Mobile robotics: Kalman Filter
https://www.ensta-bretagne.fr/kalmooc/

Luc Jaulin

October 12, 2023

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 collect 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, underwater robots [1], 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. 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*, *i.e.*, 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 [2] and [3] for more details 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.

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 6 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 1 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 5 presents the *Kalman filter*. This filter can be seen as a state observer for dynamic linear systems with Gaussian noise with coefficients that vary in time. A generalization of the Kalman filter to the case where the functions are nonlinear and the noise is non Gaussian is provided in Chapter 8. The resulting observer, will is called the Bayes filter, computes the probability density function of the state vector at each time.

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

Chapter 1

Least-square method

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.

1.1 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.

1.1.1 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:

$$f(\mathbf{x}) = (x_1 \;\; x_2) \begin{pmatrix} 2 & -3 \\ -3 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + (-2 \;\; 1) \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + 1. \tag{1.1}$$

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:

$$\begin{array}{rcl} \frac{df}{d\mathbf{x}}(\mathbf{x}) & = & \left(\frac{\partial f}{\partial x_1}(\mathbf{x}) \quad \frac{\partial f}{\partial x_2}(\mathbf{x}) \right) \\ & = & (4x_1 - 6x_2 - 2 \quad -6x_1 + 2x_2 + 1) \end{array}$$

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*.

1.1.2 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:

$$\begin{array}{rcl} f(\mathbf{x} + \delta\mathbf{x}) & = & (\mathbf{x} + \delta\mathbf{x})^{\mathrm{T}} \cdot \mathbf{Q} \cdot (\mathbf{x} + \delta\mathbf{x}) \\ & = & \mathbf{x}^{\mathrm{T}} \cdot \mathbf{Q} \cdot \mathbf{x} + \mathbf{x}^{\mathrm{T}} \cdot \mathbf{Q} \cdot \delta\mathbf{x} + \delta\mathbf{x}^{\mathrm{T}} \cdot \mathbf{Q} \cdot \mathbf{x} + \delta\mathbf{x}^{\mathrm{T}} \cdot \mathbf{Q} \cdot \delta\mathbf{x} \\ & = & \mathbf{x}^{\mathrm{T}} \cdot \mathbf{Q} \cdot \mathbf{x} + 2\mathbf{x}^{\mathrm{T}} \cdot \mathbf{Q} \cdot \delta\mathbf{x} + o(||\delta\mathbf{x}||) \end{array}$$

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 (1.1) is:

$$2(x_1 \;\; x_2) \begin{pmatrix} 2 & -3 \\ -3 & 1 \end{pmatrix} + (-2 \;\; 1) = \begin{pmatrix} 4x_1 - 6x_2 - 2 & -6x_1 + 2x_2 + 1 \end{pmatrix}.$$

1.1.3 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: s

$$\mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x} + \mathbf{L}\mathbf{x} = \alpha - c$$

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.

1.1.4 Minimizing a quadratic function

Theorem 1. *If $\mathbf{Q}$ is positive definite, the function $f(\mathbf{x}) = \mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{L}\mathbf{x} + c$ has one and only one minimizer $\mathbf{x}^*$ given by*

$$\mathbf{x}^* = -\frac{1}{2}\mathbf{Q}^{-1}\mathbf{L}^T$$

and the minimum is $f(\mathbf{x}^*) = -\frac{1}{4}\mathbf{L}\mathbf{Q}^{-1}\mathbf{L}^T + c$.

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. The quadratic function:

$$f(\mathbf{x}) = (x_1 \;\; x_2)\begin{pmatrix} 2 & -1 \\ -1 & 1 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 3 & 4 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + 5$$

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. 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 $f(x^*) = 3 - 6 + 7 = 4$.

1.2 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.

1.2.1 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 (*i.e.*, $\dim(\mathbf{y}) > \dim(\mathbf{p})$). 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.

Exercises

EXERCISE 1.– *Representation of a quadratic function*

See the correction video at https://youtu.be/8xgIbnlRZ8s

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 $\begin{pmatrix} x & y \end{pmatrix} \cdot \mathbf{Q} \cdot \begin{pmatrix} x & y \end{pmatrix}^{\mathrm{T}} + \mathbf{L} \begin{pmatrix} x & y \end{pmatrix}^{\mathrm{T}} + c$, where $\mathbf{Q}$ is a symmetric matrix. Verify that the gradient found in question 1) is given by $2 \begin{pmatrix} x & y \end{pmatrix} \mathbf{Q}$. Draw the vector field associated with this gradient. Discuss.

3) 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 2.– *Identification of a parabola*

See the correction video at https://youtu.be/NXLx02n2PJs

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 3.– *Identifying the parameters of a DC motor*

See the correction video at https://youtu.be/UQzDMr2VGbY

The angular speed Ω 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
Ω(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 4.– *Estimation of a transfer function*

See the correction video at https://youtu.be/88BnsOpVZOk

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.

Chapter 2

Parameter estimation

Estimation theory deals with estimating the values of parameters based on measured empirical data. An estimator attempts to approximate the unknown parameters using the measurements.

2.1 Principle

Denote by $\mathbf{y}$ is the vector of measurements and by $\mathbf{p}$ the vector of parameters to be estimated. If $\mathbf{f}(\mathbf{p})$ is the output generated by a model, where $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^p$ is nonlinear, 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 2.1):

- The vector of the filtered measurements $\hat{\mathbf{y}}$ represents the projection of $\mathbf{y}$ on the set $\mathbf{f}(\mathbb{R}^n)$;
- 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 $\mathbf{f}(.)$.

When $\mathbf{f}(\mathbf{p})$ is nonlinear, we can use a local optimization algorithm to try to obtain $\hat{\mathbf{p}}$. We propose now two different approaches to perform a local minimization: the Newton method and the Monte-Carlo algorithm.

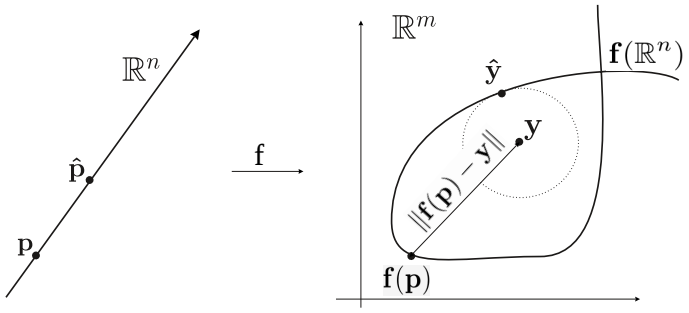

Figure 2.1: Illustration of the least squares method in the nonlinear case

2.2 Newton method

In order to minimize $j(\mathbf{p}) = ||\mathbf{f}(\mathbf{p}) - \mathbf{y}||^2$, the Newton method assumes that an approximation $\mathbf{p}_0$ of the minimizer is available. Around $\mathbf{p}_0$, we have:

$$\mathbf{f}(\mathbf{p}) \simeq \mathbf{f}(\mathbf{p}_0) + \frac{d\mathbf{f}}{d\mathbf{p}}(\mathbf{p}_0) \cdot (\mathbf{p} - \mathbf{p}_0).$$

Therefore

$$\begin{array}{rcl} j(\mathbf{p}) & = & ||\mathbf{f}(\mathbf{p}) - \mathbf{y}||^2 \\ & \simeq & ||\mathbf{f}(\mathbf{p}_0) + \underbrace{\frac{d\mathbf{f}}{d\mathbf{p}}(\mathbf{p}_0)}_{=\mathbf{M}} \cdot (\mathbf{p} - \mathbf{p}_0) - \mathbf{y}||^2 \\ & = & ||\mathbf{M} \cdot \mathbf{p} + \underbrace{\mathbf{f}(\mathbf{p}_0) - \mathbf{M} \cdot \mathbf{p}_0 - \mathbf{y}}_{=-\mathbf{z}}||^2 \\ & = & ||\mathbf{M} \cdot \mathbf{p} - \mathbf{z}||^2 \end{array}$$

The minimizer is

$$\begin{array}{rcl} \mathbf{p}_1 & = & \underbrace{\left(\mathbf{M}^{\mathrm{T}}\mathbf{M}\right)^{-1}\mathbf{M}^{\mathrm{T}}}_{=\mathbf{K}} \cdot \mathbf{z} \\ & = & \mathbf{K} \cdot (\mathbf{y} - \mathbf{f}(\mathbf{p}_0) + \mathbf{M} \cdot \mathbf{p}_0) \\ & = & \mathbf{p}_0 + \mathbf{K} \cdot (\mathbf{y} - \mathbf{f}(\mathbf{p}_0)) \end{array}$$

which is expected to be closer to the solution than $\mathbf{p}_0$. We apply the procedure several times and we get the following Newton algorithm.

	Algorithm NEWTON (in : $\mathbf{p}_0, \mathbf{y}$)
1	for $k = 0$ to k_{max}
2	$\mathbf{M} = \frac{d\mathbf{f}}{d\mathbf{p}}(\mathbf{p}_k)$
3	$\mathbf{K} = \left(\mathbf{M}^{\mathrm{T}}\mathbf{M}\right)^{-1}\mathbf{M}^{\mathrm{T}}$
4	$\mathbf{p}_{k+1} = \mathbf{p}_k + \mathbf{K} \cdot (\mathbf{y} - \mathbf{f}(\mathbf{p}_k))$

The Newton algorithm is illustrated by Figure 2.2. In the left figure, the sequence is represented on the $\mathbf{p}$-$\mathbf{y}$ space. We observe that we converge to the point $\hat{\mathbf{p}}$ which satisfies $\mathbf{f}(\hat{\mathbf{p}}) = \mathbf{y}$ but this is mainly due to the fact that we have a two dimensional representation. In practice we only have $||\mathbf{f}(\mathbf{p}) - \mathbf{y}||^2 \simeq 0$. On the right figure, the representation is made on the $\mathbf{p}$-j space.

Unfortunately, even if the solution of our minimization problem is unique, the Newton algorithm may diverge or converge to a point which is not the solution of our problem.

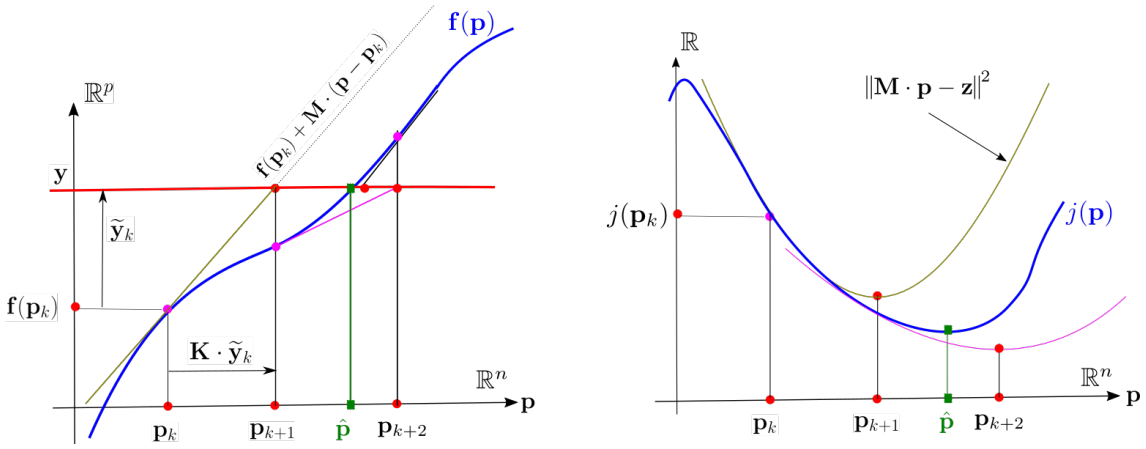

Figure 2.2: Illustration of the Newton method to minimize $||\mathbf{f}(\mathbf{p}) - \mathbf{y}||^2$

2.3 Monte-Carlo method

When we have no reliable prior estimation of the parameter vector $\mathbf{p}$ to be estimated, the Newton method may be trapped by a local minimized. Random based minimization method may be used to find the global minimizer. The following algorithm proposes a simple version of a Monte-Carlo algorithm:

	Algorithm MINIMIZE(input: $\mathbf{p}$)
1	take a random movement $\mathbf{d}$
2	$\mathbf{q} = \mathbf{p} + \mathbf{d}$
3	if $j(\mathbf{q}) < j(\mathbf{p})$ then $\mathbf{p} = \mathbf{q}$
4	go to 1

This algorithm is expected to converge toward 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 5.– *Newton method for localization*

See the correction video at https://youtu.be/f4ID4iyEEZc

We consider a robot which is at unknown position $\mathbf{p} = (p_1, p_2)$. It measures distances to 4 landmarks at position

$$\mathbf{m}(1) = \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \mathbf{m}(2) = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \mathbf{m}(3) = \begin{pmatrix} 3 \\ 2 \end{pmatrix}, \mathbf{m}(4) = \begin{pmatrix} 4 \\ 5 \end{pmatrix}.$$

as illustrated by Figure 2.3.

1) Assume that the robot is at position $\mathbf{p}$, give an expression for the vector $\mathbf{f}(\mathbf{p}) \in \mathbb{R}^4$ of all distances to the landmarks. Compute $\mathbf{f}(\mathbf{p})$ for $\mathbf{p} = (2, -3)$.

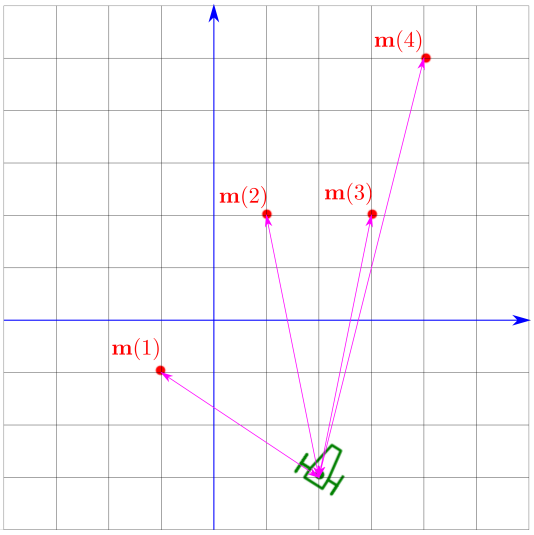

Figure 2.3: The robot measures the distances to the four landmarks

2) Assume that the distance to the landmarks that have been measured are given by the vector

$\mathbf{y} = (4, 5, 5, 8)$.

To estimate the position of the robot, we propose to use a least-square approach. For this, we build

the criterion

$$j(\mathbf{p}) = \|\mathbf{f}(\mathbf{p}) - \mathbf{y}\|^2.$$

Draw the level curves for $j(\mathbf{p})$. Discuss.

3) Using a Newton minimization method, provide a least-square estimation of $\mathbf{p}$. The initial point will be chosen as $\mathbf{p}(0)=(4,3)$.

EXERCISE 6.– *Monte-Carlo method*

See the correction video at https://youtu.be/K6PeWq1AYwM

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) Write a 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 ε 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 7.– *Localization by simulated annealing*

See the correction video at https://youtu.be/oHbTrxpnOHo

The localization problem that we will now consider is inspired from [4]. The robot, represented on Figure 2.4, 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 $\mathbf{y}$.

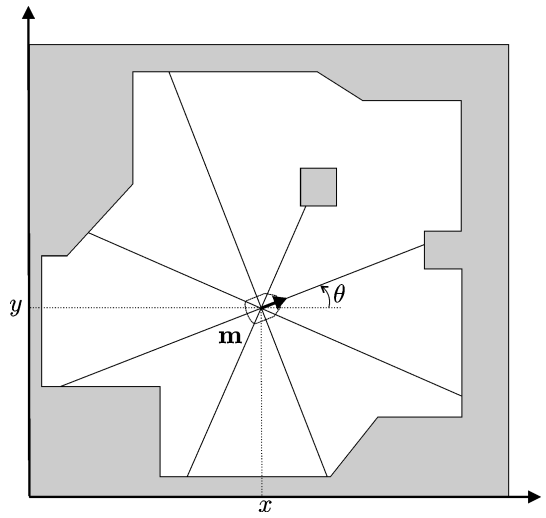

Figure 2.4: Robot equipped with eight telemeters trying to localize itself

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 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:

$$\mathbf{A} = \begin{pmatrix} 0 & 7 & 7 & 9 & 9 & 7 & 7 & 4 & 2 & 0 & 5 & 6 & 6 & 5 \\ 0 & 0 & 2 & 2 & 4 & 4 & 7 & 7 & 5 & 5 & 2 & 2 & 3 & 3 \end{pmatrix}$$

$$\mathbf{B} = \begin{pmatrix} 7 & 7 & 9 & 9 & 7 & 7 & 4 & 2 & 0 & 0 & 6 & 6 & 5 & 5 \\ 0 & 2 & 2 & 4 & 4 & 7 & 7 & 5 & 5 & 0 & 2 & 3 & 3 & 2 \end{pmatrix}$$

$$\mathbf{y} = (6.4,\ 3.6,\ 2.3,\ 2.1,\ 1.7,\ 1.6,\ 3.0,\ 3.1)^{\mathrm{T}}$$

Chapter 3

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.

3.1 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 $\widetilde{\mathbf{x}} = \mathbf{x} - \bar{\mathbf{x}}$ and $\widetilde{\mathbf{y}} = \mathbf{y} - \bar{\mathbf{y}}$. The *covariance matrix* is given by:

$$\mathbf{\Gamma_{xy}} = E\left(\widetilde{\mathbf{x}} \cdot \widetilde{\mathbf{y}}^{\mathrm{T}}\right) = E\left((\mathbf{x} - \bar{\mathbf{x}})(\mathbf{y} - \bar{\mathbf{y}})^{\mathrm{T}}\right).$$

The covariance matrix for $\mathbf{x}$ is defined by:

$$\mathbf{\Gamma_x} = \mathbf{\Gamma_{xx}} = E\left(\widetilde{\mathbf{x}} \cdot \widetilde{\mathbf{x}}^{\mathrm{T}}\right) = E\left((\mathbf{x} - \bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}})^{\mathrm{T}}\right).$$

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 program:

$$
\begin{array}{ll}
1 & N := 1000;\ \mathbf{x} := 2 \cdot \mathbf{1}_{N\times 1} + \mathrm{randn}_{N\times 1} \\
2 & \mathbf{y} := 2 \cdot \begin{pmatrix} x_1^2 \\ \vdots \\ x_N^2 \end{pmatrix} + \mathrm{randn}_{N\times 1} \\
3 & \bar{x} := \frac{1}{N}\sum_i x_i;\ \ \bar{y} = \frac{1}{N}\sum_i y_i \\
4 & \widetilde{\mathbf{x}} := \mathbf{x} - \bar{x} \cdot \mathbf{1}_{N\times 1}; \widetilde{\mathbf{y}} = \mathbf{y} - \bar{y} \cdot \mathbf{1}_{N\times 1} \\
5 & \begin{pmatrix} \Gamma_x & \Gamma_{xy} \\ \Gamma_{xy} & \Gamma_y \end{pmatrix} := \frac{1}{N} \cdot \begin{pmatrix} \sum_i \widetilde{x}_i^2 & \sum_i \widetilde{x}_i\widetilde{y}_i \\ \sum_i \widetilde{x}_i\widetilde{y}_i & \sum_i \widetilde{y}_i^2 \end{pmatrix}
\end{array}
$$

This yields Figure 3.1, 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:

$$\bar{x} \simeq 1.99, \bar{y} \simeq 9.983, \Gamma_x \simeq 1.003, \Gamma_y \simeq 74.03, \Gamma_{xy} \simeq 8.082.$$

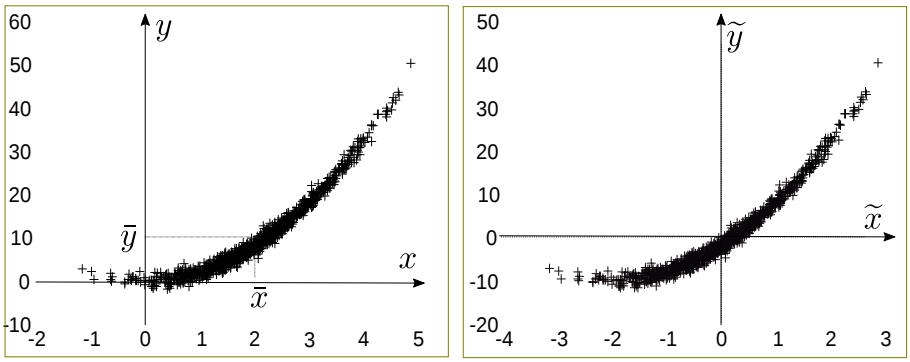

Figure 3.1: Cloud of points that represents a pair of two random variables

Two random vectors $\mathbf{x}$ and $\mathbf{y}$ are linearly independent (or non-correlated or orthogonal) if $\mathbf{\Gamma_{xy}} = \mathbf{0}$. On Figure 3.2, the two clouds of points correspond to non-correlated variables. Only the figure on the right corresponds to independent variables.

