is not rotating, there is a forward rotation, but it is not angular. Piezoelectric gyros, very widely used for low-cost robotics, form a subclass of Coriolis vibratory gyroscopes. These gyros exploit the variation of the amplitude of a piezoelectric oscillator induced by the Coriolis force, due to the rotation applied to the sensor. Mechanical gyros make use of the fact that a rotating body tends to preserve its rotational axis if no torque is subjected to it. A well-known example is the gimbal gyroscope invented by Foucault, represented on the right side of Figure 5.2. A flywheel at the center rotates with high speed. If the base of the gyroscope moves, the two gimbal angles $\psi, \theta$ will change, but the rotation axis of the flywheel will not. From the values of $\psi, \theta, \dot{\psi}, \dot{\theta}$, we can find the rotation speed of the base (which is fixed on the robot). If the rotation axis of the flywheel is initialized correctly, and in a perfect situation in which no torque is exerted on this flywheel, such a system would theoretically give us the orientation of the robot. Unfortunately, there is always a small drift and only the rotation speed can be given in a reliable and drift-free way.

More recent, optical gyroscopes can be as precise as mechanical ones. They make use of the Sagnac effect (for a circular optical path, the time taken by light to make a complete lap depends on the direction of the path) and have a precision of around 0.001 deg/s. Their principle is illustrated in Figure 5.3. On figure (a), the laser leaves the light source represented by the black disk. On figure (b), the beam splitter creates two other beams which travel in opposite directions in the optical loop. After rebounding several times on the three mirrors represented in grey, the two beams meet. Since the beams intersect on the left, the gyro rotates in the opposite trigonometric direction (c). The beams are separated again on figure (d). The two beams that arrive at the receiver are not in phase. Their phase offset allows to find the rotation speed of the gyro, which is fixed on the robot.

**Inertial unit.** An *inertial measurement unit* associates a gyro and an accelerometer in order to increase the precision of the estimation. More recent ones merge other types of information such as

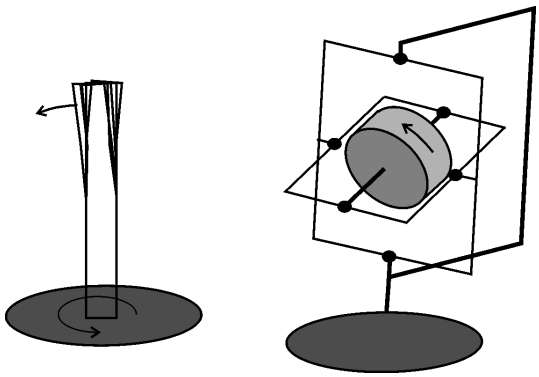

Figure 5.2: Coriolis vibratory gyroscope and gimbal gyroscope

the estimated speed or even take into account Earth's rotation. For instance, the Octans III unit of the IXBLUE company uses the Sagnac effect together with Earth's rotation in order to deduce the direction of the Earth's North-South axis in the robot's coordinate system. Knowing this direction gives us two equations involving the Euler angles of the robot (see paragraph 1.2) which are the bank $\phi$, the elevation $\theta$ and the heading $\psi$ of the robot, expressed in a local coordinate system. Due to the accelerometer included in the unit, it is possible to deduce the gravity vector from the above and thus to generate an additional equation which will allow to calculate the three Euler angles. Let us note that the accelerometers also give us the accelerations in all directions (the *surge* in the direction of the robot, the *heave* (in the vertical direction), the *sway* for lateral accelerations). The knowledge of the gravity vector and the axis of rotation theoretically allows, using a simple scalar product, to find the latitude of the robot. However, the obtained precision is too low to be taken into account in localization.

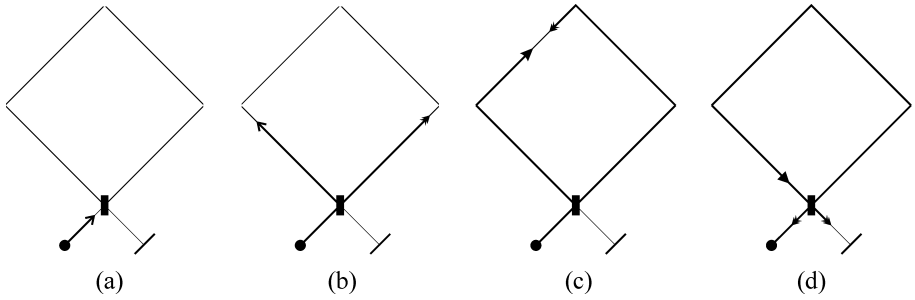


Figure 5.3: Principle of the Sagnac effect for optical gyroscopes

**Barometer**. This measures pressure. In the case of underwater robots, it allows to deduce the depth of the robot with a precision of 1 cm. For indoor flying robots, the barometer is used to measure the altitude with a precision of around one meter.

**GPS** (*Global Positioning System*). The GNSS (*Global Navigation Satellite System*) is a satellite navigation system that provides a geolocalization service covering the entire world. Nowadays, the American NAVSTAR system (*NAVigation System by Timing And Ranging*) and the Russian GLONASS system (GLObalnaya NAvigazionnaya Sputnikovaya Sistema) are operational. Two other systems are being developed: the Chinese *Compass* system and the European *Galileo* system. In practice, our mobile robots will use the American system, operational since 1995, that we will refer to

as GPS. Originally designed for exclusive military use, the precision of civil applications was limited to several hundreds of meters by a deliberate degradation of civil signals. The deactivation of this degradation in 2000 allowed the precision to increase to about ten meters. Given that electromagnetic waves (here around 1.2 MHz) do not propagate under water or across walls, the GPS does not function within buildings or in water. Thus, during a diving experiment, an underwater robot can only be localized by GPS when it begins its dive or when it resurfaces. When a georeferenced station is near the robot and advises it about the errors in distance calculated for each satellite, a localization with a precision of $\pm 1$ m is possible. This operating mode forms the so-called differential GPS or DGPS. Finally, by using the phase, it us possible to achieve even a centimeter precision. This is the principle of the *kinematic GPS*. A detailed and educational presentation of the GPS can be found in the thesis of Vincent Drevelle [DRE 11]. In practice, a GPS gives us a longitude $\ell_x$ and a latitude $\ell_y$ and it is often comfortable to convert it to Cartesian coordinates in a local coordinate system $(\mathbf{o}, \mathbf{i}, \mathbf{j}, \mathbf{k})$ fixed within the area in which the robot is evolving. Let us denote by $\ell_x^0$ and $\ell_y^0$ the longitude and the latitude expressed in radians at the origin $\mathbf{o}$ of this coordinate system. We will assume that the vector $\mathbf{i}$ indicates the North, $\mathbf{j}$ indicates the East and $\mathbf{k}$ is oriented towards the center of the Earth. Let $\mathbf{p} = (p_x, p_y, p_z)$ be the coordinates of the robot expressed in the coordinate system $(\mathbf{o}, \mathbf{i}, \mathbf{j}, \mathbf{k})$. From the latitude and longitude given by the GPS, we can deduce the first two coordinates of the robot, expressed in meters in the local coordinate system, by using the following relation (see Equation [4.4] on page 108):

$$\begin{pmatrix} p_x \\ p_y \end{pmatrix} = \rho \begin{pmatrix} 0 & 1 \\ \cos(\ell_y) & 0 \end{pmatrix} \begin{pmatrix} \ell_x - \ell_x^0 \\ \ell_y - \ell_y^0 \end{pmatrix} = \begin{pmatrix} \rho \left(\ell_y - \ell_y^0\right) \\ \rho \cos(\ell_y) (\ell_x - \ell_x^0) \end{pmatrix}$$

where $\rho$ corresponds to the distance between $\mathbf{o}$ and the center of the Earth ($\rho \simeq 6\ 371$ km, if $\mathbf{o}$ is not too far from sea level). This formula is valid everywhere on Earth, if we assume that the Earth is spherical and if the robot is in the neighborhood of the origin $\mathbf{o}$ (let us say a distance inferior to 100 km). In order to understand this formula, we must note that $\rho \cos(\ell_y)$ corresponds to the distance between $\mathbf{o}$ and the rotational axis of the Earth. Thus, if a robot is moving on a latitude parallel $\ell_y$, by modifying its longitude by an angle $\alpha > 0$, it will have traveled $\alpha\rho\cos(\ell_y)$ meters. Similarly, if this robot is moving on a meridian with an angle $\beta$ in latitude, it will have traveled $\beta\rho$ meters.

**Radar or sonar.** The robot emits electromagnetic or ultrasound waves. It recovers their echoes and builds an image that it interprets in order to map its surroundings. The radar is mainly used by surface or flying robots. The sonar is used as a low-cost rangefinder by robots with wheels as well as in underwater robotics.

**Cameras**. Cameras are low-cost sensors used for the recognition of objects. In localization, they are used as goniometers, in other words they allow to find the angles relative to landmarks that will then be used for localization. Cameras are also used for recognizing objects in an underwater context [BAZ 12].

## 5.2 Goniometric localization

### 5.2.1 Formulation of the problem

The problem consists of using angles measured between the robot and the landmarks, whose position as a function of time is known, for localization. Let us consider the robot in Figure 5.4 moving on a plane. We call *bearing* the angle $\alpha_i$ between the axis of the robot and the vector pointing towards the landmark. These angles could have been obtained, for instance, using a camera.

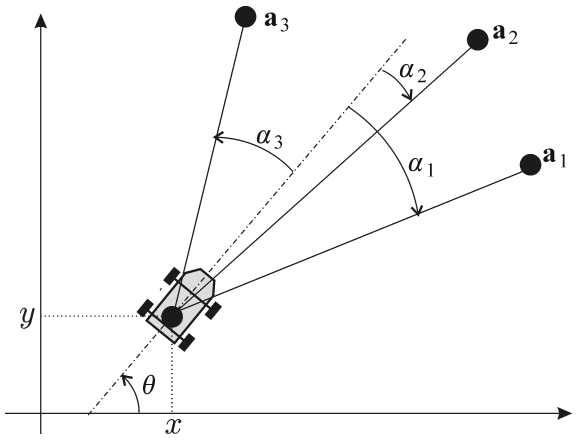


Figure 5.4: A robot moving on a plane, measures the angles in order to locate itself

Recall that two vectors $\mathbf{u}, \mathbf{v}$ of $\mathbb{R}^2$ are collinear if their determinant is zero, in other words if $\det(\mathbf{u}, \mathbf{v}) = 0$. Thus, for each landmark, we have the relation:

$$\det\left(\begin{pmatrix} x_i - x \\ y_i - y \end{pmatrix}, \begin{pmatrix} \cos(\theta + \alpha_i) \\ \sin(\theta + \alpha_i) \end{pmatrix}\right) = 0$$

in other words:

$$(x_i - x)\sin(\theta + \alpha_i) - (y_i - y)\cos(\theta + \alpha_i) = 0 \tag{5.1}$$

where $(x_i, y_i)$ are the coordinates of the landmark $\mathbf{a}_i$ and $\theta$ is the robot's heading.

### 5.2.2 Inscribed angles

THEOREM 5.1.– ***Inscribed Angle Theorem****: Consider a triangle* **abm** *as represented on Figure 5.5. Let us denote by* **c** *the center of the circle circumscribed to this triangle (in other words that* **c** *is at the intersection of the three perpendicular bisectors). Let* $\alpha = \widehat{\mathbf{amc}}$, $\beta = \widehat{\mathbf{cmb}}$, $\gamma = \widehat{\mathbf{acb}}$. *We have the angular relation:*

$$\gamma = 2(\alpha + \beta)$$

PROOF.– First of all, the two triangles **amc** and **cmb** are isosceles. We thus find the angles $\alpha$ and

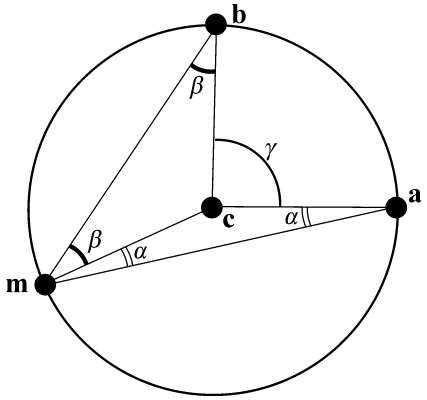


Figure 5.5: Illustration of the Inscribed Angle Theorem

$\beta$ as shown on the figure. By going around the point **c** we obtain:

$$\gamma + (\pi - 2\beta) + (\pi - 2\alpha) = 2\pi$$

In other words $\gamma = 2\alpha + 2\beta$. ■

A consequence of this theorem is that if **m** moves on the circle, the angle $\alpha + \beta$ will not move.

**Inscribed arcs**. Let us consider two points $\mathbf{a}_1$ and $\mathbf{a}_2$. The set of points **m** such that the angle $\widehat{\mathbf{a}_1\mathbf{m}\mathbf{a}_2}$ is equal to $\alpha$ is a circle arc, referred to as an *inscribed arc*. We can show this from Relations [5.1] or from the Inscribed Angle Theorem. Goniometric localization often breaks down to intersecting arcs. Figure 5.6 illustrates the concept of an inscribed arc.

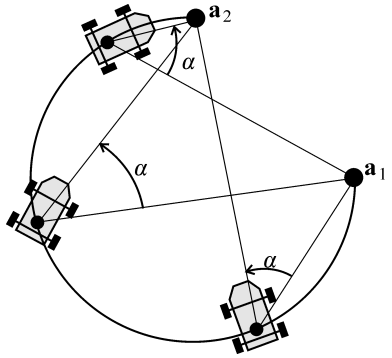


Figure 5.6: The three cars perceive the landmarks with the same angle

### 5.2.3 Static triangulation of a plane robot

#### 5.2.3.1 Two landmarks and a compass

In the case where we have two landmarks and a compass, we have, following [5.1], the two relations:

$$\begin{cases} (x_1 - x)\sin(\theta + \alpha_1) - (y_1 - y)\cos(\theta + \alpha_1) &= 0 \\ (x_2 - x)\sin(\theta + \alpha_2) - (y_2 - y)\cos(\theta + \alpha_2) &= 0 \end{cases}$$

in other words:

$$\underbrace{\begin{pmatrix} \sin(\theta+\alpha_1) & -\cos(\theta+\alpha_1) \\ \sin(\theta+\alpha_2) & -\cos(\theta+\alpha_2) \end{pmatrix}}_{\mathbf{A}(\theta,\alpha_1,\alpha_2)} \begin{pmatrix} x \\ y \end{pmatrix} = \underbrace{\begin{pmatrix} x_1\sin(\theta+\alpha_1) - y_1\cos(\theta+\alpha_1) \\ x_2\sin(\theta+\alpha_2) - y_2\cos(\theta+\alpha_2) \end{pmatrix}}_{\mathbf{b}(\theta,\alpha_1,\alpha_2,x_1,y_1,x_2,y_2)}$$

i.e.:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \mathbf{A}^{-1}(\theta,\alpha_1,\alpha_2)\cdot\mathbf{b}(\theta,\alpha_1,\alpha_2,x_1,y_1,x_2,y_2)$$

The problem of localization is therefore a linear one, which can be solved analytically. We have an identifiability problem if the matrix to invert has zero determinant, in other words:

$$\begin{aligned} & \sin(\theta+\alpha_1)\cos(\theta+\alpha_2) = \cos(\theta+\alpha_1)\sin(\theta+\alpha_2) \\ \Leftrightarrow\ & \tan(\theta+\alpha_2) = \tan(\theta+\alpha_1) \\ \Leftrightarrow\ & \theta+\alpha_2 = \theta+\alpha_1+k\pi,\ k\in\mathbb{N} \\ \Leftrightarrow\ & \alpha_2 = \alpha_1+k\pi,\ k\in\mathbb{N} \end{aligned}$$

This corresponds to a situation in which the two landmarks and the robot are aligned.

#### 5.2.3.2 Three landmarks

If we no longer have a compass, we need at least three landmarks. We then need to solve the system of three equations and three unknowns:

$$(x_i - x)\sin(\theta+\alpha_i) - (y_i - y)\cos(\theta+\alpha_i) = 0,\ i\in\{1,2,3\}$$

It can be shown that this system always has a unique solution, except when the robot is located on the circle that passes through all three landmarks. Indeed, in such a case the inscribed angles are superimposed.

### 5.2.4 Dynamic triangulation

#### 5.2.4.1 One landmark, a compass, several odometers

In the case of dynamic state observation, we are looking for the relation that connects the position of the robot to the derivatives of the values measured. For localization, we will assume that a single landmark is available to us. We will use the equations:

$$\begin{cases} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \end{cases} \tag{5.2}$$

where $v$ represents the speed of the robot measured by the odometer and $\theta$ its heading measured by the compass. These equations, which are kinematic in nature, are not supposed to describe the behavior of a particular robot with the aim of controlling it. The inputs $v$ and $\theta$ are not necessarily the real inputs of the system that we can act on. These equations have to be understood as a simple

differential relation between the variables of a plane robot. By differentiating Relation (5.1), we obtain:

$$\begin{array}{l} (\dot{x}_i - \dot{x}) \sin(\theta + \alpha_i) + (x_i - x) \left(\dot{\theta} + \dot{\alpha}_i\right) \cos(\theta + \alpha_i) \\ -(\dot{y}_i - \dot{y}) \cos(\theta + \alpha_i) + (y_i - y) \left(\dot{\theta} + \dot{\alpha}_i\right) \sin(\theta + \alpha_i) \quad = \quad 0 \end{array} \tag{5.3}$$

Let us take the relations [5.1] and [5.3] for $i = 1$. By isolating $x$ and $y$, we obtain:

$$\begin{array}{rcl} \begin{pmatrix} x \\ y \end{pmatrix} & = & \begin{pmatrix} \sin(\theta + \alpha_1) & \cos(\theta + \alpha_1) \\ -\cos(\theta + \alpha_1) & \sin(\theta + \alpha_1) \end{pmatrix} \\ & & \cdot \begin{pmatrix} -y_1 & x_1 \\ x_1 - \frac{\dot{y}_1 - v\sin\theta}{\dot{\theta} + \dot{\alpha}_1} & \frac{\dot{x}_1 - v\cos\theta}{\dot{\theta} + \dot{\alpha}_1} + y_1 \end{pmatrix} \begin{pmatrix} \cos(\theta + \alpha_1) \\ \sin(\theta + \alpha_1) \end{pmatrix} \end{array} \tag{5.4}$$

This relation can allow us to be located using a single mobile or fixed landmark and other proprioceptive sensors. For instance, in the case where we have a compass and several odometers (for a robot on wheels), we are able to measure the heading $\theta$ using the compass, the speed $v$ and $\dot{\theta}$ using the odometers. Relation (5.4) then allows us to calculate the position $x$ and $y$ at the given moment in time.

#### 5.2.4.2 One landmark, no compass

In the situation where the compass is not present, we are missing an equation. We either need to add a second landmark, or differentiate again. Let us remain with a single landmark and differentiate Relation (5.3), we obtain:

$$\begin{array}{l} (\ddot{x}_1 - \ddot{x}) \sin(\theta + \alpha_1) - (\ddot{y}_1 - \ddot{y}) \cos(\theta + \alpha_1) \\ + (x_1 - x) \left(\ddot{\theta} + \ddot{\alpha}_1\right) \cos(\theta + \alpha_1) + (y_1 - y) \left(\ddot{\theta} + \ddot{\alpha}_1\right) \sin(\theta + \alpha_1) \\ +2 (\dot{x}_1 - \dot{x}) \left(\dot{\theta} + \dot{\alpha}_1\right) \cos(\theta + \alpha_1) + 2 (\dot{y}_1 - \dot{y}) \left(\dot{\theta} + \dot{\alpha}_1\right) \sin(\theta + \alpha_1) \\ -(x_1 - x) \left(\dot{\theta} + \dot{\alpha}_1\right)^2 \sin(\theta + \alpha_1) + (y_1 - y) \left(\dot{\theta} + \dot{\alpha}_1\right)^2 \cos(\theta + \alpha_1) \quad = 0 \end{array}$$

Moreover:

$$\left\{ \begin{array}{rcl} \ddot{x} & = & \dot{v}\cos\theta - v\dot{\theta}\sin\theta \\ \ddot{y} & = & \dot{v}\sin\theta + v\dot{\theta}\cos\theta \end{array} \right.$$

We thus obtain a system of three equations with three unknowns $x, y, \theta$:

$$\begin{aligned}
&(x_1 - x)\sin(\theta+\alpha_1) - (y_1 - y)\cos(\theta+\alpha_1) && = 0\\
&(\dot{x}_1 - v\cos\theta)\sin(\theta+\alpha_1) + (x_1 - x)\left(\dot{\theta}+\dot{\alpha}_1\right)\cos(\theta+\alpha_1)\\
&\quad - (\dot{y}_1 - v\sin\theta)\cos(\theta+\alpha_1) + (y_1 - y)\left(\dot{\theta}+\dot{\alpha}_1\right)\sin(\theta+\alpha_1) && = 0\\
&\left(\ddot{x}_1 - \dot{v}\cos\theta + v\dot{\theta}\sin\theta\right)\sin(\theta+\alpha_1) - \left(\ddot{y}_1 - \dot{v}\sin\theta - v\dot{\theta}\cos\theta\right)\cos(\theta+\alpha_1)\\
&\quad + (x_1 - x)\left(\ddot{\theta}+\ddot{\alpha}_1\right)\cos(\theta+\alpha_1) + (y_1 - y)\left(\ddot{\theta}+\ddot{\alpha}_1\right)\sin(\theta+\alpha_1)\\
&\quad + 2(\dot{x}_1 - v\cos\theta)\left(\dot{\theta}+\dot{\alpha}_1\right)\cos(\theta+\alpha_1) + 2(\dot{y}_1 - v\sin\theta)\left(\dot{\theta}+\dot{\alpha}_1\right)\sin(\theta+\alpha_1)\\
&\quad - (x_1 - x)\left(\dot{\theta}+\dot{\alpha}_1\right)^2\sin(\theta+\alpha_1) + (y_1 - y)\left(\dot{\theta}+\dot{\alpha}_1\right)^2\cos(\theta+\alpha_1) && = 0
\end{aligned}$$

The quantities $x_1, y_1, \dot{x}_1, \dot{y}_1, \ddot{x}_1, \ddot{y}_1$ are calculated from the path of landmark 1, for which we know an analytic expression. The quantities $\alpha_1$, $\dot{\alpha}_1, \ddot{\alpha}_1$ are assumed to be measured. The quantities $\dot{\theta}$, $\ddot{\theta}$ can be obtained using a gyro. The speed $v$ can be measured using odometers. It is clear that this system is not easy to solve analytically and does not always admit a unique solution. For instance, if the landmark is fixed, by rotational symmetry we can see that we will not be able to find the angle $\theta$. In such a case, we need at least two landmarks for localization.

## 5.3 Multilateration

*Multilateration* is a localization technique based on measuring the difference of the distances between the robot and the landmarks. Indeed, in a number of situations (such as in GPS localization), the clocks between the landmarks and the robot are not synchronized and we can not therefore directly measure the absolute distance between the landmarks and the robot (by the propagation time of air- or soundwaves), but we can measure the difference between these distances. We will now give the principles of this technique.

Four landmarks emit a brief signal at the same time $t_0$ which propagates with a speed $c$. Each emitted signal contains the identifier of the landmark, its position and the emission time $t_0$. The robot (which does not have an accurate clock, only an accurate chronometer) receives the four signals at times $t_i$. From this it easily deduces the offsets between the reception times $\tau_2 = t_2 - t_1$, $\tau_3 = t_3 - t_1$, $\tau_4 = t_4 - t_1$ (see Figure 5.7). We thus obtain the four equations:

$$\begin{aligned}
\sqrt{(x-x_1)^2 + (y-y_1)^2 + (z-z_1)^2} &= c(t_1 - t_0)\\
\sqrt{(x-x_2)^2 + (y-y_2)^2 + (z-z_2)^2} &= c(\tau_2 + t_1 - t_0)\\
\sqrt{(x-x_3)^2 + (y-y_3)^2 + (z-z_3)^2} &= c(\tau_3 + t_1 - t_0)\\
\sqrt{(x-x_4)^2 + (y-y_4)^2 + (z-z_4)^2} &= c(\tau_4 + t_1 - t_0)
\end{aligned}$$

where the parameters, whose values are known with high precision, are $c$, $t_0$, $x_1, y_1, z_1, \ldots, x_4, y_4, z_4$, $\tau_2$, $\tau_3$, $\tau_4$. The four unknowns are $x, y, z, t_1$. Solving this system allows it to be localized and also to readjust its clock (through $t_1$). In the case of the GPS, the landmarks are mobile. They use a similar principle to be localized and synchronized, from fixed landmarks on the ground.

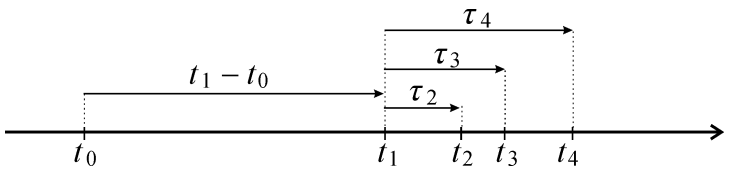


Figure 5.7: The emission time $t_0$ and the offsets between the arrival times $\tau_2, \tau_3, \tau_4$ are known

# Exercises

EXERCISE 5.1.– *Instantaneous state estimation*

Localization consists of finding the position and the orientation of the robot. This problem can sometimes be reduced to a state estimation problem, if the state model for our robot is available. In this exercise, we will give a method that is sometimes used for the state estimation of nonlinear systems. Let us consider the tricycle given in paragraph 2.5 on page 50, described by the state equations:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \\ \dot{\delta} \end{pmatrix} = \begin{pmatrix} v\cos\delta\cos\theta \\ v\cos\delta\sin\theta \\ v\sin\delta \\ u_1 \\ u_2 \end{pmatrix}$$

We measure the positions $x$ and $y$ with such high precision that we may assume that $\dot{x}, \dot{y}, \ddot{x}, \ddot{y}$ are known. Express the other state variables $\theta, v, \delta$ in function of $x$, $y$, $\dot{x}$, $\dot{y}$, $\ddot{x}$, $\ddot{y}$.

---

EXERCISE 5.2.– *Localization by lidar*

Here we are interested in developing a fast localization method for a robot using a rotating laser rangefinder, or lidar (*light radar*) of type Hokuyo, in a rectangular room whose length and width are unknown.

1) Let $\mathbf{a}_1, \ldots, \mathbf{a}_{n_p}$ be points of $\mathbb{R}^2$ located on the same line. Find this line using a least squares method. Represent this line in normal form:

$$x\cos\alpha + y\sin\alpha = d, \text{ with } d \geq 0$$

where $\alpha, d$ are the parameters of the line.

2) Consider one hundred measurements $\theta_i$ of the same angle $\theta$ of a robot by one hundred compasses placed on the robots, of which thirty are defective. Estimate $\theta$ using the median method. For this, we define the *disambiguation* function:

$$\mathbf{f} : \begin{cases} \mathbb{R} & \to & \mathbb{R}^2 \\ \theta & \to & \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix} \end{cases}$$

Such a function, which must be continuous, aims to associate a single image to two equivalent angles (i.e. congruent to $2\pi$), in other words:

$$\theta_1 \sim \theta_2 \Leftrightarrow f(\theta_1) = f(\theta_2)$$

3) The robot's lidar, which has an aperture angle of 180 degrees, gives us 512 points that belong to the rectangle representing our room. These points can be found in the file `lidar_data.mat`. We will take them in groups of ten (i.e. 51 groups) and try to find the line that passes the best through each group (using the least squares method). We will only keep the groups with a small residue. We thus obtain $m$ lines, represented by $n$ points of the form $(\alpha_i, d_i)$, $i \in \{1, \ldots, m\}$ in the so-called *Hough space*. A pair $(\alpha_i, d_i)$ is called an *alignment*. By using a median estimator, find the four possible directions for our room (knowing that it is rectangular). Why is this median estimator considered robust ?

4) How are the angles $\alpha_i$ filtered from the alignments ?

5) How are the $d_i$ filtered ?

6) Deduce from the above a method for localizing the robot.

---

EXERCISE 5.3.– *Instantaneous goniometric localization*

Consider a robot boat $\mathcal{R}$ described by the state equations:

$$\begin{cases} \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= u_1 \\ \dot{v} &= u_2 \end{cases}$$

where $v$ is the speed of the robot, $\theta$ its orientation and $(x, y)$ the coordinates of its center. Its state vector is given by $\mathbf{x} = (x, y, \theta, v)$. In the surroundings of the robot there is a point landmark (a lighthouse for instance) $\mathbf{m} = (x_m, y_m)$ whose position is known (see Figure 5.8). The robot $\mathcal{R}$ is equipped with five sensors: an omnidirectional camera (allowing it to measure the bearing angle $\alpha$ of the landmark's direction), odometers for measuring its speed, a compass that gives its heading $\theta$, an accelerometer for $u_2$ and a gyro for $u_1$.

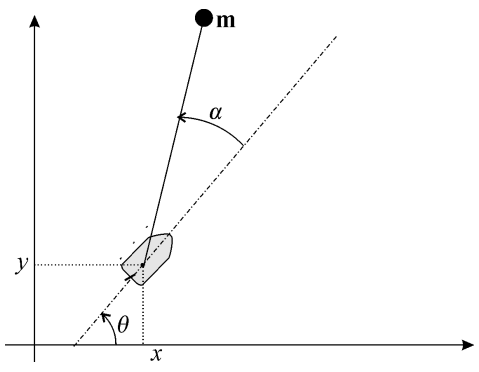


Figure 5.8: Goniometric localization

1) Simulate this system in a situation where the robot is moving around **m** in order to generate the signals $\alpha, \dot{\alpha}, \theta, v, u_1, u_2$. For generating $\alpha$, use the two-argument arctangent function *atan2*. As we have seen in Equation [1.8] on page 16, the atan2$(b, a)$ function returns the argument of the coordinate vector $(a, b)$. For generating $\dot{\alpha}$ we could use the fact that:

$$\frac{\partial \text{atan2}\,(b,a)}{\partial a} = -\frac{b}{a^2+b^2} \qquad \text{and} \qquad \frac{\partial \text{atan2}\,(b,a)}{\partial b} = \frac{a}{a^2+b^2}$$

2) Design an instantaneous localization system for $\mathcal{R}$. Verify this localization system using a simulation. Add a small white Gaussian noise in your measurements to test its robustness.

3) The robot $\mathcal{R}$ is no longer equipped with odometers. How would you adjust the previous approach to allow localization ?

4) Assuming once again that there are no odometers and that $\dot{\alpha}$ is not available, use a Kalman filter for localization.

5) Let us add a second robot $\mathcal{R}_b$ to the scene which is of the same type as the first (which we will now refer to as $\mathcal{R}_a$), with the exception that it has no odometers, contrarily to $\mathcal{R}_a$. Robot $\mathcal{R}_b$ can see $\mathcal{R}_a$ (see Figure 5.9) which gives it an angle $\beta_b$. Both robots can communicate via WIFI. Design a localization system for the robot $\mathcal{R}_b$ which also allows it to find its speed.

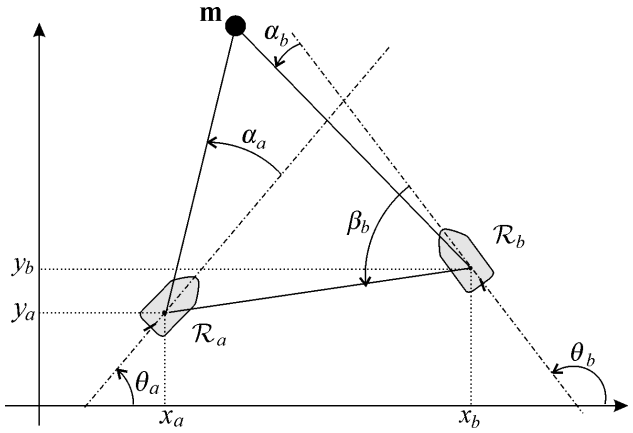


Figure 5.9: The robots $\mathcal{R}_a$ and $\mathcal{R}_a$ can see each other, which will aid them in the localization

6) Robot $\mathcal{R}_a$ can see $\mathcal{R}_b$, which gives it a new angular measurement $\beta_a$. Deduce a localization method from this for $\mathcal{R}_a$ and $\mathcal{R}_b$ that is more reliable (in other words one that presents less singularities) than that developed in the first question.

---

EXERCISE 5.4.– *Localization by distance measuring*

We consider a robot described by the state equations:

$$\begin{cases} \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= u_1 \\ \dot{v} &= u_2 \end{cases}$$

where $v$ is the speed of the robot, $\theta$ its orientation and $(x, y)$ the coordinates of its center $\mathbf{c}$. Its state vector is given by $\mathbf{x} = (x, y, \theta, v)$.

In the surroundings of the robot there is a fixed point landmark $\mathbf{m}$ whose position is known (see Figure 5.10). Each instant, the robot measures the distance $d$ between its center $\mathbf{c}$ and the landmark. If the landmark and the robot both have synchronized clocks, such a system for measuring the distance can be done by measuring the propagation time of a sound wave between the landmark and the robot. Moreover, by using the Doppler effect, the robot is also capable of measuring $\dot{d}$ with very high precision. In addition to the microphone that allows it to measure $d$ and $\dot{d}$, the robot is equipped with odometers to measure its speed $v$, a compass that gives its heading $\theta$. In this exercise, we are looking to build an instantaneous localization system in order to determine its position $(x, y)$ from $d, \dot{d}, \theta, v$.

1) For a given vector $\left(d, \dot{d}, \theta, v\right)$, there may be several configurations possible for the robot. Draw, on the picture, all the configurations for the robot that are compatible with the one that is represented.

2) Consider the coordinate system $\mathcal{R}_1$ centered at $\mathbf{m}$ as represented on the figure. Give the expression of the coordinates $(x_1, y_1)$ of the center $\mathbf{c}$ of the robot in function of $x, y, \theta, x_m, y_m$. This is in fact a coordinate system change equation.

3) Express $x_1$ and $y_1$ in function of $d, \dot{d}, v$.

4) From this, deduce the position(s) of the robot $(x, y)$ in function of $(d, \dot{d}, \theta, v, x_m, y_m)$. What are the singularities of this localization system ? In which case do we have a single solution ?

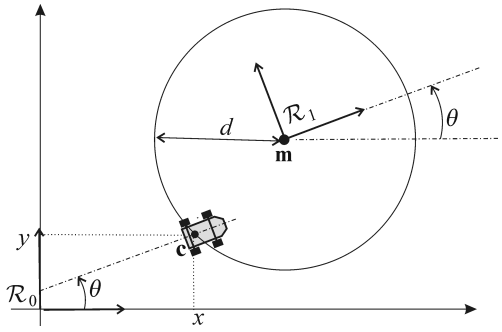


Figure 5.10: The robot measures the distance to the landmark

# Chapter 6

# Identification

The aim of identification is to estimate unmeasured quantities from other measured values, with high precision. In the particular case in which the quantity to estimate is the state vector of an invariant linear system, state observers using pole placement (or Luenberger observers) can be considered efficient tools for identification. In this chapter, we will present several basic concepts of estimation, with the aim of introducing Kalman filtering in the next chapter. In summary, this filtering can be seen as state observation for dynamic linear systems with time-variable coefficients. However, in contrast to more standard observers using a pole placement method, Kalman filtering uses the probabilistic properties of signals. Here we will consider the static (as opposed to the dynamic) case. The unknowns to estimate are all stored in a vector of parameters $\mathbf{p}$ while the measurements are stored in a vector of measurements $\mathbf{y}$. In order to perform this estimation, we will mainly look at the so-called *least squares* approach which seeks to find the vector $\mathbf{p}$ that minimizes the sum of the squares of the errors.

## 6.1 Quadratic functions

In the case in which the dependency between the vectors $\mathbf{p}$ and $\mathbf{y}$ is linear, the least squares method is used to minimize a quadratic function. This paragraph recalls several concepts attached to these functions, which are of a particular nature.

### 6.1.1 Definition

A quadratic function $f : \mathbb{R}^n \to \mathbb{R}$ is a function of the form:

$$f(\mathbf{x}) = \mathbf{x}^{\mathrm{T}} \cdot \mathbf{Q} \cdot \mathbf{x} + \mathbf{L}\mathbf{x} + c$$

where $\mathbf{Q}$ is a symmetric matrix. This definition is equivalent to stating that $f(\mathbf{x})$ is a linear combination of a constant $c$, of the $x_i$, of their squares $x_i^2$ and of the cross products $x_i x_j$ where $i \neq j$. For instance, the function $f(x_1, x_2) = 2x_1^2 - 6x_1x_2 + x_2^2 - 2x_1 + x_2 + 1$ is a quadratic function. We have:

$$f(\mathbf{x}) = \begin{pmatrix} x_1 & x_2 \end{pmatrix} \begin{pmatrix} 2 & -3 \\ -3 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} -2 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + 1. \tag{6.1}$$

We will show below that the derivative of $f$ at point $\mathbf{x}$ is an affine function. In our example, the derivative of $f$ at point $\mathbf{x}$ is given by:

$$\frac{df}{d\mathbf{x}}(\mathbf{x}) = \left( \frac{\partial f}{\partial x_1}(\mathbf{x}) \quad \frac{\partial f}{\partial x_2}(\mathbf{x}) \right)$$

with $\frac{\partial f}{\partial x_1}(\mathbf{x}) = 4x_1 - 6x_2 - 2$ and $\frac{\partial f}{\partial x_2}(\mathbf{x}) = -6x_1 + 2x_2 + 1$, in other words:

$$\frac{df}{d\mathbf{x}}(\mathbf{x}) = (4x_1 - 6x_2 - 2 \ ; \ -6x_1 + 2x_2 + 1)$$

This is an affine function in $\mathbf{x}$. The function $\mathbf{x} \mapsto \mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x}$ which composes $f(\mathbf{x})$ has terms only in $x_i x_j$ and in $x_i^2$. Such a function is called a *quadratic form*.

### 6.1.2 Derivative of a quadratic form

Let us consider the following quadratic form:

$$f(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}\mathbf{Q}\ \mathbf{x}.$$

The first-order Taylor development of $f$ at point $\mathbf{x}$ in the neighborhood of $\mathbf{x}$ yields:

$$f(\mathbf{x} + \delta\mathbf{x}) = f(\mathbf{x}) + \frac{df}{d\mathbf{x}}(\mathbf{x}) \cdot \delta\mathbf{x} + o(||\delta\mathbf{x}||)$$

where $o(||\delta\mathbf{x}||)$ means *negligible compared to* $||\delta\mathbf{x}||$, when $\delta\mathbf{x}$ is infinitely small. Of course, here $\frac{df}{d\mathbf{x}}(\mathbf{x})$ will be represented by a $1 \times n$ matrix since, just like the function we are linearizing, it goes from $\mathbb{R}^n$ to $\mathbb{R}$. However:

$$\begin{aligned} f(\mathbf{x} + \delta\mathbf{x}) &= (\mathbf{x} + \delta\mathbf{x})^{\mathrm{T}} \cdot \mathbf{Q} \cdot (\mathbf{x} + \delta\mathbf{x}) \\ &= \mathbf{x}^{\mathrm{T}} \cdot \mathbf{Q} \cdot \mathbf{x} + \mathbf{x}^{\mathrm{T}} \cdot \mathbf{Q} \cdot \delta\mathbf{x} + \delta\mathbf{x}^{\mathrm{T}} \cdot \mathbf{Q} \cdot \mathbf{x} + \delta\mathbf{x}^{\mathrm{T}} \cdot \mathbf{Q} \cdot \delta\mathbf{x} \\ &= \mathbf{x}^{\mathrm{T}} \cdot \mathbf{Q} \cdot \mathbf{x} + 2\mathbf{x}^{\mathrm{T}} \cdot \mathbf{Q} \cdot \delta\mathbf{x} + o(||\delta\mathbf{x}||) \end{aligned}$$

since $\mathbf{Q}$ is symmetric and $\delta\mathbf{x}^{\mathrm{T}} \cdot \mathbf{Q} \cdot \delta\mathbf{x} = o(||\delta\mathbf{x}||)$. By uniqueness of the Taylor development and given the expressions for $f(\mathbf{x} + \delta\mathbf{x})$, we have:

$$\frac{df}{d\mathbf{x}}(\mathbf{x}) = \left( \frac{\partial f}{\partial x_1}(\mathbf{x}), \ldots, \frac{\partial f}{\partial x_n}(\mathbf{x}) \right) = 2\mathbf{x}^{\mathrm{T}}\mathbf{Q}.$$

For instance, the derivative of the quadratic function [6.1] is:

$$2\,(x_1 \ \ x_2) \begin{pmatrix} 2 & -3 \\ -3 & 1 \end{pmatrix} + (-2 \ \ 1) = \begin{pmatrix} 4x_1 - 6x_2 - 2 & -6x_1 + 2x_2 + 1 \end{pmatrix}.$$

### 6.1.3 Eigenvalues of a quadratic function

These are the eigenvalues of $\mathbf{Q}$. The eigenvalues are all real and the eigenvectors are all orthogonal two-by-two. The contour lines of a quadratic function $f(\mathbf{x}) = \alpha$ are of the form:

$$\mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x} + \mathbf{L}\mathbf{x} = \alpha - c$$

and are called *quadrics*. These are ellipsoid if all the eigenvalues have the same sign or hyperboloid if they have different signs. If all the eigenvalues of $\mathbf{Q}$ are positive, we say that the quadratic form $\mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x}$ is positive. If they are all non-zero, we say that the quadratic form is definite. If they are all strictly positive, we say that the quadratic form is positive definite. The quadratic function $f$ has one and only one minimizer if and only if its associated quadratic form is positive definite.

### 6.1.4 Minimizing a quadratic function

**Theorem**. If $\mathbf{Q}$ is positive definite, the function $f(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x} + \mathbf{L}\mathbf{x} + c$ has one and only one minimizer $\mathbf{x}^*$ given by

$$\mathbf{x}^* = -\frac{1}{2}\mathbf{Q}^{-1}\mathbf{L}^{\mathrm{T}}$$

and the minimum is $f(\mathbf{x}^*) = -\frac{1}{4}\mathbf{L}\mathbf{Q}^{-1}\mathbf{L}^{\mathrm{T}} + c$.

**Proof**. The function $f$ is convex and differentiable. At the minimizer $\mathbf{x}^*$, we have

$$\frac{df}{d\mathbf{x}}(\mathbf{x}^*) = 2\mathbf{x}^{*\mathrm{T}}\mathbf{Q} + \mathbf{L} = \mathbf{0}.$$

Thus $\mathbf{x}^* = -\frac{1}{2}\mathbf{Q}^{-1}\mathbf{L}^{\mathrm{T}}$. The corresponding minimum is given by:

$$\begin{aligned} f(\mathbf{x}^*) &= \left(-\frac{1}{2}\mathbf{Q}^{-1}\mathbf{L}^{\mathrm{T}}\right)^{\mathrm{T}}\mathbf{Q}\left(-\frac{1}{2}\mathbf{Q}^{-1}\mathbf{L}^{\mathrm{T}}\right) + \mathbf{L}\left(-\frac{1}{2}\mathbf{Q}^{-1}\mathbf{L}^{\mathrm{T}}\right) + c \\ &= \frac{1}{4}\mathbf{L}\mathbf{Q}^{-1}\mathbf{L}^{\mathrm{T}} - \frac{1}{2}\mathbf{L}\mathbf{Q}^{-1}\mathbf{L}^{\mathrm{T}} + c \\ &= -\frac{1}{4}\mathbf{L}\mathbf{Q}^{-1}\mathbf{L}^{\mathrm{T}} + c. \end{aligned}$$

EXAMPLE 6.1.– *The quadratic function:*

$$f(\mathbf{x}) = \begin{pmatrix} x_1 & x_2 \end{pmatrix}\begin{pmatrix} 2 & -1 \\ -1 & 1 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 3 & 4 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + 5$$

*has a minimum since the matrix of its quadratic form* $\mathbf{Q}$ *is positive definite (its eigenvalues* $\frac{3}{2} \pm \frac{1}{2}\sqrt{5}$ *are both positive). The function has the following vector as minimizer:*

$$\mathbf{x}^* = -\frac{1}{2}\begin{pmatrix} 2 & -1 \\ -1 & 1 \end{pmatrix}^{-1}\begin{pmatrix} 3 \\ 4 \end{pmatrix} = \begin{pmatrix} -\frac{7}{2} \\ -\frac{11}{2} \end{pmatrix}.$$

*Its minimum is:*

$$f(\mathbf{x}^*) = -\frac{1}{4}\begin{pmatrix} 3 & 4 \end{pmatrix}\begin{pmatrix} 2 & -1 \\ -1 & 1 \end{pmatrix}^{-1}\begin{pmatrix} 3 \\ 4 \end{pmatrix} + 5 = -\frac{45}{4}$$

EXAMPLE 6.2.– *The function $f(x) = 3x^2 + 6x + 7$ has a minimum since the matrix of its quadratic form (which here corresponds to the scalar 3) is positive definite (since $3 > 0$). Its minimizer is the scalar:*

$$x^* = -\frac{1}{2} \cdot \frac{1}{3} \cdot 6 = -1$$

*and its minimum is $f(x^*) = 3 - 6 + 7 = 4$.*

## 6.2 The least squares method

Estimating means obtaining an order of magnitude for certain quantities of a system from measurements of other quantities of the same system. The estimation problem we will consider in this chapter is the following. Consider a system for which we have made various measurements $\mathbf{y} = (y_1, \ldots, y_p)$ and a model $\mathcal{M}(\mathbf{p})$ depending on a vector of parameters $\mathbf{p}$. We need to estimate $\mathbf{p}$ such that the outputs $\mathbf{f}(\mathbf{p})$ generated by $\mathcal{M}(\mathbf{p})$ resemble $\mathbf{y}$ as much as possible.

### 6.2.1 Linear case

Let us assume that the vector of the outputs can be written in the form:

$$\mathbf{f}(\mathbf{p}) = \mathbf{M}\mathbf{p}.$$

The model is then referred to as *linear with respect to the parameters*. We would like to have:

$$\mathbf{f}(\mathbf{p}) = \mathbf{y}$$

but this is generally not possible due to the presence of noise and the fact that the number of measurements is generally higher than the number of parameters (in other words $\dim(\mathbf{y}) > \dim(\mathbf{p})$). We will therefore try to find the best $\mathbf{p}$, i.e. the one that minimizes the so-called *least squares* criterion:

$$j(\mathbf{p}) = ||\mathbf{f}(\mathbf{p}) - \mathbf{y}||^2,$$

We have:

$$\begin{aligned} j(\mathbf{p}) &= ||\mathbf{f}(\mathbf{p}) - \mathbf{y}||^2 = ||\mathbf{M}\mathbf{p} - \mathbf{y}||^2 \\ &= (\mathbf{M}\mathbf{p} - \mathbf{y})^{\mathrm{T}} (\mathbf{M}\mathbf{p} - \mathbf{y}) = \left(\mathbf{p}^{\mathrm{T}}\mathbf{M}^{\mathrm{T}} - \mathbf{y}^{\mathrm{T}}\right)(\mathbf{M}\mathbf{p} - \mathbf{y}) \\ &= \mathbf{p}^{\mathrm{T}}\mathbf{M}^{\mathrm{T}}\mathbf{M}\mathbf{p} - \mathbf{p}^{\mathrm{T}}\mathbf{M}^{\mathrm{T}}\mathbf{y} - \mathbf{y}^{\mathrm{T}}\mathbf{M}\mathbf{p} + \mathbf{y}^{\mathrm{T}}\mathbf{y} \\ &= \mathbf{p}^{\mathrm{T}}\mathbf{M}^{\mathrm{T}}\mathbf{M}\mathbf{p} - 2\mathbf{y}^{\mathrm{T}}\mathbf{M}\mathbf{p} + \mathbf{y}^{\mathrm{T}}\mathbf{y}. \end{aligned}$$

However, $\mathbf{M}^{\mathrm{T}}\mathbf{M}$ is symmetric (since $\left(\mathbf{M}^{\mathrm{T}}\mathbf{M}\right)^{\mathrm{T}} = \mathbf{M}^{\mathrm{T}}\mathbf{M}$). We therefore have a quadratic function. Moreover, all the eigenvalues of $\mathbf{M}^{\mathrm{T}}\mathbf{M}$ are positive or zero. The minimizer $\hat{\mathbf{p}}$ is obtained as follows:

$$\begin{array}{rcll} \frac{dj}{d\mathbf{p}}(\hat{\mathbf{p}}) = \mathbf{0} & \Leftrightarrow & 2\hat{\mathbf{p}}^{\mathrm{T}}\mathbf{M}^{\mathrm{T}}\mathbf{M} - 2\mathbf{y}^{\mathrm{T}}\mathbf{M} = \mathbf{0} & \Leftrightarrow \quad \hat{\mathbf{p}}^{\mathrm{T}}\mathbf{M}^{\mathrm{T}}\mathbf{M} = \mathbf{y}^{\mathrm{T}}\mathbf{M} \\ & \Leftrightarrow & \mathbf{M}^{\mathrm{T}}\mathbf{M}\hat{\mathbf{p}} = \mathbf{M}^{\mathrm{T}}\mathbf{y} & \Leftrightarrow \quad \hat{\mathbf{p}} = \left(\mathbf{M}^{\mathrm{T}}\mathbf{M}\right)^{-1}\mathbf{M}^{\mathrm{T}}\mathbf{y} \end{array}$$

The matrix:

$$\mathbf{K} = \left(\mathbf{M}^{\mathrm{T}}\mathbf{M}\right)^{-1}\mathbf{M}^{\mathrm{T}}$$

is called the *generalized inverse* of the rectangular matrix $\mathbf{M}$. The vector $\hat{\mathbf{p}}$ is called the least squares *estimate*. The function:

$$\mathbf{y} \mapsto \mathbf{K}\mathbf{y}$$

is called the *estimator*. Note that this estimator is linear since the model function $\mathbf{f}$ is also linear. The vector:

$$\hat{\mathbf{y}} = \mathbf{M}\hat{\mathbf{p}} = \mathbf{M}\mathbf{K}\mathbf{y}$$

is the vector of the *filtered measurements* and the quantity:

$$\mathbf{r} = \hat{\mathbf{y}} - \mathbf{y} = (\mathbf{M}\mathbf{K} - \mathbf{I})\,\mathbf{y}$$

is called the *vector of residuals*. The norm of this vector represents the distance between $\mathbf{y}$ and the hyperplane $\mathbf{f}(\mathbb{R}^n)$. If this norm is large, it often means that there is an error in the model or inaccuracies in the data.

### 6.2.2 Nonlinear case

If $\mathbf{y}$ is the vector of measurements and if $\mathbf{f}(\mathbf{p})$ is the output generated by the model, then the least squares estimate is defined by:

$$\hat{\mathbf{p}} = \arg\min_{\mathbf{p}\in\mathbb{R}^n} ||\mathbf{f}(\mathbf{p}) - \mathbf{y}||^2.$$

When $\mathbf{f}(\mathbf{p})$ is linear with respect to $\mathbf{p}$, *i.e.*, $\mathbf{f}(\mathbf{p}) = \mathbf{M}\mathbf{p}$ then the vector of parameters $\hat{\mathbf{p}}$ estimated using the least squares method is $\hat{\mathbf{p}} = \left(\mathbf{M}^{\mathrm{T}}\mathbf{M}\right)^{-1}\mathbf{M}^{\mathrm{T}}\mathbf{y}$ and the vector of the filtered measurements

is $\hat{\mathbf{y}} = \mathbf{M}\left(\mathbf{M}^{\mathrm{T}}\mathbf{M}\right)^{-1}\mathbf{M}^{\mathrm{T}}\mathbf{y}$. In general, and even when $\mathbf{f}(\mathbf{p})$ is nonlinear, we can have the following geometric interpretation (see Figure 6.1):

- The vector of the filtered measurements $\hat{\mathbf{y}}$ represents the projection of $\mathbf{y}$ on the set $\mathbf{f}\left(\mathbb{R}^n\right)$ ;
- The vector estimated using the least squares method $\hat{\mathbf{p}}$ represents the inverse image of the vector of the filtered measurements $\mathbf{y}$ by $\mathbf{f}\left(.\right)$.

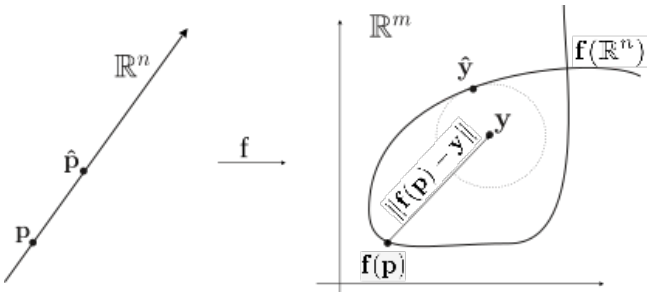


Figure 6.1: Illustration of the least squares method in the nonlinear case

When $\mathbf{f}(\mathbf{p})$ is nonlinear, we can use a local optimization algorithm to try to obtain $\hat{\mathbf{p}}$.

**Newton Method**. In order to minimize $j(\mathbf{p}) = ||\mathbf{f}\left(\mathbf{p}\right) - \mathbf{y}||^2$, the Newton method assumes that an approximation $\mathbf{p}_0$ of the minimizer is available. Around $\mathbf{p}_0$, we have:

$$\mathbf{f}\left(\mathbf{p}\right) \simeq \mathbf{f}\left(\mathbf{p}_0\right) + \frac{d\mathbf{f}}{d\mathbf{p}}(\mathbf{p}_0) \cdot (\mathbf{p} - \mathbf{p}_0).$$

Therefore

$$\begin{array}{rcl} j(\mathbf{p}) & = & ||\mathbf{f}\left(\mathbf{p}\right) - \mathbf{y}||^2 \\ & \simeq & ||\mathbf{f}\left(\mathbf{p}_0\right) + \frac{d\mathbf{f}}{d\mathbf{p}}(\mathbf{p}_0) \cdot (\mathbf{p} - \mathbf{p}_0) - \mathbf{y}||^2 \\ & = & ||\frac{d\mathbf{f}}{d\mathbf{p}}(\mathbf{p}_0) \cdot \mathbf{p} + \mathbf{f}\left(\mathbf{p}_0\right) - \frac{d\mathbf{f}}{d\mathbf{p}}(\mathbf{p}_0) \cdot \mathbf{p}_0 - \mathbf{y}||^2 \\ & = & ||\mathbf{M} \cdot \mathbf{p} - \mathbf{z}||^2 \end{array}$$

with

$$\mathbf{M} = \frac{d\mathbf{f}}{d\mathbf{p}}(\mathbf{p}_0) \text{ and } \mathbf{z} = \mathbf{y} - \mathbf{f}\left(\mathbf{p}_0\right) + \mathbf{M} \cdot \mathbf{p}_0.$$

The minimizer is

$$\mathbf{p}_1 = \left(\mathbf{M}^{\mathrm{T}}\mathbf{M}\right)^{-1}\mathbf{M}^{\mathrm{T}}\mathbf{z} = \left(\mathbf{M}^{\mathrm{T}}\mathbf{M}\right)^{-1}\mathbf{M}^{\mathrm{T}}(\mathbf{y} - \mathbf{f}\left(\mathbf{p}_0\right) + \mathbf{M} \cdot \mathbf{p}_0) = \mathbf{p}_0 + \left(\mathbf{M}^{\mathrm{T}}\mathbf{M}\right)^{-1}\mathbf{M}^{\mathrm{T}}(\mathbf{y} - \mathbf{f}\left(\mathbf{p}_0\right))$$

which is expected to me more closed to the solution than $\mathbf{p}_0$. We get the Newton algorithm:

| | **Algorithm** NEWTON (input : $\mathbf{p}_0, \mathbf{y}$) |
|---|---|
| 1 | for $k = 0$ to $k_{max}$ |
| 2 | $\mathbf{M} = \frac{d\mathbf{f}}{d\mathbf{p}}(\mathbf{p}_k)$, |
| 3 | $\mathbf{p}_{k+1} = \mathbf{p}_k + \left(\mathbf{M}^{\mathrm{T}}\mathbf{M}\right)^{-1} \mathbf{M}^{\mathrm{T}}(\mathbf{y} - \mathbf{f}\left(\mathbf{p}_k\right))$ |
| 4 | next $k$ |

Unfortunately, even if the solution of our minimization problem is unique the Newton algorithm may diverge and may converge to a point which is not the solution of our problem.

**Monté-Carlo method**. The algorithm below proposes a simple version of a Monté-Carlo algorithm:

| | **Algorithm** MINIMIZE(input: $\mathbf{p}$) |
|---|---|
| 1 | take a random movement$\mathbf{d}$ |
| 2 | $\mathbf{q} = \mathbf{p} + \mathbf{d}$ |
| 3 | if $j\left(\mathbf{q}\right) < j(\mathbf{p})$then $\mathbf{p} = \mathbf{q}$ |
| 4 | go to 1 |

Again, this algorithm is expected to converge towards a local optimum of the criterion $j(\mathbf{p}) = ||\mathbf{f}(\mathbf{p}) - \mathbf{y}||^2$. The quantity $\mathbf{d}$ is the step that represents a small vector taken randomly from $\mathbb{R}^n$. In the case of the *simulated annealing* method, the amplitude of this step decreases with the iterations in function of a parameter called *temperature* which decreases with time. If the initial temperature, is high enough and if the temperature decreases sufficiently slowly, then we generally converge to the global minimum.

# Exercises

EXERCISE 6.1.– *Representation of a quadratic function*

Consider the quadratic function $f(x, y) = x \cdot y$.

1) Find the gradient of $f$ at point $(x_0, y_0)$.

2) Put $f$ in the form $(x \;\; y) \cdot \mathbf{Q} \cdot (x \;\; y)^{\mathrm{T}} + \mathbf{L}\,(x \;\; y)^{\mathrm{T}} + c$, where $\mathbf{Q}$ is a symmetric matrix. Verify that the gradient found in question 1) is given by $2\,(x \;\; y)\,\mathbf{Q}$. Draw the vector field associated with this gradient in MATLAB using the `quiver` instruction. Discuss.

3) By using the `contour` instruction in MATLAB, draw the contour lines of $f$ then draw the graph of $f$. Does $f$ have a minimum ?

4) Restart this exercise with the function $g(x, y) = 2x^2 + xy + 4y^2 + y - x + 3$.