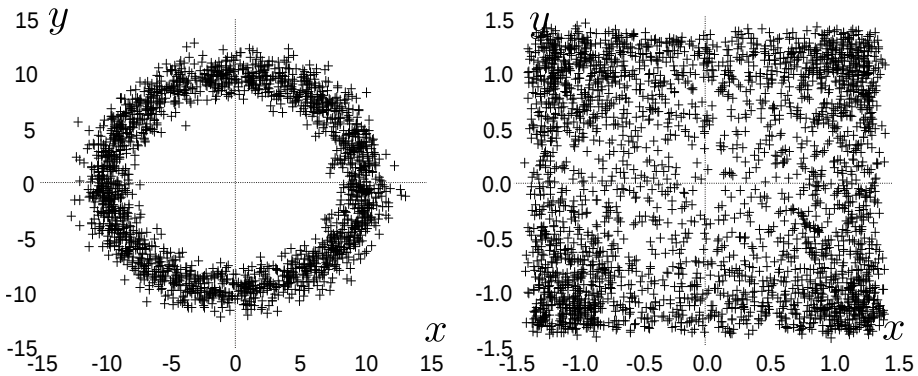

Figure 3.2: Left : dependent but non-correlated variables (x, y) ; Right : independent variables

The figure on the left was generated by:

$$\begin{array}{|ll|}\hline 1 & N := 2000; \\ 2 & \boldsymbol{\rho} := 10 \cdot \mathbf{1}_{N\times 1} + \mathrm{randn}_{N\times 1} \\ 3 & \boldsymbol{\theta} := 2\pi \cdot \mathrm{rand}_{N\times 1} \\ 4 & \mathbf{x} := \begin{pmatrix} \rho_1 \sin\theta_1 \\ \vdots \\ \rho_N \sin\theta_N \end{pmatrix}; \mathbf{y} := \begin{pmatrix} \rho_1 \cos\theta_1 \\ \vdots \\ \rho_N \cos\theta_N \end{pmatrix} \\ \hline \end{array}$$

And the figure on the right was generated by:

$$\begin{array}{|ll|}\hline 1 & N := 2000; \\ 2 & \mathbf{x} := \mathrm{atan}(2 \cdot \mathrm{randn}_{N\times 1}) \\ 3 & \mathbf{y} := \mathrm{atan}(2 \cdot \mathrm{randn}_{N\times 1}) \\ \hline \end{array}$$

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 $\boldsymbol{\Gamma}_{\mathbf{x}}$ of $\mathbf{x}$ is diagonal.

3.2 Properties

Covariance matrices are symmetric and positive, *i.e.*, all of their eigenvalues are real and positive. The set of all covariance matrices of $\mathbb{R}^{n\times n}$ will be denoted by $\mathcal{S}^+(\mathbb{R}^n)$.

Decomposition. Every symmetric matrix $\boldsymbol{\Gamma}$ can be put into a diagonal form and has an orthonormal eigenvector basis. We may therefore write:

$$\boldsymbol{\Gamma} = \mathbf{R} \cdot \mathbf{D} \cdot \mathbf{R}^{-1}$$

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 $\boldsymbol{\Gamma}$ of $\mathcal{S}^+(\mathbb{R}^n)$ has a square root in $\mathcal{S}^+(\mathbb{R}^n)$. This square root will be denoted by $\boldsymbol{\Gamma}^{\frac{1}{2}}$. Following the eigenvalue correspondence theorem, the eigenvalues of $\boldsymbol{\Gamma}^{\frac{1}{2}}$ are the square roots of those of the eigenvalues of $\boldsymbol{\Gamma}$.

Example. Consider the following script:

$$\begin{array}{|ll|}\hline 1 & \mathbf{A} := \mathrm{rand}_{3\times 3}; \mathbf{S}_1 := \mathbf{A} \cdot \mathbf{A}^{\mathrm{T}} \\ 2 & [\mathbf{R}, \mathbf{D}] := \mathrm{eig}(\mathbf{S}_1);\ \mathbf{S}_2 := \mathbf{R} \cdot \mathbf{D} \cdot \mathbf{R}^{\mathrm{T}} \\ 3 & \mathbf{A}_2 := \mathbf{S}_2^{\frac{1}{2}};\ \mathbf{S}_3 := \mathbf{A}_2 \cdot \mathbf{A}_2^{\mathrm{T}} \\ \hline \end{array}$$

The matrix $\mathbf{D}$ is diagonal and the matrix $\mathbf{R}$ is a rotation matrix that contains the eigenvectors of $\mathbf{S}_1$. The three matrices $\mathbf{S}_1, \mathbf{S}_2, \mathbf{S}_3$ are equal. This is not the case for matrices $\mathbf{A}$ and $\mathbf{A}_2$ since only $\mathbf{A}_2$ is symmetric. Note that $\mathbf{A}_2$ 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.

3.3 Linear transformation

The following result shows how a covariance matrix can easily be propagated through any linear (or affine) transformation.

Theorem 2. *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:*

$$\begin{array}{rcl} \bar{\mathbf{y}} & = & \mathbf{A}\bar{\mathbf{x}} + \mathbf{b} \\ \mathbf{\Gamma}_{\mathbf{y}} & = & \mathbf{A}\cdot\mathbf{\Gamma}_{\mathbf{x}}\cdot\mathbf{A}^T + \mathbf{\Gamma}_{\boldsymbol{\alpha}} \end{array} \tag{3.1}$$

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:

$$\begin{aligned} \mathbf{\Gamma}_{\mathbf{y}} &= E\left((\mathbf{y}-\bar{\mathbf{y}})(\mathbf{y}-\bar{\mathbf{y}})^{\mathrm{T}}\right) \\ &= E\left((\mathbf{A}\mathbf{x}+\boldsymbol{\alpha}+\mathbf{b}-\mathbf{A}\bar{\mathbf{x}}-\mathbf{b})(\mathbf{A}\mathbf{x}+\boldsymbol{\alpha}+\mathbf{b}-\mathbf{A}\bar{\mathbf{x}}-\mathbf{b})^{\mathrm{T}}\right) \\ &= E\left((\mathbf{A}\widetilde{\mathbf{x}}+\boldsymbol{\alpha})\cdot(\mathbf{A}\widetilde{\mathbf{x}}+\boldsymbol{\alpha})^{\mathrm{T}}\right) \\ &= \mathbf{A}\cdot\underbrace{E\left(\widetilde{\mathbf{x}}\cdot\widetilde{\mathbf{x}}^{\mathrm{T}}\right)}_{=\mathbf{\Gamma}_{\mathbf{x}}}\cdot\mathbf{A}^{\mathrm{T}} + \mathbf{A}\cdot\underbrace{E\left(\widetilde{\mathbf{x}}\cdot\boldsymbol{\alpha}^{\mathrm{T}}\right)}_{=\mathbf{0}} + \underbrace{E\left(\boldsymbol{\alpha}\cdot\widetilde{\mathbf{x}}^{\mathrm{T}}\right)}_{=\mathbf{0}}\cdot\mathbf{A}^{\mathrm{T}} + \underbrace{E\left(\boldsymbol{\alpha}\cdot\boldsymbol{\alpha}^{\mathrm{T}}\right)}_{=\mathbf{\Gamma}_{\boldsymbol{\alpha}}} \\ &= \mathbf{A}\cdot\mathbf{\Gamma}_{\mathbf{x}}\cdot\mathbf{A}^{\mathrm{T}} + \mathbf{\Gamma}_{\boldsymbol{\alpha}} \end{aligned}$$

which concludes the proof. □

3.4 Generating Gaussian random vectors

In this section we show, how to generate samples of a random vector $\mathbf{y}$ of $\mathbb{R}^n$ with an expectation $\bar{\mathbf{y}}$ and covariance matrix $\mathbf{\Gamma}_{\mathbf{y}}$. From Theorem 2, if $\mathbf{x}$ is a standard normal random vector (*i.e.*, $\bar{\mathbf{x}} = \mathbf{0}$

and $\mathbf{\Gamma_x} = \mathbf{I}$), we can generate a normal vector $\mathbf{y} : \mathcal{N}(\bar{\mathbf{y}}, \mathbf{\Gamma_y})$ as follows:

$$\left\{ \begin{array}{rcl} \mathbf{y} & = & \mathbf{Ax} + \mathbf{b} \\ \bar{\mathbf{y}} & = & \mathbf{b} \\ \mathbf{\Gamma_y} & = & \mathbf{A} \cdot \mathbf{A}^{\mathrm{T}} \end{array} \right.$$

by choosing $\mathbf{A} = \mathbf{\Gamma}_{\mathbf{y}}^{\frac{1}{2}}$ and $\mathbf{b} = \bar{\mathbf{y}}$. Indeed, we get that the random vector $\mathbf{y} = \mathbf{Ax} + \mathbf{b}$ will have an expectation of $\bar{\mathbf{y}}$ and a covariance matrix equal to $\mathbf{\Gamma_y}$ (see Figure 3.3). The right side of Figure 3.3 was thus obtained by the script:

```
1  N := 1000; Γ_y := [3 2; 2 3]; ȳ = [2; 3]; X := randn_{2×n}
2  Y := ȳ + Γ_y^{1/2} · X
3  plot(Y)
```

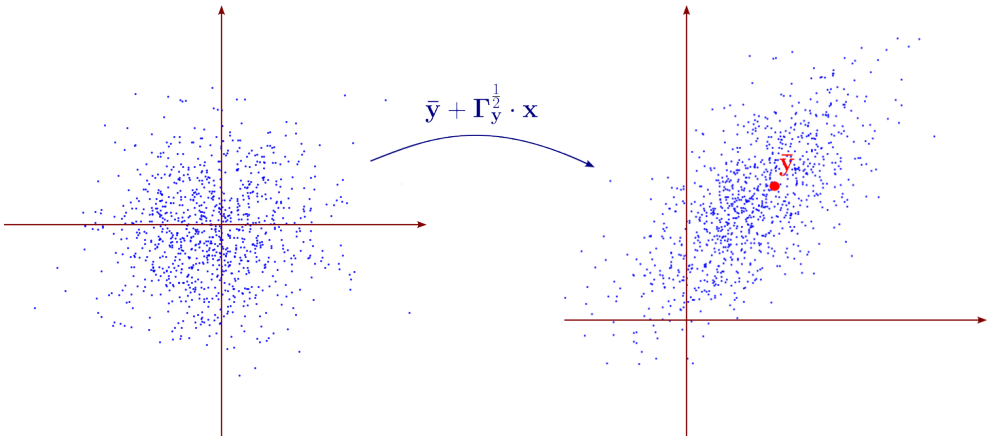

Figure 3.3: The Gaussian random vector $\mathbf{y} : (\bar{\mathbf{y}}, \mathbf{\Gamma_y})$ is the image by an affine application of a unit Gaussian white random vector $\mathbf{x}$

3.5 Confidence ellipse

We define the *confidence region* of the random vector $\mathbf{p}$ of $\mathbb{R}^n$ as the minimal volume set which contains $\mathbf{p}$ with a given probability, say η. If $\mathbf{p}$ is Gaussian, the probability distribution function can be fully characterized by the pair $(\bar{\mathbf{p}}, \mathbf{\Gamma_p})$. The associated confidence region is an ellipsoid of $\mathbb{R}^n$. In practice, for purely graphical reasons, we often only look at two components $\mathbf{y} = (p_i, p_j)$ of $\mathbf{p}$ (a computer screen is indeed two-dimensional). The average $\bar{\mathbf{y}}$ can be directly deduced from $\bar{\mathbf{p}}$ by extracting the i^{th} and j^{th} components. The covariance matrix $\mathbf{\Gamma_y} \in \mathcal{S}^+(\mathbb{R}^2)$ can also be obtained from $\mathbf{\Gamma_p} \in \mathcal{S}^+(\mathbb{R}^n)$ by extracting the i^{th} and j^{th} lines and columns. This is why in this section we consider a two dimensional Gaussian vector and we show how to get the confidence ellipse of probability η.

3.5.1 Standard normal random vector

Consider first the standard normal random vector $\mathbf{x}$ represented in Figure 3.3 left. Equivalently, $\mathbf{x}$ is a standard normal random vector. The confidence ellipse of probability η is a disk of radius a. To compute the relation between η and a, consider the random variable $z = \mathbf{x}^{\mathrm{T}}\mathbf{x}$. It follows a χ^2 law. In 2 dimensions, the corresponding probability density is given by:

$$\pi_z(z) = \begin{cases} \frac{1}{2}\exp\left(-\frac{z}{2}\right) & \text{if } z \geq 0 \\ 0 \text{ otherwise} \end{cases}$$

We have:

$$\begin{aligned} \eta = \text{prob}\left(\|\mathbf{x}\| \leq a\right) =& \text{prob}\left(\mathbf{x}^{\mathrm{T}}\mathbf{x} \leq a^2\right) = \text{prob}\left(z \leq a^2\right) \\ &= \int_0^{a^2} \frac{1}{2}\exp\left(-\frac{z}{2}\right) dz = 1 - e^{-\frac{1}{2}a^2}. \end{aligned}$$

Therefore:

$$a = \sqrt{-2\ln\left(1-\eta\right)}.$$

Figure 3.4 illustrates this relation. On the left figure, since $\eta = 0.5$, half of the samples are inside the disk. On the right, since $\eta = 0.9$, 90% of the samples are inside the disk. Note that when η increases, the radius a increases also.

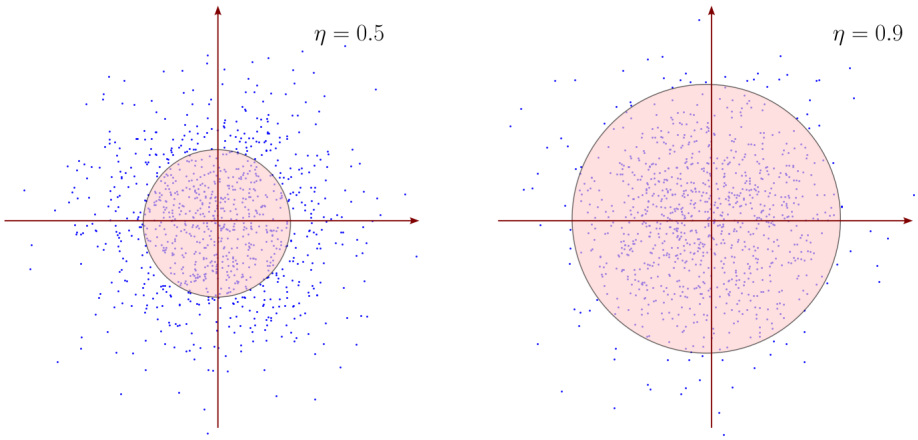

Figure 3.4: Left: confidence disk associated with the standard normal random vector $\mathbf{x}$ with $\eta = 0.5$. Right: the same with $\eta = 0.9$.

3.5.2 General case

Since a Gaussian vector random $\mathbf{y}$ can be generated a linear transformation $\mathbf{y} := \bar{\mathbf{y}} + \mathbf{\Gamma}_{\mathbf{y}}^{\frac{1}{2}} \cdot \mathbf{x}$ of a standard normal random vector $\mathbf{x}$, we can easily get the confidence ellipse. This is illustrated by

Figure 3.5 which reconsiders the example of Figure 3.3.

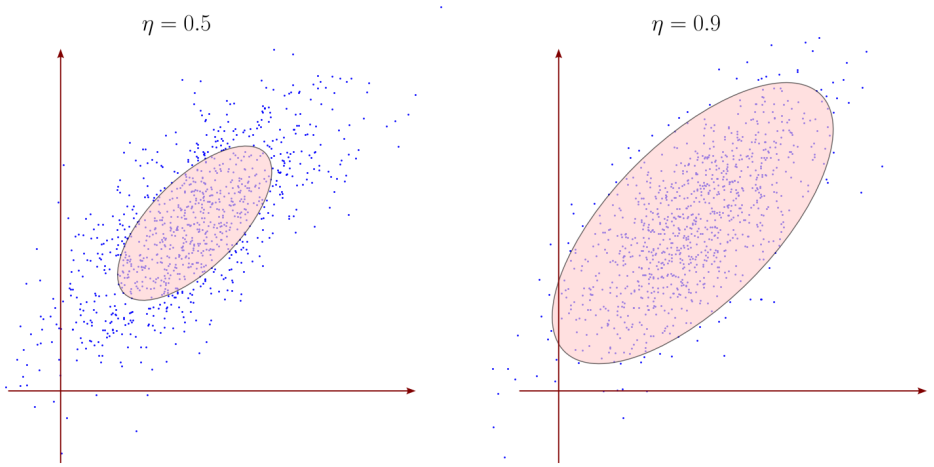

Figure 3.5: Left: confidence ellipse associated with $\mathbf{y}$ for $\eta = 0.5$. Right: the same for $\eta = 0.9$.

As a consequence, the *confidence ellipse* of probability η associated with the Gaussian vector $\mathbf{y}$ of $\mathbb{R}^2$ is described by:

$$\begin{aligned} \mathcal{E}_{\mathbf{y}} &= \{\mathbf{y} | \exists \mathbf{x}, \|\mathbf{x}\| \leq a, \mathbf{y} = \bar{\mathbf{y}} + \mathbf{\Gamma}_{\mathbf{y}}^{\frac{1}{2}} \cdot \mathbf{x}\} \\ &= \{\mathbf{y} | \exists \mathbf{s}, \|\mathbf{s}\| \leq 1, \mathbf{y} = \bar{\mathbf{y}} + \mathbf{\Gamma}_{\mathbf{y}}^{\frac{1}{2}} \cdot a\mathbf{s}\} \end{aligned}$$

where $a = \sqrt{-2\ln(1-\eta)}$. This means that, $\mathcal{E}_{\mathbf{y}}$ can be defined as the image of the unit disk by the affine function $\mathbf{s} \mapsto \mathbf{y} = \bar{\mathbf{y}} + \mathbf{\Gamma}_{\mathbf{y}}^{\frac{1}{2}} \cdot a\mathbf{s}$.

As illustrated by Figure 3.6, the following function draws the ellipse $\mathcal{E}_{\mathbf{y}}$ which contains the Gaussian random vector $\mathbf{y}$ with a given probability η.

Function DRAWELLIPSE $(\bar{\mathbf{y}}, \mathbf{\Gamma}_{\mathbf{y}}, \eta)$	
1	$\mathbf{s} = \begin{pmatrix} \cos 0 & \cos 0.1 & \cdots & \cos 2\pi \\ \sin 0 & \sin 0.1 & \cdots & \sin 2\pi \end{pmatrix}$
2	$\mathbf{y} = \bar{\mathbf{y}} + \mathbf{\Gamma}_{\mathbf{y}}^{\frac{1}{2}} \cdot \sqrt{-2\ln(1-\eta)} \cdot \mathbf{s}$
3	plot$(\mathbf{y})$

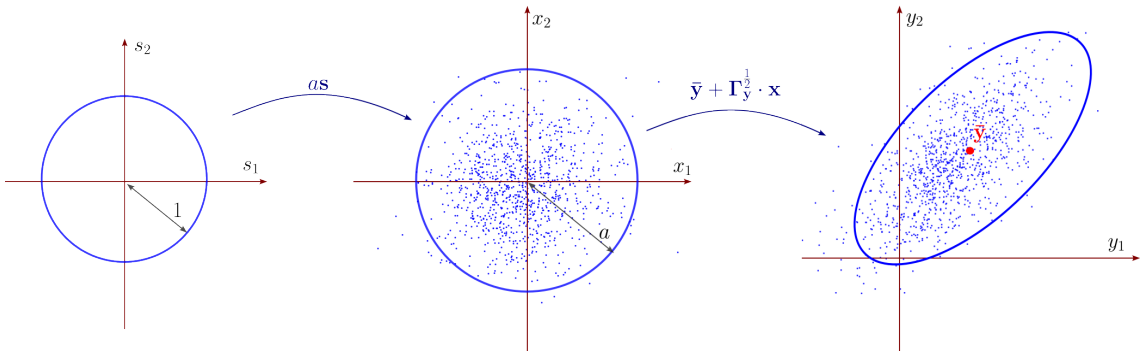

Figure 3.6: Procedure to draw a confidence ellipse of a normal random vector in $\mathbb{R}^2$

Note also that the ellipse $\mathcal{E}_{\mathbf{y}}$ can also be expresses as an inequality. Indeed,

$$\begin{aligned}
\mathcal{E}_{\mathbf{y}} &= \{\mathbf{y} | \exists \mathbf{x}, \|\mathbf{x}\| \leq a, \mathbf{x} = \mathbf{\Gamma}_{\mathbf{y}}^{-\frac{1}{2}} (\mathbf{y} - \bar{\mathbf{y}})\} \\
&= \{\mathbf{y} | \exists \mathbf{x}, \mathbf{x}^{\mathrm{T}} \mathbf{x} \leq a^2, \mathbf{x} = \mathbf{\Gamma}_{\mathbf{y}}^{-\frac{1}{2}} (\mathbf{y} - \bar{\mathbf{y}})\} \\
&= \{\mathbf{y} | (\mathbf{y} - \bar{\mathbf{y}})^{\mathrm{T}} \mathbf{\Gamma}_{\mathbf{y}}^{-\frac{1}{2}} \mathbf{\Gamma}_{\mathbf{y}}^{-\frac{1}{2}} (\mathbf{y} - \bar{\mathbf{y}}) \leq a^2\}
\end{aligned}$$

Thus

$$\mathcal{E}_{\mathbf{y}} : (\mathbf{y} - \bar{\mathbf{y}})^{\mathrm{T}} \mathbf{\Gamma}_{\mathbf{y}}^{-1} (\mathbf{y} - \bar{\mathbf{y}}) \leq -2 \ln (1 - \eta) .$$

Since $\mathbf{y}$ is a Gaussian random vector, this ellipse corresponds to a contour line of the probability density for $\mathbf{y}$.