---

EXERCISE 6.2.– *Identification of a parabola*

We would like to find a parabola $p_1 t^2 + p_2 t + p_3$ that passes through $n$ points given by:

| $t$ | $-3$ | $-1$ | 0 | 2 | 3 | 6 |
|---|---|---|---|---|---|---|
| $y$ | 17 | 3 | 1 | 5 | 11 | 46 |

1) Give a least squares estimation of the parameters $p_1, p_2, p_3$.

2) What are the corresponding filtered measurements ? Give the vector of residuals.

---

EXERCISE 6.3.– *Identifying the parameters of a DC motor*

The angular speed $\Omega$ of a DC motor in permanent regime depends linearly on the supply voltage $U$ and the resistive torque $T_r$:

$$\Omega = p_1 U + p_2 T_r.$$

We perform a series of experiments on a particular motor. We measure:

| $U$(V) | 4 | 10 | 10 | 13 | 15 |
|---|---|---|---|---|---|
| $T_r$(Nm) | 0 | 1 | 5 | 5 | 3 |
| $\Omega$(rad/ sec) | 5 | 10 | 8 | 14 | 17 |

1) Give a least squares estimation of the parameters $p_1, p_2$. Give the filtered measurements and

the corresponding vector of residuals.

2) Deduce from the above an estimation of the angular speed of the motor $U = 20\,\mathrm{V}$ and $T_r = 10\,\mathrm{Nm}$.

---

EXERCISE 6.4.– *Estimation of a transfer function*

Consider the system described by the recurrence equations:

$$y(k) + a_1 y(k-1) + a_0 y(k-2) = b_1 u(k-1) + b_0 u(k-2).$$

We perform noisy measurements on the input $u(k)$ and output $y(k)$ of this system for $k$ varying from 0 to 7. We obtain:

| $k$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| $u(k)$ | 1 | $-1$ | 1 | $-1$ | 1 | $-1$ | 1 | $-1$ |
| $y(k)$ | 0 | $-1$ | $-2$ | 3 | 7 | 11 | 16 | 36 |

Estimate the vector of parameters $\mathbf{p} = (a_1, a_0, b_1, b_0)$ by the least squares method. Discuss.

---

EXERCISE 6.5.– *Monté-Carlo method*

Consider the discrete-time system given by its state representation:

$$\left\{ \begin{array}{rcl} \mathbf{x}(k+1) & = & \begin{pmatrix} 1 & 0 \\ a & 0.3 \end{pmatrix} \mathbf{x}(k) + \begin{pmatrix} b \\ 1-b \end{pmatrix} u(k) \\ y(k) & = & \begin{pmatrix} 1 & 1 \end{pmatrix} \mathbf{x}(k) \end{array} \right.$$

where $a, b$ are two parameters to be estimated. The initial state is given by $\mathbf{x}(0) = (0,0)$ and $u(k) = 1$. We collect six measurements:

$$(y(0), \cdots, y(5)) = \begin{pmatrix} 0, & 1, & 2.5, & 4.1, & 5.8, & 7.5 \end{pmatrix}.$$

Let us note that these values were obtained for the values $a^* = 0.9$ and $b^* = 0.75$, but we are not supposed to know them. We will only assume that $a \in [0,2]$ and $b \in [0,2]$.

1) Propose a MATLAB program that estimates the parameters $a$ and $b$ using a Monte Carlo method. For this, generate a cloud of vectors $\mathbf{p} = (a,b)$ using a uniform random law. Then, by simulating the state equations, calculate for all the $\mathbf{p}$ the corresponding outputs $y_m(\mathbf{p},k)$. Draw on the screen the vectors $\mathbf{p}$ such that for each $k \in \{0,\ldots,5\}$, $|y_m(k) - y(k)| < \varepsilon$, where $\varepsilon$ is a small positive number.

2) Calculate the transfer function of the system in function of $a$ and $b$.

3) Let us assume that the real values $a^* = 0.9$ and $b^* = 0.75$ for $a$ and $b$ are known. Calculate the set of all pairs $(a,b)$ that generate the same transfer function as the pair $(a^*, b^*)$. Deduce from this an interpretation of the results obtained in question 1).

---

EXERCISE 6.6.– *Localization by simulated annealing*

The localization problem that we will now consider is inspired from [JAU 02]. The robot, represented on Figure 6.2, is equipped with eight laser telemeters capable of measuring its distance from the walls for angles equal to $\frac{k\pi}{4}$, $k \in \{0, \ldots, 7\}$. We assume that the obstacles are composed of $n$ segments $[\mathbf{a}_i\mathbf{b}_i]$, $i = 1, \ldots, n$, where the coordinates of $\mathbf{a}_i$ and $\mathbf{b}_i$ are known. The eight distances are stored in the vector $\mathbf{y}$ and the localization problem amounts to estimating the position and orientation of the robot from $\mathbf{y}$.

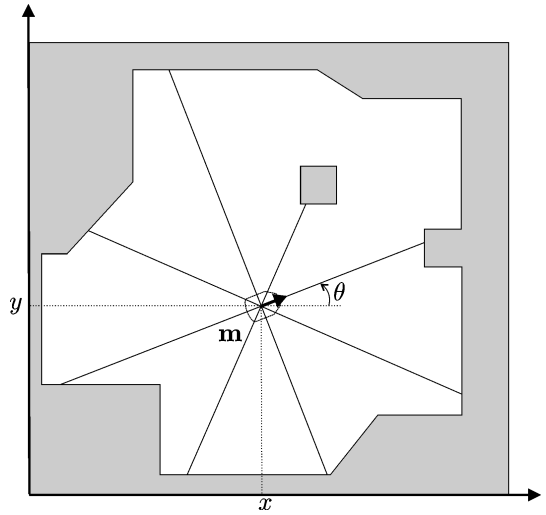


Figure 6.2: Robot equipped with eight telemeters trying to localize itself

1) Let $\mathbf{m}$, $\mathbf{a}$, $\mathbf{b}$ be three points of $\mathbb{R}^2$ and $\overrightarrow{\mathbf{u}}$ a unit vector. Show that the ray $\mathcal{E}\left(\mathbf{m}, \overrightarrow{\mathbf{u}}\right)$ intersects the segment $[\mathbf{ab}]$ if and only if:

$$\begin{cases} \det\left(\mathbf{a}-\mathbf{m}, \overrightarrow{\mathbf{u}}\right) \cdot \det\left(\mathbf{b}-\mathbf{m}, \overrightarrow{\mathbf{u}}\right) & \leq 0 \\ \det(\mathbf{a}-\mathbf{m}, \mathbf{b}-\mathbf{a}) \cdot \det(\overrightarrow{\mathbf{u}}, \mathbf{b}-\mathbf{a}) & \geq 0 \end{cases}$$

If this condition is verified, show that the distance from $\mathbf{m}$ to $[\mathbf{ab}]$ following $\overrightarrow{\mathbf{u}}$ is:

$$d = \frac{\det(\mathbf{a}-\mathbf{m}, \mathbf{b}-\mathbf{a})}{\det(\overrightarrow{\mathbf{u}}, \mathbf{b}-\mathbf{a})}.$$

2) Design a simulator $\mathbf{f}(\mathbf{p})$ that calculates the directional distances between the pose $\mathbf{p} = (x, y, \theta)$ and the walls.

3) Using a global simulated annealing-type optimization method, design a MATLAB program that gives a least squares estimation $\hat{\mathbf{p}}$ of the pose $\mathbf{p}$ from $\mathbf{y}$. For the segments $[\mathbf{a}_i, \mathbf{b}_i]$ of the room and for the vector of the measured distances, take the following quantities:

```
A=[0 7 7 9 9 7 7 4 2 0 5 6 6 5; 0 0 2 2 4 4 7 7 5 5 2 2 3 3 ];
B=[7 7 9 9 7 7 4 2 0 0 6 6 5 5 ; 0 2 2 4 4 7 7 5 5 0 2 3 3 2];
y=[6.4;3.6;2.3;2.1;1.7;1.6;3.0;3.1];
```

# Chapter 7

# Kalman filter

In chapters 2 and 3 we have looked at tools to control robots in a nonlinear manner. For this purpose, we have assumed that the state vector was completely known. However, this is not the case in practice. This vector must be estimated from sensor measurements. In the case where the only unknown variables are associated with the position of the robot, Chapter 5 gives guidelines to find them. In the more general case, *filtering* or *state observation* seeks to reconstruct this state vector as well as possible from all the data measured on the robot throughout time by taking into account the state equations. The aim of this chapter is to show how such reconstruction is performed, within a stochastic context in which the system to observe is assumed to be linear. This is the purpose of the Kalman filter [KAL 60] which will be developed in this chapter. The Kalman filter is used in numerous mobile robotics applications, even though the robots in question are strongly nonlinear. For such applications, the initial conditions are assumed to be relatively well known in order to allow a reliable linearization.

## 7.1 Covariance matrices

The Kalman filter is mainly based on the concept of covariance matrix which is important to grasp in order to understand the design and the utilization of the observer. This section recalls the fundamental concepts surrounding covariance matrices.

### 7.1.1 Definitions and interpretations

Let us consider two random vectors $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{y} \in \mathbb{R}^m$. The mathematical expectations of $\mathbf{x}$ and $\mathbf{y}$ are denoted by $\bar{\mathbf{x}} = E(\mathbf{x})$, $\bar{\mathbf{y}} = E(\mathbf{y})$. Let us define the *variations* of $\mathbf{x}$ and $\mathbf{y}$ by $\tilde{\mathbf{x}} = \mathbf{x} - \bar{\mathbf{x}}$ and $\tilde{\mathbf{y}} = \mathbf{y} - \bar{\mathbf{y}}$. The *covariance matrix* is given by:

$$\mathbf{\Gamma_{xy}} = E\left(\tilde{\mathbf{x}} \cdot \tilde{\mathbf{y}}^{\mathrm{T}}\right) = E\left((\mathbf{x} - \bar{\mathbf{x}})(\mathbf{y} - \bar{\mathbf{y}})^{\mathrm{T}}\right).$$

The covariance matrix for $\mathbf{x}$ is defined by:

$$\mathbf{\Gamma_x} = \mathbf{\Gamma_{xx}} = E\left(\tilde{\mathbf{x}} \cdot \tilde{\mathbf{x}}^{\mathrm{T}}\right) = E\left((\mathbf{x} - \bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}})^{\mathrm{T}}\right).$$

The one for $\mathbf{y}$ is:

$$\mathbf{\Gamma_y} = \mathbf{\Gamma_{yy}} = E\left(\widetilde{\mathbf{y}} \cdot \widetilde{\mathbf{y}}^{\mathrm{T}}\right) = E\left((\mathbf{y} - \bar{\mathbf{y}})(\mathbf{y} - \bar{\mathbf{y}})^{\mathrm{T}}\right).$$

Let us note that $\mathbf{x}$, $\mathbf{y}$, $\widetilde{\mathbf{x}}$, $\widetilde{\mathbf{y}}$ are random vectors whereas $\bar{\mathbf{x}}, \bar{\mathbf{y}}, \mathbf{\Gamma_x}, \mathbf{\Gamma_y}, \mathbf{\Gamma_{xy}}$ are deterministic. A covariance matrix $\mathbf{\Gamma_x}$ of a random vector $\mathbf{x}$ is always positive definite (we will write $\mathbf{\Gamma_x} \succ \mathbf{0}$), except in the degenerate case. In a computer, a random vector can be represented by a cloud of points associated with realizations. Let us consider the following MATLAB program:

```
x=2+randn(1000,1); e=randn(1000,1); y=2*x.^2+e; plot(x,y);
xbar=mean(x); ybar=mean(y); xtilde=x-xbar; ytilde=y-ybar; plot(xtilde,ytilde);
Gx=mean(xtilde.^2); Gy=mean(ytilde.^2); Gxy=mean(xtilde.*ytilde);
```

This yields Figure 7.1, which gives us a representation of the random variables $x, y$ (on the left) and of $\widetilde{x}, \widetilde{y}$ (on the right). The program also gives us the estimations:

$$\bar{x} \simeq 1.99, \bar{y} \simeq 9.983, \Gamma_x \simeq 1.003, \Gamma_y \simeq 74.03, \Gamma_{xy} \simeq 8.082$$

where $\bar{x}, \bar{y}, \Gamma_x, \Gamma_y, \Gamma_{xy}$ correspond to `xbar`, `ybar`, `Gx`, `Gy`, `Gxy`.

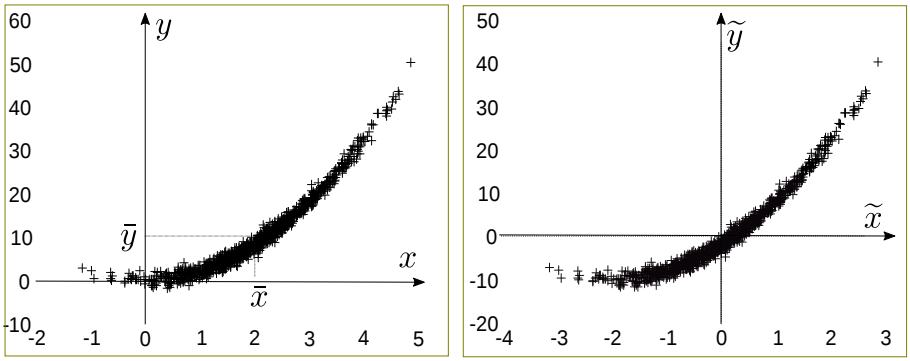


Figure 7.1: Cloud of points that represents a pair of two random variables

Two random vectors $\mathbf{x}$ and $\mathbf{y}$ are linearly independent (or non-correlated or orthogonal) if $\mathbf{\Gamma_{xy}} = \mathbf{0}$. On Figure 7.2, the two point clouds correspond to non-correlated variables. Only the figure on the right corresponds to independent variables.

The figure on the left was generated by:

```
rho=10+randn(2000,1);  theta=2*pi*rand(2000,1);  x=rho.*sin(theta);
y=rho.*cos(theta);
```

And the figure on the right was generated by:

```
x=atan(2*randn(3000,1)); y=atan(2*randn(3000,1));
```

**Whiteness**. A random vector $\mathbf{x}$ is called *white* if all of its components $x_i$ are independent from one another. In such a case, the covariance vector $\mathbf{\Gamma_x}$ of $\mathbf{x}$ is diagonal.

### 7.1.2 Properties

Covariance matrices are symmetric and positive, in other words all of their eigenvalues are real and positive. The set of all covariance matrices of $\mathbb{R}^{n \times n}$ will be denoted by $\mathcal{S}^+(\mathbb{R}^n)$.

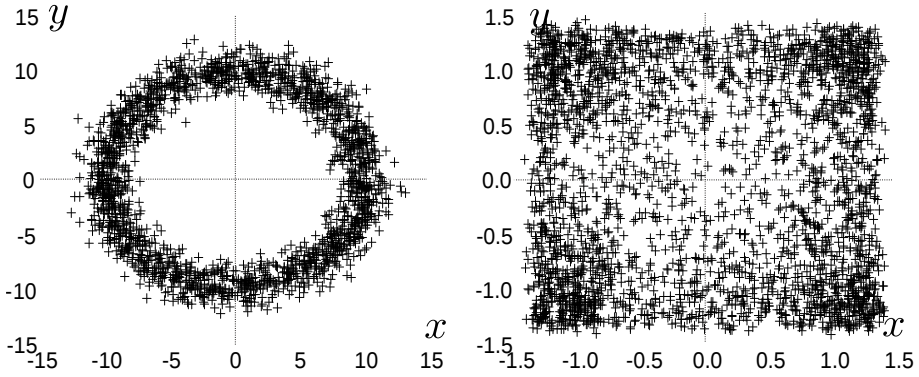


Figure 7.2: Left : dependent but non-correlated variables $(x, y)$ ; Right : independent variables

**Decomposition**. Every symmetric matrix $\mathbf{\Gamma}$ can be put into a diagonal form and has an orthonormal eigenvector basis. We may therefore write:

$$\mathbf{\Gamma} = \mathbf{R} \cdot \mathbf{D} \cdot \mathbf{R}^{-1}$$

where $\mathbf{R}$ is a rotation matrix (i.e. $\mathbf{R}^{\mathrm{T}}\mathbf{R} = \mathbf{I}$ and $\det \mathbf{R} = 1$). The matrix $\mathbf{R}$ corresponds to the eigenvectors and $\mathbf{D}$ is a diagonal matrix whose elements are the eigenvalues. For the matrices of $\mathcal{S}^+(\mathbb{R}^n)$, these eigenvalues are positive.

**Square root**. Every matrix $\mathbf{\Gamma}$ of $\mathcal{S}^+(\mathbb{R}^n)$ has a square root in $\mathcal{S}^+(\mathbb{R}^n)$. This square root will be denoted by $\mathbf{\Gamma}^{\frac{1}{2}}$. Following the eigenvalue correspondence theorem, the eigenvalues of $\mathbf{\Gamma}^{\frac{1}{2}}$ are the square roots of those of the eigenvalues of $\mathbf{\Gamma}$.

EXAMPLE 7.1.– *Consider the following* MATLAB *script:*

```
A=rand(3,3); S1=A*A'; [R,D]=eig(S1); S2=R*D*R'; A2=sqrtm(S2); S3=A2*A2'.
```

*The matrix* `D` *is diagonal and the matrix* `R` *is a rotation matrix that contains the eigenvectors of* `S1`*. The three matrices* `S1`*,* `S2` *and* `S3` *are equal. This is not the case for matrices* `A` *and* `A2` *since only* `A2` *is symmetric. Here,* `sqrt` *returns the square root of* `S2` *and therefore* `A2` *is a covariance matrix.*

**Order**. If $\mathbf{\Gamma}_1$ and $\mathbf{\Gamma}_2$ belong to $\mathcal{S}^+(\mathbb{R}^n)$, then $\mathbf{\Gamma} = \alpha_1\mathbf{\Gamma}_1 + \alpha_2\mathbf{\Gamma}_2$ also belongs to $\mathcal{S}^+(\mathbb{R}^n)$ if $\alpha_1 \geq 0$ and $\alpha_2 \geq 0$. This is equivalent to saying that $\mathcal{S}^+(\mathbb{R}^n)$ is a convex cone of $\mathbb{R}^{n\times n}$. Let us define the order relation:

$$\mathbf{\Gamma}_1 \leq \mathbf{\Gamma}_2 \;\Leftrightarrow\; \mathbf{\Gamma}_2 - \mathbf{\Gamma}_1 \in \mathcal{S}^+(\mathbb{R}^n)\,.$$

It can be easily verified that it is reflexive, antisymmetric and transitive. If $\mathbf{\Gamma}_1 \leq \mathbf{\Gamma}_2$ then the $a$-level confidence ellipse (see following paragraph) of $\mathbf{\Gamma}_1$ is (in general) included in the one that corresponds to $\mathbf{\Gamma}_2$. The smaller the covariance matrix (in the sense of this order relation), the more precise it is.

### 7.1.3 Confidence ellipse

A random vector $\mathbf{x}$ of $\mathbb{R}^n$ can be characterized by the pair $(\bar{\mathbf{x}}, \mathbf{\Gamma}_{\mathbf{x}})$, to which we can associate an ellipse of $\mathbb{R}^n$ which encloses the consistent values for $\mathbf{x}$. In practice, for purely graphical reasons, we

often only look at two components $\mathbf{w} = (x_i, x_j)$ of $\mathbf{x}$ (a computer screen is in fact two-dimensional). The average $\bar{\mathbf{w}}$ can be directly deduced from $\bar{\mathbf{x}}$ by extracting the $i^{\text{th}}$ and $j^{\text{th}}$ components. The covariance matrix $\mathbf{\Gamma}_{\mathbf{w}} \in \mathcal{S}^+(\mathbb{R}^2)$ can also be obtained from $\mathbf{\Gamma}_{\mathbf{x}} \in \mathcal{S}^+(\mathbb{R}^n)$ by extracting the $i^{\text{th}}$ and $j^{\text{th}}$ lines and columns. The *confidence ellipse* associated with $\mathbf{w}$ is described by the inequality:

$$\mathcal{E}_{\mathbf{w}} : (\mathbf{w} - \bar{\mathbf{w}})^{\mathrm{T}} \mathbf{\Gamma}_{\mathbf{w}}^{-1} (\mathbf{w} - \bar{\mathbf{w}}) \leq a^2$$

where $a$ is an arbitrary positive real number. Therefore if $\mathbf{w}$ is a Gaussian random vector, this ellipse corresponds to a contour line of the probability density for $\mathbf{w}$. Since $\mathbf{\Gamma}_{\mathbf{w}}^{-1} \succ \mathbf{0}$, it has a square root $\mathbf{\Gamma}_{\mathbf{w}}^{-\frac{1}{2}}$ which is also positive definite. We may therefore write:

$$\begin{aligned}
\mathcal{E}_{\mathbf{w}} &= \left\{ \mathbf{w} \mid (\mathbf{w} - \bar{\mathbf{w}})^{\mathrm{T}} \ \mathbf{\Gamma}_{\mathbf{w}}^{-\frac{1}{2}} \cdot \mathbf{\Gamma}_{\mathbf{w}}^{-\frac{1}{2}} (\mathbf{w} - \bar{\mathbf{w}}) \leq a^2 \right\} \\
&= \left\{ \mathbf{w} \mid \left\| \tfrac{1}{a} \cdot \mathbf{\Gamma}_{\mathbf{w}}^{-\frac{1}{2}} (\mathbf{w} - \bar{\mathbf{w}}) \right\| \leq 1 \right\} \\
&= \left\{ \mathbf{w} \mid \tfrac{1}{a} \cdot \mathbf{\Gamma}_{\mathbf{w}}^{-\frac{1}{2}} (\mathbf{w} - \bar{\mathbf{w}}) \in \mathcal{U} \right\}, \text{ where } \mathcal{U} \text{ is the unit disk} \\
&= \left\{ \mathbf{w} \mid \mathbf{w} \in \bar{\mathbf{w}} + a\mathbf{\Gamma}_{\mathbf{w}}^{\frac{1}{2}} \mathcal{U} \right\} \\
&= \bar{\mathbf{w}} + a\mathbf{\Gamma}_{\mathbf{w}}^{\frac{1}{2}} \mathcal{U}.
\end{aligned}$$

The ellipse $\mathcal{E}_{\mathbf{w}}$ can therefore be defined as the image of the unit disk by the affine function $\mathbf{w}(\mathbf{s}) = \bar{\mathbf{w}} + a \cdot \mathbf{\Gamma}_{\mathbf{w}}^{\frac{1}{2}} \mathbf{s}$, as illustrated by Figure 7.3.

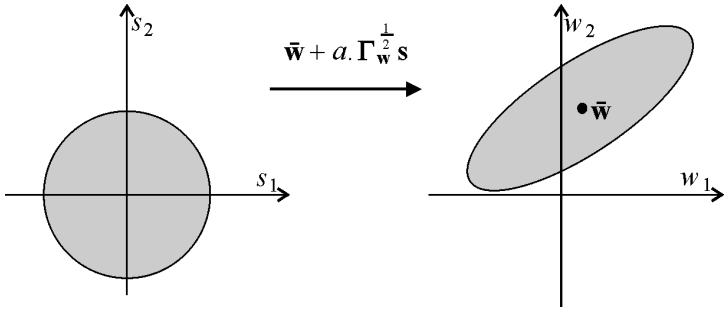


Figure 7.3: A confidence ellipse is the image of the unit circle by an affine function

Recall that for a centered, unit Gaussian random vector $\mathbf{s}$, the random variable $z = \mathbf{s}^{\mathrm{T}}\mathbf{s}$ follows a $\chi^2$ law. In 2 dimensions, this probability density is given by:

$$\pi_z(z) = \begin{cases} \frac{1}{2} \exp\left(-\frac{z}{2}\right) & \text{if } \ z \geq 0 \\ 0 \text{ otherwise} \end{cases}$$

Thus, for a given $a > 0$, we have:

$$\begin{aligned}
\eta \stackrel{\text{def}}{=} \text{prob}\left(||\mathbf{s}|| \leq a\right) &= \text{prob}\left(\mathbf{s}^{\mathrm{T}}\mathbf{s} \leq a^2\right) = \text{prob}\left(z \leq a^2\right) \\
&= \int_0^{a^2} \frac{1}{2} \exp\left(-\frac{z}{2}\right) dz = 1 - e^{-\frac{1}{2}a^2}.
\end{aligned}$$

And therefore:

$$a = \sqrt{-2\ln(1-\eta)}$$

This relation allows us to calculate the threshold $a$ that we need to choose in order to have a probability of being in the ellipse of $\eta$. But careful, this probabilistic interpretation only makes sense in the Gaussian case. The following MATLAB function draws $\mathcal{E}_{\mathbf{w}}$ for a given probability $\eta$:

```
function draw_ellipse(wbar,Gw,eta);
s=0:0.01:2*pi;
w=wbar*ones(size(s))+sqrtm(-2*log(1-eta)*Gw)*[cos(s);sin(s)];
plot(w(1,:),w(2,:));
```

### 7.1.4 Generating Gaussian random vectors

If we generate $n$ centered Gaussian random numbers, we obtain the realization of a random vector whose center is $\bar{\mathbf{x}} = \mathbf{0}$ and whose covariance matrix $\mathbf{\Gamma}_{\mathbf{x}}$ is the identity matrix. In this section we will show, given a centered Gaussian random number generator allowing us to realize $\mathbf{x}$, how we can obtain a Gaussian random vector $\mathbf{y}$ of dimension $n$ with an expectation and covariance matrix $\mathbf{\Gamma}_{\mathbf{y}}$. The main principle of this generation is based on the following theorem.

THEOREM 7.1.– *If $\mathbf{x}, \boldsymbol{\alpha}$ and $\mathbf{y}$ are three random vectors connected by the relation $\mathbf{y} = \mathbf{A}\mathbf{x} + \boldsymbol{\alpha} + \mathbf{b}$ (where $\mathbf{A}$ and $\mathbf{b}$ are deterministic), and assuming that $\mathbf{x}, \boldsymbol{\alpha}$ are independent and that $\boldsymbol{\alpha}$ is centered, we have:*

$$\begin{array}{rcl} \bar{\mathbf{y}} & = & \mathbf{A}\bar{\mathbf{x}} + \mathbf{b} \\ \mathbf{\Gamma}_{\mathbf{y}} & = & \mathbf{A} \cdot \mathbf{\Gamma}_{\mathbf{x}} \cdot \mathbf{A}^T + \mathbf{\Gamma}_{\boldsymbol{\alpha}} \end{array} \tag{7.1}$$

PROOF.– We have:

$$\bar{\mathbf{y}} = E(\mathbf{A}\mathbf{x} + \boldsymbol{\alpha} + \mathbf{b}) = \mathbf{A}E(\mathbf{x}) + E(\boldsymbol{\alpha}) + \mathbf{b} = \mathbf{A}\bar{\mathbf{x}} + \mathbf{b}$$

Moreover:

$$\begin{aligned} \mathbf{\Gamma}_{\mathbf{y}} &= E\left((\mathbf{y}-\bar{\mathbf{y}})(\mathbf{y}-\bar{\mathbf{y}})^{\mathrm{T}}\right) \\ &= E\left((\mathbf{A}\mathbf{x}+\boldsymbol{\alpha}+\mathbf{b}-\mathbf{A}\bar{\mathbf{x}}-\mathbf{b})(\mathbf{A}\mathbf{x}+\boldsymbol{\alpha}+\mathbf{b}-\mathbf{A}\bar{\mathbf{x}}-\mathbf{b})^{\mathrm{T}}\right) \\ &= E\left((\mathbf{A}\tilde{\mathbf{x}}+\boldsymbol{\alpha})\cdot(\mathbf{A}\tilde{\mathbf{x}}+\boldsymbol{\alpha})^{\mathrm{T}}\right) \\ &= \mathbf{A}\cdot\underbrace{E\left(\tilde{\mathbf{x}}\cdot\tilde{\mathbf{x}}^{\mathrm{T}}\right)}_{=\mathbf{\Gamma}_{\mathbf{x}}}\cdot\mathbf{A}^{\mathrm{T}} + \mathbf{A}\cdot\underbrace{E\left(\tilde{\mathbf{x}}\cdot\boldsymbol{\alpha}^{\mathrm{T}}\right)}_{=\mathbf{0}} + \underbrace{E\left(\boldsymbol{\alpha}\cdot\tilde{\mathbf{x}}^{\mathrm{T}}\right)}_{=\mathbf{0}}\cdot\mathbf{A}^{\mathrm{T}} + \underbrace{E\left(\boldsymbol{\alpha}\cdot\boldsymbol{\alpha}^{\mathrm{T}}\right)}_{=\mathbf{\Gamma}_{\boldsymbol{\alpha}}} \\ &= \mathbf{A}\cdot\mathbf{\Gamma}_{\mathbf{x}}\cdot\mathbf{A}^{\mathrm{T}} + \mathbf{\Gamma}_{\boldsymbol{\alpha}} \end{aligned}$$

which concludes the proof. ■

Thus, if $\mathbf{x}$ is a centered, unit Gaussian white random noise (in other words $\bar{\mathbf{x}} = \mathbf{0}$ and $\mathbf{\Gamma}_{\mathbf{x}} = \mathbf{I}$), the random vector $\mathbf{y} = \mathbf{\Gamma}_{\mathbf{y}}^{\frac{1}{2}}\mathbf{x} + \bar{\mathbf{y}}$ will have an expectation of $\bar{\mathbf{y}}$ and a covariance matrix equal to $\mathbf{\Gamma}_{\mathbf{y}}$

(see Figure 7.4). To generate a Gaussian random vector with covariance matrix $\mathbf{\Gamma_y}$ and expectation $\bar{\mathbf{y}}$, we will use this property. The right side of Figure 7.4 was thus obtained by the script:

```
n=1000;Gy=[3,1;1,3]; ybar=[2;3]; x=randn(2,n);
y=ybar*ones(1,n)+sqrtm(Gy)*x; plot(y(1,:),y(2,:),'.');
```

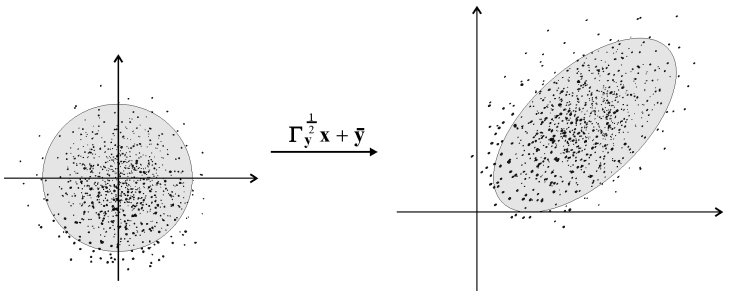


Figure 7.4: The Gaussian random vector $\mathbf{y} : (\bar{\mathbf{y}}, \mathbf{\Gamma_y})$ is the image by an affine application of a unit Gaussian white random vector $\mathbf{x}$

## 7.2 Unbiased orthogonal estimator

Let us consider two random vectors $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{y} \in \mathbb{R}^m$. Vector $\mathbf{y}$ corresponds to the measurement vector which is for the moment a random vector, and will only become available when the measurements have been made. The random vector $\mathbf{x}$ is the vector we need to estimate. An *estimator* is a function $\phi(\mathbf{y})$ that gives us an estimation of $\mathbf{x}$ given the knowledge of the measurement $\mathbf{y}$. Figure 7.5 shows a nonlinear estimator corresponding to $E(x|y)$.

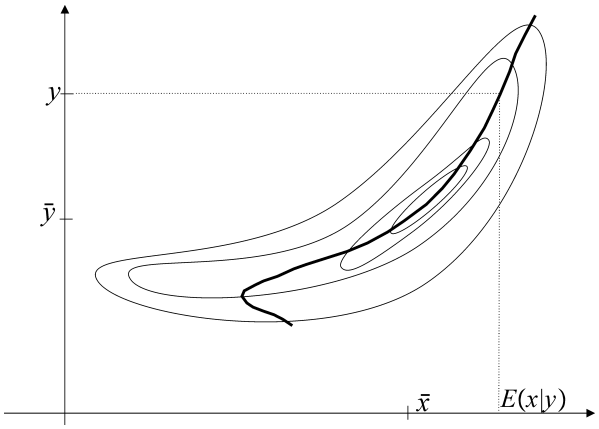


Figure 7.5: Nonlinear estimator $E(x|y)$

However, obtaining an analytic expression for such an estimator in generally not a simple task and it is preferable to limit ourselves to linear estimators. A *linear estimator* is a linear function of $\mathbb{R}^m \to \mathbb{R}^n$ of the form:

$$\hat{\mathbf{x}} = \mathbf{K}\mathbf{y} + \mathbf{b} \tag{7.2}$$

where $\mathbf{K} \in \mathbb{R}^{n \times m}$ and $\mathbf{b} \in \mathbb{R}^n$. In this section, we will propose a method capable of finding a *good* $\mathbf{K}$ and a *good* $\mathbf{b}$ from the sole knowledge of the first-order moments $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ and the second-order moments $\mathbf{\Gamma_x}, \mathbf{\Gamma_x}, \mathbf{\Gamma_{xy}}$. The *estimation error* is:

$$\boldsymbol{\varepsilon} = \hat{\mathbf{x}} - \mathbf{x}$$

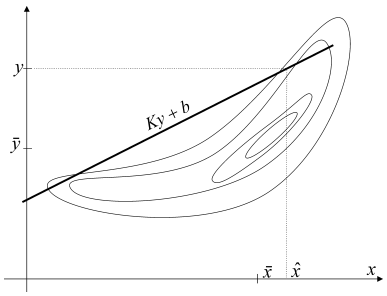


Figure 7.6: Biased linear estimator

The estimator is said to be *unbiased* if $E(\boldsymbol{\varepsilon}) = \mathbf{0}$. It is *orthogonal* if $E\left(\boldsymbol{\varepsilon}\tilde{\mathbf{y}}^{\mathrm{T}}\right) = \mathbf{0}$. This naming comes from the fact that the space of random variables of $\mathbb{R}$ can be equipped with a scalar product defined by $\langle a, b \rangle = E\left((a - \bar{a})\left(b - \bar{b}\right)\right)$ and that if this scalar product is zero, the two random variables $a$ and $b$ are called orthogonal. In the vector case (which is that of our paragraph since $\boldsymbol{\varepsilon}$ and $\tilde{\mathbf{y}}$ are vectors), we say that the two random vectors $\mathbf{a}$ and $\mathbf{b}$ are orthogonal if their components are, in other words $E\left((a_i - \bar{a}_i)\left(b_j - \bar{b}_j\right)\right) = 0$ for all $(i, j)$, or equivalently $E\left((\mathbf{a} - \bar{\mathbf{a}})\left(\mathbf{b} - \bar{\mathbf{b}}\right)^{\mathrm{T}}\right) = \mathbf{0}$. Figure 7.6 represents the contour lines of a probability law for the pair $(x, y)$. The line illustrates a linear estimator. Let us randomly pick a pair $(x, y)$ while respecting its probability law. It is clear that the probability to be above the line is high, in other words the probability to have $\hat{x} < x$ is high, or even that $E(\varepsilon) < 0$. The estimator is thus biased. Figure 7.7 represents four different linear estimators. For estimator (a), $E(\varepsilon) < 0$ and for estimator (c), $E(\varepsilon) > 0$. For estimators (b) and (d), $E(\varepsilon) = 0$ and therefore the two estimators are unbiased. However, it is evident that estimator (b) is better. What differentiates these two is orthogonality. For (d), we have $E(\varepsilon\widetilde{y}) < 0$ (if $\widetilde{y} > 0$, $\varepsilon$ tends to be negative whereas if $\widetilde{y} < 0$, $\varepsilon$ tends to be positive).

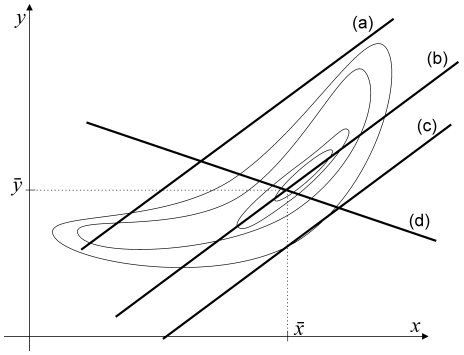


Figure 7.7: Among these four linear estimators, estimator (b), which is unbiased and orthogonal, seems to be the best

THEOREM 7.2.– *Consider two random vectors* $\mathbf{x}$ *and* $\mathbf{y}$. *A unique unbiased orthogonal estimator exists. It is given by:*

$$\hat{\mathbf{x}} = \bar{\mathbf{x}} + \mathbf{K} \cdot (\mathbf{y} - \bar{\mathbf{y}}) \tag{7.3}$$

*where:*

$$\mathbf{K} = \mathbf{\Gamma}_{\mathbf{xy}} \mathbf{\Gamma}_{\mathbf{y}}^{-1} \tag{7.4}$$

*is referred to as the* Kalman gain.

**Example.** Let us consider once again the example in Section 7.1.1. We obtain:

$$\hat{x} = \bar{x} + \mathbf{\Gamma}_{xy} \mathbf{\Gamma}_{y}^{-1} \cdot (y - \bar{y}) = 2 + 0.1 \cdot (y - 10) .$$

The corresponding estimator is illustrated on Figure 7.8.

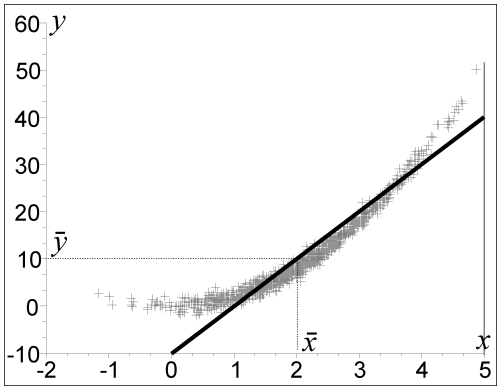


Figure 7.8: Unbiased orthogonal linear estimator

**Proof of the theorem**. We have:

$$E(\boldsymbol{\varepsilon}) = E(\hat{\mathbf{x}} - \mathbf{x}) \overset{[7.2]}{=} E(\mathbf{K}\mathbf{y} + \mathbf{b} - \mathbf{x}) = \mathbf{K}E(\mathbf{y}) + \mathbf{b} - E(\mathbf{x}) = \mathbf{K}\bar{\mathbf{y}} + \mathbf{b} - \bar{\mathbf{x}}$$

The estimator is unbiased if $E(\boldsymbol{\varepsilon}) = \mathbf{0}$, *i.e.*:

$$\mathbf{b} = \bar{\mathbf{x}} - \mathbf{K}\bar{\mathbf{y}} \tag{7.5}$$

which give us [7.3]. In this case:

$$\boldsymbol{\varepsilon} = \hat{\mathbf{x}} - \mathbf{x} \overset{[7.3]}{=} \bar{\mathbf{x}} + \mathbf{K} \cdot (\mathbf{y} - \bar{\mathbf{y}}) - \mathbf{x} = \mathbf{K}\tilde{\mathbf{y}} - \tilde{\mathbf{x}} \tag{7.6}$$

The estimator is orthogonal if:

$$\begin{aligned} E\left(\boldsymbol{\varepsilon}\cdot\widetilde{\mathbf{y}}^{\mathrm{T}}\right)=\mathbf{0} &\overset{[7.6]}{\Leftrightarrow} E\left((\mathbf{K}\widetilde{\mathbf{y}}-\widetilde{\mathbf{x}})\cdot\widetilde{\mathbf{y}}^{\mathrm{T}}\right)=\mathbf{0}\\ &\Leftrightarrow E\left(\mathbf{K}\widetilde{\mathbf{y}}\widetilde{\mathbf{y}}^{\mathrm{T}}-\widetilde{\mathbf{x}}\widetilde{\mathbf{y}}^{\mathrm{T}}\right)=\mathbf{0}\\ &\Leftrightarrow \mathbf{K}\boldsymbol{\Gamma}_{\mathbf{y}}-\boldsymbol{\Gamma}_{\mathbf{xy}}=\mathbf{0}\\ &\Leftrightarrow \mathbf{K}=\boldsymbol{\Gamma}_{\mathbf{xy}}\cdot\boldsymbol{\Gamma}_{\mathbf{y}}^{-1}\end{aligned}$$

which concludes the proof. ■

THEOREM 7.3.– *The covariance matrix of the error associated with the unbiased orthogonal linear estimator is:*

$$\boldsymbol{\Gamma}_{\boldsymbol{\varepsilon}}=\boldsymbol{\Gamma}_{\mathbf{x}}-\mathbf{K}\cdot\boldsymbol{\Gamma}_{\mathbf{yx}}. \tag{7.7}$$

PROOF.– The covariance matrix of $\boldsymbol{\varepsilon}$ in the unbiased case is written as:

$$\begin{aligned}\boldsymbol{\Gamma}_{\boldsymbol{\varepsilon}} &= E\left(\boldsymbol{\varepsilon}\cdot\boldsymbol{\varepsilon}^{\mathrm{T}}\right)\overset{[7.6]}{=}E\left((\mathbf{K}\widetilde{\mathbf{y}}-\widetilde{\mathbf{x}})\cdot(\mathbf{K}\widetilde{\mathbf{y}}-\widetilde{\mathbf{x}})^{\mathrm{T}}\right)\\ &= E\left((\mathbf{K}\widetilde{\mathbf{y}}-\widetilde{\mathbf{x}})\cdot\left(\widetilde{\mathbf{y}}^{\mathrm{T}}\mathbf{K}^{\mathrm{T}}-\widetilde{\mathbf{x}}^{\mathrm{T}}\right)\right)\\ &= E\left(\mathbf{K}\widetilde{\mathbf{y}}\widetilde{\mathbf{y}}^{\mathrm{T}}\mathbf{K}^{\mathrm{T}}-\widetilde{\mathbf{x}}\widetilde{\mathbf{y}}^{\mathrm{T}}\mathbf{K}^{\mathrm{T}}-\mathbf{K}\widetilde{\mathbf{y}}\widetilde{\mathbf{x}}^{\mathrm{T}}+\widetilde{\mathbf{x}}\widetilde{\mathbf{x}}^{\mathrm{T}}\right).\end{aligned}$$

Using the linearity of the expectation operator:

$$\boldsymbol{\Gamma}_{\boldsymbol{\varepsilon}}=\left(\mathbf{K}\boldsymbol{\Gamma}_{\mathbf{y}}-\boldsymbol{\Gamma}_{\mathbf{xy}}\right)\mathbf{K}^{\mathrm{T}}-\mathbf{K}\boldsymbol{\Gamma}_{\mathbf{yx}}+\boldsymbol{\Gamma}_{\mathbf{x}}. \tag{7.8}$$

However, following [7.4], in the orthogonal case $\mathbf{K}\boldsymbol{\Gamma}_{\mathbf{y}}-\boldsymbol{\Gamma}_{\mathbf{xy}}=0$, which concludes the proof. ■

We will now present a theorem that shows that the unbiased orthogonal linear estimator is the best among all unbiased estimators. In order to understand this concept of *best*, we need to recall the inequalities on the covariance matrices (see paragraph 7.1), which tells us that $\boldsymbol{\Gamma}_1\leq\boldsymbol{\Gamma}_2$ if and only if $\Delta=\boldsymbol{\Gamma}_2-\boldsymbol{\Gamma}_1$ is a covariance matrix.

THEOREM 7.4.– *No unbiased linear estimator exists allowing to obtain a smaller covariance matrix on the error $\boldsymbol{\Gamma}_{\boldsymbol{\varepsilon}}$ than the one given by the orthogonal estimator.*

PROOF.– Every possible matrix $\mathbf{K}$ for our unbiased linear estimator is written in the form $\mathbf{K}=\mathbf{K}_0+\boldsymbol{\Delta}$ with $\mathbf{K}_0=\boldsymbol{\Gamma}_{\mathbf{xy}}\boldsymbol{\Gamma}_{\mathbf{y}}^{-1}$ and $\boldsymbol{\Delta}$ being an arbitrary matrix. Following (7.8), the covariance matrix for the error is:

$$\begin{aligned}\boldsymbol{\Gamma}_{\boldsymbol{\varepsilon}} &= \left(\left(\mathbf{K}_0+\boldsymbol{\Delta}\right)\boldsymbol{\Gamma}_{\mathbf{y}}-\boldsymbol{\Gamma}_{\mathbf{xy}}\right)\left(\mathbf{K}_0+\boldsymbol{\Delta}\right)^{\mathrm{T}}-\left(\mathbf{K}_0+\boldsymbol{\Delta}\right)\boldsymbol{\Gamma}_{\mathbf{yx}}+\boldsymbol{\Gamma}_{\mathbf{x}}\\ &= \left(\mathbf{K}_0+\boldsymbol{\Delta}\right)\left(\underbrace{\boldsymbol{\Gamma}_{\mathbf{y}}\mathbf{K}_0^{\mathrm{T}}}_{=\boldsymbol{\Gamma}_{\mathbf{yx}}}+\boldsymbol{\Gamma}_{\mathbf{y}}\boldsymbol{\Delta}^{\mathrm{T}}\right)-\left(\underbrace{\boldsymbol{\Gamma}_{\mathbf{xy}}\mathbf{K}_0^{\mathrm{T}}}_{=\mathbf{K}_0\boldsymbol{\Gamma}_{\mathbf{yx}}}+\boldsymbol{\Gamma}_{\mathbf{xy}}\boldsymbol{\Delta}^{\mathrm{T}}\right)-\left(\mathbf{K}_0\boldsymbol{\Gamma}_{\mathbf{yx}}+\boldsymbol{\Delta}\boldsymbol{\Gamma}_{\mathbf{yx}}\right)+\boldsymbol{\Gamma}_{\mathbf{x}}\\ &= \mathbf{K}_0\boldsymbol{\Gamma}_{\mathbf{yx}}+\boldsymbol{\Delta}\boldsymbol{\Gamma}_{\mathbf{yx}}+\underbrace{\mathbf{K}_0\boldsymbol{\Gamma}_{\mathbf{y}}}_{=\boldsymbol{\Gamma}_{\mathbf{xy}}}\boldsymbol{\Delta}^{\mathrm{T}}+\boldsymbol{\Delta}\boldsymbol{\Gamma}_{\mathbf{y}}\boldsymbol{\Delta}^{\mathrm{T}}-\mathbf{K}_0\boldsymbol{\Gamma}_{\mathbf{yx}}-\boldsymbol{\Gamma}_{\mathbf{xy}}\boldsymbol{\Delta}^{\mathrm{T}}-\mathbf{K}_0\boldsymbol{\Gamma}_{\mathbf{yx}}-\boldsymbol{\Delta}\boldsymbol{\Gamma}_{\mathbf{yx}}+\boldsymbol{\Gamma}_{\mathbf{x}}\\ &= -\mathbf{K}_0\boldsymbol{\Gamma}_{\mathbf{yx}}+\boldsymbol{\Delta}\boldsymbol{\Gamma}_{\mathbf{y}}\boldsymbol{\Delta}^{\mathrm{T}}+\boldsymbol{\Gamma}_{\mathbf{x}}.\end{aligned}$$

Since $\boldsymbol{\Delta}\boldsymbol{\Gamma}_{\mathbf{y}}\boldsymbol{\Delta}^{\mathrm{T}}$ is always positive symmetric, the covariance matrix $\boldsymbol{\Gamma}_{\boldsymbol{\varepsilon}}$ is minimal for $\boldsymbol{\Delta}=\mathbf{0}$, i.e.

for $\mathbf{K} = \mathbf{\Gamma_{xy}}\mathbf{\Gamma_y}^{-1}$, which corresponds to the orthogonal unbiased estimator. ■

## 7.3 Application to linear estimation

Let us assume that $\mathbf{x}$ and $\mathbf{y}$ are connected by the relation:

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \boldsymbol{\beta}$$

where $\boldsymbol{\beta}$ is a centered random vector non-correlated with $\mathbf{x}$. The covariance matrices of $\mathbf{x}$ and $\boldsymbol{\beta}$ are denoted by $\mathbf{\Gamma_x}$ and $\mathbf{\Gamma}_\beta$. Let us utilize the results obtained in the previous section in order to find the best unbiased linear estimator for $\mathbf{x}$ (refer to [WAL 14] for more details on linear estimation). We have:

$$\begin{array}{rcl}
\bar{\mathbf{y}} & = & \mathbf{C}\bar{\mathbf{x}} + \bar{\boldsymbol{\beta}} = \mathbf{C}\bar{\mathbf{x}} \\
\mathbf{\Gamma_y} & \overset{[7.1]}{=} & \mathbf{C}\mathbf{\Gamma_x}\mathbf{C}^{\mathrm{T}} + \mathbf{\Gamma}_\beta \\
\mathbf{\Gamma_{xy}} & = & E\left(\tilde{\mathbf{x}} \cdot \tilde{\mathbf{y}}^{\mathrm{T}}\right) = E\left(\tilde{\mathbf{x}} \cdot \left(\mathbf{C}\tilde{\mathbf{x}} + \tilde{\boldsymbol{\beta}}\right)^{\mathrm{T}}\right) = E\left(\tilde{\mathbf{x}} \cdot \tilde{\mathbf{x}}^{\mathrm{T}}\mathbf{C}^{\mathrm{T}} + \tilde{\mathbf{x}} \cdot \tilde{\boldsymbol{\beta}}^{\mathrm{T}}\right) \\
& = & E\left(\tilde{\mathbf{x}} \cdot \tilde{\mathbf{x}}^{\mathrm{T}}\right)\mathbf{C}^{\mathrm{T}} + \underbrace{E\left(\tilde{\mathbf{x}} \cdot \tilde{\boldsymbol{\beta}}^{\mathrm{T}}\right)}_{=\mathbf{0}} = \mathbf{\Gamma_x}\mathbf{C}^{\mathrm{T}}.
\end{array} \tag{7.9}$$

Consequently, the best unbiased estimator for $\mathbf{x}$ and covariance matrix of the error can be obtained from $\mathbf{\Gamma_x}, \mathbf{\Gamma}_\beta, \mathbf{C}, \bar{\mathbf{x}}$ by using the following formulas:

$$\begin{array}{llcll}
\text{(i)} & \hat{\mathbf{x}} & \overset{[7.3]}{=} & \bar{\mathbf{x}} + \mathbf{K}\tilde{\mathbf{y}} & \text{(estimation)} \\
\text{(ii)} & \mathbf{\Gamma}_\varepsilon & \overset{[7.7]}{=} & \mathbf{\Gamma_x} - \mathbf{K}\mathbf{C}\mathbf{\Gamma_x} & \text{(covariance of the error)} \\
\text{(iii)} & \tilde{\mathbf{y}} & \overset{[7.9]}{=} & \mathbf{y} - \mathbf{C}\bar{\mathbf{x}} & \text{(innovation)} \\
\text{(iv)} & \mathbf{\Gamma_y} & \overset{[7.9]}{=} & \mathbf{C}\mathbf{\Gamma_x}\mathbf{C}^{\mathrm{T}} + \mathbf{\Gamma}_\beta & \text{(covariance of the innovation)} \\
\text{(v)} & \mathbf{K} & \overset{[7.4,7.9]}{=} & \mathbf{\Gamma_x}\mathbf{C}^{\mathrm{T}}\mathbf{\Gamma_y}^{-1} & \text{(Kalman gain)}
\end{array} \tag{7.10}$$

REMARK 7.1.– *Figure 7.5 illustrates a situation in which it could be advantageous not to use a linear estimator. Here, the chosen estimator corresponds to* $\hat{x} = E(x|y)$. *In the particular case where the pair* $(\mathbf{x}, \mathbf{y})$ *is Gaussian, the estimator* $\hat{\mathbf{x}} = E(\mathbf{x}|\mathbf{y})$ *corresponds to the unbiased orthogonal estimator. In this case, we have, following [7.10]:*

$$\begin{array}{rcl}
E(\mathbf{x}|\mathbf{y}) & = & \bar{\mathbf{x}} + \mathbf{\Gamma_{xy}}\mathbf{\Gamma_y}^{-1}(\mathbf{y} - \bar{\mathbf{y}}) \\
E\left(\varepsilon \cdot \varepsilon^T|\mathbf{y}\right) & = & E\left((\hat{\mathbf{x}} - \mathbf{x})(\hat{\mathbf{x}} - \mathbf{x})^T|\mathbf{y}\right) = \mathbf{\Gamma_x} - \mathbf{\Gamma_{xy}}\mathbf{\Gamma_y}^{-1}\mathbf{\Gamma_{yx}}.
\end{array}$$

## 7.4 Kalman filter

This paragraph presents the Kalman filter (refer to [DEL 93] for more information). Let us consider the system described by the following state equations:

$$\left\{ \begin{array}{rcl} \mathbf{x}_{k+1} & = & \mathbf{A}_k \mathbf{x}_k + \mathbf{u}_k + \boldsymbol{\alpha}_k \\ \mathbf{y}_k & = & \mathbf{C}_k \mathbf{x}_k + \boldsymbol{\beta}_k \end{array} \right.$$

where $\boldsymbol{\alpha}_k$ and $\boldsymbol{\beta}_k$ are random, independent Gaussian noises white in time. By white in time we mean that the vectors $\boldsymbol{\alpha}_{k_1}$ and $\boldsymbol{\alpha}_{k_2}$ (or $\boldsymbol{\beta}_{k_1}$ and $\boldsymbol{\beta}_{k_2}$) are independent of one another if $k_1 \neq k_2$. The Kalman filter alternates between two phases: *correction* and *prediction*. To understand the mechanism of the filter, let us position ourselves at time $k$ and assume that we have already processed the measurements $\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_{k-1}$. At this stage, the state vector is a random vector that we will denote by $\mathbf{x}_{k|k-1}$ (since we are at time $k$ and the measurements have been processed until $k-1$). This random vector is represented by an estimation denoted by $\hat{\mathbf{x}}_{k|k-1}$ and a covariance matrix $\boldsymbol{\Gamma}_{k|k-1}$.

**Correction**. Let us take the measurement $\mathbf{y}_k$. The random vector representing the state is now $\mathbf{x}_{k|k}$, which is different than $\mathbf{x}_{k|k-1}$ since $\mathbf{x}_{k|k}$ has knowledge of the measurement $\mathbf{y}$. The expectation $\hat{\mathbf{x}}_{k|k}$ and the covariance matrix $\boldsymbol{\Gamma}_{k|k}$ associated to $\mathbf{x}_{k|k}$ are given by Equations [7.10]. We therefore have:

$$\begin{array}{llcll} \text{(i)} & \hat{\mathbf{x}}_{k|k} & = & \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \cdot \widetilde{\mathbf{y}}_k & \text{(corrected estimation)} \\ \text{(ii)} & \boldsymbol{\Gamma}_{k|k} & = & \boldsymbol{\Gamma}_{k|k-1} - \mathbf{K}_k \cdot \mathbf{C}_k \boldsymbol{\Gamma}_{k|k-1} & \text{(corrected covariance)} \\ \text{(iii)} & \widetilde{\mathbf{y}}_k & = & \mathbf{y}_k - \mathbf{C}_k \hat{\mathbf{x}}_{k|k-1} & \text{(innovation)} \\ \text{(iv)} & \mathbf{S}_k & = & \mathbf{C}_k \boldsymbol{\Gamma}_{k|k-1} \mathbf{C}_k^{\mathrm{T}} + \boldsymbol{\Gamma}_{\beta_k} & \text{(covariance of the innovation)} \\ \text{(v)} & \mathbf{K}_k & = & \boldsymbol{\Gamma}_{k|k-1} \mathbf{C}_k^{\mathrm{T}} \mathbf{S}_k^{-1} & \text{(Kalman gain)} \end{array} \tag{7.11}$$

**Prediction**. Given the measurements $\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_k$, the random vector representing the state is now $\mathbf{x}_{k+1|k}$. Let us calculate its expectation $\hat{\mathbf{x}}_{k+1|k}$ and its covariance matrix $\boldsymbol{\Gamma}_{k+1|k}$. Since

$$\mathbf{x}_{k+1} = \mathbf{A}_k \mathbf{x}_k + \mathbf{u}_k + \boldsymbol{\alpha}_k,$$

we have, following [7.1]:

$$\hat{\mathbf{x}}_{k+1|k} = \mathbf{A}_k \hat{\mathbf{x}}_{k|k} + \mathbf{u}_k \tag{7.12}$$

and

$$\boldsymbol{\Gamma}_{k+1|k} = \mathbf{A}_k \cdot \boldsymbol{\Gamma}_{k|k} \cdot \mathbf{A}_k^{\mathrm{T}} + \boldsymbol{\Gamma}_{\boldsymbol{\alpha}_k}. \tag{7.13}$$

**Kalman filter**. The complete Kalman filter is given by the following equations :

$$
\begin{array}{lcll}
\hat{\mathbf{x}}_{k+1|k} & \overset{[7.12]}{=} & \mathbf{A}_k\hat{\mathbf{x}}_{k|k} + \mathbf{u}_k & \text{(predicted estimation)} \\
\mathbf{\Gamma}_{k+1|k} & \overset{[7.13]}{=} & \mathbf{A}_k \cdot \mathbf{\Gamma}_{k|k} \cdot \mathbf{A}_k^{\mathrm{T}} + \mathbf{\Gamma}_{\boldsymbol{\alpha}_k} & \text{(predicted covariance)} \\
\hat{\mathbf{x}}_{k|k} & \overset{[7.10,\mathrm{i}]}{=} & \hat{\mathbf{x}}_{k|k-1}+\mathbf{K}_k \cdot \widetilde{\mathbf{y}}_k & \text{(corrected estimation)} \\
\mathbf{\Gamma}_{k|k} & \overset{[7.10,\mathrm{ii}]}{=} & (\mathbf{I} - \mathbf{K}_k\mathbf{C}_k)\,\mathbf{\Gamma}_{k|k-1} & \text{(corrected covariance)} \\
\widetilde{\mathbf{y}}_k & \overset{[7.10,\mathrm{iii}]}{=} & \mathbf{y}_k - \mathbf{C}_k\hat{\mathbf{x}}_{k|k-1} & \text{(innovation)} \\
\mathbf{S}_k & \overset{[7.10,\mathrm{iv}]}{=} & \mathbf{C}_k\mathbf{\Gamma}_{k|k-1}\mathbf{C}_k^{\mathrm{T}} + \mathbf{\Gamma}_{\boldsymbol{\beta}_k} & \text{(covariance of the innovation)} \\
\mathbf{K}_k & \overset{[7.10,\mathrm{v}]}{=} & \mathbf{\Gamma}_{k|k-1}\mathbf{C}_k^{\mathrm{T}}\mathbf{S}_k^{-1} & \text{(Kalman gain)}
\end{array}
$$

Figure 7.9 illustrates the fact that the Kalman filter stores the vector $\hat{\mathbf{x}}_{k+1|k}$ and the matrix $\mathbf{\Gamma}_{k+1|k}$. Its inputs are $\mathbf{y}_k$, $\mathbf{u}_k$, $\mathbf{A}_k$, $\mathbf{C}_k$, $\mathbf{\Gamma}_{\boldsymbol{\alpha}_k}$ and $\mathbf{\Gamma}_{\boldsymbol{\beta}_k}$. The quantities $\hat{\mathbf{x}}_{k|k}$, $\mathbf{\Gamma}_{k|k}$, $\widetilde{\mathbf{y}}_k$, $\mathbf{S}_k$, $\mathbf{K}_k$ are auxiliary variables. The delay is activated by a clock and when activated, $k$ is incremented by 1.