Exercises

EXERCISE 8.– *Gaussian distribution*

See the correction video at https://youtu.be/IZBcA8xzH0c

The probability distribution of a random Gaussian vector $\mathbf{x}$ is fully characterized by its expectation $\bar{\mathbf{x}}$ and its covariance matrix $\mathbf{\Gamma_x}$. More precisely, it is given by:

$$\pi_{\mathbf{x}}(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^n \det(\mathbf{\Gamma_x})}} \cdot \exp\left(-\frac{1}{2}(\mathbf{x}-\bar{\mathbf{x}})^{\mathrm{T}} \cdot \mathbf{\Gamma_x}^{-1} \cdot (\mathbf{x}-\bar{\mathbf{x}})\right).$$

1) Draw the graph and the contour lines of $\pi_{\mathbf{x}}$ with:

$$\bar{\mathbf{x}} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \text{ and } \mathbf{\Gamma_x} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

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 9.– *Confidence ellipses*

See the correction video at https://youtu.be/jfQnFSmfKT8

Let us generate six covariance matrices as follows:

$$\mathbf{A}_1 = \begin{pmatrix} 1 & 0 \\ 0 & 3 \end{pmatrix} \qquad \mathbf{A}_2 = \begin{pmatrix} \cos\frac{\pi}{4} & -\sin\frac{\pi}{4} \\ \sin\frac{\pi}{4} & \cos\frac{\pi}{4} \end{pmatrix}$$

$$\mathbf{\Gamma}_1 = \mathbf{I}_{2\times 2} \qquad \mathbf{\Gamma}_2 = 3\mathbf{\Gamma}_1 \qquad \mathbf{\Gamma}_3 = \mathbf{A}_1 \cdot \mathbf{\Gamma}_2 \cdot \mathbf{A}_1^{\mathrm{T}} + \mathbf{\Gamma}_1$$

$$\mathbf{\Gamma}_4 = \mathbf{A}_2 \cdot \mathbf{\Gamma}_3 \cdot \mathbf{A}_2^{\mathrm{T}} \qquad \mathbf{\Gamma}_5 = \mathbf{\Gamma}_4 + \mathbf{\Gamma}_3 \qquad \mathbf{\Gamma}_6 = \mathbf{A}_2 \cdot \mathbf{\Gamma}_5 \cdot \mathbf{A}_2^{\mathrm{T}}$$

Here, $\mathbf{A}_2$ 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 3.7.

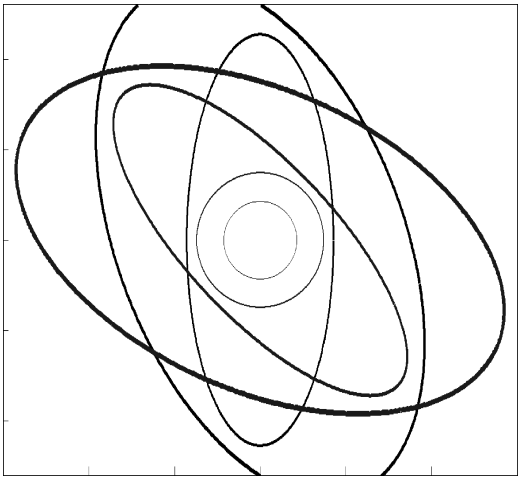

Figure 3.7: Confidence ellipses associated with the six covariance matrices

1) Associate each covariance matrix with its confidence ellipse on the figure.

2) Verify the result by regenerating these ellipses.

EXERCISE 10.– *Confidence ellipse: prediction*

See the correction video at https://youtu.be/yHG1GlvuZ7E

1) Generate a cloud of $N = 1000$ points representing a random Gaussian vector centered at $\mathbb{R}^2$ whose covariance matrix is the identity matrix. Deduce from the latter a cloud of points for the random vector $\mathbf{x}$ such that:

$$\bar{\mathbf{x}} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \text{ and } \mathbf{\Gamma}_{\mathbf{x}} = \begin{pmatrix} 4 & 3 \\ 3 & 3 \end{pmatrix}$$

Use a two-lines, n-columns matrix $\mathbf{X}$ to store the cloud.

2) Assuming we know $\bar{\mathbf{x}}$ and $\mathbf{\Gamma}_{\mathbf{x}}$ draw the confidence ellipses for the probabilities $\eta \in \{0.9, 0.99, 0.999\}$. Show graphically the consistency with the cloud $\mathbf{X}$.

3) Find an estimation of $\bar{\mathbf{x}}$ and $\mathbf{\Gamma}_{\mathbf{x}}$ from the cloud $\mathbf{X}$.

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 $dt = 0.01$. For the discretization, we take an Euler

integration:

$$\mathbf{x}(k+1) = \mathbf{x}(k) + dt \left(\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \mathbf{x} + \begin{pmatrix} 2 \\ 3 \end{pmatrix} u(k) \right) + \boldsymbol{\alpha}(k)$$

where $\boldsymbol{\alpha}(k) : \mathcal{N}(\mathbf{0}, dt \cdot \mathbf{I}_2)$ is a white Gaussian noise. Draw the cloud for $t \in \{0, 1, 2, 3, 4, 5\}$.

5) Represent this evolution using only a Kalman prediction. Compare the resulting confidence ellipses with the cloud computed at Question 4.

Exercise 11.– *Brownian noise*

See the correction video at https://youtu.be/mFzoloe8hT8

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 σ_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 δ and in function of t_k ? Discuss. Validate the result with a simulation.

2) We now tend δ towards 0. What standard deviation σ_x do we have to choose in function of δ in order for the variances $\sigma_y^2(t_k)$ to remain unchanged ? Illustrate this program that generates Brownian noises $y(t)$ that are insensitive to sampling period changes.

Chapter 4

Unbiased orthogonal estimator

Let us consider two random vectors $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{y} \in \mathbb{R}^m$. The vector $\mathbf{y}$ corresponds to the measurement vector which is a random vector. Later, when the measurements will have been made, it will become deterministic. 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 $\mathbf{y}$. In this chapter, we propose a linear estimator which provides a minimal covariance for the estimation error.

4.1 Linear estimator

Figure 4.1 shows a nonlinear estimator corresponding to $E(x|y)$.

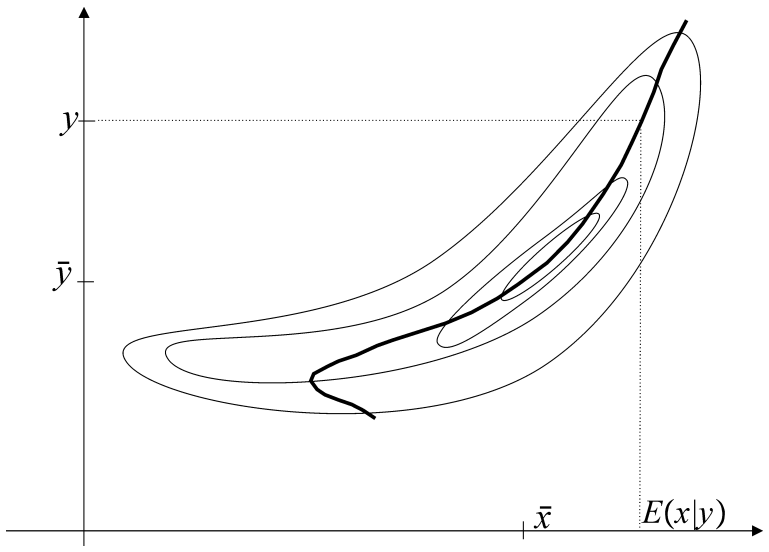

Figure 4.1: Nonlinear estimator $E(x|y)$

However, obtaining an analytic expression for such an estimator is 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:

$$\hat{\mathbf{x}} = \mathbf{K}\mathbf{y} + \mathbf{b} \tag{4.1}$$

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:

$$\boldsymbol{\varepsilon} = \hat{\mathbf{x}} - \mathbf{x}$$

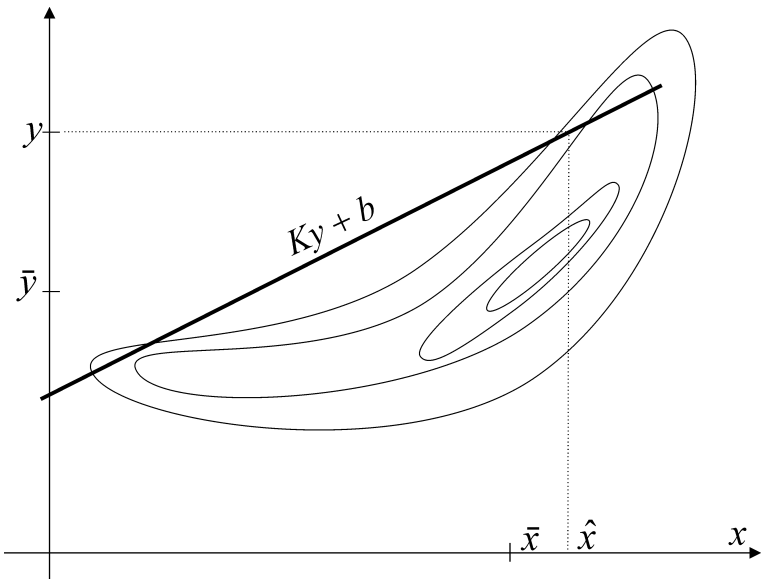

Figure 4.2: Biased linear estimator

The estimator is said to be *unbiased* if $E(\boldsymbol{\varepsilon}) = \mathbf{0}$. It is *orthogonal* if $E\left(\boldsymbol{\varepsilon}\widetilde{\mathbf{y}}^{\mathrm{T}}\right) = \mathbf{0}$. 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 $\widetilde{\mathbf{y}}$ are vectors), we say that the two random vectors $\mathbf{a}$ and $\mathbf{b}$ are orthogonal if their components are, *i.e.,* $E\left((a_i - \bar{a}_i)\left(b_j - \bar{b}_j\right)\right) = 0$ 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 4.2 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, *i.e.,* the probability to have $\hat{x} < x$ is high, or even that $E(\varepsilon) < 0$. The estimator is thus biased. Figure 4.3 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 $\widetilde{y} > 0$, ε tends to be negative whereas if $\widetilde{y} < 0$, ε tends to be positive).

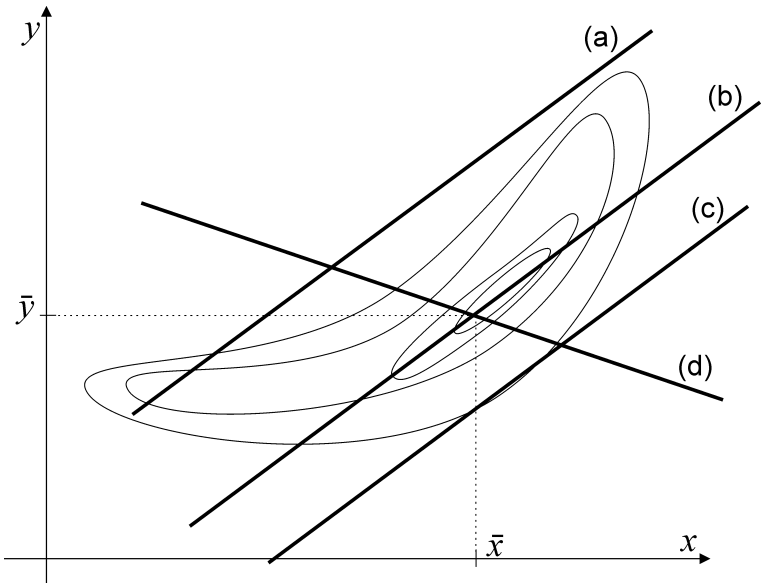

Figure 4.3: Among these four linear estimators, estimator (b), which is unbiased and orthogonal, seems to be the best

Theorem 3. *Consider two random vectors* $\mathbf{x}$ *and* $\mathbf{y}$*. A unique unbiased orthogonal estimator exists. It is given by:*

$$\hat{\mathbf{x}} = \bar{\mathbf{x}} + \mathbf{K} \cdot (\mathbf{y} - \bar{\mathbf{y}}) \tag{4.2}$$

where:

$$\mathbf{K} = \mathbf{\Gamma}_{\mathbf{xy}} \mathbf{\Gamma}_{\mathbf{y}}^{-1} \tag{4.3}$$

is referred to as the Kalman gain.

Example. Let us consider once again the example in Section 3.1. 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 4.4.

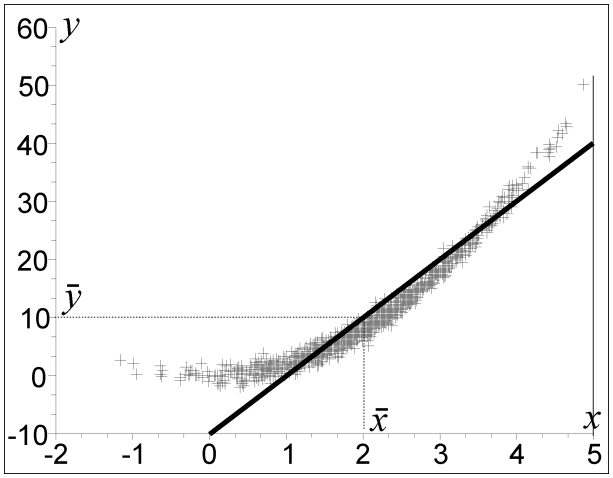

Figure 4.4: Unbiased orthogonal linear estimator

Proof. We have:

$$\begin{array}{rcl} E(\boldsymbol{\varepsilon}) & = & E(\hat{\mathbf{x}}-\mathbf{x}) \\ & \overset{(4.1)}{=} & E(\mathbf{K}\mathbf{y}+\mathbf{b}-\mathbf{x}) \\ & = & \mathbf{K}E(\mathbf{y})+\mathbf{b}-E(\mathbf{x}) \\ & = & \mathbf{K}\bar{\mathbf{y}}+\mathbf{b}-\bar{\mathbf{x}} \end{array}$$

The estimator is unbiased if $E(\boldsymbol{\varepsilon})=\mathbf{0}$, *i.e.*:

$$\mathbf{b}=\bar{\mathbf{x}}-\mathbf{K}\bar{\mathbf{y}}, \tag{4.4}$$

which gives us (4.2). In this case:

$$\boldsymbol{\varepsilon}=\hat{\mathbf{x}}-\mathbf{x}\overset{(4.2)}{=}\bar{\mathbf{x}}+\mathbf{K}\cdot(\mathbf{y}-\bar{\mathbf{y}})-\mathbf{x}=\mathbf{K}\widetilde{\mathbf{y}}-\widetilde{\mathbf{x}}. \tag{4.5}$$

The estimator is orthogonal if:

$$\begin{array}{rcl} E\left(\boldsymbol{\varepsilon}\cdot\widetilde{\mathbf{y}}^{\mathrm{T}}\right)=\mathbf{0} & \overset{(4.5)}{\Leftrightarrow} & E\left((\mathbf{K}\widetilde{\mathbf{y}}-\widetilde{\mathbf{x}})\cdot\widetilde{\mathbf{y}}^{\mathrm{T}}\right)=\mathbf{0} \\ & \Leftrightarrow & E\left(\mathbf{K}\widetilde{\mathbf{y}}\widetilde{\mathbf{y}}^{\mathrm{T}}-\widetilde{\mathbf{x}}\widetilde{\mathbf{y}}^{\mathrm{T}}\right)=\mathbf{0} \\ & \Leftrightarrow & \mathbf{K}\boldsymbol{\Gamma}_{\mathbf{y}}-\boldsymbol{\Gamma}_{\mathbf{xy}}=\mathbf{0} \\ & \Leftrightarrow & \mathbf{K}=\boldsymbol{\Gamma}_{\mathbf{xy}}\cdot\boldsymbol{\Gamma}_{\mathbf{y}}^{-1} \end{array}$$

which concludes the proof. □

Theorem 4. *The covariance matrix of the error associated with the unbiased orthogonal linear estimator is:*

$$\boldsymbol{\Gamma}_{\boldsymbol{\varepsilon}}=\boldsymbol{\Gamma}_{\mathbf{x}}-\mathbf{K}\cdot\boldsymbol{\Gamma}_{\mathbf{yx}}. \tag{4.6}$$

Proof. The covariance matrix of $\boldsymbol{\varepsilon}$ in the unbiased case is written as:

$$\begin{aligned}\boldsymbol{\Gamma}_{\boldsymbol{\varepsilon}} &= E\left(\boldsymbol{\varepsilon}\cdot\boldsymbol{\varepsilon}^{\mathrm{T}}\right) \overset{(4.5)}{=} E\left((\mathbf{K}\widetilde{\mathbf{y}}-\widetilde{\mathbf{x}})\cdot(\mathbf{K}\widetilde{\mathbf{y}}-\widetilde{\mathbf{x}})^{\mathrm{T}}\right)\\ &= E\left((\mathbf{K}\widetilde{\mathbf{y}}-\widetilde{\mathbf{x}})\cdot(\widetilde{\mathbf{y}}^{\mathrm{T}}\mathbf{K}^{\mathrm{T}}-\widetilde{\mathbf{x}}^{\mathrm{T}})\right)\\ &= E\left(\mathbf{K}\widetilde{\mathbf{y}}\widetilde{\mathbf{y}}^{\mathrm{T}}\mathbf{K}^{\mathrm{T}}-\widetilde{\mathbf{x}}\widetilde{\mathbf{y}}^{\mathrm{T}}\mathbf{K}^{\mathrm{T}}-\mathbf{K}\widetilde{\mathbf{y}}\widetilde{\mathbf{x}}^{\mathrm{T}}+\widetilde{\mathbf{x}}\widetilde{\mathbf{x}}^{\mathrm{T}}\right).\end{aligned}$$

Using the linearity of the expectation operator, we get

$$\boldsymbol{\Gamma}_{\boldsymbol{\varepsilon}} = \underbrace{(\mathbf{K}\boldsymbol{\Gamma}_{\mathbf{y}}-\boldsymbol{\Gamma}_{\mathbf{xy}})}_{\overset{(4.3)}{=}\mathbf{0}}\mathbf{K}^{\mathrm{T}}-\mathbf{K}\boldsymbol{\Gamma}_{\mathbf{yx}}+\boldsymbol{\Gamma}_{\mathbf{x}}. \tag{4.7}$$

□

4.2 Optimality

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, 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 5. *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 (4.7), the covariance matrix for the error is:

$$\begin{aligned}\boldsymbol{\Gamma}_{\boldsymbol{\varepsilon}} &= ((\mathbf{K}_0+\boldsymbol{\Delta})\boldsymbol{\Gamma}_{\mathbf{y}}-\boldsymbol{\Gamma}_{\mathbf{xy}})(\mathbf{K}_0+\boldsymbol{\Delta})^{\mathrm{T}}-(\mathbf{K}_0+\boldsymbol{\Delta})\boldsymbol{\Gamma}_{\mathbf{yx}}+\boldsymbol{\Gamma}_{\mathbf{x}}\\ &= (\mathbf{K}_0+\boldsymbol{\Delta})(\underbrace{\boldsymbol{\Gamma}_{\mathbf{y}}\mathbf{K}_0^{\mathrm{T}}}_{=\boldsymbol{\Gamma}_{\mathbf{yx}}}+\boldsymbol{\Gamma}_{\mathbf{y}}\boldsymbol{\Delta}^{\mathrm{T}})-(\underbrace{\boldsymbol{\Gamma}_{\mathbf{xy}}\mathbf{K}_0^{\mathrm{T}}}_{=\mathbf{K}_0\boldsymbol{\Gamma}_{\mathbf{yx}}}+\boldsymbol{\Gamma}_{\mathbf{xy}}\boldsymbol{\Delta}^{\mathrm{T}})-(\mathbf{K}_0\boldsymbol{\Gamma}_{\mathbf{yx}}+\boldsymbol{\Delta}\boldsymbol{\Gamma}_{\mathbf{yx}})+\boldsymbol{\Gamma}_{\mathbf{x}}\\ &= \mathbf{K}_0\boldsymbol{\Gamma}_{\mathbf{yx}}+\boldsymbol{\Delta}\boldsymbol{\Gamma}_{\mathbf{yx}}+\underbrace{\mathbf{K}_0\boldsymbol{\Gamma}_{\mathbf{y}}}_{=\boldsymbol{\Gamma}_{\mathbf{xy}}}\boldsymbol{\Delta}^{\mathrm{T}}+\boldsymbol{\Delta}\boldsymbol{\Gamma}_{\mathbf{y}}\boldsymbol{\Delta}^{\mathrm{T}}-\mathbf{K}_0\boldsymbol{\Gamma}_{\mathbf{yx}}-\boldsymbol{\Gamma}_{\mathbf{xy}}\boldsymbol{\Delta}^{\mathrm{T}}-\mathbf{K}_0\boldsymbol{\Gamma}_{\mathbf{yx}}-\boldsymbol{\Delta}\boldsymbol{\Gamma}_{\mathbf{yx}}+\boldsymbol{\Gamma}_{\mathbf{x}}\\ &= -\mathbf{K}_0\boldsymbol{\Gamma}_{\mathbf{yx}}+\boldsymbol{\Delta}\boldsymbol{\Gamma}_{\mathbf{y}}\boldsymbol{\Delta}^{\mathrm{T}}+\boldsymbol{\Gamma}_{\mathbf{x}}.\end{aligned}$$

□

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 $\mathbf{K} = \boldsymbol{\Gamma}_{\mathbf{xy}}\boldsymbol{\Gamma}_{\mathbf{y}}^{-1}$, which corresponds to the orthogonal unbiased estimator.