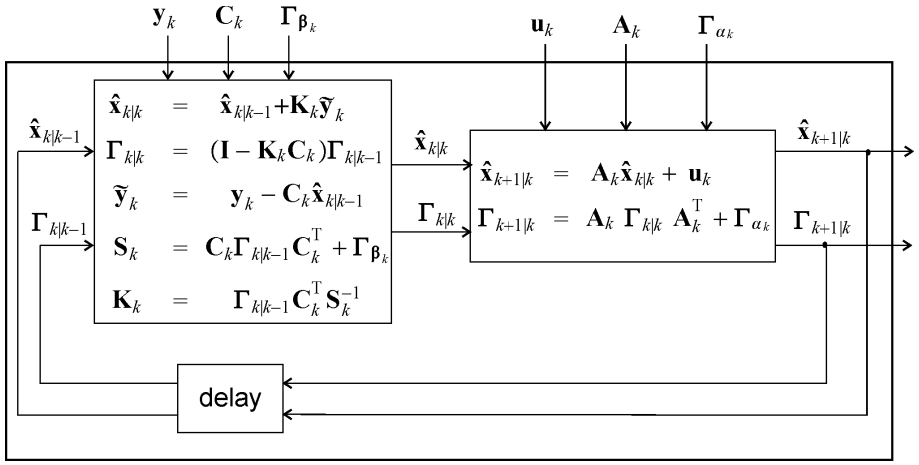


Figure 7.9: The Kalman filter is composed of a corrector followed by a predictor

The following MATLAB function implements the Kalman filter. In this program, we have the following correspondences: `x0`$\leftrightarrow \hat{\mathbf{x}}_{k|k-1}$, `G1`$\leftrightarrow \mathbf{\Gamma}_{k|k-1}$, `x1`$\leftrightarrow \hat{\mathbf{x}}_{k+1|k}$, `G1`$\leftrightarrow \mathbf{\Gamma}_{k+1|k}$ , `xup`$\leftrightarrow \hat{\mathbf{x}}_{k|k}$, `Gup`$\leftrightarrow \mathbf{\Gamma}_{k|k}$ (the term `up` refers to *update*, in other words correction).

```
function [x1,G1]=kalman(x0,G0,u,y,Galpha,Gbeta,A,C);
   S=C*G0*C'+Gbeta;
   K=G0*C'*inv(S);
   ytilde=y-C*x0;
   xup=x0+K*ytilde;
   Gup=G0-K*C*G0;
   x1=A*xup + u;
   G1=A*Gup*A'+Galpha;
end
```

**Positivity lost**. Due to numerical problems, the covariance of the innovation $\mathbf{S}_k$ can sometimes loose its positivity. If such a problem arises, it is preferable to replace the corrected covariance equation

by :

$$\mathbf{\Gamma}_{k|k} = \sqrt{\left(\mathbf{I} - \mathbf{K}_k\mathbf{C}_k\right)\mathbf{\Gamma}_{k|k-1}\mathbf{\Gamma}_{k|k-1}^{\mathrm{T}}\left(\mathbf{I} - \mathbf{K}_k\mathbf{C}_k\right)^{\mathrm{T}}}$$

which will always be positive definite, even when the matrix $\mathbf{\Gamma}_{k|k-1}$ is not. The Kalman filter equations will then be more stable in the sense that a slight error on the positive character of the covariance matrices is removed at the next iteration.

**Predictor.** When no measurement is available, the Kalman filter operates in *predictor* mode. In order to be able to use the `kalman` function, $\mathbf{y}$, $\mathbf{\Gamma}_\beta, \mathbf{C}$ have to become empty quantities. However, they have to have correct dimensions in order to allow, in MATLAB, to perform matrix operations. The function call will then be as follows:

```
[xhat,Gx]=kalman(xhat,Gx,u,eye(0,1),Galpha,eye(0,0),A,eye(0,length(x)))
```

**Initialization** : Most of the time, we have no idea of the initial state $\mathbf{x}_0$. In this case, we generally set

$$\hat{\mathbf{x}}_0 = (0, 0, \ldots, 0) \text{ et } \mathbf{\Gamma}_{\mathbf{x}}(0) = \begin{pmatrix} \frac{1}{\varepsilon^2} & 0 & 0 & \\ 0 & \frac{1}{\varepsilon^2} & & \\ & & \ddots & 0 \\ 0 & & 0 & \frac{1}{\varepsilon^2} \end{pmatrix},$$

where $\varepsilon$ in a small positive number (for instance 0.001). More or less, this assumption amounts to say that $\mathbf{x}_0$ is inside a sphere centered in zero and a radius $\frac{1}{\varepsilon}$.

**Stationnary case**. For time independent systems $\mathbf{\Gamma}_{k+1|k}$ converges to a matrix $\mathbf{\Gamma}$. For large $k$ , $\mathbf{\Gamma} = \mathbf{\Gamma}_{k+1|k} = \mathbf{\Gamma}_{k|k-1}$. Thus:

$$\begin{aligned}
\mathbf{\Gamma}_{k+1|k} &= \mathbf{A}\cdot\mathbf{\Gamma}_{k|k}\cdot\mathbf{A}^{\mathrm{T}} + \mathbf{\Gamma}_{\boldsymbol{\alpha}} \\
&= \mathbf{A}\cdot\left(\mathbf{I} - \mathbf{K}_k\mathbf{C}\right)\mathbf{\Gamma}_{k|k-1}\cdot\mathbf{A}^{\mathrm{T}} + \mathbf{\Gamma}_{\boldsymbol{\alpha}} \\
&= \mathbf{A}\cdot\left(\mathbf{I} - \left(\mathbf{\Gamma}_{k|k-1}\mathbf{C}^{\mathrm{T}}\mathbf{S}^{-1}\right)\mathbf{C}\right)\mathbf{\Gamma}_{k|k-1}\cdot\mathbf{A}^{\mathrm{T}} + \mathbf{\Gamma}_{\boldsymbol{\alpha}} \\
&= \mathbf{A}\cdot\left(\mathbf{I} - \left(\mathbf{\Gamma}_{k|k-1}\mathbf{C}^{\mathrm{T}}\left(\mathbf{C}\mathbf{\Gamma}_{k|k-1}\mathbf{C}^{\mathrm{T}} + \mathbf{\Gamma}_\beta\right)^{-1}\right)\mathbf{C}\right)\mathbf{\Gamma}_{k|k-1}\cdot\mathbf{A}^{\mathrm{T}} + \mathbf{\Gamma}_{\boldsymbol{\alpha}}
\end{aligned}$$

Therefor, for large $k$,

$$\mathbf{\Gamma} = \mathbf{A}\cdot\left(\mathbf{\Gamma} - \mathbf{\Gamma}\cdot\mathbf{C}^{\mathrm{T}}\cdot\left(\mathbf{C}\cdot\mathbf{\Gamma}\cdot\mathbf{C}^{\mathrm{T}} + \mathbf{\Gamma}_\beta\right)^{-1}\cdot\mathbf{C}\cdot\mathbf{\Gamma}\right)\cdot\mathbf{A}^{\mathrm{T}} + \mathbf{\Gamma}_{\boldsymbol{\alpha}}$$

which is an equation of Ricatti in $\mathbf{\Gamma}$. It can be solved by the sequence

$$\mathbf{\Gamma}(k+1) = \mathbf{A}\cdot\left(\mathbf{\Gamma}(k) - \mathbf{\Gamma}(k)\cdot\mathbf{C}^{\mathrm{T}}\cdot\left(\mathbf{C}\cdot\mathbf{\Gamma}(k)\cdot\mathbf{C}^{\mathrm{T}} + \mathbf{\Gamma}_\beta\right)^{-1}\cdot\mathbf{C}\cdot\mathbf{\Gamma}(k)\right)\cdot\mathbf{A}^{\mathrm{T}} + \mathbf{\Gamma}_{\boldsymbol{\alpha}}$$

up to the equilibrium and starting with $\mathbf{\Gamma}(0)$ large (e.g. $10^{10}\cdot\mathbf{I}$). This computation can be done before the implementation of the filter, which allows us to avoid unnecessary real time computation of $\mathbf{\Gamma}_{k+1|k}$. The corresponding stationary Kalman filter is thus :

$$\hat{\mathbf{x}}_{k+1|k} = \mathbf{A}\hat{\mathbf{x}}_{k|k-1} + \mathbf{A}\mathbf{K}\cdot\left(\mathbf{y}_k - \mathbf{C}\hat{\mathbf{x}}_{k|k-1}\right) + \mathbf{u}_k$$

with $\mathbf{K} = \mathbf{\Gamma}\mathbf{C}^{\mathrm{T}}\left(\mathbf{C}\mathbf{\Gamma}\mathbf{C}^{\mathrm{T}} + \mathbf{\Gamma}\right)^{-1}$.

## 7.5 Kalman-Bucy

We now consider the linear continuous time system described by

$$\left\{\begin{array}{rcl} \dot{\mathbf{x}}_t &=& \mathbf{A}_t\mathbf{x}_t + \mathbf{u}_t + \boldsymbol{\alpha}_t \\ \mathbf{y}_t &=& \mathbf{C}_t\mathbf{x}_t + \boldsymbol{\beta}_t \end{array}\right.$$

where $\mathbf{u}_t$ and $\mathbf{y}_t$ are measured. The noise $\boldsymbol{\alpha}_t$ and $\boldsymbol{\beta}_t$ are assumed to be white and Gaussian. After a discretization with a sampling time $dt$ which is assumed to be infinitely small, we get :

$$\left\{\begin{array}{rcl} \mathbf{x}_{k+1} &=& \underbrace{(\mathbf{I} + dt\cdot\mathbf{A}_t)}_{\mathbf{A}_k}\ \mathbf{x}_k + \underbrace{dt\cdot\mathbf{u}_t}_{\mathbf{u}_k} + \boldsymbol{\alpha}_\mathbf{k} \\ \mathbf{y}_k &=& \mathbf{C}_t\mathbf{x}_k + \boldsymbol{\beta}_k \end{array}\right.$$

For this discrete-time system, we can apply the classical Kalman filter given at Section 7.4. We have the following correspondences.

$$\begin{array}{rclcrclcrcl} \mathbf{A}_k & \rightarrow & (\mathbf{I} + dt\mathbf{A}_t) & & \mathbf{u}_k & \rightarrow & dt\cdot\mathbf{u}_t & & & & \\ \hat{\mathbf{x}}_{k+1|k} & \rightarrow & \hat{\mathbf{x}}_{t+dt} & & \hat{\mathbf{x}}_{k|k-1} & \rightarrow & \hat{\mathbf{x}}_t & & \hat{\mathbf{x}}_{k|k} & \rightarrow & \mathring{\mathbf{x}}_t \\ \mathbf{\Gamma}_{k+1|k} & \rightarrow & \mathbf{\Gamma}_{t+dt} & & \mathbf{\Gamma}_{k|k-1} & \rightarrow & \mathbf{\Gamma}_t & & \mathbf{\Gamma}_{k|k} & \rightarrow & \mathring{\mathbf{\Gamma}}_t \\ \mathbf{\Gamma}_{\boldsymbol{\alpha}} & \rightarrow & \frac{1}{dt}\cdot\mathbf{\Gamma}_{\boldsymbol{\alpha}} & & \mathbf{\Gamma}_{\boldsymbol{\beta}} & \rightarrow & \frac{1}{dt}\cdot\mathbf{\Gamma}_{\boldsymbol{\beta}} & & & & \end{array}$$

Note that, as explained in Exercise 7.6, the covariance for the state noise should of the same order as $\frac{1}{dt}$ (denoted by $O(\frac{1}{dt})$ ) and this is why it has the form $\frac{1}{dt}\cdot\mathbf{\Gamma}_{\boldsymbol{\alpha}}$. Without this coefficient of $\frac{1}{dt}$, the noise would have no effect on the evolution of the system when $dt$ tends to zero. In a dual manner, the covariance for the measurements should also be of order $\frac{1}{dt}$ and this is why we took the covariance matrix $\frac{1}{dt}\cdot\mathbf{\Gamma}_{\boldsymbol{\beta}}$. Otherwise, the Kalman filter would immediately find the true state when $dt \rightarrow 0$, which is not realistic. The Kalman filter is now:

$$\left\{\begin{array}{rcll} \hat{\mathbf{x}}_{t+dt} &=& (\mathbf{I} + dt\mathbf{A}_t)\mathring{\mathbf{x}}_t + dt\cdot\mathbf{u}_t & (\mathrm{i}) \\ \mathbf{\Gamma}_{t+dt} &=& (\mathbf{I} + dt\mathbf{A}_t)\cdot\mathring{\mathbf{\Gamma}}_t\cdot(\mathbf{I} + dt\mathbf{A}_t)^T + \frac{1}{dt}\cdot\mathbf{\Gamma}_{\boldsymbol{\alpha}} & (ii) \\ \mathring{\mathbf{x}}_t &=& \hat{\mathbf{x}}_t + \mathbf{K}_t\cdot\widetilde{\mathbf{y}}_t & (iii) \\ \mathring{\mathbf{\Gamma}}_t &=& (\mathbf{I} - \mathbf{K}_t\mathbf{C}_t)\cdot\mathbf{\Gamma}_t & (iv) \\ \widetilde{\mathbf{y}}_t &=& \mathbf{y}_t - \mathbf{C}_t\hat{\mathbf{x}}_t & (v) \\ \mathbf{S}_t &=& \mathbf{C}_t\mathbf{\Gamma}_t\mathbf{C}_t^{\mathrm{T}} + \frac{1}{dt}\cdot\mathbf{\Gamma}_{\boldsymbol{\beta}} & (vi) \\ \mathbf{K}_t &=& \mathbf{\Gamma}_t\mathbf{C}_t^{\mathrm{T}}\mathbf{S}_t^{-1} & (viii) \end{array}\right.$$

Taking into account the fact that $\mathbf{C}_t\mathbf{\Gamma}_t\mathbf{C}_t^{\mathrm{T}}$ is small (line (vi)) compared to $\frac{1}{dt}\cdot\mathbf{\Gamma}_{\boldsymbol{\beta}}$ (which is infinitely large) and we can simplify it. We conclude that $\mathbf{S}_t$ is huge ($= O(1/dt)$) and thus that $\mathbf{S}_t^{-1}$ at line

(viii) is small ($= O(dt)$). The Kalman filter becomes

$$\left\{\begin{array}{lcl}\hat{\mathbf{x}}_{t+dt} &=& (\mathbf{I}+dt\mathbf{A}_t)\left(\hat{\mathbf{x}}_t+\mathbf{K}_t\cdot(\mathbf{y}_t-\mathbf{C}_t\hat{\mathbf{x}}_t)\right)+dt\cdot\mathbf{u}_t\\ \mathbf{\Gamma}_{t+dt} &=& (\mathbf{I}+dt\mathbf{A}_t)\cdot(\mathbf{I}-\mathbf{K}_t\mathbf{C}_t)\,\mathbf{\Gamma}_t\cdot(\mathbf{I}+dt\mathbf{A}_t)^T+\frac{1}{dt}\cdot\mathbf{\Gamma}_{\boldsymbol{\alpha}}\\ \mathbf{K}_t &=& dt\mathbf{\Gamma}_t\mathbf{C}_t^{\mathrm{T}}\mathbf{\Gamma}_{\boldsymbol{\beta}}^{-1}.\end{array}\right.$$

Since $\mathbf{K}_t$ is small ($= O(dt)$), it is more convenient to introduce the *Kalman-Bucy* gain given by $\mathbf{K}_b = \mathbf{\Gamma}_t\mathbf{C}_t^{\mathrm{T}}\mathbf{\Gamma}_{\boldsymbol{\beta}}^{-1}$ which is $O(1)$, *i.e.*, neither infinitely small nor infinitely large. We have

$$\left\{\begin{array}{lcl}\hat{\mathbf{x}}_{t+dt} &=& \underbrace{(\mathbf{I}+dt\mathbf{A}_t)\left(\hat{\mathbf{x}}_t+dt\mathbf{K}_b\cdot(\mathbf{y}_t-\mathbf{C}_t\hat{\mathbf{x}}_t)\right)}_{=(\mathbf{I}+dt\mathbf{A}_t)\hat{\mathbf{x}}_t+dt\mathbf{K}_b\cdot(\mathbf{y}_t-\mathbf{C}_t\hat{\mathbf{x}}_t)+O(dt)^2}+dt\cdot\mathbf{u}_t\\ \mathbf{\Gamma}_{t+dt} &=& \underbrace{(\mathbf{I}+dt\mathbf{A}_t)\cdot(\mathbf{I}-dt\mathbf{K}_b\mathbf{C}_t)\,\mathbf{\Gamma}_t\cdot(\mathbf{I}+dt\mathbf{A}_t)^T}_{=\mathbf{\Gamma}_t+dt\left(\mathbf{A}_t\mathbf{\Gamma}_t-\mathbf{K}_b\mathbf{C}_t\mathbf{\Gamma}_t+\mathbf{\Gamma}_t\mathbf{A}_t^T\right)+O(dt)^2}+\frac{1}{dt}\cdot\mathbf{\Gamma}_{\boldsymbol{\alpha}}\\ \mathbf{K}_b &=& \mathbf{\Gamma}_t\mathbf{C}_t^{\mathrm{T}}\mathbf{\Gamma}_{\boldsymbol{\beta}}^{-1}\end{array}\right.$$

Or equivalently:

$$\left\{\begin{array}{lcl}\frac{\hat{\mathbf{x}}_{t+dt}-\hat{\mathbf{x}}_t}{dt} &=& \mathbf{A}_t\hat{\mathbf{x}}_t+\mathbf{K}_b\cdot(\mathbf{y}_t-\mathbf{C}_t\hat{\mathbf{x}}_t)+\mathbf{u}_t\\ \frac{\mathbf{\Gamma}_{t+dt}-\mathbf{\Gamma}_t}{dt} &=& \left(\mathbf{A}_t\mathbf{\Gamma}_t-\mathbf{K}_b\mathbf{C}_t\mathbf{\Gamma}_t+\mathbf{\Gamma}_t\mathbf{A}_t^{\mathrm{T}}\right)+\frac{1}{(dt)^2}\cdot\mathbf{\Gamma}_{\boldsymbol{\alpha}}\\ \mathbf{K}_b &=& \mathbf{\Gamma}_t\mathbf{C}_t^{\mathrm{T}}\mathbf{\Gamma}_{\boldsymbol{\beta}}^{-1}.\end{array}\right.$$

Now, $\mathbf{C}_t\mathbf{\Gamma}_t = \left(\mathbf{\Gamma}_t\mathbf{C}_t^{\mathrm{T}}\right)^T = (\mathbf{K}_b\mathbf{\Gamma}_{\boldsymbol{\beta}})^T = \mathbf{\Gamma}_{\boldsymbol{\beta}}\mathbf{K}_b^{\mathrm{T}}$. Therefore, we the Kalman-Busy filter

$$\left\{\begin{array}{lcl}\frac{d}{dt}\hat{\mathbf{x}}_t &=& \mathbf{A}_t\hat{\mathbf{x}}_t+\mathbf{K}_b\left(\mathbf{y}_t-\mathbf{C}_t\hat{\mathbf{x}}_t\right)+\mathbf{u}_t\\ \frac{d}{dt}\mathbf{\Gamma}_t &=& \mathbf{A}_t\mathbf{\Gamma}_t+\mathbf{\Gamma}_t\cdot\mathbf{A}_t^{\mathrm{T}}-\mathbf{K}_{kb}\mathbf{\Gamma}_{\boldsymbol{\beta}}\mathbf{K}_b^T+\frac{1}{(dt)^2}\cdot\mathbf{\Gamma}_{\boldsymbol{\alpha}}\\ \mathbf{K}_b &=& \mathbf{\Gamma}_t\mathbf{C}_t^{\mathrm{T}}\mathbf{\Gamma}_{\boldsymbol{\beta}}^{-1}.\end{array}\right.$$

This Kalman-Busy filter corresponds to a continuous version of the Kalman filter.

## 7.6 Extended Kalman Filter

As we have seen in previous chapters, a robot is described by continuous state equations of the form

$$\left\{\begin{array}{lcl}\dot{\mathbf{x}} &=& \mathbf{f}_c(\mathbf{x},\mathbf{u})\\ \mathbf{y} &=& \mathbf{g}(\mathbf{x})\end{array}\right.$$

which are nonlinear. A Kalman filter can still be used to estimate the state vector $\mathbf{x}$, but we need to discretize the system and to linearize it. A possible discretization can be performed with an Euler

method. We get

$$\left\{ \begin{array}{ll} \mathbf{x}_{k+1} & = \mathbf{x}_k + dt \cdot \mathbf{f}_c(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \\ \mathbf{y}_k & = \mathbf{g}(\mathbf{x}_k). \end{array} \right.$$

To linearize the system, we assume that we have an estimation $\hat{\mathbf{x}}_k$ of the state vector. Therefore,

$$\left\{ \begin{array}{rcl} \mathbf{x}_{k+1} & \simeq & \mathbf{f}(\hat{\mathbf{x}}_k, \mathbf{u}_k) + \frac{\partial \mathbf{f}(\hat{\mathbf{x}}_k, \mathbf{u}_k)}{\partial \mathbf{x}} \cdot (\mathbf{x}_k - \hat{\mathbf{x}}_k) \\ \mathbf{y}_k & \simeq & \mathbf{g}(\hat{\mathbf{x}}_k) + \frac{d\mathbf{g}(\hat{\mathbf{x}}_k)}{d\mathbf{x}} \cdot (\mathbf{x}_k - \hat{\mathbf{x}}_k) \, . \end{array} \right.$$

If we set

$$\mathbf{A}_k = \frac{\partial \mathbf{f}(\hat{\mathbf{x}}_k, \mathbf{u}_k)}{\partial \mathbf{x}} \text{ and } \mathbf{C}_k = \frac{d\mathbf{g}(\hat{\mathbf{x}}_k)}{d\mathbf{x}}$$

we get,

$$\left\{ \begin{array}{rcl} \mathbf{x}_{k+1} & \simeq & \mathbf{A}_k \cdot \mathbf{x}_k + \underbrace{(\mathbf{f}(\hat{\mathbf{x}}_k, \mathbf{u}_k) - \mathbf{A}_k \cdot \hat{\mathbf{x}}_k)}_{\mathbf{v}_k} \\ \underbrace{(\mathbf{y}_k - \mathbf{g}(\hat{\mathbf{x}}_k) + \mathbf{C}_k \cdot \hat{\mathbf{x}}_k)}_{\mathbf{z}_k} & \simeq & \mathbf{C}_k \cdot \mathbf{x}_k. \end{array} \right.$$

This approximation can be written as

$$\left\{ \begin{array}{rcl} \mathbf{x}_{k+1} & = & \mathbf{A}_k \mathbf{x}_k + \mathbf{v}_k + \boldsymbol{\alpha}_k \\ \mathbf{z}_k & = & \mathbf{C}_k \mathbf{x}_k + \boldsymbol{\beta}_k. \end{array} \right.$$

The noises $\boldsymbol{\alpha}_k$ and $\boldsymbol{\beta}_k$ are non Gaussian and are not white, since they include some linearization errors. A classical Kalman filter can be used (with $\mathbf{v}_k$ and $\mathbf{z}_k$, which are known, take the role of $\mathbf{u}_k$ and $\mathbf{y}_k$) but its behavior is not reliable anymore. If we are lucky, the results will not be so bad even if they are too optimistic in general. But very often, we are not lucky and the filter provides wrong results. Moreover, it is difficult to quantify the linearization errors in order to deduce covariance matrices for the noises.

In our context, $\hat{\mathbf{x}}_k$ corresponds to $\hat{\mathbf{x}}_{k|k-1}$, the estimation we have for the state $\mathbf{x}_k$, taking into account all the past measurements. But it can also correspond to $\hat{\mathbf{x}}_{k|k}$, when available. Therefore,

the linearized Kalman filter, called the *extended Kalman filter*, is:

$$
\begin{array}{lcll}
\mathbf{A}_k & = & \frac{\partial \mathbf{f}(\hat{\mathbf{x}}_{k|k}, \mathbf{u}_k)}{\partial \mathbf{x}} & \text{(evolution matrix)} \\
\mathbf{C}_k & = & \frac{d\mathbf{g}(\hat{\mathbf{x}}_{k|k-1})}{d\mathbf{x}} & \text{(observation matrix)} \\
\hat{\mathbf{x}}_{k+1|k} & = & \mathbf{A}_k \hat{\mathbf{x}}_{k|k} + \underbrace{\left(\mathbf{f}(\hat{\mathbf{x}}_{k|k}, \mathbf{u}_k) - \mathbf{A}_k \cdot \hat{\mathbf{x}}_{k|k}\right)}_{\mathbf{v}_k} = \mathbf{f}(\hat{\mathbf{x}}_{k|k}, \mathbf{u}_k) & \text{(predicted estimation)} \\
\mathbf{\Gamma}_{k+1|k} & = & \mathbf{A}_k \cdot \mathbf{\Gamma}_{k|k} \cdot \mathbf{A}_k^{\mathrm{T}} + \mathbf{\Gamma}_{\boldsymbol{\alpha}_k} & \text{(predicted covariance)} \\
\hat{\mathbf{x}}_{k|k} & = & \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \cdot \tilde{\mathbf{z}}_k & \text{(corrected estimation)} \\
\mathbf{\Gamma}_{k|k} & = & (\mathbf{I} - \mathbf{K}_k \mathbf{C}_k)\, \mathbf{\Gamma}_{k|k-1} & \text{(corrected covariance)} \\
\widetilde{\mathbf{z}}_k & = & \underbrace{\left(\mathbf{y}_k - \mathbf{g}(\hat{\mathbf{x}}_{k|k-1}) + \mathbf{C}_k \cdot \hat{\mathbf{x}}_{k|k-1}\right)}_{\mathbf{z}_k} - \mathbf{C}_k \hat{\mathbf{x}}_{k|k-1} = \mathbf{y}_k - \mathbf{g}(\hat{\mathbf{x}}_{k|k-1}) & \text{(innovation)} \\
\mathbf{S}_k & = & \mathbf{C}_k \mathbf{\Gamma}_{k|k-1} \mathbf{C}_k^{\mathrm{T}} + \mathbf{\Gamma}_{\boldsymbol{\beta}_k} & \text{(covariance of the innovation)} \\
\mathbf{K}_k & = & \mathbf{\Gamma}_{k|k-1} \mathbf{C}_k^{\mathrm{T}} \mathbf{S}_k^{-1} & \text{(Kalman gain)}
\end{array}
$$

The extended Kalman filter takes as inputs $\hat{\mathbf{x}}_{k|k-1}$ , $\mathbf{\Gamma}_{k|k-1}$, $\mathbf{y}_k$, $\mathbf{u}_k$ and returns $\hat{\mathbf{x}}_{k+1|k}$ and $\mathbf{\Gamma}_{k+1|k}$.

# Exercises

Exercise 7.1.– *Gaussian distribution*

The probability distribution of a random Gaussian vector $\mathbf{x}$ is fully characterized by its expectation $\bar{\mathbf{x}}$ and its covariance matrix $\mathbf{\Gamma}_{\mathbf{x}}$. More precisely, it is given by:

$$\pi_{\mathbf{x}}(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^n \det(\mathbf{\Gamma}_{\mathbf{x}})}} \cdot \exp\left(-\frac{1}{2}(\mathbf{x}-\bar{\mathbf{x}})^{\mathrm{T}} \cdot \mathbf{\Gamma}_{\mathbf{x}}^{-1} \cdot (\mathbf{x}-\bar{\mathbf{x}})\right).$$

1) Draw the graph and the contour lines of $\pi_{\mathbf{x}}$ with:

$$\bar{\mathbf{x}} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \text{ and } \mathbf{\Gamma}_{\mathbf{x}} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

2) We define the random vector:

$$\mathbf{y} = \begin{pmatrix} \cos\frac{\pi}{6} & -\sin\frac{\pi}{6} \\ \sin\frac{\pi}{6} & \cos\frac{\pi}{6} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 3 \end{pmatrix} \mathbf{x} + \begin{pmatrix} 2 \\ -5 \end{pmatrix}.$$

Draw the graph and the contour lines of $\pi_{\mathbf{y}}$. Discuss.

---

Exercise 7.2.– *Confidence ellipses*

Let us generate six covariance matrices in Matlab as follows:

```
A1=[1 0;0 3]; A2=[cos(pi/4) -sin(pi/4);sin(pi/4) cos(pi/4)];
G1=eye(2,2); G2=3*eye(2,2); G3=A1*G2*A1'+G1; G4=A2*G3*A2'; G5=G4+G3; G6=A2*G5*A2';
```

Here, `A2` corresponds to a rotation matrix of angle $\frac{\pi}{4}$. Then, we draw the six confidence ellipses at 90 % associated with these matrices by centering them around 0. We thus obtain Figure 7.10.

1) Associate each covariance matrix with its confidence ellipse on the figure.

2) Verify the result by regenerating these ellipses in Matlab.

---

Exercise 7.3.– *Confidence ellipse: prediction*

1) Generate a cloud of $n = 1\,000$ points in Matlab representing a random Gaussian vector centered at $\mathbb{R}^2$ whose covariance matrix is the identity matrix. Deduce from the latter a cloud of

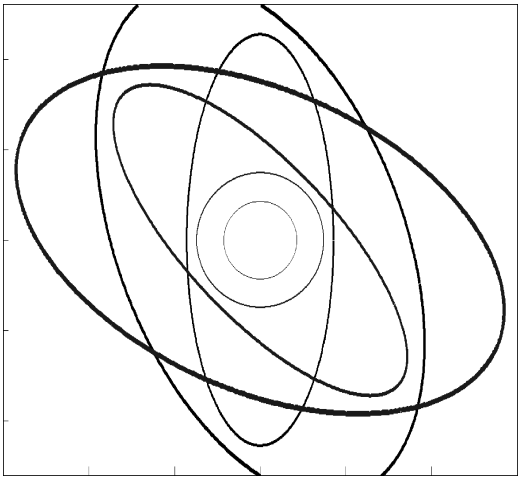

Figure 7.10: Confidence ellipses associated with the six covariance matrices

points for the random vector $\mathbf{x}$ such that:

$$\bar{\mathbf{x}} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \text{ and } \mathbf{\Gamma}_{\mathbf{x}} = \begin{pmatrix} 3 & 1 \\ 1 & 3 \end{pmatrix}$$

Use a two-line, $n$-column matrix to store the clouds.

2) Draw the confidence ellipses for the probabilities $\eta \in \{0.9, 0.99, 0.999\}$.

3) Find an estimation of $\bar{\mathbf{x}}$ and $\mathbf{\Gamma}_{\mathbf{x}}$ from the cloud of $\mathbf{x}$.

4) This distribution represents the knowledge we have of the initial conditions of a system (a robot, for instance) described by state equations of the form:

$$\dot{\mathbf{x}} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \mathbf{x} + \begin{pmatrix} 2 \\ 3 \end{pmatrix} u$$

where the input $u(t) = \sin t$ is known. Write a program that illustrates the evolution of this particle cloud with time. Use a sampling period of $\delta = 0.01\,\text{sec}$.

5) Represent this evolution using only the confidence ellipses.

---

EXERCISE 7.4.– *Confidence ellipse: correction*

1) As in the previous exercise, generate a Gaussian point cloud of $n = 1\ 000$ points associated with the random vector $\mathbf{x}$ with:

$$\bar{\mathbf{x}} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \text{ and } \mathbf{\Gamma}_{\mathbf{x}} = \begin{pmatrix} 3 & 1 \\ 1 & 3 \end{pmatrix}$$

Use a two-line, $n$-column matrix to store the clouds. Verify the coherence of the random generator by comparing it to MATLAB's `mvnrnd` generator (where *mvn* refers to *multivariate normal distribution*).

2) Find an unbiased and orthogonal linear estimator which allows to find $x_1$ from $x_2$. Draw this

estimator.

3) Same question as above, but one that allows to find $x_2$ from $x_1$. Draw the estimator and discuss the difference with the previous question.

---

EXERCISE 7.5.– *Covariance matrix propagation*

Consider three centered random vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$ with covariance matrices equal to the identity matrix. These three vectors are independent of each other. Let $\mathbf{x}, \mathbf{y}$ be two random vectors defined as follows:

$$\begin{aligned} \mathbf{x} &= \mathbf{A} \cdot \mathbf{a} - \mathbf{b} \\ \mathbf{y} &= \mathbf{C} \cdot \mathbf{x} + \mathbf{c} \end{aligned}$$

where $\mathbf{A}, \mathbf{C}$ are matrices that are known.

1) Give the expression of the mathematical expectations $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ and of the covariance matrices $\mathbf{\Gamma_x}, \mathbf{\Gamma_y}$ of these two vectors, in function of $\mathbf{A}$ and $\mathbf{C}$.

2) We form the vector $\mathbf{v} = (\mathbf{x}\ ,\ \mathbf{y})$. Calculate the mathematical expectation $\bar{\mathbf{v}}$ and the covariance matrix $\mathbf{\Gamma_v}$ for $\mathbf{v}$.

3) Deduce from the previous question the covariance matrix of the random vector $\mathbf{z} = \mathbf{y} - \mathbf{x}$. We assume of course that $\mathbf{x}$ and $\mathbf{y}$ are of same dimension.

4) We measure $\mathbf{y}$, which means that now the random vector $\mathbf{y}$ becomes deterministic and is well-known. Give an estimation $\hat{\mathbf{x}}$ for $\mathbf{x}$ using an unbiased and orthogonal linear estimator.

---

EXERCISE 7.6.– *Brownian noise*

We consider a random stationary, discretized, white and centered random signal. This signal is denoted by $x(t_k)$ with $k \in \mathbb{N}$. More precisely, for every $t_k = k\delta$ the random variables $x(t_k)$ with variance $\sigma_x^2$ are independent of one other. A Brownian noise is defined as the integral of a white noise. In our case, we form the Brownian noise as follows:

$$y(t_k) = \delta \cdot \sum_{i=0}^{k} x(t_k).$$

1) Calculate, in function of time, the variance $\sigma_y^2(t_k)$ of the signal $y(t_k)$. How does the standard deviation $\sigma_y(t_k)$ evolve in function of $\delta$ and in function of $t_k$ ? Discuss. Validate the result with a MATLAB simulation.

2) We now tend $\delta$ towards 0. What standard deviation $\sigma_x$ do we have to choose in function of $\delta$ in order for the variances $\sigma_y^2(t_k)$ to remain unchanged ? Illustrate this with a MATLAB program that generates Brownian noises $y(t)$ that are insensitive to sampling period changes.

---

EXERCISE 7.7.– *Solving three equations using a linear estimator*

The linear estimator can be used to solve problems that can be translated as linear equations.

Let us consider as an illustration the system:

$$\begin{cases} 2x_1 + 3x_2 &= 8 \\ 3x_1 + 2x_2 &= 7 \\ x_1 - x_2 &= 0 \end{cases}$$

Since we have more equations than unknowns, the linear estimator must find some sort of compromise between all of these equations. Let us assume that the errors $\varepsilon_i$ over the $i^{th}$ equation are centered and with variances : $\sigma_1^2 = 1$, $\sigma_2^2 = 4$ and $\sigma_3^2 = 4$. Solve the system by using a linear estimator and find the associated covariance matrix.

---

EXERCISE 7.8.– *Estimating the parameters of an electric motor using a linear estimator*

Let us consider a DC motor whose parameters have been estimated with a least squares method (see Exercise 6.3). Recall that in that example, the angular speed $\Omega$ of a DC motor verifies the relation:

$$\Omega = x_1 U + x_2 T_r$$

where $U$ is the input voltage, $T_r$ is the resistive torque and $\mathbf{x} = (x_1, x_2)$ is the vector of parameters that we need to estimate. The following table recalls the measurements made on the motor for various experimental conditions:

| $U$(V) | 4 | 10 | 10 | 13 | 15 |
|---|---|---|---|---|---|
| $T_r$(Nm) | 0 | 1 | 5 | 5 | 3 |
| $\Omega$(rad/ sec) | 5 | 10 | 8 | 14 | 17 |

We assume that the variance of the measurement error is equal to 9 and does not depend on the experimental conditions. Moreover, we assume that we know *a priori* that $x_1 \simeq 1$ and $x_2 \simeq -1$ with a variance of 4. Estimate the parameters of the motor and find the associated covariance matrix.

---

EXERCISE 7.9.– *Trochoid*

1) A point mass (placed on a wheel) is moving following a trochoid of the form:

$$\begin{cases} x(t) &= p_1 t - p_2 \sin t \\ y(t) &= p_1 - p_2 \cos t \end{cases}$$

where $x$ corresponds to the abscissa and $y$ to the altitude of the mass. We measure $y$ for various instants $t$:

| $t$(sec) | 1 | 2 | 3 | 7 |
|---|---|---|---|---|
| $y$(m) | 0.38 | 3.25 | 4.97 | $-0.26$ |

The measurement errors have a standard deviation of 10 cm. By using an unbiased orthogonal filter, calculate an estimation for $p_1$ and $p_2$.

2) Draw the estimated path of the mass in MATLAB.

---

EXERCISE 7.10.– *Solving three equations using a Kalman filter*

Let us consider once again the linear equations of Exercise 7.7:

$$\begin{cases} 2x_1 + 3x_2 &= 8 + \beta_1 \\ 3x_1 + 2x_2 &= 7 + \beta_2 \\ x_1 - x_2 &= 0 + \beta_3 \end{cases}$$

where $\beta_1, \beta_2, \beta_3$ are three independent, centered random variables with respective variances $1, 4, 4$.

1) Solve this system in MATLAB by calling the Kalman filter three times. Give an estimation of the solution and find the covariance matrix of the error.

2) Draw the confidence ellipses associated with each call.

3) Compare these with the results obtained for Exercise 7.7.

---

EXERCISE 7.11.– *Three-step Kalman filter*

Let us consider the discrete-time system:

$$\begin{cases} \mathbf{x}_{k+1} &= \mathbf{A}_k \mathbf{x}_k + \mathbf{u}_k + \boldsymbol{\alpha}_k \\ y_k &= \mathbf{C}_k \mathbf{x}_k + \beta_k \end{cases}$$

with $k \in \{0, 1, 2\}$. The values for the quantities $\mathbf{A}_k, \mathbf{C}_k, \mathbf{u}_k, y_k$ are given by:

| $k$ | $\mathbf{A}_k$ | $\mathbf{u}_k$ | $\mathbf{C}_k$ | $y_k$ |
|---|---|---|---|---|
| 0 | $\begin{pmatrix} 0.5 & 0 \\ 0 & 1 \end{pmatrix}$ | $\begin{pmatrix} 8 \\ 16 \end{pmatrix}$ | $\begin{pmatrix} 1 & 1 \end{pmatrix}$ | 7 |
| 1 | $\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$ | $\begin{pmatrix} -6 \\ -18 \end{pmatrix}$ | $\begin{pmatrix} 1 & 1 \end{pmatrix}$ | 30 |
| 2 | $\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$ | $\begin{pmatrix} 32 \\ -8 \end{pmatrix}$ | $\begin{pmatrix} 1 & 1 \end{pmatrix}$ | $-6$ |

Let us assume that the signals $\boldsymbol{\alpha}_k$ and $\beta_k$ are white Gaussian signals with a unitary covariance matrix, in other words :

$$\boldsymbol{\Gamma}_{\boldsymbol{\alpha}} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \text{ and } \boldsymbol{\Gamma}_{\beta} = 1$$

The initial state vector is unknown and is represented by an estimation $\hat{\mathbf{x}}_{0|-1}$ and a covariance

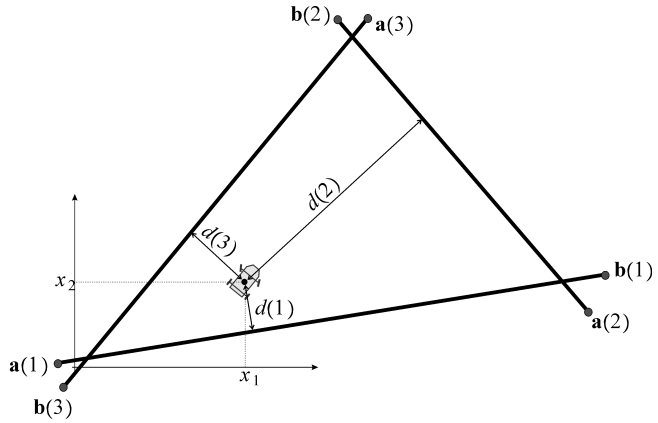


Figure 7.11: The robot must locate itself by measuring its distance to the three walls

matrix $\mathbf{\Gamma}_{0|-1}$. We will take:

$$\hat{\mathbf{x}}_{0|-1} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \ \mathbf{\Gamma}_{0|-1} = \begin{pmatrix} 100 & 0 \\ 0 & 100 \end{pmatrix}$$

Draw the confidence ellipses with center $\hat{\mathbf{x}}_{k|k}$ and covariance matrix $\mathbf{\Gamma}_{k|k}$ obtained by the Kalman filter, in MATLAB.

---

EXERCISE 7.12.– *Estimating the parameters of an electric motor*

Let us consider once more the DC motor with angular speed $\Omega$ (see Exercises 6.3 to 7.8). We have:

$$\Omega = x_1 U + x_2 T_r$$

where $U$ is the input voltage, $T_r$ is the resistive torque and $\mathbf{x} = (x_1, x_2)$ is the vector of parameters to estimate. The following table presents the measurements obtained for various experimental conditions:

| $k$ | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| $U$(V) | 4 | 10 | 10 | 13 | 15 |
| $T_r$(Nm) | 0 | 1 | 5 | 5 | 3 |
| $\Omega$(rad/ sec) | 5 | 10 | 11 | 14 | 17 |

We still assume that the variance of the measurement error is equal to 9 and that $x_1 \simeq 1$ and $x_2 \simeq -1$ with a variance of 4. Using the Kalman filter, calculate an estimation of the parameters $x_1, x_2$ and give the associated covariance matrix.

---

EXERCISE 7.13.– *Localization from wall distance measurements*

Consider a punctual robot positioned at $\mathbf{x} = (x_1, x_2)$. This robot measures its distance to the

three walls, as shown in Figure 7.11. The $i^{\text{th}}$ wall corresponds to a line defined by two points $\mathbf{a}(i)$ and $\mathbf{b}(i)$. The distance to the $i^{\text{th}}$ wall is:

$$d(i) = \det(\mathbf{u}(i), \mathbf{x} - \mathbf{a}(i)) + \beta_i$$

with $\mathbf{u}(i) = \frac{\mathbf{b}(i)-\mathbf{a}(i)}{\|\mathbf{b}(i)-\mathbf{a}(i)\|}$. Each distance is measured with a centered error $\beta_i$ with variance 1 and all the errors are independent of one other. Before taking any measurements, the robot assumes it is in position $\bar{\mathbf{x}} = (1, 2)$ with the associated covariance matrix given by $100 \cdot \mathbf{I}$ where $\mathbf{I}$ is the identity matrix.

1) Give, in function of the $\mathbf{a}(i), \mathbf{b}(i), d(i)$, an estimation of the robot's position as well as the covariance matrix for the error. For this you can use the expression of the unbiased orthogonal linear estimator or equivalently the expression of the Kalman filter in correction mode.

2) The coordinates of the points as well as the distances are given by:

| $i$ | 1 | 2 | 3 |
|---|---|---|---|
| $\mathbf{a}(i)$ | $\begin{pmatrix} 2 \\ 1 \end{pmatrix}$ | $\begin{pmatrix} 15 \\ 5 \end{pmatrix}$ | $\begin{pmatrix} 3 \\ 12 \end{pmatrix}$ |
| $\mathbf{b}(i)$ | $\begin{pmatrix} 15 \\ 5 \end{pmatrix}$ | $\begin{pmatrix} 3 \\ 12 \end{pmatrix}$ | $\begin{pmatrix} 2 \\ 1 \end{pmatrix}$ |
| $d(i)$ | 2 | 5 | 4 |

Write a MATLAB program that gives us the required estimation.

---

EXERCISE 7.14.– *Temperature estimation*

The temperature in a room has to verify (after temporal discretization) the state equation:

$$\begin{cases} x_{k+1} &= x_k + \alpha_k \\ y_k &= x_k + \beta_k \end{cases}$$

We assume that the state noise $\alpha_k$ and the measurement noise $\beta_k$ are independent and Gaussian with covariance $\Gamma_\alpha = 4$ and $\Gamma_\beta = 3$.

1) Give the expression of the Kalman filter that allows to estimate the temperature $x_k$ from the measurement $y_k$. From this deduce an expression of $\hat{x}_{k+1|k}$ and $\Gamma_{k+1|k}$ in function of $\hat{x}_{k|k-1}, \Gamma_{k|k-1}, y_k$.

2) For large enough $k$, we may assume that $\Gamma_{k+1|k} = \Gamma_{k|k-1} = \Gamma_\infty$. We then obtain the so-called *asymptotic* Kalman filter. Give the expression of the asymptotic Kalman filter. How would you characterize the precision of this filter ?

3) Going back to the non-asymptotic case, but now assuming that $\Gamma_{\alpha_k} = 0$, what is the value of $\Gamma_\infty$ ? Discuss.

---

EXERCISE 7.15.– *Blind walker*

We consider a blind walker moving on a horizontal line. Its movement is described by the

discretized state equation:

$$\left\{ \begin{array}{rcl} x_1(k+1) & = & x_1(k) + x_2(k) \cdot u(k) \\ x_2(k+1) & = & x_2(k) + \alpha_2(k) \end{array} \right.$$

where $x_1(k)$ is the position of the walker, $x_2(k)$ is the length of a step (referred to as *scale factor*) and $u(k)$ is the number of steps per time unit. We measure the quantity $u(k)$. Thus, at each unit of time, the walker moves a distance of $x_2(k)\,u(k)$. At the initial moment, we know that $x_1$ is zero and that $x_2$ is close to 1. $x_2(0)$ will be represented by a Gaussian distribution whose mean is equal to 1 and whose standard deviation is 0.02. The scale factor $x_2$ evolves slowly by means of $\alpha_2(k)$ that we will assume to be centered, white and of standard deviation 0.01.

1) We apply an input $u(k) = 1$ for $k = 0, \ldots, 9$ and $u(k) = -1$ for $k = 10, \ldots, 19$. Write a MATLAB program that implements a predictive Kalman filter capable of estimating the position $x_1(k)$.

2) Draw the confidence ellipses associated with the probability $\eta = 0.99$. How does the uncertainty evolve for $x_1$ in function of $k$ ?

3) In function of $k$, draw the determinant of the covariance matrix $\mathbf{\Gamma_x}$. Discuss.

---

EXERCISE 7.16.– *Simple pendulum*

Let us consider the pendulum in Figure 7.12. The input of this system is the torque $u$ exerted on the pendulum.

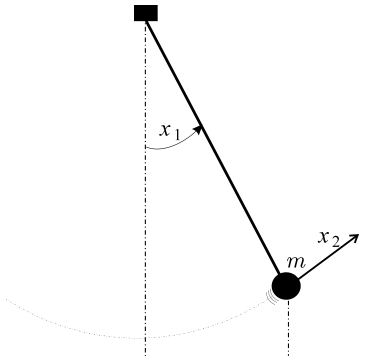


Figure 7.12: Simple pendulum with state vector $\mathbf{x} = (x_1, x_2)$

Its state representation is assumed to be:

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_2 \\ -\sin x_1 + u \end{pmatrix}.$$

This is of course a normalized model in which the coefficients (mass, gravity, length) have all been set to 1.

1) We would like the position of the pendulum $x_1(t)$ to be equal to a setpoint $w(t)$ that varies with time. Using a feedback linearization method, suggest a state feedback controller such that the

error $e = w - x_1$ converges towards 0 at $\exp(-t)$ (which means that we place the poles at $-1$). For this the expression of $u$ in function of $\mathbf{x}$, $w, \dot{w}, \ddot{w}$ must be written.

2) We would like the angle of the pendulum $x_1$ to be equal to $w(t) = \sin t$, once the transient regime has passed. What expression do we need to choose for the control $u(t)$ ? Give the expression of $u$ in function of $\mathbf{x}$ and of $t$. Provide a MATLAB simulation.

3) In order to implement the control proposed in the previous question, it is necessary for the complete state to be available. However, we only have a gyro placed on the axis of the pendulum that gives us a measurement of $x_2$ every $\delta = 0.01$ seconds, with a white Gaussian error of standard deviation 0.1rad/ sec. We must therefore reconstruct the state $\mathbf{x}$ of the pendulum to implement our control. Build an extended Kalman filter to realize the observer and illustrate the behavior of the corresponding controlled system with MATLAB.

---

EXERCISE 7.17.– *State estimation of the inverted rod pendulum*

We consider an inverted rod pendulum whose state equations are given by:

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} x_3 \\ x_4 \\ \dfrac{m \sin x_2 (g \cos x_2 - \ell x_4^2) + u}{M + m \sin^2 x_2} \\ \dfrac{\sin x_2 ((M+m)g - m\ell x_4^2 \cos x_2) + \cos x_2 u}{\ell (M + m \sin^2 x_2)} \end{pmatrix} \quad \text{and} \quad y = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

Here, we have taken as state vector $\mathbf{x} = \left(x, \theta, \dot{x}, \dot{\theta}\right)$, where the input $u$ is the force exerted on the cart of mass $M$, $x$ is the position of the cart and $\theta$ is the angle between the pendulum and the vertical direction. We will assume here that only the position of the cart $x$ and the angle $\theta$ of the pendulum are measured.

1) Linearize this system around the state $\mathbf{x} = \mathbf{0}$.

2) Suggest a state feedback controller of the form $u = -\mathbf{K} \cdot \mathbf{x} + h\ w$ that stabilizes the system. Use a pole placement method to achieve this (the `place` instruction in MATLAB). All the poles will be equal to $-2$. For the precompensator $h$, take a setpoint $w$ that corresponds to the desired position for the cart. Following [JAU 13], we must take:

$$h = -\left(\mathbf{E} \cdot (\mathbf{A} - \mathbf{B} \cdot \mathbf{K})^{-1} \cdot \mathbf{B}\right)^{-1}$$

where $\mathbf{E}$ is the setpoint matrix given by:

$$\mathbf{E} = \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}$$

Simulate the system controlled by this state feedback.

3) In order to preform output feedback, we need an estimation $\hat{\mathbf{x}}$ of the state vector $\mathbf{x}$. For this, we will use a Kalman filter (see Figure 7.13):

Discretize the system using steps of $dt = 0.01\,\text{sec}$, then propose a Kalman filter for observing the state.

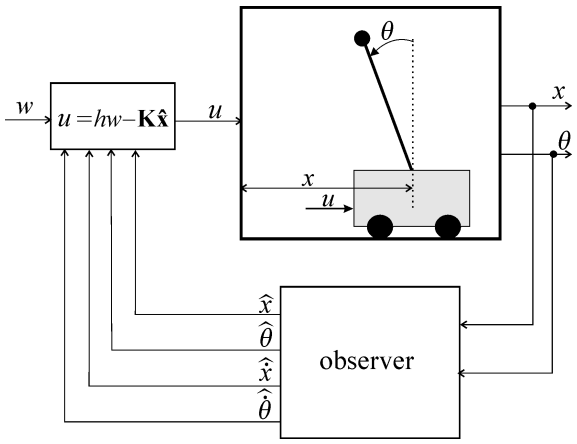


Figure 7.13: Kalman filter used to estimate the state of the inverted rod pendulum

4) Implement this filter in MATLAB. We will take a centered Gaussian white noise for the state with variance $\sigma_x^2 = dt \cdot 0.01$ for the variables $x, \theta$. For the measurement noise, we will take a centered Gaussian white noise with variance $\sigma_y^2 = 0.01^2$. Study the robustness of the observer when the measurement noise is increased. On a separate figure, draw the confidence ellipses in the space $(x, \theta)$ and verify whether their true value $(x^*, \theta^*)$ is within the ellipse.

5) An extended Kalman filter can be obtained by replacing, in the prediction step of the Kalman filter, the instruction:

$$\hat{\mathbf{x}}_{k+1|k} = \mathbf{A}_k \hat{\mathbf{x}}_{k|k} + \mathbf{B}_k \mathbf{u}_k$$

by:

$$\hat{\mathbf{x}}_{k+1|k} = \hat{\mathbf{x}}_{k|k} + \mathbf{f}\left(\hat{\mathbf{x}}_{k|k}, \mathbf{u}_k\right) \cdot dt$$

Here we have replaced the prediction performed on the linearized model by a prediction performed on the initial nonlinear model that is closer to reality. Propose an implementation of this extended Kalman filter.

---

EXERCISE 7.18.– *Dead reckoning*

*Dead reckoning* corresponds to the problem of localization in which only proprioceptive sensors are available. This type of navigation was used by early navigators who were trying to locate themselves during long journeys. They were able to do this in a very approximate way by measuring the heading of the boat, the speed at various instants and integrating all the corresponding variations in position over the entire journey. In a more general context, we may consider that using a state observer in prediction mode and without correction (in the particular case in which the state is the position of the robot) corresponds to dead reckoning. Let us consider the robot represented on Figure 7.14 and

whose state equations are:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \\ \dot{\delta} \end{pmatrix} = \begin{pmatrix} v\cos\delta\cos\theta \\ v\cos\delta\sin\theta \\ \frac{v\sin\delta}{3} + \alpha_\theta \\ u_1 + \alpha_v \\ u_2 + \alpha_\delta \end{pmatrix}$$

where $\alpha_\theta, \alpha_v, \alpha_\delta$ are independent continuous-time Gaussian white noises. In a more rigorous way, these are random distributions with infinite power, but once they are discretized, the mathematical difficulties disappear.

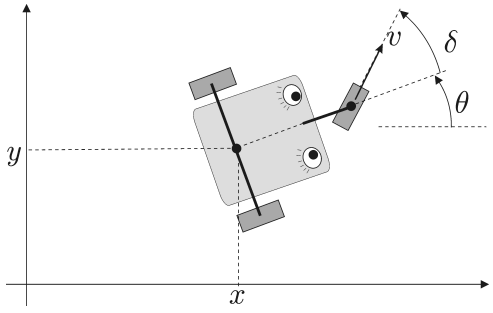


Figure 7.14: Dead reckoning for a tricycle robot

The robot is equipped with a compass that returns $\theta$ with high precision and an angle sensor that returns the angle $\delta$ of the front wheel.

1) Discretize this system with an Euler method. Simulate this system in MATLAB for an arbitrary input $\mathbf{u}(t)$ and initial vector. For the variance of the discretized noises $\alpha_\theta, \alpha_v, \alpha_\delta$ we will take $0.01 \cdot dt$, where $dt$ is the discretization step.

2) Express this localization problem in a linear and discretized form.

3) Using a Kalman filter, predict the position of the robot as well as the associated covariance matrix.

4) How does the localization program change if we assume that, using odometers, the robot is capable of measuring its speed $v$ with a variance of 0.01 ?

---

EXERCISE 7.19.– *Goniometric localization*

Let us consider once again a robot vehicle described by the state equations:

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \end{pmatrix} = \begin{pmatrix} x_4\cos x_5\cos x_3 \\ x_4\cos x_5\sin x_3 \\ \frac{x_4\sin x_5}{3} \\ u_1 \\ u_2 \end{pmatrix}$$

The vector $(x_1, x_2)$ represents the coordinates of the center of the robot, $x_3$ is the heading of the

robot, $x_4$ its speed and $x_5$ the angle of its front wheels. The robot is surrounded by point landmarks $\mathbf{m}(1), \mathbf{m}(2), \ldots$ whose positions are known. The robot can only detect these landmarks $\mathbf{m}(i)$ if the distance to them is sufficiently small (smaller than 15 m). In such a case, the robot measures the angle $\delta_i$ with high precision. We will also assume that the robot knows the angles $x_3$ and $x_5$ at all times, without any error. Finally, it measures its speed $x_4$ with an error of variance 1. Figure 7.15 illustrates a situation in which two landmarks $\mathbf{m}(1)$ and $\mathbf{m}(2)$ are detected by the robot.

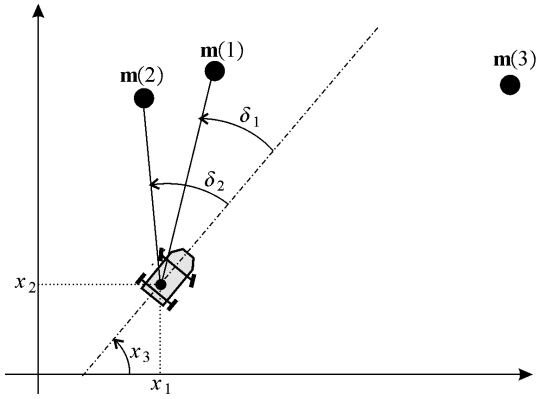


Figure 7.15: Goniometric localization

In order for the robot to locate itself, we would like to use a Kalman filter. For this, we need linear equations, which we do not have here. Since $x_3$ and $x_5$ are known, the nonlinearity can be based on a temporal dependency. Let us take for this $\mathbf{z} = (x_1, x_2, x_4)$.

1) Show that $\mathbf{z}$ satisfies a linear state evolution equation. Find the associated observation equation.

2) Find a discretization for the evolution of $\mathbf{z}$ in order to feed a Kalman filter.

3) Implement a simulator in MATLAB with the robot surrounded by the following four landmarks:

$$\mathbf{a}(1) = \begin{pmatrix} 0 \\ 25 \end{pmatrix}, \ \mathbf{a}(2) = \begin{pmatrix} 15 \\ 30 \end{pmatrix}, \ \mathbf{a}(3) = \begin{pmatrix} 30 \\ 15 \end{pmatrix}, \ \mathbf{a}(4) = \begin{pmatrix} 15 \\ 20 \end{pmatrix}$$

As stated above, the robot can only goniometrically detect landmarks once they are close.