4.3 Application to linear estimation

Let us assume that $\mathbf{x}$ and $\mathbf{y}$ are related 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 $\boldsymbol{\Gamma}_{\mathbf{x}}$ and $\boldsymbol{\Gamma}_{\boldsymbol{\beta}}$. Let us use the results obtained in the previous section in order to find the best unbiased linear estimator for $\mathbf{x}$ (refer to [5] for more details on linear estimation).

We have:

$$\begin{array}{rcl}
\bar{\mathbf{y}} & = & \mathbf{C}\bar{\mathbf{x}} + \bar{\boldsymbol{\beta}} = \mathbf{C}\bar{\mathbf{x}} \\
\boldsymbol{\Gamma}_{\mathbf{y}} & \overset{(3.1)}{=} & \mathbf{C}\boldsymbol{\Gamma}_{\mathbf{x}}\mathbf{C}^{\mathrm{T}} + \boldsymbol{\Gamma}_{\boldsymbol{\beta}} \\
\boldsymbol{\Gamma}_{\mathbf{xy}} & = & E\left(\widetilde{\mathbf{x}} \cdot \widetilde{\mathbf{y}}^{\mathrm{T}}\right) = E\left(\widetilde{\mathbf{x}} \cdot \left(\mathbf{C}\widetilde{\mathbf{x}} + \widetilde{\boldsymbol{\beta}}\right)^{\mathrm{T}}\right) = E\left(\widetilde{\mathbf{x}} \cdot \widetilde{\mathbf{x}}^{\mathrm{T}}\mathbf{C}^{\mathrm{T}} + \widetilde{\mathbf{x}} \cdot \widetilde{\boldsymbol{\beta}}^{\mathrm{T}}\right) \\
& = & E\left(\widetilde{\mathbf{x}} \cdot \widetilde{\mathbf{x}}^{\mathrm{T}}\right)\mathbf{C}^{\mathrm{T}} + \underbrace{E\left(\widetilde{\mathbf{x}} \cdot \widetilde{\boldsymbol{\beta}}^{\mathrm{T}}\right)}_{=\,\mathbf{0}} = \boldsymbol{\Gamma}_{\mathbf{x}}\mathbf{C}^{\mathrm{T}}.
\end{array} \tag{4.8}$$

Consequently, the best unbiased estimator for $\mathbf{x}$ and covariance matrix of the error can be obtained from $\boldsymbol{\Gamma}_{\mathbf{x}}, \boldsymbol{\Gamma}_{\boldsymbol{\beta}}, \mathbf{C}, \bar{\mathbf{x}}$ by using the following formulas:

$$\begin{array}{llcll}
\text{(i)} & \hat{\mathbf{x}} & \overset{(4.2)}{=} & \bar{\mathbf{x}} + \mathbf{K}\widetilde{\mathbf{y}} & \text{(estimation)} \\
\text{(ii)} & \boldsymbol{\Gamma}_{\varepsilon} & \overset{(4.6)}{=} & \boldsymbol{\Gamma}_{\mathbf{x}} - \mathbf{K}\mathbf{C}\boldsymbol{\Gamma}_{\mathbf{x}} & \text{(covariance of the error)} \\
\text{(iii)} & \widetilde{\mathbf{y}} & \overset{(4.8)}{=} & \mathbf{y} - \mathbf{C}\bar{\mathbf{x}} & \text{(innovation)} \\
\text{(iv)} & \boldsymbol{\Gamma}_{\mathbf{y}} & \overset{(4.8)}{=} & \mathbf{C}\boldsymbol{\Gamma}_{\mathbf{x}}\mathbf{C}^{\mathrm{T}} + \boldsymbol{\Gamma}_{\boldsymbol{\beta}} & \text{(covariance of the innovation)} \\
\text{(v)} & \mathbf{K} & \overset{(4.3,4.8)}{=} & \boldsymbol{\Gamma}_{\mathbf{x}}\mathbf{C}^{\mathrm{T}}\boldsymbol{\Gamma}_{\mathbf{y}}^{-1} & \text{(Kalman gain)}
\end{array} \tag{4.9}$$

Remark. Figure 4.1 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 (4.9):

$$\begin{array}{rcl}
E(\mathbf{x}|\mathbf{y}) & = & \bar{\mathbf{x}} + \boldsymbol{\Gamma}_{\mathbf{xy}}\boldsymbol{\Gamma}_{\mathbf{y}}^{-1}(\mathbf{y} - \bar{\mathbf{y}}) \\
E\left(\varepsilon \cdot \varepsilon^{\mathrm{T}}|\mathbf{y}\right) & = & E\left((\hat{\mathbf{x}} - \mathbf{x})(\hat{\mathbf{x}} - \mathbf{x})^{\mathrm{T}}|\mathbf{y}\right) \\
& = & \boldsymbol{\Gamma}_{\mathbf{x}} - \mathbf{K}\mathbf{C}\boldsymbol{\Gamma}_{\mathbf{x}} \\
& = & \boldsymbol{\Gamma}_{\mathbf{x}} - \boldsymbol{\Gamma}_{\mathbf{xy}}\boldsymbol{\Gamma}_{\mathbf{y}}^{-1}\boldsymbol{\Gamma}_{\mathbf{yx}}
\end{array}$$

Exercises

EXERCISE 12.– *Confidence ellipse: correction*

See the correction video at https://youtu.be/Rmx7nIEwpBg

1) Generate a Gaussian point cloud of $n = 1\ 000$ points associated with the random vector $\mathbf{x}$ with:

$$\bar{\mathbf{x}} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \text{ and } \mathbf{\Gamma_x} = \begin{pmatrix} 3 & 1 \\ 1 & 3 \end{pmatrix}.$$

Use a two-line, n-column matrix to store the clouds.

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 13.– *Covariance matrix propagation*

See the correction video at https://youtu.be/e0iRudPE8-Y

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 that $\mathbf{x}$ and $\mathbf{y}$ have 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 14.– *Solving three equations using a linear estimator*

See the correction video at https://youtu.be/Rst52D2GXM0

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 ε_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 15.– *Estimating the parameters of an electric motor using a linear estimator*

See the correction video at https://youtu.be/vCCbSvANT7M

Let us consider a DC motor whose parameters have been estimated with a least squares method (see Exercise 3). Recall that in that example, the angular speed Ω 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
Ω(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 16.– *Trochoid*

See the correction video at https://youtu.be/tzkliJQQj7g

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.

1) Using an unbiased orthogonal filter, compute an estimation for p_1 and p_2.

2) Draw the estimated path of the mass.

Chapter 5

Kalman filter

When we control a system, such as a robot, we generally assume 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 6 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 [6] 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.

5.1 Equations of the Kalman filter

This paragraph presents the Kalman filter (refer to [7] for more details). 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 $\mathbf{\Gamma}_{k|k}$ associated to $\mathbf{x}_{k|k}$ are given by Equations (4.9). We therefore have:

$$\begin{array}{lllll}
\text{(i)} & \hat{\mathbf{x}}_{k|k} & = & \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \cdot \widetilde{\mathbf{y}}_k & \text{(corrected estimation)} \\
\text{(ii)} & \mathbf{\Gamma}_{k|k} & = & \mathbf{\Gamma}_{k|k-1} - \mathbf{K}_k \cdot \mathbf{C}_k \mathbf{\Gamma}_{k|k-1} & \text{(corrected covariance)} \\
\text{(iii)} & \widetilde{\mathbf{y}}_k & = & \mathbf{y}_k - \mathbf{C}_k \hat{\mathbf{x}}_{k|k-1} & \text{(innovation)} \\
\text{(iv)} & \mathbf{S}_k & = & \mathbf{C}_k \mathbf{\Gamma}_{k|k-1} \mathbf{C}_k^{\mathrm{T}} + \mathbf{\Gamma}_{\boldsymbol{\beta}_k} & \text{(covariance of the innovation)} \\
\text{(v)} & \mathbf{K}_k & = & \mathbf{\Gamma}_{k|k-1} \mathbf{C}_k^{\mathrm{T}} \mathbf{S}_k^{-1} & \text{(Kalman gain)}
\end{array} \tag{5.1}$$

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 $\mathbf{\Gamma}_{k+1|k}$. Since

$$\mathbf{x}_{k+1} = \mathbf{A}_k \mathbf{x}_k + \mathbf{u}_k + \boldsymbol{\alpha}_k,$$

we have, following (3.1):

$$\hat{\mathbf{x}}_{k+1|k} = \mathbf{A}_k \hat{\mathbf{x}}_{k|k} + \mathbf{u}_k \tag{5.2}$$

and

$$\mathbf{\Gamma}_{k+1|k} = \mathbf{A}_k \cdot \mathbf{\Gamma}_{k|k} \cdot \mathbf{A}_k^{\mathrm{T}} + \mathbf{\Gamma}_{\boldsymbol{\alpha}_k}. \tag{5.3}$$

Kalman filter. The complete Kalman filter is given by the following equations :

$$\begin{array}{lll}
\hat{\mathbf{x}}_{k+1|k} \overset{(5.2)}{=} & \mathbf{A}_k \hat{\mathbf{x}}_{k|k} + \mathbf{u}_k & \text{(predicted estimation)} \\
\mathbf{\Gamma}_{k+1|k} \overset{(5.3)}{=} & \mathbf{A}_k \cdot \mathbf{\Gamma}_{k|k} \cdot \mathbf{A}_k^{\mathrm{T}} + \mathbf{\Gamma}_{\boldsymbol{\alpha}_k} & \text{(predicted covariance)} \\
\hat{\mathbf{x}}_{k|k} \overset{(4.9,\mathrm{i})}{=} & \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \cdot \widetilde{\mathbf{y}}_k & \text{(corrected estimation)} \\
\mathbf{\Gamma}_{k|k} \overset{(4.9,\mathrm{ii})}{=} & (\mathbf{I} - \mathbf{K}_k \mathbf{C}_k)\, \mathbf{\Gamma}_{k|k-1} & \text{(corrected covariance)} \\
\widetilde{\mathbf{y}}_k \overset{(4.9,\mathrm{iii})}{=} & \mathbf{y}_k - \mathbf{C}_k \hat{\mathbf{x}}_{k|k-1} & \text{(innovation)} \\
\mathbf{S}_k \overset{(4.9,\mathrm{iv})}{=} & \mathbf{C}_k \mathbf{\Gamma}_{k|k-1} \mathbf{C}_k^{\mathrm{T}} + \mathbf{\Gamma}_{\boldsymbol{\beta}_k} & \text{(covariance of the innovation)} \\
\mathbf{K}_k \overset{(4.9,\mathrm{v})}{=} & \mathbf{\Gamma}_{k|k-1} \mathbf{C}_k^{\mathrm{T}} \mathbf{S}_k^{-1}. & \text{(Kalman gain)}
\end{array}$$

Figure 5.1 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.

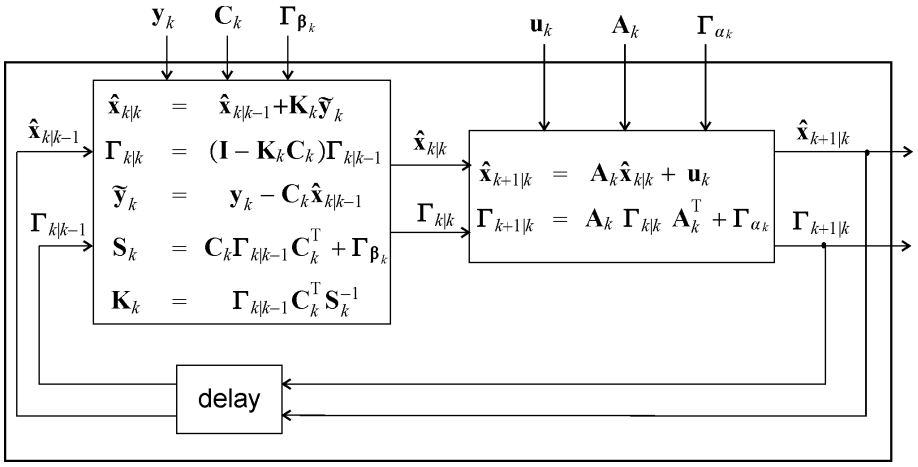

Figure 5.1: The Kalman filter is composed of a corrector followed by a predictor

The function KALMAN below implements the Kalman filter and returns $\hat{\mathbf{x}}_{k+1|k}, \mathbf{\Gamma}_{k+1|k}$. In this program, we have the following correspondences: $\hat{\mathbf{x}}^{\mathrm{pred}} \leftrightarrow \hat{\mathbf{x}}_{k|k-1}$, $\mathbf{\Gamma}^{\mathrm{pred}} \leftrightarrow \mathbf{\Gamma}_{k|k-1}$, $\hat{\mathbf{x}}^{\mathrm{up}} \leftrightarrow \hat{\mathbf{x}}_{k|k}$, $\mathbf{\Gamma}^{\mathrm{up}} \leftrightarrow \mathbf{\Gamma}_{k|k}$ (the term `up` refers to *update*, *i.e.*, correction).

	Function KALMAN $(\hat{\mathbf{x}}^{\mathrm{pred}}, \mathbf{\Gamma}^{\mathrm{pred}}, \mathbf{u}, \mathbf{y}, \mathbf{\Gamma}_{\boldsymbol{\alpha}}, \mathbf{\Gamma}_{\boldsymbol{\beta}}, \mathbf{A}, \mathbf{C})$
1	$\mathbf{S} := \mathbf{C} \cdot \mathbf{\Gamma}^{\mathrm{pred}} \cdot \mathbf{C}^{\mathrm{T}} + \mathbf{\Gamma}_{\boldsymbol{\beta}}$
2	$\mathbf{K} := \mathbf{\Gamma}^{\mathrm{pred}} \cdot \mathbf{C}^{\mathrm{T}} \cdot \mathbf{S}^{-1}$
3	$\widetilde{\mathbf{y}} := \mathbf{y} - \mathbf{C} \cdot \hat{\mathbf{x}}^{\mathrm{pred}}$
4	$\hat{\mathbf{x}}^{\mathrm{up}} := \hat{\mathbf{x}}^{\mathrm{pred}} + \mathbf{K} \cdot \widetilde{\mathbf{y}}$
5	$\mathbf{\Gamma}^{\mathrm{up}} := (\mathbf{I} - \mathbf{K} \cdot \mathbf{C})\, \mathbf{\Gamma}^{\mathrm{pred}}$
6	$\hat{\mathbf{x}}^{\mathrm{pred}} := \mathbf{A} \cdot \hat{\mathbf{x}}^{\mathrm{up}} + \mathbf{u}$
7	$\mathbf{\Gamma}^{\mathrm{pred}} := \mathbf{A} \cdot \mathbf{\Gamma}^{\mathrm{up}} \cdot \mathbf{A}^{\mathrm{T}} + \mathbf{\Gamma}_{\boldsymbol{\alpha}}$
8	Return$\left(\hat{\mathbf{x}}^{\mathrm{pred}}, \mathbf{\Gamma}^{\mathrm{pred}}\right)$

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 :

$$\mathbf{\Gamma}_{k|k} := \sqrt{(\mathbf{I} - \mathbf{K}_k \mathbf{C}_k)\, \mathbf{\Gamma}_{k|k-1} \mathbf{\Gamma}_{k|k-1}^{\mathrm{T}}\, (\mathbf{I} - \mathbf{K}_k \mathbf{C}_k)^{\mathrm{T}}}$$

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, $\mathbf{y}$, $\mathbf{\Gamma}_{\boldsymbol{\beta}}, \mathbf{C}$ have to become empty quantities. However, they have to have correct dimensions in order to perform matrix operations.

Initialization : Most of the time, we have no idea of the initial state $\mathbf{x}_0$. In this case, we generally set

$$\hat{\mathbf{x}}_0 = (0, 0, \ldots, 0) \text{ et } \mathbf{\Gamma}_{\mathbf{x}}(0) = \begin{pmatrix} \frac{1}{\varepsilon^2} & 0 & 0 & \\ 0 & \frac{1}{\varepsilon^2} & & \\ 0 & & \ddots & 0 \\ & & 0 & \frac{1}{\varepsilon^2} \end{pmatrix},$$

where ε 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 $\frac{1}{\varepsilon}$.

Stationary 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:

$$\begin{array}{rcl} \mathbf{\Gamma}_{k+1|k} & = & \mathbf{A} \cdot \mathbf{\Gamma}_{k|k} \cdot \mathbf{A}^{\mathrm{T}} + \mathbf{\Gamma}_{\boldsymbol{\alpha}} \\ & = & \mathbf{A} \cdot (\mathbf{I} - \mathbf{K}_k \mathbf{C}) \, \mathbf{\Gamma}_{k|k-1} \cdot \mathbf{A}^{\mathrm{T}} + \mathbf{\Gamma}_{\boldsymbol{\alpha}} \\ & = & \mathbf{A} \cdot \left(\mathbf{I} - \left(\mathbf{\Gamma}_{k|k-1} \mathbf{C}^{\mathrm{T}} \mathbf{S}^{-1}\right) \mathbf{C}\right) \mathbf{\Gamma}_{k|k-1} \cdot \mathbf{A}^{\mathrm{T}} + \mathbf{\Gamma}_{\boldsymbol{\alpha}} \\ & = & \mathbf{A} \cdot \left(\mathbf{I} - \left(\mathbf{\Gamma}_{k|k-1} \mathbf{C}^{\mathrm{T}} \left(\mathbf{C} \mathbf{\Gamma}_{k|k-1} \mathbf{C}^{\mathrm{T}} + \mathbf{\Gamma}_{\boldsymbol{\beta}}\right)^{-1}\right) \mathbf{C}\right) \mathbf{\Gamma}_{k|k-1} \cdot \mathbf{A}^{\mathrm{T}} + \mathbf{\Gamma}_{\boldsymbol{\alpha}} \end{array}$$

Therefore, for large k,

$$\mathbf{\Gamma} = \mathbf{A} \cdot \left(\mathbf{\Gamma} - \mathbf{\Gamma} \cdot \mathbf{C}^{\mathrm{T}} \cdot \left(\mathbf{C} \cdot \mathbf{\Gamma} \cdot \mathbf{C}^{\mathrm{T}} + \mathbf{\Gamma}_{\boldsymbol{\beta}}\right)^{-1} \cdot \mathbf{C} \cdot \mathbf{\Gamma}\right) \cdot \mathbf{A}^{\mathrm{T}} + \mathbf{\Gamma}_{\boldsymbol{\alpha}}$$

which is an equation of Ricatti in $\mathbf{\Gamma}$. It can be solved by the sequence

$$\mathbf{\Gamma}(k+1) = \mathbf{A} \cdot \left(\mathbf{\Gamma}(k) - \mathbf{\Gamma}(k) \cdot \mathbf{C}^{\mathrm{T}} \cdot \left(\mathbf{C} \cdot \mathbf{\Gamma}(k) \cdot \mathbf{C}^{\mathrm{T}} + \mathbf{\Gamma}_{\boldsymbol{\beta}}\right)^{-1} \cdot \mathbf{C} \cdot \mathbf{\Gamma}(k)\right) \cdot \mathbf{A}^{\mathrm{T}} + \mathbf{\Gamma}_{\boldsymbol{\alpha}}$$

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 $\mathbf{K} = \mathbf{\Gamma}\mathbf{C}^{\mathrm{T}} \left(\mathbf{C}\mathbf{\Gamma}\mathbf{C}^{\mathrm{T}} + \mathbf{\Gamma}\right)^{-1}$.

Exercises

Exercise 17.– *Solving three equations using a Kalman filter*

See the correction video at https://youtu.be/Z8x7b1Owdko

Let us consider once again the linear equations of Exercise 14:

$$\begin{cases} 2x_1 + 3x_2 &= 8 + \beta_1 \\ 3x_1 + 2x_2 &= 7 + \beta_2 \\ x_1 - x_2 &= 0 + \beta_3 \end{cases}$$

where $\beta_1, \beta_2, \beta_3$ are three independent, centered random variables with respective variances $1, 4, 4$.

1) Solve this system 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 14.