4) Implement a Kalman filter for the localization. The initial state will be assumed unknown.

5) We now have two robots $\mathcal{R}_a$ and $\mathcal{R}_b$ capable of communicating wirelessly while measuring the landmark angles (see Figure 7.16). When the distances are small (i.e. smaller than 20 m), the robots can measure the angles $\varphi_a$ and $\varphi_b$ with high precision using cameras (see figure). Suggest a centralized Kalman filter for the localization of the two robots.

---

EXERCISE 7.20.– *Following a boat with two radars*

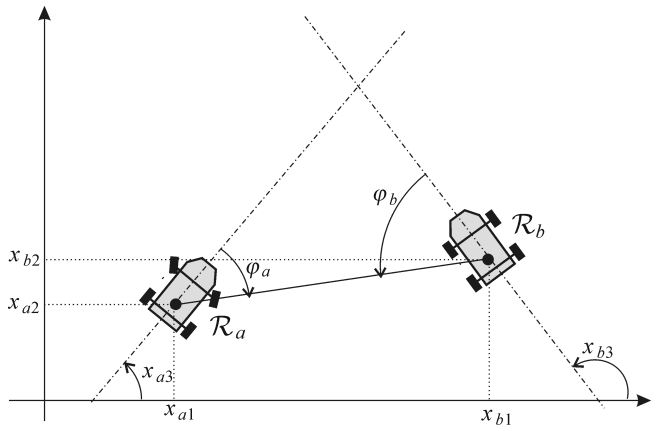


Figure 7.16: Goniometric localization for two communicating robots

The movement of a boat that we are seeking to follow is described by the state equations:

$$\begin{cases} p_x(k+1) &= p_x(k) + dt \cdot v_x(k) \\ v_x(k+1) &= v_x(k) - dt \cdot v_x(k) + \alpha_x(k) \\ p_y(k+1) &= p_y(k) + dt \cdot v_y(k) \\ v_y(k+1) &= v_y(k) - dt \cdot v_y(k) + \alpha_y(k) \end{cases}$$

where $dt = 0.01$ and $\alpha_x$ and $\alpha_y$ are Gaussian white noises with variance matrix $dt$. The state vector is therefore $\mathbf{x} = (p_x, v_x, p_y, v_y)$.

1) Write a MATLAB program that simulates the boat.

2) Two radars placed at $\mathbf{a} : (a_x, a_y) = (0, 0)$ and $\mathbf{b} : (b_x, b_y) = (1, 0)$ measure the square of the distance to the boat. The observation equation is:

$$\mathbf{y}_k = \underbrace{\begin{pmatrix} (p_x(k) - a_x)^2 + (p_y(k) - a_y)^2 \\ (p_x(k) - b_x)^2 + (p_y(k) - b_y)^2 \end{pmatrix}}_{\mathbf{g}(\mathbf{x}_k)} + \boldsymbol{\beta}_k$$

where $\beta_1(k)$ and $\beta_2(k)$ are independent unit Gaussian white noises. Adjust the simulation in order to visualize the radars and generate the measurement vector $\mathbf{y}(k)$.

3) Linearize this observation equation around the current estimation $\hat{\mathbf{x}}_k$ of the state vector $\mathbf{x}_k$. Deduce from this an equation of the form $\mathbf{z}_k = \mathbf{C}_k \mathbf{x}_k$ where $\mathbf{z}_k = \mathbf{h}(\mathbf{y}_k, \hat{\mathbf{x}}_k)$ takes the role of the measurement taken at time $k$.

4) Implement a Kalman filter that allows the localization of the boat.

---

EXERCISE 7.21.– *Robot localization in a pool*

Consider an underwater robot moving within a rectangular pool of length $2R_y$ and width $2R_y$. A sonar placed just above the robot rotates with a constant angular speed. The depth $z$ is easily obtained using a pressure sensor and we will therefore assume this quantity to be known. The robot is weighted in such a way that the bank and elevation angles may be assumed zero. In our context,

localizing the robot means estimating the coordinates $(x, y)$ of the robot. The origin of the coordinate system will be middle of the pool. For this localization, we will assume that the angle $\alpha$ of the sonar is measured relative to the body of the robot, the heading angle $\theta$ is measured with a compass and the tangential $a_{\mathrm{T}}$ and normal $a_{\mathrm{N}}$ accelerations with accelerometers. Every 0.1s, the sonar returns the length $\ell$ of the sonar beam. Figure 7.17 represents the length $\ell(t)$ of the sonar beam, obtained by simulation when the sonar performs seven rotations around itself while the robot is moving.

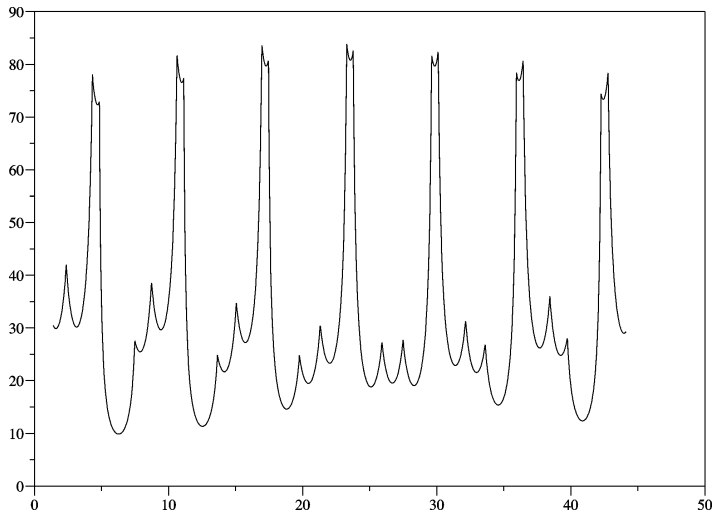


Figure 7.17: Telemetric measurements collected by the robot

1) Given the signal collected by the sonar, suggest a robust method for detecting local minima that correspond to the situation where the sonar is perpendicularly pointing towards one of the four walls of the pool.

2) Let $v_x = \dot{x}$, $v_y = \dot{y}$. Show that we can write the following state equations to connect the measurements:

$$\begin{cases} \dot{x} &= v_x \\ \dot{y} &= v_y \\ \dot{v}_x &= a_{\mathrm{T}} \cos\theta - a_{\mathrm{N}} \sin\theta \\ \dot{v}_y &= a_{\mathrm{T}} \sin\theta + a_{\mathrm{N}} \cos\theta \end{cases}$$

3) Let us now assume that we have collected (either by simulation or real experience) for each $t \in [0, t_{\max}]$, data relative to the accelerations $(a_{\mathrm{T}}, a_{\mathrm{N}})$, to the heading angle $\theta$ and to the angle of the sonar $\alpha$. Propose a recursive method based on the Kalman filter in order to localize the robot.

# Chapter 8

# Bayes filter

## 8.1 Introduction

This chapter proposes to generalize the Kalman filter to the case where the functions are nonlinear and the noise is non Gaussian. The resulting observer will be called the *Bayes filter*. Instead of computing for each time $k$ the covariance and the estimate of the state, the Bayes filter directly computes the probability density function of the state vector. As for the Kalman filter, it consists of two parts: the prediction and the correction. In the linear and Gaussian case, the Bayes filter is equivalent to the Kalman filter.

By increasing the level of abstraction, the Bayes filter will allow us to have a better understanding of the Kalman filter, and some proofs become easier and more intuitive. As an illustration we will consider the smoothing problem where the estimation is made more accurate by taking all future measurements, when available. Of course, the smoothing is mainly used for offline applications.

## 8.2 Basic notions on probabilities

**Marginal density**. If $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{y} \in \mathbb{R}^m$ are two random vectors with a joint probability density function $\pi(\mathbf{x}, \mathbf{y})$. Note that $\pi(\mathbf{x}, \mathbf{y})$ is a function which associates to $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) \in \mathbb{R}^n \times \mathbb{R}^m$ an element of $\mathbb{R}^+$ denoted by $\pi(\mathbf{x} = \tilde{\mathbf{x}}, \mathbf{y} = \tilde{\mathbf{y}})$. The *marginal density* for $\mathbf{x}$ is

$$\pi(\mathbf{x}) = \int \pi(\mathbf{x}, \mathbf{y}) \cdot d\mathbf{y}. \tag{8.1}$$

Note that, to be rigorous, we should have written

$$\pi(\mathbf{x} = \tilde{\mathbf{x}}) = \int_{\tilde{\mathbf{y}} \in \mathbb{R}^m} \pi(\mathbf{x} = \tilde{\mathbf{x}}, \mathbf{y} = \tilde{\mathbf{y}}) \cdot d\tilde{\mathbf{y}},$$

but this notation would become too heavy for our applications. In the same manner, the marginal density for $\mathbf{y}$ is

$$\pi(\mathbf{y}) = \int \pi(\mathbf{x}, \mathbf{y}) \cdot d\mathbf{x}.$$

The two random vectors $\mathbf{x}$ and $\mathbf{y}$ are *independent* if

$$\pi(\mathbf{x}, \mathbf{y}) = \pi(\mathbf{x}) \cdot \pi(\mathbf{y}).$$

**Conditional density**. The *conditional density* for $\mathbf{x}$ given $\mathbf{y}$ is

$$\pi(\mathbf{x}|\mathbf{y}) = \frac{\pi(\mathbf{x}, \mathbf{y})}{\pi(\mathbf{y})}. \tag{8.2}$$

Again, the quantity $\pi(\mathbf{x}|\mathbf{y})$ is a function which associates to the pair $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) \in \mathbb{R}^n \times \mathbb{R}^m$ a positive real number. But, $\mathbf{y}$ a not the same role: it is a parameter of the density for $\mathbf{x}$. We also have

$$\pi(\mathbf{y}|\mathbf{x}) = \frac{\pi(\mathbf{x}, \mathbf{y})}{\pi(\mathbf{x})}. \tag{8.3}$$

**Bayes rule**. Combining the two equations (8.2) and (8.3), we get

$$\pi(\mathbf{x}, \mathbf{y}) = \pi(\mathbf{y}|\mathbf{x}) \cdot \pi(\mathbf{x}) = \pi(\mathbf{x}|\mathbf{y}) \cdot \pi(\mathbf{y}).$$

The *Bayes rule* obtained from the previous equation:

$$\pi(\mathbf{x}|\mathbf{y}) = \frac{\pi(\mathbf{y}|\mathbf{x}) \cdot \pi(\mathbf{x})}{\pi(\mathbf{y})} = \eta \cdot \pi(\mathbf{y}|\mathbf{x}) \cdot \pi(\mathbf{x}). \tag{8.4}$$

The quantity $\eta = \frac{1}{\pi(\mathbf{y})}$ called the *normaliser* allows to have an integral for $\mathbf{x}$ equal to one. Sometimes, we use the following notation

$$\pi(\mathbf{x}|\mathbf{y}) \propto \pi(\mathbf{y}|\mathbf{x}) \cdot \pi(\mathbf{x})$$

to indicate that the two functions $\pi(\mathbf{x}|\mathbf{y})$ and $\pi(\mathbf{y}|\mathbf{x}) \cdot \pi(\mathbf{x})$ are proportional for a given $\mathbf{y}$.

**Total probability law**. The marginal density for $\mathbf{x}$ is

$$\pi(\mathbf{x}) \overset{(8.1,8.2)}{=} \int \pi(\mathbf{x}|\mathbf{y}) \cdot \pi(\mathbf{y}) \cdot d\mathbf{y}. \tag{8.5}$$

This corresponds to the *law of total probability*.

**Parametric case**. If $\mathbf{z}$ is a parameter (which can be random vector and any other deterministic quantity), the parametric total probability law is given by

$$\pi(\mathbf{x}|\mathbf{z}) \overset{(8.5)}{=} \int \pi(\mathbf{x}|\mathbf{y}, \mathbf{z}) \cdot \pi(\mathbf{y}|\mathbf{z}) \cdot d\mathbf{y} \tag{8.6}$$

and the parametric Bayes rule is

$$\pi(\mathbf{x}|\mathbf{y}, \mathbf{z}) \overset{(8.4)}{=} \frac{\pi(\mathbf{y}|\mathbf{x}, \mathbf{z}) \cdot \pi(\mathbf{x}|\mathbf{z})}{\pi(\mathbf{y}|\mathbf{z})}. \tag{8.7}$$

**Bayes network**. A Bayes network is a probabilistic graphical model that represents a set of random vectors and their conditional dependencies. Formally, Bayes networks are directed acyclic

graph whose nodes represent random vectors. Arcs represent dependencies. More precisely the two vectors $\mathbf{x}, \mathbf{y}$ are connected by an arc there exists a relation linking them. Nodes that are not connected (there is no path from one of the variables to the other in the Bayes network) represent variables that are independent. To have a better understanding, consider 5 vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{e}$ that are linked by the relations

$$\begin{aligned} \mathbf{b} &= \mathbf{a} + \boldsymbol{\alpha}_1 \\ \mathbf{e} &= \mathbf{a} + 2\mathbf{b} + \boldsymbol{\alpha}_2 \\ \mathbf{c} &= \mathbf{x} + \boldsymbol{\alpha}_3 \\ \mathbf{d} &= 3\mathbf{c} + \boldsymbol{\alpha}_4 \end{aligned}$$

where the $\boldsymbol{\alpha}_i$ are all independent vector with a known density for instance $\mathcal{N}(0,1)$, which mean Gaussian centred with a unit variance. It is clear that if we know a density $\pi(\mathbf{a})$ for $\mathbf{a}$, we can compute the probabilities for all other vectors. Equivalently, we can easily write a program which generates the generates some realizations for all the vectors. For instance in the scalar case, if $\pi(a) = N(1, 9)$, the program could be :

```
a=1+3*randn();b=a+randn();e=a+2*b+randn();c=e+randn();d=3*c+randn();
```

From this program, we easily see that $\pi(\mathbf{c}|\mathbf{a}, \mathbf{e}, \mathbf{b}) = \pi(\mathbf{c}|\mathbf{e})$, which means that given $e$, the vector $\mathbf{c}$ is independent of $\mathbf{a}$ and $\mathbf{b}$. The corresponding network is depicted on Figure 8.1 (left). The network will help us to simplify conditional densities. For instance, from the network we can conclude that $\pi(\mathbf{c}|\mathbf{a}, \mathbf{e}, \mathbf{b}) = \pi(\mathbf{c}|\mathbf{e})$, since all paths from $\mathbf{a}$ to $\mathbf{c}$ and all paths from $\mathbf{b}$ to $\mathbf{c}$ have to go through $\mathbf{e}$. Equivalently, this means that if we know $e$ the knowledge of $\mathbf{a}$ and $\mathbf{b}$ does not bring any new information on $\mathbf{c}$.

Consider 3 vectors $\mathbf{x}, \mathbf{y}, \mathbf{z}$ that are linked by the relations

$$\begin{aligned} \mathbf{y} &= 2\mathbf{x} + \boldsymbol{\alpha}_1 \\ \mathbf{z} &= 3\mathbf{y} + \boldsymbol{\alpha}_2 \\ \mathbf{x} &= 4\mathbf{z} + \boldsymbol{\alpha}_3 \end{aligned}$$

where the $\boldsymbol{\alpha}_i$ are all independent vectors with density $\mathcal{N}(0,1)$. The graph (see Figure 8.1 (right) has a cycle and is not considered a a Bayes network anymore. It is now much more difficult (or even impossible) to compute the marginal densities or to build a program that generates a cloud of points consistent with the probabilistic assumptions.

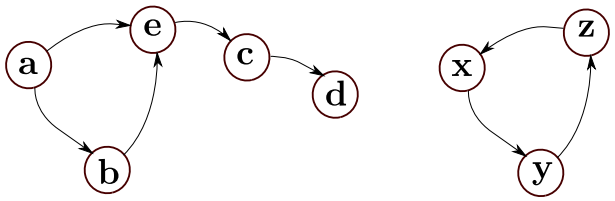


Figure 8.1: Left: a Bayes network; Right, the graph has a cycle and thus is not a Bayes network

## 8.3 Bayes filter

Consider a system described by the following state equations

$$\left\{ \begin{array}{rcl} \mathbf{x}_{k+1} & = & \mathbf{f}_k(\mathbf{x}_k) + \boldsymbol{\alpha}_k \\ \mathbf{y}_k & = & \mathbf{g}_k(\mathbf{x}_k) + \boldsymbol{\beta}_k \end{array} \right. \tag{8.8}$$

where $\boldsymbol{\alpha}_k$ and $\boldsymbol{\beta}_k$ are random vectors white and mutually independent. The dependency with respect to some known inputs $\mathbf{u}_k$ is taken into account via the functions $\mathbf{f}_k$ and $\mathbf{g}_k$. We have here a random process which satisfies the Markov assumptions which tells us that the future of the system only depends of the past through the state at the current time $t$. This can be written as:

$$\begin{array}{rcl} \pi(\mathbf{y}_k|\mathbf{x}_k, \mathbf{y}_{0:k-1}) & = & \pi(\mathbf{y}_k|\mathbf{x}_k) \\ \pi(\mathbf{x}_{k+1}|\mathbf{x}_k, \mathbf{y}_{0:k}) & = & \pi(\mathbf{x}_{k+1}|\mathbf{x}_k). \end{array} \tag{8.9}$$

The notation $\mathbf{y}_{0:k}$ has to be understood as follows

$$\mathbf{y}_{0:k} = (\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_k)\,.$$

This is illustrated by the Bayes network of Figure 8.2. An arc of this graph corresponds to a dependency. For instance, the arc between $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$ corresponds to the knowledge that $\mathbf{x}_{k+1} - \mathbf{f}_k(\mathbf{x}_k)$ has a density which corresponds to that of $\boldsymbol{\alpha}_k$.

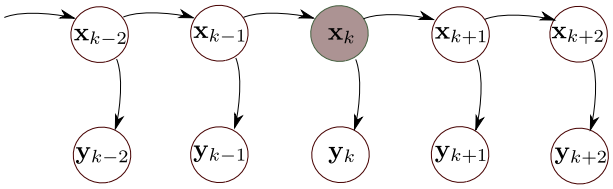


Figure 8.2: Bayes network associated with the state equation

**Theorem**. The two densities

$$\begin{array}{rcl} \mathrm{pred}\,(\mathbf{x}_k) & \stackrel{\mathrm{def}}{=} & \pi\,(\mathbf{x}_k|\mathbf{y}_{0:k-1}) \\ \mathrm{bel}\,(\mathbf{x}_k) & \stackrel{\mathrm{def}}{=} & \pi\,(\mathbf{x}_k|\mathbf{y}_{0:k})\,. \end{array} \tag{8.10}$$

satisfie

$$\begin{array}{lrcll} \text{(i)} & \mathrm{pred}\,(\mathbf{x}_{k+1}) & = & \int \pi\,(\mathbf{x}_{k+1}|\mathbf{x}_k) \cdot \mathrm{bel}\,(\mathbf{x}_k) \cdot d\mathbf{x}_k & \text{(prediction)} \\ \text{(ii)} & \mathrm{bel}\,(\mathbf{x}_k) & = & \frac{\pi(\mathbf{y}_k|\mathbf{x}_k) \cdot \mathrm{pred}(\mathbf{x}_k)}{\pi(\mathbf{y}_k|\mathbf{y}_{0:k-1})}. & \text{(correction)} \end{array} \tag{8.11}$$

**Remark**. These relations correspond to a *Bayes observer* or *Bayesian filter*. The prediction equation is known as the equation of *Chapman-Kolmogorov*.

**Proof** : Let us prove relation (ii). We have

$$\begin{array}{lll} \text{bel}\,(\mathbf{x}_k) & \overset{(8.10)}{=} & \pi\,(\mathbf{x}_k|\mathbf{y}_{0:k}) \\ & = & \pi\,(\mathbf{x}_k|\mathbf{y}_k, \mathbf{y}_{0:k-1}) \\ & \overset{(8.7)}{=} & \frac{1}{\pi(\mathbf{y}_k|\mathbf{y}_{0:k-1})} \cdot \underbrace{\pi(\mathbf{y}_k|\mathbf{x}_k, \mathbf{y}_{0:k-1})}_{\overset{(8.9)}{=}\pi(\mathbf{y}_k|\mathbf{x}_k)} \cdot \underbrace{\pi(\mathbf{x}_k|\mathbf{y}_{0:k-1})}_{\overset{(8.10)}{=}\text{pred}(\mathbf{x}_k)} \end{array} \qquad \left\{\pi(\mathbf{x}|\mathbf{y},\mathbf{z}) = \frac{\pi(\mathbf{y}|\mathbf{x},\mathbf{z})\cdot\pi(\mathbf{x}|\mathbf{z})}{\pi(\mathbf{y}|\mathbf{z})}\right\}$$

Let us now prove (i). From the total probability rule (8.6), we get

$$\begin{array}{lll} \text{pred}\,(\mathbf{x}_{k+1}) & \overset{(8.10)}{=} & \pi\,(\mathbf{x}_{k+1}|\mathbf{y}_{0:k}) \\ & \overset{(8.6)}{=} & \int \underbrace{\pi(\mathbf{x}_{k+1}|\mathbf{x}_k, \mathbf{y}_{0:k})}_{\overset{(8.9)}{=}\pi(\mathbf{x}_{k+1}|\mathbf{x}_k)} \underbrace{\pi(\mathbf{x}_k|\mathbf{y}_{0:k})}_{\overset{(8.10)}{=}\text{bel}\,(\mathbf{x}_k)} d\mathbf{x}_k \end{array} \qquad \{\pi(\mathbf{x}|\mathbf{z}) = \int \pi(\mathbf{x}|\mathbf{y},\mathbf{z})\cdot\pi(\mathbf{y}|\mathbf{z})\cdot d\mathbf{y}\}$$

This corresponds to relation (i). ■

**Remark 1**. From (8.11, ii), we have

$$\underbrace{\int \text{bel}\,(\mathbf{x}_k)\cdot d\mathbf{x}_k}_{=1} = \int \frac{\pi(\mathbf{y}_k|\mathbf{x}_k)\cdot\text{pred}\,(\mathbf{x}_k)}{\pi\,(\mathbf{y}_k|\mathbf{y}_{0:k-1})}\cdot d\mathbf{x}_k.$$

Thus

$$\pi\,(\mathbf{y}_k|\mathbf{y}_{0:k-1}) = \int \pi(\mathbf{y}_k|\mathbf{x}_k)\cdot\text{pred}\,(\mathbf{x}_k)\cdot d\mathbf{x}_k.$$

is a normaliser coefficient which can be computed directly from $\pi(\mathbf{y}_k|\mathbf{x}_k)$ and $\text{pred}(\mathbf{x}_k)$.

The relations (8.11) can be interpreted as an algorithm where the variables $\text{pred}(\mathbf{x}_k)$ and $\text{bel}(\mathbf{x}_k)$ are densities, *i.e.*, functions from $\mathbb{R}^n \to \mathbb{R}$. Such an algorithm cannot be implemented in a computer in the general case, and can only be poorly approximated. In practice different approaches can be used to implement a Bayesian filter.

- If the random vector $\mathbf{x}$ is discrete, the process can be represented by a Markov chain and the densities $\text{pred}(\mathbf{x}_k)$ and $\text{bel}(\mathbf{x}_k)$ can be represented by a vector of $\mathbb{R}^n$. The Bayesian filter can be implemented exactly.

- If the densities $\text{pred}(\mathbf{x}_k)$ and $\text{bel}(\mathbf{x}_k)$ are Gaussian, they can be represented exactly by their expectation and their covariance matrices. We get the Kalman filter [KAL 60].

- We can approximate $\text{pred}(\mathbf{x}_k)$ and $\text{bel}(\mathbf{x}_k)$ by cloud of points (called *particles*) with different weight. The corresponding observer is called *Particle filter* .

- We can represent the densities $\text{pred}(\mathbf{x}_k)$ and $\text{bel}(\mathbf{x}_k)$ by a subset of $\mathbb{R}^n$ which contain the support a the density. We obtain an observer called *set-membership filter* .

## 8.4 Bayes smoother

A filter is causal. This means that the estimation $\hat{\mathbf{x}}_{k|k-1}$ only takes into account the past. The *smoothing* process consists of a state estimation when all the measurements (future, present and past) are available. Let us denote by $N$ the maximum time $k$. This time can correspond, for instance, to the end date of a mission performed by the robot and for which we are trying to estimate its path. In order to perform smoothing, we simply need to run once more a Kalman filter in the backward direction and merge, for each $k$, the information from the future with that of the past.

**Theorem**. The three densities

$$\begin{array}{lcl} \text{pred}\left(\mathbf{x}_k\right) & \overset{\text{def}}{=} & \pi\left(\mathbf{x}_k|\mathbf{y}_{0:k-1}\right) \\ \text{bel}\left(\mathbf{x}_k\right) & \overset{\text{def}}{=} & \pi\left(\mathbf{x}_k|\mathbf{y}_{0:k}\right). \\ \text{back}\left(\mathbf{x}_k\right) & \overset{\text{def}}{=} & \pi\left(\mathbf{x}_k|\mathbf{y}_{0:N}\right) \end{array} \tag{8.12}$$

can be defined recursively as follows

$$\begin{array}{llcll} \text{(i)} & \text{pred}\left(\mathbf{x}_k\right) & = & \int \pi\left(\mathbf{x}_k|\mathbf{x}_{k-1}\right) \cdot \text{bel}\left(\mathbf{x}_{k-1}\right) \cdot d\mathbf{x}_{k-1} & \text{(prediction)} \\ \text{(ii)} & \text{bel}\left(\mathbf{x}_k\right) & = & \frac{\pi(\mathbf{y}_k|\mathbf{x}_k)\cdot\text{pred}(\mathbf{x}_k)}{\pi(\mathbf{y}_k|\mathbf{y}_{0:k-1})} & \text{(correction)} \\ \text{(iii)} & \text{back}\left(\mathbf{x}_k\right) & = & \text{bel}\left(\mathbf{x}_k\right) \int \frac{\pi(\mathbf{x}_{k+1}|\mathbf{x}_k)\cdot\text{back}(\mathbf{x}_{k+1})}{\text{pred}(\mathbf{x}_{k+1})} \cdot d\mathbf{x}_{k+1} & \text{(smoother)} \end{array} \tag{8.13}$$

**Proof**. The prediction equation (i) and the correction equation (ii) have already been proven. It remains to prove (iii). We have

$$\begin{array}{lcll} & \text{back}\left(\mathbf{x}_k\right) & & \\ = & \pi\left(\mathbf{x}_k|\mathbf{y}_{0:N}\right) & & \\ \overset{(8.6)}{=} & \int \underbrace{\pi\left(\mathbf{x}_k|\mathbf{x}_{k+1},\mathbf{y}_{0:N}\right)}_{\|} \cdot \pi\left(\mathbf{x}_{k+1}|\mathbf{y}_{0:N}\right) \cdot d\mathbf{x}_{k+1} & \{\pi(a|c) = \int \pi(a|b,c)\cdot\pi(b|c)\cdot db\} \\ \overset{(\text{Markov})}{=} & \int \overbrace{\pi\left(\mathbf{x}_k|\mathbf{x}_{k+1},\mathbf{y}_{0:k}\right)}^{\|} \cdot \text{back}\left(\mathbf{x}_{k+1}\right) \cdot d\mathbf{x}_{k+1} & \\ \overset{(\text{Bayes})}{=} & \int \overbrace{\frac{\pi\left(\mathbf{x}_{k+1}|\mathbf{x}_k,\mathbf{y}_{0:k}\right)\cdot\pi\left(\mathbf{x}_k|\mathbf{y}_{0:k}\right)}{\pi\left(\mathbf{x}_{k+1}|\mathbf{y}_{0:k}\right)}}^{\|} \cdot \text{back}\left(\mathbf{x}_{k+1}\right) \cdot d\mathbf{x}_{k+1} & \left\{\pi(a|b,c) = \pi(b|a,c)\cdot\frac{\pi(a|c)}{\pi(b|c)}\right\} \\ \overset{(\text{Markov})}{=} & \int \frac{\pi(\mathbf{x}_{k+1}|\mathbf{x}_k)\cdot\pi(\mathbf{x}_k|\mathbf{y}_{0:k})}{\text{pred}(\mathbf{x}_{k+1})} \cdot \text{back}\left(\mathbf{x}_{k+1}\right) \cdot d\mathbf{x}_{k+1} & \\ = & \underbrace{\pi\left(\mathbf{x}_k|\mathbf{y}_{0:k}\right)}_{\text{bel}(\mathbf{x}_k)} \int \frac{\pi(\mathbf{x}_{k+1}|\mathbf{x}_k)}{\text{pred}(\mathbf{x}_{k+1})} \cdot \text{back}\left(\mathbf{x}_{k+1}\right) \cdot d\mathbf{x}_{k+1} & \end{array}$$

## 8.5 Kalman smoother

### 8.5.1 Equations of the Kalman smoother

Consider a linear discrete time system described by the state equation.

$$\left\{\begin{array}{lcl} \mathbf{x}_{k+1} & = & \mathbf{A}_k\mathbf{x}_k + \mathbf{u}_k + \boldsymbol{\alpha}_k \\ \mathbf{y}_k & = & \mathbf{C}_k\mathbf{x}_k + \boldsymbol{\beta}_k \end{array}\right.$$

where $\boldsymbol{\alpha}_k$ and $\boldsymbol{\beta}_k$ are Gaussian, white and independent.