Exercise 18.– *Three-step Kalman filter*

See the correction video at https://youtu.be/beIEAc2QI4s

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 β_k are white Gaussian signals with a unitary covariance matrix. We have

$$\boldsymbol{\Gamma}_{\boldsymbol{\alpha}} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \text{ and } \boldsymbol{\Gamma}_{\beta} = 1.$$

The initial state vector is unknown and is represented by an estimation $\hat{\mathbf{x}}_{0|-1}$ and a covariance matrix $\boldsymbol{\Gamma}_{0|-1}$. We will take:

$$\hat{\mathbf{x}}_{0|-1} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \ \boldsymbol{\Gamma}_{0|-1} = \begin{pmatrix} 100 & 0 \\ 0 & 100 \end{pmatrix}.$$

Draw the $\eta = 0.9$ confidence ellipses with center $\hat{\mathbf{x}}_{k|k}$ and covariance matrix $\boldsymbol{\Gamma}_{k|k}$ obtained by the Kalman filter.

EXERCISE 19.– *Estimating the parameters of an electric motor*

See the correction video at https://youtu.be/_05sBBiw3ac

Let us consider once more the DC motor with angular speed Ω (see Exercises 3 to 15). 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
Ω(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 20.– *Temperature estimation*

See the correction video at https://youtu.be/6sTAJvH3PO8

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 α_k and the measurement noise β_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 Γ_∞ ? Discuss.

Chapter 6

Localization

Localization consists of finding the position of the robot (*i.e.*, 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, *i.e.*, 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.

6.1 Sensors

The robots are equipped with numerous sensors that are used for their localization. We will now present some of these.

Compass. It measures the magnetic field of its environment. If no magnetic disturbance exists, we get the magnetic north. In practice magnetic disturbances exists in the robot and a calibration procedure is required to compensate the hard-iron and soft-iron disturbances.

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).

Accelerometers. These sensors provide measurements of the instantaneous forward acceleration. The principle of the axis-based accelerometer is illustrated on Figure 6.1. Generally, three accelerometers are used by the robot. Due to gravity, the value measured according to the vertical axis must be compensated.

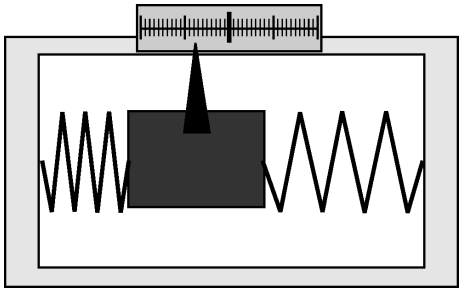

Figure 6.1: Operating principle of an accelerometer

Gyros. A gyro provides 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 6.2. 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 6.2. A flywheel at the center rotates with high speed. If the base of the gyroscope moves, the two gimbal angles ψ, θ 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.

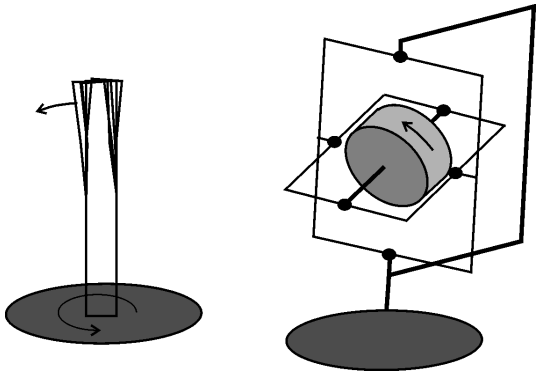

Figure 6.2: Coriolis vibratory gyroscope and gimbal gyroscope

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 6.3. 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.

IMU. 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 the estimated speed or even take into account Earth's rotation. Thus, they may 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 which are the bank ϕ, the elevation θ and the heading ψ 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.

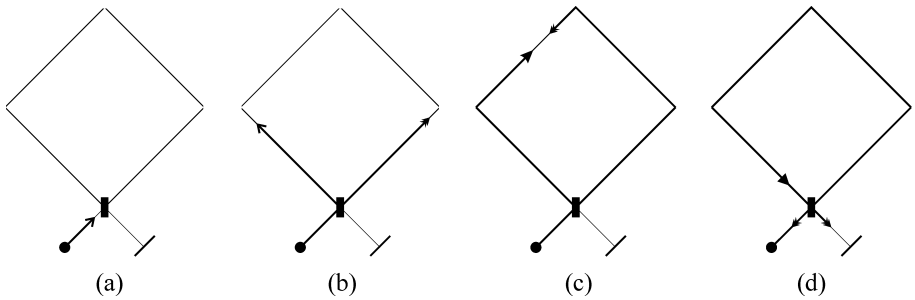

Figure 6.3: Principle of the Sagnac effect for optical gyroscopes

Barometer. It 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*) or **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 GNSS. 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 GNSS does not work within buildings or in water. Thus, during a diving experiment, an underwater robot can only be localized by GNSS 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 ± 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 GNSS can be found in the thesis of Vincent Drevelle [8]. In practice, a GNSS gives us a longitude ℓ_x and a latitude ℓ_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 ℓ_x^0 and ℓ_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 GNSS, we can deduce the first two coordinates of the robot, expressed in meters in the local coordinate system, by using the following relation:

$$\begin{pmatrix} p_x \\ p_y \end{pmatrix} = \rho \begin{pmatrix} 0 & 1 \\ \cos \ell_y & 0 \end{pmatrix} \begin{pmatrix} \ell_x - \ell_x^0 \\ \ell_y - \ell_y^0 \end{pmatrix} = \begin{pmatrix} \rho \ (\ell_y - \ell_y^0) \\ \rho \cos \ell_y \cdot (\ell_x - \ell_x^0) \end{pmatrix}$$

where ρ 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 **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 **o** and the rotational axis of the Earth. Thus, if a robot is moving on a latitude parallel ℓ_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 β 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.

Lidar. It composed of an emitter, projecting laser beams, and a receiver measuring the time-of-flight. Underwater, green lasers are used to get a lower absorption.

Cameras. Cameras are low-cost sensors used for the recognition of objects. They are also used as goniometers by finding angles relative to landmarks that will then be used for localization.

6.2 Goniometric localization

6.2.1 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 6.4 moving on a plane. We call *bearing* the angle α_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.

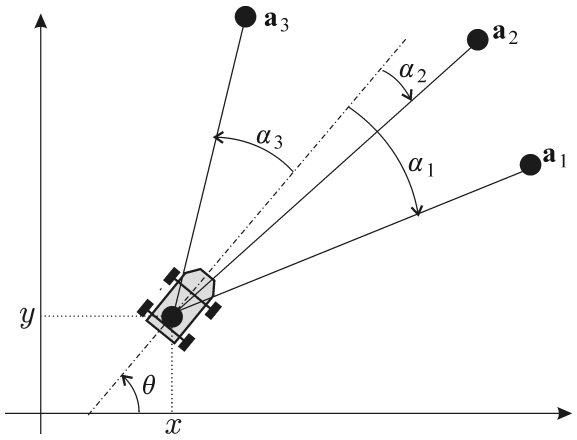

Figure 6.4: A robot moving on a plane, measures the angles in order to locate itself

Recall that two vectors $\mathbf{u}, \mathbf{v}$ of $\mathbb{R}^2$ are collinear if their determinant is zero, *i.e.*, if $\det(\mathbf{u}, \mathbf{v}) = 0$.

Thus, for each landmark, we have the relation:

$$\det\left(\begin{pmatrix} x_i - x \\ y_i - y \end{pmatrix}, \begin{pmatrix} \cos(\theta + \alpha_i) \\ \sin(\theta + \alpha_i) \end{pmatrix}\right) = 0 ,$$

i.e.,

$$(x_i - x)\sin(\theta + \alpha_i) - (y_i - y)\cos(\theta + \alpha_i) = 0, \tag{6.1}$$

where (x_i, y_i) are the coordinates of the landmark $\mathbf{a}_i$ and θ is the robot's heading.

6.2.2 Inscribed angles

Theorem 6. ***(Inscribed Angle)*** *Consider a triangle* **abm** *as represented on Figure 6.5. Let us denote by* **c** *the center of the circle circumscribed to this triangle (*i.e., **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:*

$$\gamma = 2(\alpha + \beta).$$

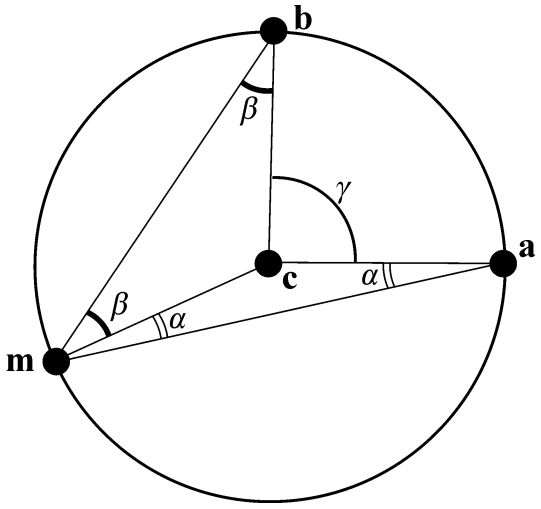

Figure 6.5: Illustration of the Inscribed Angle Theorem

Proof. Since the two triangles **amc** and **cmb** are isosceles. We thus find the angles α and β as shown on the figure. By going around the point **c** we obtain:

$$\gamma + (\pi - 2\beta) + (\pi - 2\alpha) = 2\pi.$$

Thus $\gamma = 2\alpha + 2\beta$. □

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 $\mathbf{m}$ such that the angle $\widehat{\mathbf{a}_1\mathbf{m}\mathbf{a}_2}$ is equal to α is a circle arc, referred to as an *inscribed arc*. We can show this from Relations (6.1) or from the Inscribed Angle Theorem. Goniometric localization often breaks down to intersecting arcs. Figure 6.6 illustrates the concept of an inscribed arc.

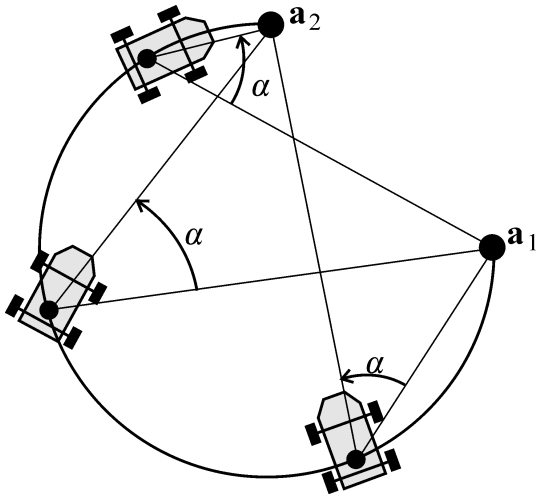

Figure 6.6: The three cars perceive the landmarks with the same angle

6.2.3 Static triangulation of a plane robot

6.2.3.1 Two landmarks and a compass

In the case where we have two landmarks and a compass, we have, following (6.1), the two relations:

$$\begin{cases} (x_1 - x)\sin(\theta+\alpha_1) - (y_1 - y)\cos(\theta+\alpha_1) &= 0 \\ (x_2 - x)\sin(\theta+\alpha_2) - (y_2 - y)\cos(\theta+\alpha_2) &= 0 \end{cases}$$

Or equivalently

$$\underbrace{\begin{pmatrix} \sin(\theta+\alpha_1) & -\cos(\theta+\alpha_1) \\ \sin(\theta+\alpha_2) & -\cos(\theta+\alpha_2) \end{pmatrix}}_{\mathbf{A}(\theta,\alpha_1,\alpha_2)} \begin{pmatrix} x \\ y \end{pmatrix} = \underbrace{\begin{pmatrix} x_1\sin(\theta+\alpha_1) - y_1\cos(\theta+\alpha_1) \\ x_2\sin(\theta+\alpha_2) - y_2\cos(\theta+\alpha_2) \end{pmatrix}}_{\mathbf{b}(\theta,\alpha_1,\alpha_2,x_1,y_1,x_2,y_2)}$$

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, *i.e.*,

$$\begin{array}{ll} & \sin(\theta+\alpha_1)\cos(\theta+\alpha_2) = \cos(\theta+\alpha_1)\sin(\theta+\alpha_2) \\ \Leftrightarrow & \tan(\theta+\alpha_2) = \tan(\theta+\alpha_1) \\ \Leftrightarrow & \theta+\alpha_2 = \theta+\alpha_1+k\pi,\ k\in\mathbb{N} \\ \Leftrightarrow & \alpha_2 = \alpha_1+k\pi,\ k\in\mathbb{N} \end{array}$$

This corresponds to a situation in which the two landmarks and the robot are aligned.

6.2.3.2 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.

6.2.4 Dynamic triangulation

6.2.4.1 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:

$$\left\{\begin{array}{l} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \end{array}\right. \tag{6.2}$$

where v represents the speed of the robot measured by the odometer and θ 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 θ 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 (6.1), we obtain:

$$\begin{array}{l} (\dot{x}_i-\dot{x})\sin(\theta+\alpha_i) + (x_i-x)\left(\dot{\theta}+\dot{\alpha}_i\right)\cos(\theta+\alpha_i) \\ -(\dot{y}_i-\dot{y})\cos(\theta+\alpha_i) + (y_i-y)\left(\dot{\theta}+\dot{\alpha}_i\right)\sin(\theta+\alpha_i) \ = \ 0 \end{array} \tag{6.3}$$

Let us take the relations (6.1) and (6.3) for $i=1$. By isolating x and y, we obtain:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \sin(\theta+\alpha_1) & \cos(\theta+\alpha_1) \\ -\cos(\theta+\alpha_1) & \sin(\theta+\alpha_1) \end{pmatrix} \cdot \begin{pmatrix} -y_1 & x_1 \\ x_1 - \frac{\dot{y}_1 - v\sin\theta}{\dot{\theta}+\dot{\alpha}_1} & \frac{\dot{x}_1 - v\cos\theta}{\dot{\theta}+\dot{\alpha}_1} + y_1 \end{pmatrix} \begin{pmatrix} \cos(\theta+\alpha_1) \\ \sin(\theta+\alpha_1) \end{pmatrix} \tag{6.4}$$

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 θ using the compass, the speed v and $\dot{\theta}$ using the odometers. Relation (6.4) then allows us to calculate the position x and y at the given moment in time.

6.2.4.2 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 (6.3), 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:

$$\begin{cases} \ddot{x} = \dot{v}\cos\theta - v\dot{\theta}\sin\theta \\ \ddot{y} = \dot{v}\sin\theta + v\dot{\theta}\cos\theta \end{cases}$$

We thus obtain a system of three equations with three unknowns x, y, θ:

$$\begin{array}{ll} (x_1-x)\sin(\theta+\alpha_1) - (y_1-y)\cos(\theta+\alpha_1) & = 0 \\ (\dot{x}_1 - v\cos\theta)\sin(\theta+\alpha_1) + (x_1-x)\left(\dot{\theta}+\dot{\alpha}_1\right)\cos(\theta+\alpha_1) & \\ \quad -(\dot{y}_1 - v\sin\theta)\cos(\theta+\alpha_1) + (y_1-y)\left(\dot{\theta}+\dot{\alpha}_1\right)\sin(\theta+\alpha_1) & = 0 \\ \left(\ddot{x}_1 - \dot{v}\cos\theta + v\dot{\theta}\sin\theta\right)\sin(\theta+\alpha_1) - \left(\ddot{y}_1 - \dot{v}\sin\theta - v\dot{\theta}\cos\theta\right)\cos(\theta+\alpha_1) & \\ \quad +(x_1-x)\left(\ddot{\theta}+\ddot{\alpha}_1\right)\cos(\theta+\alpha_1) + (y_1-y)\left(\ddot{\theta}+\ddot{\alpha}_1\right)\sin(\theta+\alpha_1) & \\ \quad +2(\dot{x}_1 - v\cos\theta)\left(\dot{\theta}+\dot{\alpha}_1\right)\cos(\theta+\alpha_1) + 2(\dot{y}_1 - v\sin\theta)\left(\dot{\theta}+\dot{\alpha}_1\right)\sin(\theta+\alpha_1) & \\ \quad -(x_1-x)\left(\dot{\theta}+\dot{\alpha}_1\right)^2\sin(\theta+\alpha_1) + (y_1-y)\left(\dot{\theta}+\dot{\alpha}_1\right)^2\cos(\theta+\alpha_1) & = 0 \end{array}$$

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 α_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 θ. In such a case, we need at least two landmarks for localization.

6.3 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 GNSS 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 6.7). 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,$x_1, y_1, z_1, \ldots, x_4, y_4, z_4$, τ_2,τ_3,τ_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 GNSS, the landmarks are mobile. They use a similar principle to be localized and synchronized, from fixed landmarks on the ground.

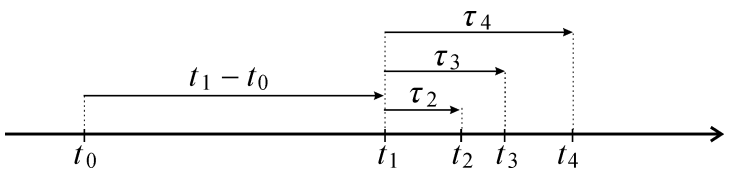

Figure 6.7: The emission time t_0 and the offsets between the arrival times τ_2, τ_3, τ_4 are known

Exercises

EXERCISE 21.– *Localization from wall distance measurements*

See the correction video at https://youtu.be/A6dYrlsGl8k

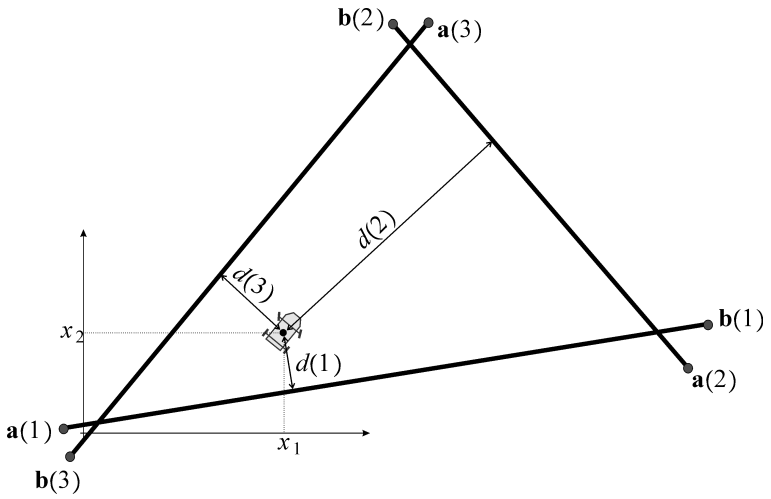

Figure 6.8: The robot must locate itself by measuring its distance to the three walls

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 6.8. The i^{th} wall corresponds to a line defined by two points $\mathbf{a}(i)$ and $\mathbf{b}(i)$. The distance to the i^{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 β_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 program that gives us the required estimation.

EXERCISE 22.– *Blind walker*

See the correction video at https://youtu.be/MAkBZiEyt3I

We consider a blind walker moving on a horizontal line. Its movement is described by the discretized state equation:

$$\begin{cases} x_1(k+1) &= x_1(k) + x_2(k) \cdot u(k) \\ x_2(k+1) &= x_2(k) + \alpha_2(k) \end{cases}$$

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 $x_2(k) \cdot u(k)$. 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 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) Draw the determinant of the covariance matrix $\mathbf{\Gamma_x}$ with respect to k. Discuss.

EXERCISE 23.– *Dead reckoning*

See the correction video at https://youtu.be/Zumje1wUOJg

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 6.9 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.

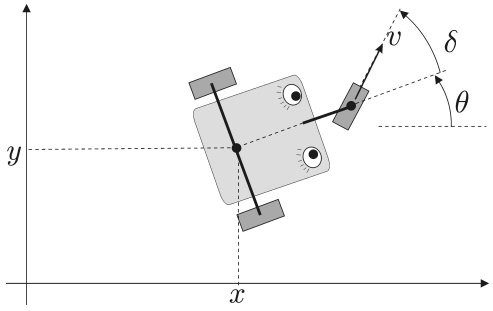

Figure 6.9: Dead reckoning for a tricycle robot

The robot is equipped with a compass that returns θ with high precision and an angle sensor that returns the angle δ of the front wheel.

1) Discretize this system with an Euler method. Simulate this system 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 24.– *Goniometric localization*

See the correction video at https://youtu.be/kIhSORI2cwg

Let u&s 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 bearing angle δ_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 6.10 illustrates a situation in which two landmarks $\mathbf{m}(1)$ and $\mathbf{m}(2)$ are detected by the robot.

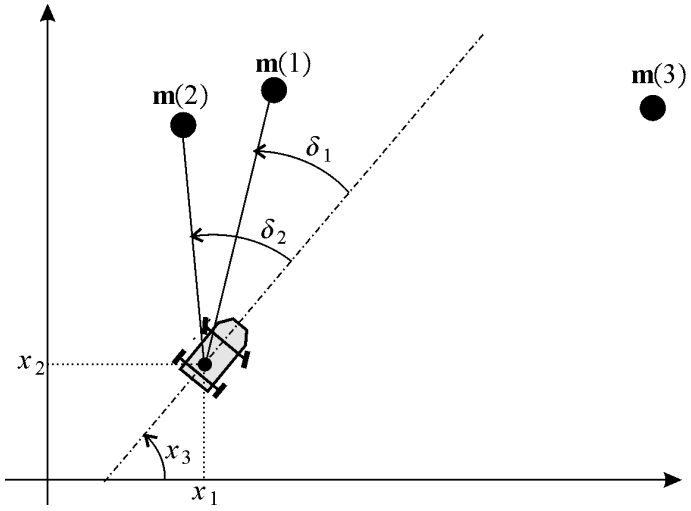

Figure 6.10: Goniometric localization

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 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 6.11). When the distances are small (i.e. smaller than 20 m), the robots can measure the angles φ_a and φ_b with high precision using cameras (see figure). Suggest a centralized Kalman filter for the localization of the two robots.

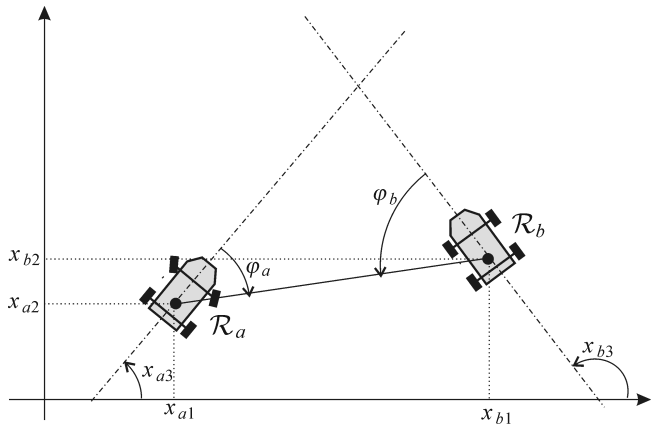

Figure 6.11: Goniometric localization for two communicating robots

---

EXERCISE 25.– *Localization by lidar*

See the correction video at https://youtu.be/V0hOejnwNEo

Here we are interested in developing a fast and robust localization method for a robot using a rotating laser rangefinder, or lidar (*light radar*) of type Hokuyo. The robot has no compass and in located inside a rectangular room whose length and width are unknown. More precisely, we want to localize the walls of the rectangular room in the robot frame.

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 α, d are the parameters of the line.

2) The lidar of the robot has an aperture angle of 180 degrees. It returns $n = 512$ points (x_i, y_i), in the robot frame as represented by Figure 6.12. These points correspond to real data and can be found in the file `lidar_data.csv`. In the world frame, most of the points, say 70%, are supposed to belong to the rectangle representing our room.

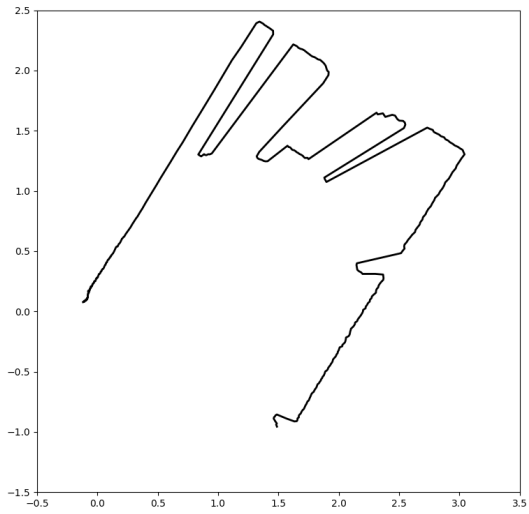

Figure 6.12: Data collected by the lidar

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. Keep only the groups with a small residue. You should thus obtain m lines, represented by m points of the form (β_j, d_j), $j \in \{1, \ldots, n\}$ in the so-called *Hough space*. A pair (β_j, d_j) is called an *alignment*. Write a program which draw all alignments with a small residue.

3) Find the four possible directions for walls of our room (knowing that it is rectangular).

4) Associate one wall to each aligment (β_j, d_j).

5) Write a program which localizes the walls of the room in the robot frame from the data set given in the file `lidar_data`.csv.

Chapter 7

Observers

7.1 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 known for all t. The noise $\boldsymbol{\alpha}_t$ and $\boldsymbol{\beta}_t$ are assumed to be white and Gaussian. For a better understanding, the order of the quantities that will be used with respect to dt (assumed to be infinitely small) are recalled below

Size	Order	Random	Deterministic
Infinitely small	$O(dt)$	$\boldsymbol{\alpha}_k$	$\mathbf{K}_t$,$\mathbf{u}_k$
Medium	$O(1)$	$\mathbf{x}_t, \mathbf{x}_k$	$\boldsymbol{\Gamma}_{\boldsymbol{\alpha}}, \boldsymbol{\Gamma}_{\boldsymbol{\beta}}, \mathbf{K}_b, \mathbf{u}_t, \mathbf{A}_t, \mathbf{C}_t$
Infinitely Large	$O\left(\frac{1}{dt}\right)$	$\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \boldsymbol{\beta}_k, \dot{\mathbf{x}}_t, \mathbf{y}_t$	$\boldsymbol{\Gamma}_{\boldsymbol{\alpha}_t}, \boldsymbol{\Gamma}_{\boldsymbol{\beta}_t}$

The covariance matrices $\boldsymbol{\Gamma}_{\boldsymbol{\alpha}_t}$,$\boldsymbol{\Gamma}_{\boldsymbol{\beta}_t}$ for $\boldsymbol{\alpha}_t$ et $\boldsymbol{\beta}_t$ and are infinite ($O\left(\frac{1}{dt}\right)$) and we write $\boldsymbol{\Gamma}_{\boldsymbol{\alpha}_t} = \frac{1}{dt} \cdot \boldsymbol{\Gamma}_{\boldsymbol{\alpha}}$ and $\boldsymbol{\Gamma}_{\boldsymbol{\beta}_t} = \frac{1}{dt} \cdot \boldsymbol{\Gamma}_{\boldsymbol{\beta}}$.

Remark. The fact that $\dot{\mathbf{x}}_t, \mathbf{y}_t$ are considered as infinite, due to the fact the noises $\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t$ are also infinite. If all would have been finite, the state could have been found in a infinitely small time. To be able to deal properly with these infinitely large quantities, distribution theory is needed. Here, we will not use these difficult notions and consider classical notions from signal processing. Note that this type of simplification is usually made in physics, as for instance when we assimilate the Dirac distribution to the Dirac function $\delta_0(t)$.

Since we have

$$\begin{array}{rcl} \mathbf{x}_{t+dt} & = & \mathbf{x}_t + \int_t^{t+dt} \dot{\mathbf{x}}_\tau d\tau \\ & = & \mathbf{x}_t + \int_t^{t+dt} (\mathbf{A}_\tau\mathbf{x}_\tau + \mathbf{u}_\tau + \boldsymbol{\alpha}_\tau)\, d\tau \\ & & \mathbf{x}_t + dt \cdot (\mathbf{A}_t\mathbf{x}_t + \mathbf{u}_t) + \int_t^{t+dt} \boldsymbol{\alpha}_\tau d\tau \end{array}$$

a discretization with a sampling time dt is

$$\left\{\begin{array}{rcl} \underbrace{\mathbf{x}_{t+dt}}_{\mathbf{x}_{k+1}} & = & \underbrace{(\mathbf{I}+dt\cdot\mathbf{A}_t)}_{\mathbf{A}_k}\cdot\underbrace{\mathbf{x}_t}_{\mathbf{x}_k} + \underbrace{dt\cdot\mathbf{u}_t}_{\mathbf{u}_k} + \underbrace{\int_t^{t+dt}\boldsymbol{\alpha}_\tau d\tau}_{\boldsymbol{\alpha}_k} \\ \underbrace{\mathbf{y}_t}_{\mathbf{y}_k} & = & \underbrace{\mathbf{C}_t}_{\mathbf{C}_k}\cdot\underbrace{\mathbf{x}_t}_{\mathbf{x}_k} + \underbrace{\boldsymbol{\beta}_t}_{\boldsymbol{\beta}_k}. \end{array}\right.$$

Note that, as explained in Exercise 11, the covariance for the state noise $\boldsymbol{\alpha}_k$ increases linearly with dt. More precisely,

$$\boldsymbol{\Gamma}_{\boldsymbol{\alpha}_k} = dt\cdot\boldsymbol{\Gamma}_{\boldsymbol{\alpha}}.$$

This means that when dt is infinitely small, $\boldsymbol{\alpha}_k$ is infinitely small, whereas $\boldsymbol{\alpha}_t$ is infinitely large (otherwise $\boldsymbol{\alpha}_k$ would be equal to zero). The measurement $\mathbf{y}_k$ represents a fusion of all measurement $\mathbf{y}_t$ with $t\in[t,t+dt]$. As for $\boldsymbol{\beta}_t$, its covariance matrix is

$$\boldsymbol{\Gamma}_{\boldsymbol{\beta}_k} = \frac{1}{dt}\cdot\boldsymbol{\Gamma}_{\boldsymbol{\beta}}.$$

We now have a linear discrete-time system, and we can apply the classical Kalman filter given at Section 5.1. We have the following correspondences.

$$\begin{array}{rclrclrcl} \mathbf{A}_k & \to & (\mathbf{I}+dt\mathbf{A}_t) & \mathbf{u}_k & \to & dt\cdot\mathbf{u}_t & \boldsymbol{\alpha}_k & \to & \int_t^{t+dt}\boldsymbol{\alpha}_\tau d\tau \\ \hat{\mathbf{x}}_{k+1|k} & \to & \hat{\mathbf{x}}_{t+dt} & \hat{\mathbf{x}}_{k|k-1} & \to & \hat{\mathbf{x}}_t & \hat{\mathbf{x}}_{k|k} & \to & \mathring{\mathbf{x}}_t \\ \boldsymbol{\Gamma}_{k+1|k} & \to & \boldsymbol{\Gamma}_{t+dt} & \boldsymbol{\Gamma}_{k|k-1} & \to & \boldsymbol{\Gamma}_t & \boldsymbol{\Gamma}_{k|k} & \to & \mathring{\boldsymbol{\Gamma}}_t \\ \boldsymbol{\Gamma}_{\boldsymbol{\alpha}_k} & \to & dt\cdot\boldsymbol{\Gamma}_{\boldsymbol{\alpha}} & \boldsymbol{\Gamma}_{\boldsymbol{\beta}_k} & \to & \frac{1}{dt}\cdot\boldsymbol{\Gamma}_{\boldsymbol{\beta}} & & & \end{array}$$

The Kalman filter becomes

$$\left\{\begin{array}{rcl} \hat{\mathbf{x}}_{t+dt} & = & (\mathbf{I}+dt\mathbf{A}_t)\mathring{\mathbf{x}}_t + dt\cdot\mathbf{u}_t \\ \boldsymbol{\Gamma}_{t+dt} & = & (\mathbf{I}+dt\mathbf{A}_t)\cdot\mathring{\boldsymbol{\Gamma}}_t\cdot(\mathbf{I}+dt\mathbf{A}_t)^T + dt\boldsymbol{\Gamma}_{\boldsymbol{\alpha}} \\ \mathring{\mathbf{x}}_t & = & \hat{\mathbf{x}}_t+\mathbf{K}_t\cdot\widetilde{\mathbf{y}}_t \\ \mathring{\boldsymbol{\Gamma}}_t & = & (\mathbf{I}-\mathbf{K}_t\mathbf{C}_t)\cdot\boldsymbol{\Gamma}_t \\ \widetilde{\mathbf{y}}_t & = & \mathbf{y}_t-\mathbf{C}_t\hat{\mathbf{x}}_t \\ \mathbf{S}_t & = & \mathbf{C}_t\boldsymbol{\Gamma}_t\mathbf{C}_t^{\mathrm{T}}+\frac{1}{dt}\cdot\boldsymbol{\Gamma}_{\boldsymbol{\beta}} \\ \mathbf{K}_t & = & \boldsymbol{\Gamma}_t\mathbf{C}_t^{\mathrm{T}}\mathbf{S}_t^{-1} \end{array}\right.$$

Taking into account the fact that $\mathbf{C}_t\boldsymbol{\Gamma}_t\mathbf{C}_t^{\mathrm{T}}$ is small (line (vi)) compared to $\frac{1}{dt}\cdot\boldsymbol{\Gamma}_{\boldsymbol{\beta}}$ (which is infinitely large) and we can simplify it, i.e., $\mathbf{S}_t=\frac{1}{dt}\cdot\boldsymbol{\Gamma}_{\boldsymbol{\beta}}(t)$, 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

$$\left\{\begin{array}{lcl} \hat{\mathbf{x}}_{t+dt} &=& (\mathbf{I}+dt\mathbf{A}_t)\,(\hat{\mathbf{x}}_t+\mathbf{K}_t\cdot(\mathbf{y}_t-\mathbf{C}_t\hat{\mathbf{x}}_t))+dt\cdot\mathbf{u}_t\\ \boldsymbol{\Gamma}_{t+dt} &=& (\mathbf{I}+dt\mathbf{A}_t)\cdot(\mathbf{I}-\mathbf{K}_t\mathbf{C}_t)\,\boldsymbol{\Gamma}_t\cdot(\mathbf{I}+dt\mathbf{A}_t)^T+dt\boldsymbol{\Gamma}_{\boldsymbol{\alpha}}\\ \mathbf{K}_t &=& dt\boldsymbol{\Gamma}_t\mathbf{C}_t^{\mathrm{T}}\boldsymbol{\Gamma}_{\boldsymbol{\beta}}^{-1}. \end{array}\right.$$

Since $\mathbf{K}_t$ is small ($=O(dt)$), it is more convenient to introduce the *Kalman-Bucy* gain given by $\mathbf{K}_b=\boldsymbol{\Gamma}_t\mathbf{C}_t^{\mathrm{T}}\boldsymbol{\Gamma}_{\boldsymbol{\beta}}^{-1}$ which is $O(1)$, *i.e.*, neither infinitely small nor infinitely large. We have

$$\left\{\begin{array}{lcl} \hat{\mathbf{x}}_{t+dt} &=& \underbrace{(\mathbf{I}+dt\mathbf{A}_t)\,(\hat{\mathbf{x}}_t+dt\mathbf{K}_b\cdot(\mathbf{y}_t-\mathbf{C}_t\hat{\mathbf{x}}_t))}_{=(\mathbf{I}+dt\mathbf{A}_t)\hat{\mathbf{x}}_t+dt\mathbf{K}_b\cdot(\mathbf{y}_t-\mathbf{C}_t\hat{\mathbf{x}}_t)+O(dt)^2}+dt\cdot\mathbf{u}_t\\ \boldsymbol{\Gamma}_{t+dt} &=& \underbrace{(\mathbf{I}+dt\mathbf{A}_t)\cdot(\mathbf{I}-dt\mathbf{K}_b\mathbf{C}_t)\,\boldsymbol{\Gamma}_t\cdot(\mathbf{I}+dt\mathbf{A}_t)^T}_{=\boldsymbol{\Gamma}_t+dt\left(\mathbf{A}_t\boldsymbol{\Gamma}_t-\mathbf{K}_b\mathbf{C}_t\boldsymbol{\Gamma}_t+\boldsymbol{\Gamma}_t\mathbf{A}_t^T\right)+O(dt)^2}+dt\boldsymbol{\Gamma}_{\boldsymbol{\alpha}}\\ \mathbf{K}_b &=& \boldsymbol{\Gamma}_t\mathbf{C}_t^{\mathrm{T}}\boldsymbol{\Gamma}_{\boldsymbol{\beta}}^{-1} \end{array}\right.$$

Or equivalently

$$\left\{\begin{array}{lcl} \frac{\hat{\mathbf{x}}_{t+dt}-\hat{\mathbf{x}}_t}{dt} &=& \mathbf{A}_t\hat{\mathbf{x}}_t+\mathbf{K}_b\cdot(\mathbf{y}_t-\mathbf{C}_t\hat{\mathbf{x}}_t)+\mathbf{u}_t\\ \frac{\boldsymbol{\Gamma}_{t+dt}-\boldsymbol{\Gamma}_t}{dt} &=& \left(\mathbf{A}_t\boldsymbol{\Gamma}_t-\mathbf{K}_b\mathbf{C}_t\boldsymbol{\Gamma}_t+\boldsymbol{\Gamma}_t\mathbf{A}_t^{\mathrm{T}}\right)+\boldsymbol{\Gamma}_{\boldsymbol{\alpha}}\\ \mathbf{K}_b &=& \boldsymbol{\Gamma}_t\mathbf{C}_t^{\mathrm{T}}\boldsymbol{\Gamma}_{\boldsymbol{\beta}}^{-1}. \end{array}\right.$$

Now, $\mathbf{C}_t\boldsymbol{\Gamma}_t=\left(\boldsymbol{\Gamma}_t\mathbf{C}_t^{\mathrm{T}}\right)^T=\left(\mathbf{K}_b\boldsymbol{\Gamma}_{\boldsymbol{\beta}}\right)^T=\boldsymbol{\Gamma}_{\boldsymbol{\beta}}\mathbf{K}_b^{\mathrm{T}}$. Therefore, we get the Kalman-Bucy filter:

$$\left\{\begin{array}{lcl} \frac{d}{dt}\hat{\mathbf{x}}_t &=& \mathbf{A}_t\hat{\mathbf{x}}_t+\mathbf{K}_b\left(\mathbf{y}_t-\mathbf{C}_t\hat{\mathbf{x}}_t\right)+\mathbf{u}_t\\ \frac{d}{dt}\boldsymbol{\Gamma}_t &=& \mathbf{A}_t\boldsymbol{\Gamma}_t+\boldsymbol{\Gamma}_t\cdot\mathbf{A}_t^{\mathrm{T}}-\mathbf{K}_b\boldsymbol{\Gamma}_{\boldsymbol{\beta}}\mathbf{K}_b^T+\boldsymbol{\Gamma}_{\boldsymbol{\alpha}}\\ \mathbf{K}_b &=& \boldsymbol{\Gamma}_t\mathbf{C}_t^{\mathrm{T}}\boldsymbol{\Gamma}_{\boldsymbol{\beta}}^{-1}. \end{array}\right.$$

The advantage of the latter formulation compared to the former is that the product $\mathbf{K}_b\boldsymbol{\Gamma}_{\boldsymbol{\beta}}\mathbf{K}_b^T$ is more stable than $\mathbf{K}_b\mathbf{C}_t\boldsymbol{\Gamma}_t$, since it preserves the symmetry of the resulting matrix. This Kalman-Bucy filter corresponds to a continuous version of the Kalman filter. In its formulation all quantities that are $O\left(1\right)$ and thus no more infinitely small ($O(dt)$) or infinitely large ($O\left(\frac{1}{dt}\right)$) quantities appear.

Remark. The Kalman-Bucy allows us to understand some effects that occur when the discrete-time Kalman filter is used for continuous time systems, such as a robot described by

$$\left\{\begin{array}{lcl} \dot{\mathbf{x}} &=& \mathbf{f}_c(\mathbf{x},\mathbf{u})\\ \mathbf{y} &=& \mathbf{g}(\mathbf{x}) \end{array}\right.$$

To perform a simulation, we discretize it, for instance using an Euler integration. We get

$$\left\{\begin{array}{ll} \mathbf{x}_{k+1} &= \mathbf{f}(\mathbf{x}_k,\mathbf{u}_k)+\boldsymbol{\alpha}_k\\ \mathbf{y}_k &= \mathbf{g}(\mathbf{x}_k)+\boldsymbol{\beta}_k \end{array}\right.$$

where $\mathbf{f}(\mathbf{x}_k,\mathbf{u}_k)=\mathbf{x}_k+dt\cdot\mathbf{f}_c(\mathbf{x}_k,\mathbf{u}_k)$ and where $\boldsymbol{\alpha}_k,\boldsymbol{\beta}_k$ is the noise. Now, a good tuning for the

covariances matrices $\mathbf{\Gamma}_{\boldsymbol{\alpha}}, \mathbf{\Gamma}_{\boldsymbol{\beta}}$ at a given sampling time dt, should still be consistent if we change dt. For have this property, the Kalman-Bucy filter tells us that the covariance matrices should depend on dt as follows:

$$\begin{array}{rcl} \mathbf{\Gamma}_{\boldsymbol{\alpha}} & = & dt \cdot \mathbf{\Gamma}^0_{\boldsymbol{\alpha}} \\ \mathbf{\Gamma}_{\boldsymbol{\beta}} & = & \frac{1}{dt} \cdot \mathbf{\Gamma}^0_{\boldsymbol{\beta}} \end{array}$$

In such a case, the behavior of the simulated system will not be affected by a change of dt. As consequence, if we change $dt \rightarrow 2 \cdot dt$ (for instance to make the simulation goes twice faster), we need to multiply the noises noise $\boldsymbol{\alpha}_k, \boldsymbol{\beta}_k$, by $\sqrt{2}, \frac{1}{\sqrt{2}}$, respectively.