An optimized version of the Bayes smoother, in the case where all densities are Gaussian and all functions are linear is referred to as *Kalman smoother* (or Rauch-Tung-Striebel Smoother). It can then be applied by adding the smoothing equation of the following theorem to those of the Kalman filter.

**Theorem**. The smoothing density is

$$\text{back}\left(\mathbf{x}_k\right) = \pi\left(\mathbf{x}_k|\mathbf{y}_{0:N}\right) = \mathcal{N}\left(\mathbf{x}_k \left\| \hat{\mathbf{x}}_{k|N}, \boldsymbol{\Gamma}_{k|N}\right.\right).$$

where

$$\begin{array}{rcl} \mathbf{J}_k & = & \boldsymbol{\Gamma}_{k|k} \cdot \mathbf{A}_k^{\mathrm{T}} \cdot \boldsymbol{\Gamma}_{k+1|k}^{-1} \\ \hat{\mathbf{x}}_{k|N} & = & \hat{\mathbf{x}}_{k|k} + \mathbf{J}_k \cdot \left(\hat{\mathbf{x}}_{k+1|N} - \hat{\mathbf{x}}_{k+1|k}\right) \\ \boldsymbol{\Gamma}_{k|N} & = & \boldsymbol{\Gamma}_{k|k} - \mathbf{J}_k \cdot \left(\boldsymbol{\Gamma}_{k+1|k} - \boldsymbol{\Gamma}_{k+1|N}\right) \cdot \mathbf{J}_k^{\mathrm{T}}. \end{array} \tag{8.14}$$

**Proof**. See exercise 8.6 on page 192.

### 8.5.2 Implementation

In order to perform the smoothing process, we first need to run the Kalman filter for $k$ ranging from 0 to $N$, then run Equations [8.14] backwards for $k$ ranging form $N$ to 0. Note that all the quantities $\hat{\mathbf{x}}_{k+1|k}, \hat{\mathbf{x}}_{k|k}, \boldsymbol{\Gamma}_{k+1|k}, \boldsymbol{\Gamma}_{k|k}, \hat{\mathbf{x}}_{k|N}$ are stored in the lists `x_forw{k}`,`x_up{k}`, `G_forw{k}`, `G_up{k}`, `x_back{k}`, `G_back{k}`, where `forw`, `up`, `back` respectively mean *forward*, *update*, *backward*. The quantities $\mathbf{u}_k, \boldsymbol{\Gamma}_{\boldsymbol{\alpha}}(k), \mathbf{y}_k, \boldsymbol{\Gamma}_{\boldsymbol{\beta}}(k), \mathbf{A}_k$ are also stored in lists. The direct, or *forward* part of the smoother corresponding to the Kalman filter is given by the script:

```
x_forw{1}=...; G_forw{1}=...; % initialization
for k=1:N,
   [x_forw{k+1},G_forw{k+1},x_up{k},G_up{k}]
   =kalman(x_forw{k},G_forw{k},u{k},y{k},Galpha{k},Gbeta{k},A{k},C{k});
end;
```

As for the *backward* part of the smoother, this is given by:

```
x_back{N}=x_up{N};
G_back{kmax}=G_up{N};
for k=N-1:-1:1,
   J=G_up{k}*A'/G_forw{k+1};
   x_back{k}=x_up{k}+J*(x_back{k+1}-x_forw{k+1});
   G_back{k}=G_up{k}-J*(G_forw{k+1}-G_back{k+1})*J';
end;
```

Note that, given the specificity of MATLAB to index lists starting from 1, the initial condition corresponds to $k = 1$.

# Exercises

EXERCISE 8.1.– *Conditional and marginal densities*

Consider two random variables $x, y$ which belong to the set $\{1, 2, 3\}$ . The table below gives $\pi(x, y)$.

| $\pi(x, y)$ | $x = 1$ | $x = 2$ | $x = 3$ |
|---|---|---|---|
| $y = 1$ | 0.1 | 0.2 | 0 |
| $y = 2$ | 0.1 | 0.3 | 0.1 |
| $y = 3$ | 0 | 0.1 | 0.1 |

1) Compute the marginal densities $\pi(x)$ and $\pi(y)$ for $x$ and $y$.

2) Compute the conditional densities $\pi(x|y)$ and $\pi(y|x)$.

---

EXERCISE 8.2.– *Weather forecast*

**Markov chain**. A *Markov chain* is a sequence of discrete random variables $x_0, x_1, \ldots, x_k, \ldots$, called the *state variables*, where $k$ is time such that the future of the system only depends on the present state. More precisely, if the current state is known precisely, a full knownledge of the past will not bring only new information to predict the future. We can write the Markov property as

$$\pi\left(x_{k+1} = j|\ x_k = i_k, x_{k-1} = i_{k-1}, \ldots\right) = \pi\left(x_{k+1} = j|\ x_k = i_k\right).$$

The set of all feasible values for $x_k$ is the *state space* . This probabilistic definition is totally consistent with all the notions of state space used in this book. Markov chains can be described by an oriented graph the label of which corresponds to the probability to go from one state to another. To each $k$ we associate the vector $\mathbf{p}_k$ the $i^{\text{th}}$ component of which is defined by

$$p_{k,i} = \pi(x_k = i).$$

Note that all components of $\mathbf{p}_k$ are positive and that their sum is equal to 1.

**Weather forecast**. In the town of Brest, two type of weathers are possible: sunny coded by 1 and rainy coded by 2. And the weather does not change during the day. Denote by $x_k$ the weather at the day $k$. We assume that the state $x_k$ corresponds to a Markov chain the conditional density

$\pi(x_{k+1}|x_k)$ of which is given by

| $\pi(x_{k+1}\|x_k)$ | $x_k = 1$ | $x_k = 2$ |
|---|---|---|
| $x_{k+1} = 1$ | 0.9 | 0.5 |
| $x_{k+1} = 2$ | 0.1 | 0.5 |

This can be represented by the following picture.

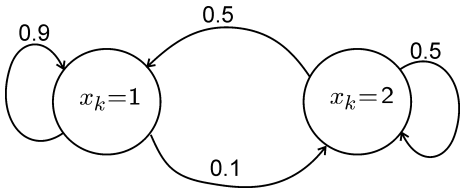


Figure 8.3: Markov chain describing the evolution of the weather in Brest

1) We represent $\pi(x_k)$ by a vector $\mathbf{p}_k$ of dimension 2. Using a Bayes filter, show that the evolution of $\mathbf{p}_k$ can be represented by a state equation of the form $\mathbf{p}_{k+1} = \mathbf{A} \cdot \mathbf{p}_k$.

2) At day $k$, the weather is sunny. What is the probability to be sunny the day $k + \ell$.

3) What about this probability when $\ell$ tends to infinity?

---

Exercise 8.3.– *Door robot*

In this exercise, we illustrate the Bayes filter in a system with a state $x \in \{0, 1\}$. It corresponds [THR 05] to a door which can be closed ($x = 0$) or open ($x = 1$) and an actuator to open and close. The system has also a sensor to measure the state. The input $u$, can be $u = -1$, 0 or 1 which mean 'close', 'do nothing' or 'open'. The require input may fail (for instance, if someone stops the door). We can represent the influence an $u$ on the state evolution by the following table

| $\pi(x_{k+1}\|x_k, u_k)$ | $u_k = -1$, | $u_k = 0$ | $u_k = 1$ |
|---|---|---|---|
| $x_k = 0$ | $\delta_0$ | $\delta_0$ | $0.2\ \delta_0 + 0.8\ \delta_1$ |
| $x_k = 1$ | $0.8\ \delta_0 + 0.2\ \delta_1$ | $\delta_1$ | $\delta_1$ |

For simplicity, the dependencies of the $\delta_i$ with respect to $x_{k+1}$ is omitted. When we read in the table

$$\pi(x_{k+1}|x_k = 0, u_k = 1) = 0.2\ \delta_0 + 0.8\ \delta_1,$$

it means that if $x_k = 0$, there is a probability of 0.8 to be at state 1 at time $k + 1$, if we apply the input $u_k = 1$. The sensor which gives us the state of the door is also uncertain. This is represented by the following conditional density

| $\pi(y_k\|x_k)$ | $x_k = 0$ | $x_k = 1$ |
|---|---|---|
| $y_k = 0$ | 0.8 | 0.4 |
| $y_k = 1$ | 0.2 | 0.6 |

The system can be represented by the graph of Figure 8.4.

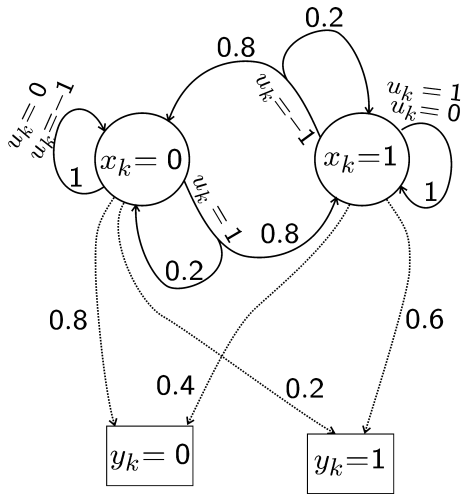


Figure 8.4: Graph representing the evolution and the observation of the door robot

1) Assume that the belief at time $k = 0$ is given by $\text{bel}(x_0) = 0.5 \cdot \delta_0 + 0.5 \cdot \delta_1$ and that $u_0 = 1$, compute $\text{pred}(x_1)$.

2) At time 1 we measure $y_1 = 1$. Compute $\text{bel}(x_1)$ .

3) If $\mathbf{p}_k^{\text{pred}}$ and $\mathbf{p}_k^{\text{bel}}$ are the stochastic vectors associated to $\text{pred}(x_k)$ et $\text{bel}(x_k)$. Give the state equation for the Bayesian filter.

---

EXERCISE 8.4.– *Robot in the forest*

The objective of this exercise if to give a graphical interpretation of the Bayesian filter. We recall that the Bayes filter is given by the following equations

$$\begin{array}{lcl} \text{pred}\left(\mathbf{x}_{k+1}\right) & = & \int \pi\left(\mathbf{x}_{k+1}|\mathbf{x}_k, \mathbf{u}_k\right) \cdot \text{bel}\left(\mathbf{x}_k\right) \cdot d\mathbf{x}_k \\ \text{bel}\left(\mathbf{x}_k\right) & = & \frac{\pi(\mathbf{y}_k|\mathbf{x}_k) \cdot \text{pred}(\mathbf{x}_k)}{\int \pi(\mathbf{y}_k|\mathbf{x}_k) \cdot \text{pred}(\mathbf{x}_k) \cdot d\mathbf{x}_k}. \end{array}$$

Figure 8.5 represents a robot with a scalar state $x$ which corresponds to its position on an horizontal line.

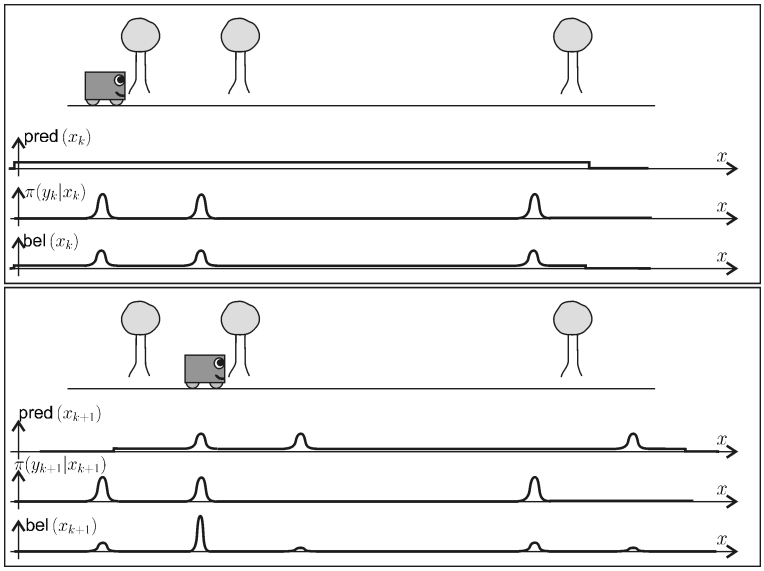


Figure 8.5: Illustration of the Bayesian filter

**Step** $k$. We assume that at time $k$, we know from the past the prediction pred$(x_k)$, which is uniform on a large interval. The robot knows that in its environment, there exists three trees that are identical. Now, at time $k$, the robot sees a tree in front of it. This corresponds the measurement $y_k$. It deduces the *likelyhood* $\pi(y_k|x_k)$ of $x$. The belief bel$(x_k)$ can thus be obtained simple multiplication of the density pred$(x_k)$ and the likelyhood $\pi(y_k|x_k)$, followed by a normalization.

**Step** $k$+1. The robot moves forward of few meters and the counter $k$ is increase by 1. From the belief at time $k$ and the knowledge of the motion, we can predict, via pred$(x_{k+1})$ the state at time $k+1$. To get bel$(x_{k+1})$ as previously at time $k$.

**Step** $k$+2. Assume now that the robot moves forward again of few meters but does not see any tree. Draw pred$(x_{k+2})$, $\pi(y_{k+2}|x_{k+2})$ and bel$(x_{k+2})$.

---

Exercise 8.5.– *Bayes rule with Kalman*

Consider two Gaussian random variables $x \sim \mathcal{N}(1,1)$ and $b \sim \mathcal{N}(0,1)$, i.e., the expectation of $x$ and $b$ are $\bar{x} = 1$ and $\bar{b} = 0$ and (2) the variances (*i.e*, the covariance matrix in the scalar case) of $x$ and $b$ are $\sigma_x^2 = 1$ and $\sigma_b^2 = 1$.

1) Define $y = x + b$. Give the expression of probabilility density function of $y$.

2) We collect the following measurement $y = 3$. Using the correction equations of the Kalman filter, compute the posterior density for $x$.

3) Provide the link with the Bayes rule.

---

Exercise 8.6.– *Derivation of the Kalman smoother*

1) Consider two normal vectors $\mathbf{a}$ and $\mathbf{b}$. Show that

$$\left.\begin{array}{rcl} \pi(\mathbf{a}) & = & \mathcal{N}\left(\mathbf{a}\,\|\hat{\mathbf{a}}, \mathbf{\Gamma_a}\right) \\ \pi(\mathbf{b}|\mathbf{a}) & = & \mathcal{N}\left(\mathbf{b}\,\|\mathbf{J}\mathbf{a}+\mathbf{u}, \mathbf{R}\right) \end{array}\right\} \implies \pi(\mathbf{b}) = \mathcal{N}\left(\mathbf{b}\,\middle\|\mathbf{J}\hat{\mathbf{a}}+\mathbf{u}, \mathbf{R}+\mathbf{J}\cdot\mathbf{\Gamma_a}\cdot\mathbf{J}^{\mathrm{T}}\right). \tag{8.15}$$

2) Consider a linear discrete time system described by the state equation.

$$\left\{\begin{array}{rcl} \mathbf{x}_{k+1} & = & \mathbf{A}_k\mathbf{x}_k + \mathbf{u}_k + \boldsymbol{\alpha}_k \\ \mathbf{y}_k & = & \mathbf{C}_k\mathbf{x}_k + \boldsymbol{\beta}_k \end{array}\right.$$

where $\boldsymbol{\alpha}_k$ and $\boldsymbol{\beta}_k$ are Gaussian, white and independent. Using the equations of the Kalman filter, show that

$$\pi\left(\mathbf{x}_k|\mathbf{x}_{k+1}, \mathbf{y}_{0:N}\right) = \mathcal{N}\left(\mathbf{x}_k\,\middle\|\hat{\mathbf{x}}_{k|k} + \mathbf{J}_k\left(\mathbf{x}_{k+1} - \hat{\mathbf{x}}_{k+1|k}\right), \mathbf{\Gamma}_{k|k} - \mathbf{J}_k\cdot\mathbf{\Gamma}_{k+1|k}\cdot\mathbf{J}_k^T\right) \tag{8.16}$$

where

$$\mathbf{J}_k = \mathbf{\Gamma}_{k|k}\cdot\mathbf{A}_k^T\cdot\mathbf{\Gamma}_{k+1|k}^{-1}.$$

3) From the two previous questions, show that the smoothing density is

$$\mathrm{back}\left(\mathbf{x}_k\right) = \pi\left(\mathbf{x}_k|\mathbf{y}_{0:N}\right) = \mathcal{N}\left(\mathbf{x}_k\,\middle\|\hat{\mathbf{x}}_{k|N}, \mathbf{\Gamma}_{k|N}\right)$$

where

$$\begin{array}{rcl} \hat{\mathbf{x}}_{k|N} & = & \hat{\mathbf{x}}_{k|k} + \mathbf{J}_k\cdot\left(\hat{\mathbf{x}}_{k+1|N} - \hat{\mathbf{x}}_{k+1|k}\right) \\ \mathbf{\Gamma}_{k|N} & = & \mathbf{\Gamma}_{k|k} - \mathbf{J}_k\cdot\left(\mathbf{\Gamma}_{k+1|k} - \mathbf{\Gamma}_{k+1|N}\right)\cdot\mathbf{J}_k^{\mathrm{T}}. \end{array}$$

---

Exercise 8.7.– *SLAM*

The *Redermor* (underwater robot built by GESMA, Brest) performed a two-hour mission in the Douarnenez bay (see Figure 8.6). During its mission, it collected data from its inertial unit (which gives us the Euler angles $\phi, \theta, \psi$), its Doppler log (which gives us the robot's speed $\mathbf{v}_r$ in the robot's coordinate system), its pressure sensor (which gives us the robot's depth $p_z$) and its altitude sensor (sonar that gives us the altitude $a$), with a sampling period of $dt = 0.1$ sec. This data can be found in the file `slam_data.txt`. The file is composed of 59 996 lines (one line per sampling period) and 9 columns which are respectively:

$$(t, \varphi, \theta, \psi, v_x, v_y, v_z, p_z, a)$$

where $p_z$ is the depth of the robot and $a$ is its altitude (in other words its distance to the seabed).

1) Given that the robot started from a position $\mathbf{p} = (0, 0, 0)$, at time $t = 0$, and using an Euler method, deduce an estimation for the path. Use for this the state equation:

$$\dot{\mathbf{p}}(t) = \mathbf{R}(\varphi(t), \theta(t), \psi(t))\cdot\mathbf{v}_r(t)$$

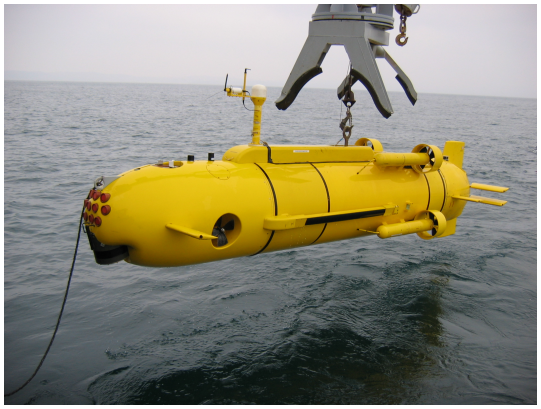

Figure 8.6: The *Redermor*, built by GESMA (Groupe d'Etude Sous-Marine de l'Atlantique), right before diving into the water

with $\mathbf{R}(\varphi, \theta, \psi)$ the Euler matrix (see [1.7] on page 15) whose expression we recall below :

$$\mathbf{R}(\varphi,\theta,\psi) = \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \sin\varphi & \cos\varphi \end{pmatrix}$$

2) The angles $\psi, \theta, \varphi$ are measured with a standard deviation of $(2 \times 10^{-4}, 2 \times 10^{-4}, 5 \times 10^{-3})$. The components of $\mathbf{v}_r$ are measured every $dt$ deconds with a standard deviation of $\sigma_v = 1$ ms$^{-1}$. We may assume that the robot satisfies the equation:

$$\mathbf{p}_{k+1} = \mathbf{p}_k + (dt \cdot \mathbf{R}(k)) \cdot \bar{\mathbf{v}}_r(k) + \boldsymbol{\alpha}_k$$

where $\boldsymbol{\alpha}_k$ is a white noise and $\bar{\mathbf{v}}_r(k)$ is a measurement of the average speed over the corresponding sampling period. Show that a realistic covariance matrix for $\boldsymbol{\alpha}_k$ is:

$$\boldsymbol{\Gamma}_{\boldsymbol{\alpha}} = dt^2\sigma_v^2 \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

**Remark**. We are not in the situation described in Section 7.5 or in Exercice 7.6 where the covariance for the state noise should have the same order as $\frac{1}{dt}$ . Here, our system in not a discretisation of a continuous random process, the behavior of which should be independent of $dt$. On the contrary, $dt$ is a sampling time of the velocity sensor. Smaller is $dt$, more accurate will be the integration.

3) Using the Kalman filter as predictor, calculate the precision with which the robot knows its position at each moment $t = k \cdot dt$. Give, in function of $t$, the standard deviation of the error over the position. What will this become after an hour ? After two hours ? Verify your calculations experimentally in MATLAB by implementing a Kalman predictor.

4) During its mission, the robot may detect several landmarks with its lateral sonar (here these will be mines). When the robot detects a landmark, it will be on its right side and in a plane perpendicular to the robot. The seabed is assumed to be flat and horizontal. The following table shows the detection times, the numbers $i$ of the landmarks and the distance $r_i$ between the robot

and the landmark:

| $t$ | 1 054 | 1 092 | 1 374 | 1 748 | 3 038 | 3 688 | 4 024 | 4 817 | 5 172 | 5 232 | 5 279 | 5 688 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $i$ | 1 | 2 | 1 | 0 | 1 | 5 | 4 | 3 | 3 | 4 | 5 | 1 |
| $r_i(t)$ | 52.42 | 12.47 | 54.40 | 52.68 | 27.73 | 26.98 | 37.90 | 36.71 | 37.37 | 31.03 | 33.51 | 15.05 |

SLAM (*Simultaneous Localization And Mapping*) seeks to use these repeated detections to improve the precision of the estimation of its path. For this, we form a large state vector $\mathbf{x}$, of dimension $3 + 2 \cdot 6 = 15$ that contains the position of the robot $\mathbf{p}$ as well as the vector $\mathbf{q}$ of dimension 12 containing the coordinates (as $x$ and $y$) of the six landmarks. Let us note that since the landmarks are immobile, we have $\dot{\mathbf{q}} = \mathbf{0}$. Give the MATLAB function `[y,C,Gbeta]=g(k)` that corresponds to the observation. This function returns the measurement vector $\mathbf{y}$, the matrix $\mathbf{C}(k)$ and the covariance matrix of the measurement noise. As for the standard deviation of the measurement noise $\beta_k$, we will take 0.1 for that of the depth and 1 for that of the robot-landmark distance.

5) Using a Kalman filter, find the position of the landmarks together with the associated uncertainty. Show how the robot was able to readjust its position.

6) Use the Kalman smoother to improve the precision over the landmark positions by taking into account the past as well as the future.

---

EXERCISE 8.8.– a priori *SLAM*

An underwater robot carries a navigation system (inertial unit and Doppler log) that gives its position as $x, y$ with a drift of 100 meters per hour. This means that if the robot knows its position with a precision of $r$ meters at time $t$, then an hour before and an hour later it knows its position with an error smaller than $r + 100$ m.

In the beginning, our robot locates itself by GPS with a precision of 10 m and proceeds to dive in a zone of flat seabed. It swims around for eight hours at constant depth (that it can measure with a pressure sensor). When it resurfaces, it locates itself again using the GPS, again with a precision of 10 m. Each hour, the robot passes above a remarkable landmark (small rock with a particular form, for instance) that can be detected by a camera places under the robot. The following table indicates the number of the detected landmark in function of time, expressed in hours:

| t(hour) | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| landmark | 1 | 2 | 1 | 3 | 2 | 1 | 4 |

We may deduce from this table that the robot encounters four remarkable landmarks in total and that it encounters landmark 1 three times, at times $t = 1H$, $t = 3H$ and $t = 6H$. With what precision is it able to localize landmarks 1, 2, 3 and 4 ?

---

# Bibliography


[BAZ 12] Bazeille S., Quidu I, Jaulin L., *Color-based underwater object recognition using water light attenuation*, Journal of Intelligent Service Robotics, vol. 5, **2**, 2012.

[BEA 12] Beard R., McLain T., *Small Unmanned Aircraft, Theory and Practice*, Princeton University Press, 2012.

[BOY 06] Boyer F., Alamir M., Chablat D., Khalil W., Leroyer A., Lemoine P., Robot anguille sous-marin en 3d, *Techniques de l'Ingénieur*, 2006.

[CHE 07] Chevallereau C., Bessonnet G., Abba G., Aoustin Y., *Les robots marcheurs bipèdes ; Modélisation, conception, synthèse de la marche, commande*, Hermès-Lavoisier, Paris, 2007.

[COR 11] @BOOK{Corke11, author = {P. Corke}, year = 2011, title = {Robotics, Vision and Control}, publisher = {Springer}, address = {Berlin Heidelberg} }

[CRE 14] Creuze V., Robots marins et sous-marins ; perception, modélisation, commande, *Techniques de l'ingénieur*, 2014.

[DEL 93] de Larminat P., *Automatique, commande des systèmes linéaires*, Hermès, Paris, France, 1993.

[DRE 11] Drevelle V., *Etude de méthodes ensemblistes robustes pour une localisation multisensorielle intègre, Application à la navigation des véhicules en milieu urbain*, PhD dissertation, Université de Technologie de Compiègne, Compiègne, France, 2011.

[DUB 57] Dubins L.E., On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents, *American Journal of Mathematics*, vol. 79, **3**, p. 497-516, 1957.

[FAN 01] Fantoni I., Lozano R., *Non-linear control for underactuated mechanical systems*, Springer-Verlag, 2001.

[FLI 13] Fliess M., Join C., Model-free control, *International Journal of Control*, vol. 86, **12**, p. 2228-2252, 2013.

[FOS 2002] Fossen T., *Marine Control Systems: Guidance, Navigation and Control of Ships, Rigs and Underwater Vehicles*, Marine Cybernetics, 2002.

[KAI 80] Kailath T., *Linear Systems*, Prentice Hall, Englewood Cliffs, 1980.

[GOR 11] Gorgues T., Ménage O., Terre T., Gaillard F., An innovative approach of the surface layer sampling, *Journal des Sciences Halieutique et Aquatique*, vol. 4, p. 105-109, 2011.

[MUR 89] Murata T., *Petri nets : properties, analysis and applications*. Proceedings of the IEEE, vol. 77, **4**, 1989.

[HER 10] Herrero P., Jaulin L., Vehi J., Sainz M.A., Guaranteed set-point computation with application to the control of a sailboat, *International Journal of Control Automation and Systems*, vol. 8, **1**, p. 1-7, 2010.

[JAU 05] Jaulin L., *Représentation d'état pour la modélisation et la commande des systèmes (Coll. Automatique de base)*, Hermès, London, 2005.

[JAU 10] Jaulin L., Commande d'un skate-car par biomimétisme, *CIFA 2010*, Nancy, France, 2010.

[JAU 12] Jaulin L., Le Bars F., An Interval Approach for Stability Analysis; Application to Sailboat Robotics, *IEEE Transaction on Robotics*, vol. 27, **5**, 2012.

[JAU 13] Jaulin L., *Automation for Robotics*, ISTE editions, 2013.

[JAU 02] Jaulin L., Kieffer M., Walter E., Meizel D., Guaranteed Robust Nonlinear Estimation with Application to Robot Localization, *IEEE Transactions on systems, man and cybernetics; Part C Applications and Reviews*, vol. 32, 2002.

[KAL 60] Kalman E.R. Contributions to the theory of optimal control. *Bol. Soc. Mat. Mex.*, vol. 5, p. 102-119, 1960.

[KHA 02] Khalil H.K., *Nonlinear Systems*, Third Edition, Prentice Hall, 2002

[KLE 06] Klein E.M.V., *Aircraft System Identification: Theory And Practice*, American Institute of Aeronautics and Astronautics, 2006.

[LAR 03] Laroche B., Martin P., Petit N., *Commande par platitude, Equations différentielles ordinaires et aux dérivées partielles*, Available at http://cas.ensmp.fr/∼petit/ensta/main.pdf, 2003.

[LAT 91] Latombe J., *Robot Motion Planning*, Kluwer Academic Publishers, Boston, MA, 1991.

[LAU 01] Laumond J., *La robotique mobile*, Hermès, Paris, France, 2001.

[LaV 06] LaValle S., *Planning algorithm*, Cambridge University Press, 2006.

[ROM 12] Romero-Ramirez M., *Contribution à la commande de voiliers robotisés*, PhD dissertation, Université Pierre et Marie Curie, France, 2012.

[SLO 91] Slotine J.J., Li W., *Applied nonlinear control*, Prentice Hall, 1991.

[SPO 01] Spong M., Corke P., Lozano R. *Nonlinear control of the Reaction Wheel Pendulum*, *Automatica*, vol. 37, p. 1845-1851, 2001

[1]

[THR 05] Thrun S., Bugard W. and Fox D., *Probabilistic Robotics*, MIT Press, Cambridge, 2005.

[WAL 14] Walter E., *Numerical Methods and Optimization ; a Consumer Guide*, Springer, London, 2014.

# Index