7.2 Extended Kalman Filter

As we have seen in previous chapters, a robot is described by continuous state equations of the form

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

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

$$\begin{array}{lcll}
\mathbf{A}_k & = & \frac{\partial \mathbf{f}(\hat{\mathbf{x}}_{k|k}, \mathbf{u}_k)}{\partial \mathbf{x}} & \text{(evolution matrix)} \\
\mathbf{C}_k & = & \frac{d\mathbf{g}(\hat{\mathbf{x}}_{k|k-1})}{d\mathbf{x}} & \text{(observation matrix)} \\
\hat{\mathbf{x}}_{k+1|k} & = & \mathbf{A}_k \hat{\mathbf{x}}_{k|k} + \underbrace{\left(\mathbf{f}(\hat{\mathbf{x}}_{k|k}, \mathbf{u}_k) - \mathbf{A}_k \cdot \hat{\mathbf{x}}_{k|k}\right)}_{\mathbf{v}_k} = \mathbf{f}(\hat{\mathbf{x}}_{k|k}, \mathbf{u}_k) & \text{(predicted estimation)} \\
\boldsymbol{\Gamma}_{k+1|k} & = & \mathbf{A}_k \cdot \boldsymbol{\Gamma}_{k|k} \cdot \mathbf{A}_k^{\mathrm{T}} + \boldsymbol{\Gamma}_{\boldsymbol{\alpha}_k} & \text{(predicted covariance)} \\
\hat{\mathbf{x}}_{k|k} & = & \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \cdot \widetilde{\mathbf{z}}_k & \text{(corrected estimation)} \\
\boldsymbol{\Gamma}_{k|k} & = & (\mathbf{I} - \mathbf{K}_k \mathbf{C}_k)\, \boldsymbol{\Gamma}_{k|k-1} & \text{(corrected covariance)} \\
\widetilde{\mathbf{z}}_k & = & \underbrace{\left(\mathbf{y}_k - \mathbf{g}(\hat{\mathbf{x}}_{k|k-1}) + \mathbf{C}_k \cdot \hat{\mathbf{x}}_{k|k-1}\right)}_{\mathbf{z}_k} - \mathbf{C}_k \hat{\mathbf{x}}_{k|k-1} = \mathbf{y}_k - \mathbf{g}(\hat{\mathbf{x}}_{k|k-1}) & \text{(innovation)} \\
\mathbf{S}_k & = & \mathbf{C}_k \boldsymbol{\Gamma}_{k|k-1} \mathbf{C}_k^{\mathrm{T}} + \boldsymbol{\Gamma}_{\boldsymbol{\beta}_k} & \text{(covariance of the innovation)} \\
\mathbf{K}_k & = & \boldsymbol{\Gamma}_{k|k-1} \mathbf{C}_k^{\mathrm{T}} \mathbf{S}_k^{-1} & \text{(Kalman gain)}
\end{array}$$

The extended Kalman filter takes as inputs $\hat{\mathbf{x}}_{k|k-1}$, $\boldsymbol{\Gamma}_{k|k-1}$, $\mathbf{y}_k$, $\mathbf{u}_k$ and returns $\hat{\mathbf{x}}_{k+1|k}$ and $\boldsymbol{\Gamma}_{k+1|k}$.

Exercises

EXERCISE 26.– *State estimation of the inverted rod pendulum*

See the correction video at https://youtu.be/wuwmIOaj7Fc

We consider an inverted rod pendulum whose state equations are given by:

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} x_3 \\ x_4 \\ \dfrac{m_r \sin x_2 (g \cos x_2 - \ell x_4^2) + u}{m_c + m_r \sin^2 x_2} \\ \dfrac{\sin x_2 ((m_c + m_r) g - m_r \ell x_4^2 \cos x_2) + \cos x_2 u}{\ell (m_c + m_r \sin^2 x_2)} \end{pmatrix} \quad \text{and} \quad \mathbf{y} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

Here, we have taken as state vector $\mathbf{x} = \left(s, \theta, \dot{s}, \dot{\theta}\right)$, where the input u is the force exerted on the cart of mass m_c, s is the position of the cart and θ is the angle between the pendulum and the vertical direction. We will assume here that only the position of the cart s and the angle θ 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. 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 [3], 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 perform an output-based feedback, we need an estimation $\hat{\mathbf{x}}$ of the state vector $\mathbf{x}$. For this, we will use a Kalman filter (see Figure 7.1):

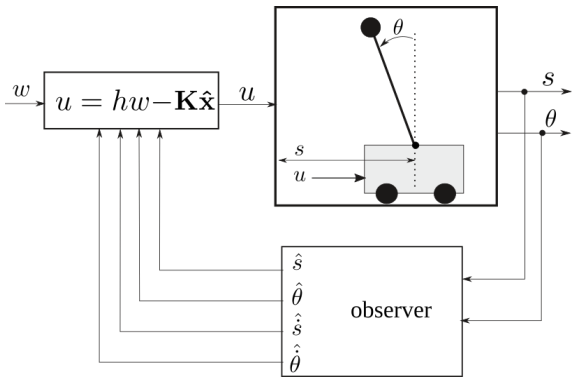

Figure 7.1: Kalman filter used to estimate the state of the inverted rod pendulum

Discretize the system using steps of $dt = 0.1\,\text{sec}$, then propose a Kalman filter for observing the state.

4) Implement this filter and study the robustness of the Kalman observer when the measurement noise is increased.

5) An extended Kalman filter can be obtained by replacing, in the prediction step of the Kalman filter, the statement:

$$\hat{\mathbf{x}}_{k+1|k} = \mathbf{A}_k \hat{\mathbf{x}}_{k|k} + \mathbf{B}_k \mathbf{u}_k$$

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 on the inverted rod pendulum.

EXERCISE 27.– *Following a boat with two radars*

See the correction video at https://youtu.be/FQSud487xIE

The movement of a boat that we are seeking to localize 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 α_x and α_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 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 Gaussian white noises with a variance equal to $\frac{1}{dt} \cdot 10^{-2}$. 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 28.– *Robot localization in a pool*

See the correction video at https://youtu.be/U3X52L9quvE

Consider an underwater robot moving within a rectangular pool of length $2R_y$ and width $2R_x$. A sonar placed at the center of the robot rotates with a constant angular speed $\dot{\delta}$. The depth is easily obtained using a barometer 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 to be zero. We want to estimate the coordinates (x, y) of the robot. The origin of the coordinate system is in middle of the pool. For this localization, we will assume that the angle δ of the sonar is known perfectly relative to the body of the robot and the heading angle θ is measured with a compass. The sonar illuminates the environment inside an emission cone with an angle $\pm\frac{\pi}{4}$, as illustrated by Figure 7.2.

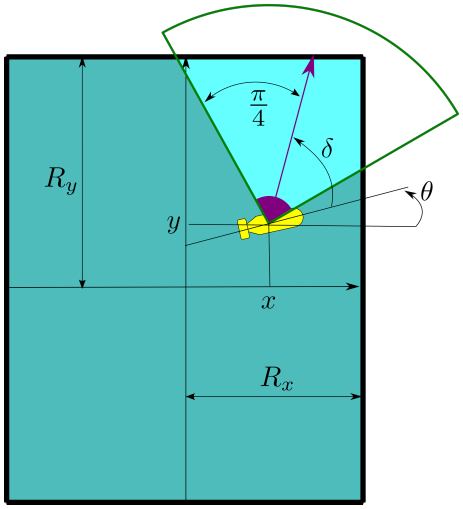

Figure 7.2: In the pool, the rotating sonar measures the distance to one of the wall

The tangential acceleration a is measured with an accelerometer. Every $dt = 0.05$s, the sonar

returns the distance ℓ to the wall in front of it. As illustrated by Figure 7.3 the number w of the wall involved in the distance measurement (called the *hit wall*) only depends on the angle $\theta + \delta$ and not on the position of the robot. In the configuration (a) of Figure 7.3 right, the sonar in on the right of the robot and since $\theta + \delta \simeq \frac{\pi}{2}$, Wall 1 is hit. In the configuration (b), the sonar is on the left of the robot and since $\theta + \delta \simeq \pi$, Wall 2 is hit.

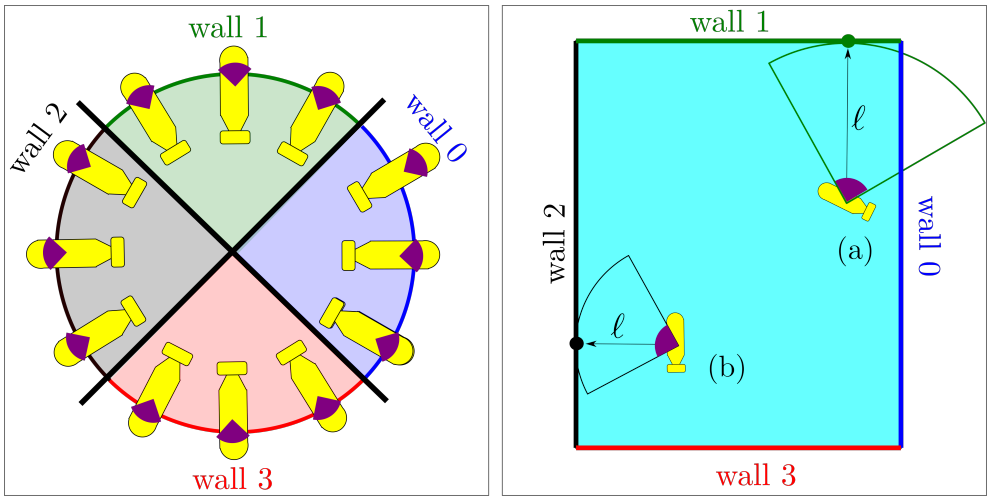

Figure 7.3: The sonar returns the distance ℓ to the wth wall where w depends on the orientation $\theta + \delta$ of the sonar

We want the build a Kalman filter which estimates the position (x, y) from the measurements $a_k, \ell_k, \theta_k, \delta_k$ at time $t_k = dt \cdot k$.

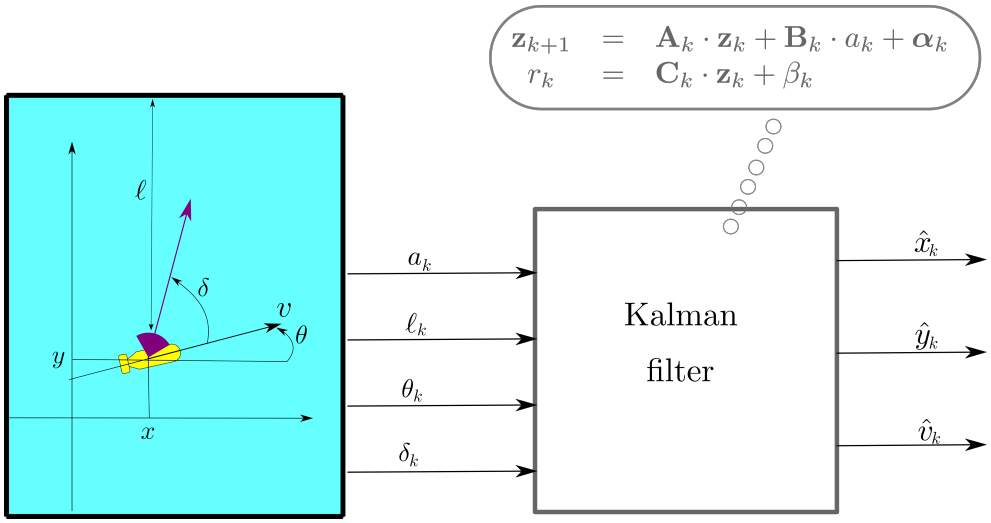

Figure 7.4: The Kalman filter estimates the position of the robot in the pool from the measurements collected by the sensors

The Kalman filter will be built on the following cinematic model

$$\left\{\begin{array}{rcl} \dot{x} &=& v\cdot\cos\theta \\ \dot{y} &=& v\cdot\sin\theta \\ \dot{v} &=& a \end{array}\right.$$

The state vector is $\mathbf{z}=(x,y,v)$ and the input is a.

1) The Kalman filter assumes the following linear state equations

$$\begin{array}{rcl} \mathbf{z}_{k+1} &=& \mathbf{A}_k\cdot\mathbf{z}_k+\mathbf{B}_k\cdot a_k+\boldsymbol{\alpha}_k \\ r_k &=& \mathbf{C}_k\cdot\mathbf{z}_k+\beta_k \end{array}$$

where $\boldsymbol{\alpha}_k$ and β_k are white Gaussian noises. Give the expressions for $\mathbf{A}_k,\mathbf{B}_k$ and $\mathbf{C}_k$ we should take to have the estimations $\hat{x}$, $\hat{y}$, $\hat{v}$ of the variables x, y, v. Explain how the quantity r_k should be chosen from the measurements.

2) Simulate a robot in the pool. Its motion will obey to the state equation

$$\begin{array}{rcl} \dot{x} &=& v\cdot\cos\theta \\ \dot{y} &=& v\cdot\sin\theta \\ \dot{\theta} &=& u_1 \\ \dot{v} &=& u_2 \\ \dot{\delta} &=& u_3 \end{array}$$

The initial state vector $\mathbf{x}_0$ and the constant input $\mathbf{u}$ are given by

$$\mathbf{x}_0=\begin{pmatrix} 10 \\ -10 \\ 1 \\ 3 \\ 0 \end{pmatrix} \text{ and } \mathbf{u}=\begin{pmatrix} 0.2 \\ 0 \\ 2 \end{pmatrix}$$

Illustrate the behavior of the Kalman filter. To avoid outliers, we will assume that the distance ℓ is reliable only when the sonar beam is orthogonal to the hit wall.

EXERCISE 29.– *Instantaneous state estimation*

See the correction video at https://youtu.be/-JIQb_0CEI4

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 described by the state equations:

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

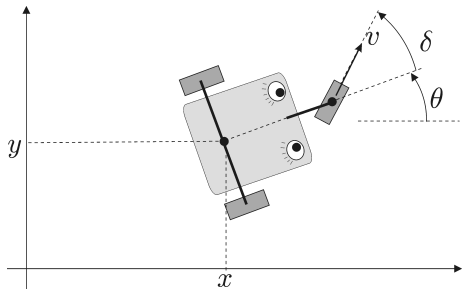

Figure 7.5: Tricycle for which we want to reconstruct the state

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 θ, v, δ in function of x,y,$\dot{x}$,$\dot{y}$,$\ddot{x}$,$\ddot{y}$.

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 with the pair $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) \in \mathbb{R}^n \times \mathbb{R}^m$ a positive real number. But, $\mathbf{y}$ has 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 *normalizer* 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 five vectors $\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d},\mathbf{e}$ that are linked by the relations

$$\begin{array}{rcl} \mathbf{b} & = & \mathbf{a} + \boldsymbol{\alpha}_1 \\ \mathbf{e} & = & \mathbf{a} + 2\mathbf{b} + \boldsymbol{\alpha}_2 \\ \mathbf{c} & = & \mathbf{e} + \boldsymbol{\alpha}_3 \\ \mathbf{d} & = & 3\mathbf{c} + \boldsymbol{\alpha}_4 \end{array}$$

where the $\boldsymbol{\alpha}_i$ are all independent vectors with a known density, for instance $\mathcal{N}(0,1)$, which means Gaussian centered 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 $\pi(a) = \mathcal{N}(1,9)$, the program could be:

```
1  a := 1+randn(9)
2  b := a + randn(1)
3  e := a + 2 · b + randn(1)
4  c := e + randn(1);
5  d := 3 · c + randn(1)
```

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 $\mathbf{e}$, 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 $\mathbf{e}$, the knowledge of $\mathbf{a}$ and $\mathbf{b}$ does not bring any new information on $\mathbf{c}$. This reasoning can be formalized and generalized under the form of the Hammersley–Clifford theorem.

Consider 3 vectors $\mathbf{x}, \mathbf{y}, \mathbf{z}$ that are linked by the relations

$$\begin{array}{rcl} \mathbf{y} & = & 2\mathbf{x} + \boldsymbol{\alpha}_1 \\ \mathbf{z} & = & 3\mathbf{y} + \boldsymbol{\alpha}_2 \\ \mathbf{x} & = & 4\mathbf{z} + \boldsymbol{\alpha}_3 \end{array}$$

where the $\boldsymbol{\alpha}_i$ are all independent vectors. The graph (see Figure 8.1 (right) has a cycle and is not considered as 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.

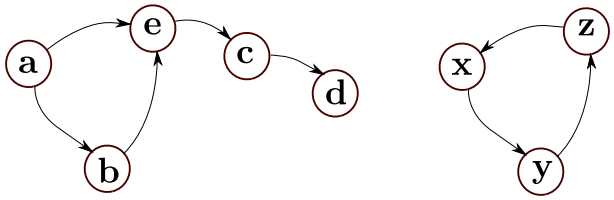

Figure 8.1: Left: a Bayes network; Right, the graph has a cycle and thus is not a Bayes network

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 $\boldsymbol{\alpha}_k$.

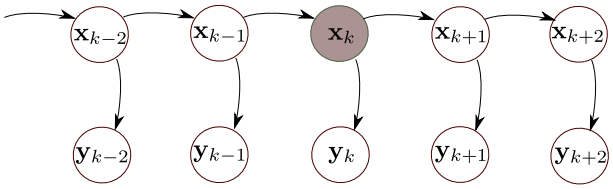

Figure 8.2: Bayes network associated with the state equation

Theorem 7. *The two densities*

$$\begin{array}{lll} pred(\mathbf{x}_k) & \stackrel{def}{=} & \pi(\mathbf{x}_k|\mathbf{y}_{0:k-1}) \\ bel(\mathbf{x}_k) & \stackrel{def}{=} & \pi(\mathbf{x}_k|\mathbf{y}_{0:k}). \end{array} \tag{8.10}$$

satisfy

$$\begin{array}{llll} (i) & pred\,(\mathbf{x}_{k+1}) & = & \int \pi\left(\mathbf{x}_{k+1}|\mathbf{x}_k\right) \cdot bel\left(\mathbf{x}_k\right) \cdot d\mathbf{x}_k \quad (prediction) \\ (ii) & bel\,(\mathbf{x}_k) & = & \frac{\pi(\mathbf{y}_k|\mathbf{x}_k)\cdot pred(\mathbf{x}_k)}{\pi(\mathbf{y}_k|\mathbf{y}_{0:k-1})}. \quad (correction) \end{array} \tag{8.11}$$

These relations correspond to a *Bayes observer* or *Bayes filter*. The prediction equation is known as the equation of *Chapman-Kolmogorov*.

Proof. Let us prove relation (ii). We have

$$\begin{array}{lll} \mathrm{bel}\left(\mathbf{x}_k\right) & \stackrel{(8.10)}{=} & \pi\left(\mathbf{x}_k|\mathbf{y}_{0:k}\right) \\ & = & \pi\left(\mathbf{x}_k|\mathbf{y}_k, \mathbf{y}_{0:k-1}\right) \\ & \stackrel{(8.7)}{=} & \frac{1}{\pi(\mathbf{y}_k|\mathbf{y}_{0:k-1})} \cdot \underbrace{\pi(\mathbf{y}_k|\mathbf{x}_k, \mathbf{y}_{0:k-1})}_{\stackrel{(8.9)}{=}\pi(\mathbf{y}_k|\mathbf{x}_k)} \cdot \underbrace{\pi(\mathbf{x}_k|\mathbf{y}_{0:k-1})}_{\stackrel{(8.10)}{=}\mathrm{pred}(\mathbf{x}_k)} \end{array} \qquad \left\{\pi(\mathbf{x}|\mathbf{y},\mathbf{z}) = \frac{\pi(\mathbf{y}|\mathbf{x},\mathbf{z})\cdot\pi(\mathbf{x}|\mathbf{z})}{\pi(\mathbf{y}|\mathbf{z})}\right\}$$

Let us now prove (i). From the total probability rule (8.6), we get

$$\begin{array}{lll} \mathrm{pred}\left(\mathbf{x}_{k+1}\right) & \stackrel{(8.10)}{=} & \pi\left(\mathbf{x}_{k+1}|\mathbf{y}_{0:k}\right) \\ & \stackrel{(8.6)}{=} & \int \underbrace{\pi(\mathbf{x}_{k+1}|\mathbf{x}_k, \mathbf{y}_{0:k})}_{\stackrel{(8.9)}{=}\pi(\mathbf{x}_{k+1}|\mathbf{x}_k)} \cdot \underbrace{\pi(\mathbf{x}_k|\mathbf{y}_{0:k})}_{\stackrel{(8.10)}{=}\mathrm{bel}\left(\mathbf{x}_k\right)} d\mathbf{x}_k \end{array} \qquad \left\{ \begin{array}{l} \pi(\mathbf{x}|\mathbf{z}) = \\ \int \pi(\mathbf{x}|\mathbf{y},\mathbf{z}) \cdot \pi(\mathbf{y}|\mathbf{z}) \cdot d\mathbf{y} \end{array} \right\}$$

This corresponds to relation (i). □

Remark. From (8.11, ii), we have

$$\underbrace{\int \mathrm{bel}\left(\mathbf{x}_k\right) \cdot d\mathbf{x}_k}_{=1} = \int \frac{\pi(\mathbf{y}_k|\mathbf{x}_k) \cdot \mathrm{pred}\left(\mathbf{x}_k\right)}{\pi\left(\mathbf{y}_k|\mathbf{y}_{0:k-1}\right)} \cdot d\mathbf{x}_k.$$

Thus

$$\pi\left(\mathbf{y}_k|\mathbf{y}_{0:k-1}\right) = \int \pi(\mathbf{y}_k|\mathbf{x}_k) \cdot \mathrm{pred}\left(\mathbf{x}_k\right) \cdot d\mathbf{x}_k$$

is a normalizer coefficient which can be computed directly from $\pi(\mathbf{y}_k|\mathbf{x}_k)$ and $\mathrm{pred}(\mathbf{x}_k)$.

The relations (8.11) can be interpreted as an algorithm where the variables $\mathrm{pred}(\mathbf{x}_k)$ and $\mathrm{bel}(\mathbf{x}_k)$ 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 Bayes filter.

- If the random vector $\mathbf{x}$ is discrete, the process can be represented by a Markov chain and the densities $\mathrm{pred}(\mathbf{x}_k)$ and $\mathrm{bel}(\mathbf{x}_k)$ can be represented by a vector of $\mathbb{R}^n$. The Bayes filter can be implemented exactly.
- If the densities $\mathrm{pred}(\mathbf{x}_k)$ and $\mathrm{bel}(\mathbf{x}_k)$ are Gaussian, they can be represented exactly by their expectation and their covariance matrices. We get the Kalman filter [6].
- We can approximate $\mathrm{pred}(\mathbf{x}_k)$ and $\mathrm{bel}(\mathbf{x}_k)$ by cloud of points (called *particles*) with different weights. The corresponding observer is called *Particle filter*.
- We can represent the densities $\mathrm{pred}(\mathbf{x}_k)$ and $\mathrm{bel}(\mathbf{x}_k)$ with a subset of $\mathbb{R}^n$ which contains the support of 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 8. *The three densities*

$$\begin{array}{lcl} pred\left(\mathbf{x}_k\right) & \stackrel{def}{=} & \pi\left(\mathbf{x}_k|\mathbf{y}_{0:k-1}\right) \\ bel\left(\mathbf{x}_k\right) & \stackrel{def}{=} & \pi\left(\mathbf{x}_k|\mathbf{y}_{0:k}\right). \\ back\left(\mathbf{x}_k\right) & \stackrel{def}{=} & \pi\left(\mathbf{x}_k|\mathbf{y}_{0:N}\right) \end{array} \tag{8.12}$$

can be defined recursively as follows

$$\begin{array}{llcll} (i) & pred\left(\mathbf{x}_k\right) & = & \int \pi\left(\mathbf{x}_k|\mathbf{x}_{k-1}\right) \cdot bel\left(\mathbf{x}_{k-1}\right) \cdot d\mathbf{x}_{k-1} & (prediction) \\ (ii) & bel\left(\mathbf{x}_k\right) & = & \frac{\pi(\mathbf{y}_k|\mathbf{x}_k) \cdot pred(\mathbf{x}_k)}{\pi(\mathbf{y}_k|\mathbf{y}_{0:k-1})} & (correction) \\ (iii) & back\left(\mathbf{x}_k\right) & = & bel\left(\mathbf{x}_k\right) \int \frac{\pi(\mathbf{x}_{k+1}|\mathbf{x}_k) \cdot back(\mathbf{x}_{k+1})}{pred(\mathbf{x}_{k+1})} \cdot d\mathbf{x}_{k+1} & (\text{smoother}) \end{array} \tag{8.13}$$

Proof. The prediction equation (i) and the correction equation (ii) have already been proven. It remains to prove (iii). We have

$$
\begin{array}{lll}
 & \mathrm{back}\left(\mathbf{x}_k\right) & \\
= & \pi\left(\mathbf{x}_k|\mathbf{y}_{0:N}\right) & \\
\overset{(8.6)}{=} & \int \underbrace{\pi\left(\mathbf{x}_k|\mathbf{x}_{k+1},\mathbf{y}_{0:N}\right)} \cdot \pi\left(\mathbf{x}_{k+1}|\mathbf{y}_{0:N}\right)\cdot d\mathbf{x}_{k+1} & \left\{\pi(a|c)=\int \pi(a|b,c)\cdot\pi(b|c)\cdot db\right\} \\
 & \qquad\qquad \| & \\
\overset{(\text{Markov})}{=} & \int \overbrace{\pi\left(\mathbf{x}_k|\mathbf{x}_{k+1},\mathbf{y}_{0:k}\right)}\cdot \mathrm{back}\left(\mathbf{x}_{k+1}\right)\cdot d\mathbf{x}_{k+1} & \\
 & \qquad\qquad \| & \\
\overset{(\text{Bayes})}{=} & \int \overbrace{\frac{\pi\left(\mathbf{x}_{k+1}|\mathbf{x}_k,\mathbf{y}_{0:k}\right)\cdot\pi\left(\mathbf{x}_k|\mathbf{y}_{0:k}\right)}{\pi\left(\mathbf{x}_{k+1}|\mathbf{y}_{0:k}\right)}}\cdot \mathrm{back}\left(\mathbf{x}_{k+1}\right)\cdot d\mathbf{x}_{k+1} & \left\{\pi(a|b,c)=\pi(b|a,c)\cdot\frac{\pi(a|c)}{\pi(b|c)}\right\} \\
\overset{(\text{Markov})}{=} & \int \frac{\pi(\mathbf{x}_{k+1}|\mathbf{x}_k)\cdot\pi(\mathbf{x}_k|\mathbf{y}_{0:k})}{\mathrm{pred}(\mathbf{x}_{k+1})}\cdot \mathrm{back}\left(\mathbf{x}_{k+1}\right)\cdot d\mathbf{x}_{k+1} & \\
= & \underbrace{\pi\left(\mathbf{x}_k|\mathbf{y}_{0:k}\right)}_{\mathrm{bel}(\mathbf{x}_k)} \int \frac{\pi(\mathbf{x}_{k+1}|\mathbf{x}_k)}{\mathrm{pred}(\mathbf{x}_{k+1})}\cdot \mathrm{back}\left(\mathbf{x}_{k+1}\right)\cdot d\mathbf{x}_{k+1} &
\end{array}
$$

□

8.5 Kalman smoother

8.5.1 Equations of the Kalman smoother

Consider a linear discrete time system described by the state equation.

$$
\left\{\begin{array}{rcl} \mathbf{x}_{k+1} &=& \mathbf{A}_k\mathbf{x}_k+\mathbf{u}_k+\boldsymbol{\alpha}_k \\ \mathbf{y}_k &=& \mathbf{C}_k\mathbf{x}_k+\boldsymbol{\beta}_k \end{array}\right.
$$

where $\boldsymbol{\alpha}_k$ and $\boldsymbol{\beta}_k$ are Gaussian, white and independent.

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 9. *he smoothing density is*

$$
back\left(\mathbf{x}_k\right)=\pi\left(\mathbf{x}_k|\mathbf{y}_{0:N}\right)=\mathcal{N}\left(\mathbf{x}_k\left\|\hat{\mathbf{x}}_{k|N},\boldsymbol{\Gamma}_{k|N}\right.\right)
$$

where

$$
\begin{array}{rcl} \mathbf{J}_k &=& \boldsymbol{\Gamma}_{k|k}\cdot\mathbf{A}_k^T\cdot\boldsymbol{\Gamma}_{k+1|k}^{-1} \\ \hat{\mathbf{x}}_{k|N} &=& \hat{\mathbf{x}}_{k|k}+\mathbf{J}_k\cdot\left(\hat{\mathbf{x}}_{k+1|N}-\hat{\mathbf{x}}_{k+1|k}\right) \\ \boldsymbol{\Gamma}_{k|N} &=& \boldsymbol{\Gamma}_{k|k}-\mathbf{J}_k\cdot\left(\boldsymbol{\Gamma}_{k+1|k}-\boldsymbol{\Gamma}_{k+1|N}\right)\cdot\mathbf{J}_k^T. \end{array} \tag{8.14}
$$

Proof. See exercise 35 on page 93. □

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 $\hat{\mathbf{x}}_{k+1|k}, \hat{\mathbf{x}}_{k|k}, \mathbf{\Gamma}_{k+1|k}, \mathbf{\Gamma}_{k|k}, \hat{\mathbf{x}}_{k|N}$ are stored in the lists $\hat{\mathbf{x}}_k^{\texttt{pred}}, \hat{\mathbf{x}}_k^{\texttt{up}}, \hat{\mathbf{x}}_k^{\texttt{back}}, \mathbf{\Gamma}_k^{\texttt{pred}}, \mathbf{\Gamma}_k^{\texttt{up}}, \mathbf{\Gamma}_k^{\texttt{back}}$, where *pred*, *up*, *back* respectively mean *prediction*, *update*, *backward*. The quantities $\mathbf{u}(k), \mathbf{y}(k), \mathbf{\Gamma}_{\boldsymbol{\alpha}_k}, \mathbf{\Gamma}_{\boldsymbol{\beta}_k}, \mathbf{A}_k$ are also stored in lists. The direct, or *forward* part of the smoother is given by the algorithm FILTER below. It uses the Kalman function given page 47.

	Function FILTER $(\hat{\mathbf{x}}_0^{\text{pred}}, \mathbf{\Gamma}_0^{\text{pred}}, \mathbf{u}(k), \mathbf{y}(k), \mathbf{\Gamma}_{\boldsymbol{\alpha}_k}, \mathbf{\Gamma}_{\boldsymbol{\beta}_k}, \mathbf{A}_k, \mathbf{C}_k, k = 0, \ldots, N)$
1	For $k = 0$ to N
2	$\left(\hat{\mathbf{x}}_{k+1}^{\text{pred}}, \mathbf{\Gamma}_{k+1}^{\text{pred}}, \hat{\mathbf{x}}_k^{\text{up}}, \mathbf{\Gamma}_k^{\text{up}}\right) =$
3	KALMAN $\left(\hat{\mathbf{x}}_k^{\text{pred}}, \mathbf{\Gamma}_k^{\text{pred}}, \mathbf{u}(k), \mathbf{y}(k), \mathbf{\Gamma}_{\boldsymbol{\alpha}_k}, \mathbf{\Gamma}_{\boldsymbol{\beta}_k}, \mathbf{A}_k, \mathbf{C}_k\right)$
4	Return $\left(\hat{\mathbf{x}}_{k+1}^{\text{pred}}, \mathbf{\Gamma}_{k+1}^{\text{pred}}, \hat{\mathbf{x}}_k^{\text{up}}, \mathbf{\Gamma}_k^{\text{up}}, k = 0, \ldots, N\right)$

The *backward* part of the smoother, this is given by the following Algorithm SMOOTHER .

	Function SMOOTHER $(\hat{\mathbf{x}}_{k+1}^{\text{pred}}, \mathbf{\Gamma}_{k+1}^{\text{pred}}, \hat{\mathbf{x}}_k^{\text{up}}, \mathbf{\Gamma}_k^{\text{up}}, \mathbf{A}_k, k = 0, \ldots, N)$
1	$\hat{\mathbf{x}}_N^{\text{back}} = \hat{\mathbf{x}}_N^{\text{up}}$
2	$\mathbf{\Gamma}_N^{\text{back}} = \mathbf{\Gamma}_N^{\text{up}}$
3	For $k = N - 1$ downto 0
4	$\mathbf{J} = \mathbf{\Gamma}_k^{\text{up}} \cdot \mathbf{A}_k^{\text{T}} \cdot \left(\mathbf{\Gamma}_{k+1}^{\text{pred}}\right)^{-1}$
5	$\hat{\mathbf{x}}_k^{\text{back}} = \hat{\mathbf{x}}_k^{\text{up}} + \mathbf{J} \cdot \left(\hat{\mathbf{x}}_{k+1}^{\text{back}} - \hat{\mathbf{x}}_{k+1}^{\text{pred}}\right)$
6	$\mathbf{\Gamma}_k^{\text{back}} = \mathbf{\Gamma}_k^{\text{up}} - \mathbf{J} \cdot \left(\mathbf{\Gamma}_{k+1}^{\text{pred}} - \mathbf{\Gamma}_{k+1}^{\text{back}}\right) \cdot \mathbf{J}^{\text{T}}$
7	Return $\left(\hat{\mathbf{x}}_k^{\text{back}}, \mathbf{\Gamma}_k^{\text{back}}, k = 0, \ldots, N\right)$

Exercises

EXERCISE 30.– *Conditional and marginal densities*

See the correction video at https://youtu.be/joN7hHLz3oU

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 31.– *Weather forecast*

See the correction video at https://youtu.be/Qa8nkC7Hdeg

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 knowledge of the past will not bring any 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^{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 kinds of weather 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 Figure 8.3.

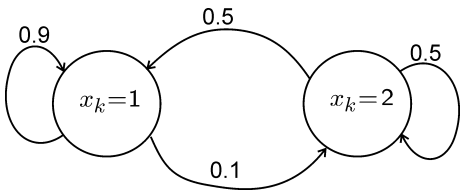

Figure 8.3: Markov chain describing the evolution of the weather in Brest

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 ℓ tends to infinity?

EXERCISE 32.– *Door robot*

See the correction video at https://youtu.be/J_0_-KTGpfM

In this exercise, we illustrate the Bayes filter in a system with a state $x \in \{0, 1\}$. It corresponds [9] 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

$\pi(x_{k+1}\|x_k, u_k)$	$u_k = -1$,	$u_k = 0$	$u_k = 1$
$x_k = 0$	δ_0	δ_0	$0.2\ \delta_0 + 0.8\ \delta_1$
$x_k = 1$	$0.8\ \delta_0 + 0.2\ \delta_1$	δ_1	δ_1

For simplicity, the dependencies of the δ_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.

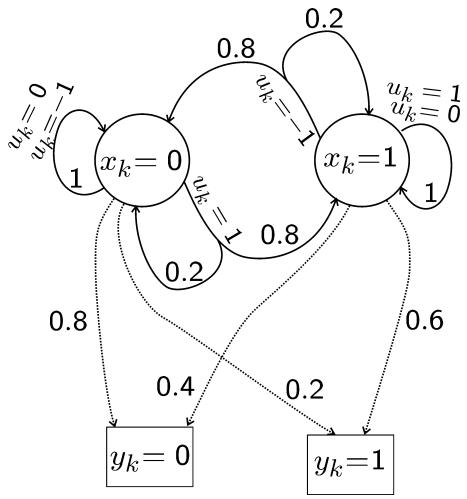

Figure 8.4: Graph representing the evolution and the observation of the door robot

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 Bayes filter.

Exercise 33.– *Robot in the forest*

See the correction video at https://youtu.be/C443JFbwBvg

The objective of this exercise if to give a graphical interpretation of the Bayes filter. We recall that the Bayes filter is given by the following equations

$$\begin{array}{rcl} \text{pred}\left(\mathbf{x}_{k+1}\right) & = & \int \pi\left(\mathbf{x}_{k+1}|\mathbf{x}_k,\mathbf{u}_k\right)\cdot \text{bel}\left(\mathbf{x}_k\right)\cdot d\mathbf{x}_k \\ \text{bel}\left(\mathbf{x}_k\right) & = & \frac{\pi(\mathbf{y}_k|\mathbf{x}_k)\cdot\text{pred}(\mathbf{x}_k)}{\int \pi(\mathbf{y}_k|\mathbf{x}_k)\cdot\text{pred}(\mathbf{x}_k)\cdot d\mathbf{x}_k}. \end{array}$$

Figure 8.5 represents a robot with a scalar state x which corresponds to its position on a horizontal line.

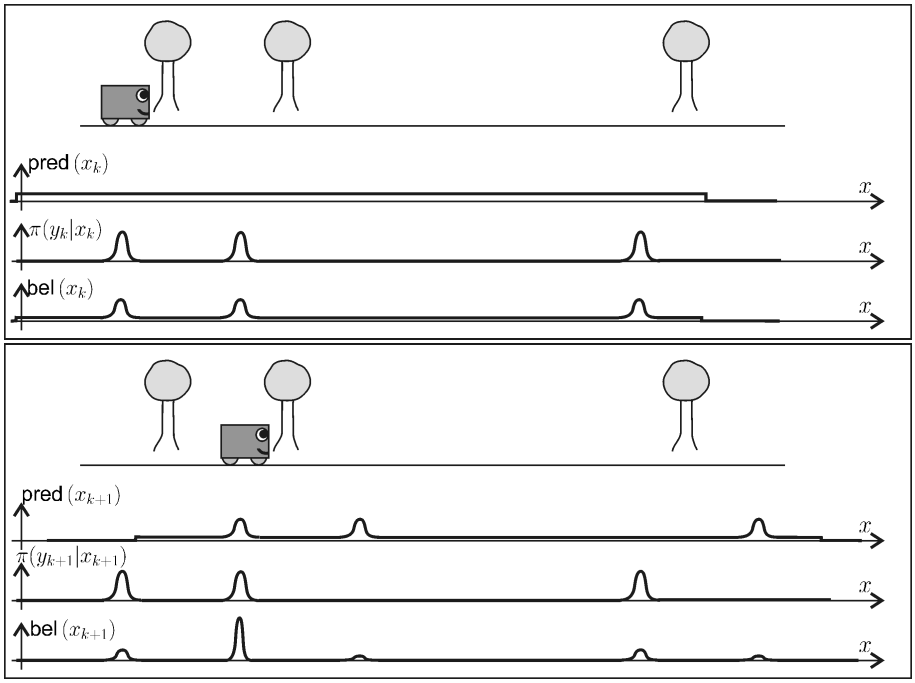

Figure 8.5: Illustration of the Bayes filter

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 increases 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}) in the same way 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 34.– *Bayes rule with Kalman*

See the correction video at https://youtu.be/aAM1nMhOKcw

Consider two Gaussian random variables $x \sim \mathcal{N}(1,1)$ and $b \sim \mathcal{N}(0,1)$, *i.e.* the expectations of x,b are $\bar{x}=1$, $\bar{b}=0$ and the variances (*i.e*, the covariance matrix in the scalar case) of x, b are $\sigma_x^2=1$, $\sigma_b^2=1$.

1) Define $y=x+b$. Give the expression of probability 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 35.– *Derivation of the Kalman smoother*

See the correction video at https://youtu.be/aJKNEstmIfU

1) Consider two normal vectors $\mathbf{a}$ and $\mathbf{b}$. Show that

$$\left.\begin{array}{rcl} \pi(\mathbf{a}) & = & \mathcal{N}\left(\mathbf{a}\,\|\hat{\mathbf{a}},\boldsymbol{\Gamma}_{\mathbf{a}}\right) \\ \pi(\mathbf{b}|\mathbf{a}) & = & \mathcal{N}\left(\mathbf{b}\,\|\mathbf{J}\mathbf{a}+\mathbf{u},\mathbf{R}\right) \end{array}\right\} \implies \pi(\mathbf{b}) = \mathcal{N}\left(\mathbf{b}\,\big\|\mathbf{J}\hat{\mathbf{a}}+\mathbf{u},\mathbf{R}+\mathbf{J}\cdot\boldsymbol{\Gamma}_{\mathbf{a}}\cdot\mathbf{J}^{\mathrm{T}}\right). \tag{8.15}$$

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

$$\pi\left(\mathbf{x}_k|\mathbf{x}_{k+1},\mathbf{y}_{0:N}\right) = \mathcal{N}\left(\mathbf{x}_k\,\big\|\hat{\mathbf{x}}_{k|k}+\mathbf{J}_k\left(\mathbf{x}_{k+1}-\hat{\mathbf{x}}_{k+1|k}\right),\boldsymbol{\Gamma}_{k|k}-\mathbf{J}_k\cdot\boldsymbol{\Gamma}_{k+1|k}\cdot\mathbf{J}_k^T\right) \tag{8.16}$$

where

$$\mathbf{J}_k = \boldsymbol{\Gamma}_{k|k}\cdot\mathbf{A}_k^T\cdot\boldsymbol{\Gamma}_{k+1|k}^{-1}.$$

3) From the two previous questions, show that the smoothing density is

$$\mathrm{back}\left(\mathbf{x}_k\right) = \pi\left(\mathbf{x}_k|\mathbf{y}_{0:N}\right) = \mathcal{N}\left(\mathbf{x}_k\,\big\|\hat{\mathbf{x}}_{k|N},\boldsymbol{\Gamma}_{k|N}\right)$$

where

$$\left.\begin{array}{rcl} \hat{\mathbf{x}}_{k|N} & = & \hat{\mathbf{x}}_{k|k}+\mathbf{J}_k\cdot\left(\hat{\mathbf{x}}_{k+1|N}-\hat{\mathbf{x}}_{k+1|k}\right) \\ \boldsymbol{\Gamma}_{k|N} & = & \boldsymbol{\Gamma}_{k|k}-\mathbf{J}_k\cdot\left(\boldsymbol{\Gamma}_{k+1|k}-\boldsymbol{\Gamma}_{k+1|N}\right)\cdot\mathbf{J}_k^{\mathrm{T}}. \end{array}\right.$$

EXERCISE 36.– *SLAM*

See the correction video at http://youtu.be/KHkP4vf_th0

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 ϕ,θ,ψ), 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 (*i.e.*, its distance to the seabed).

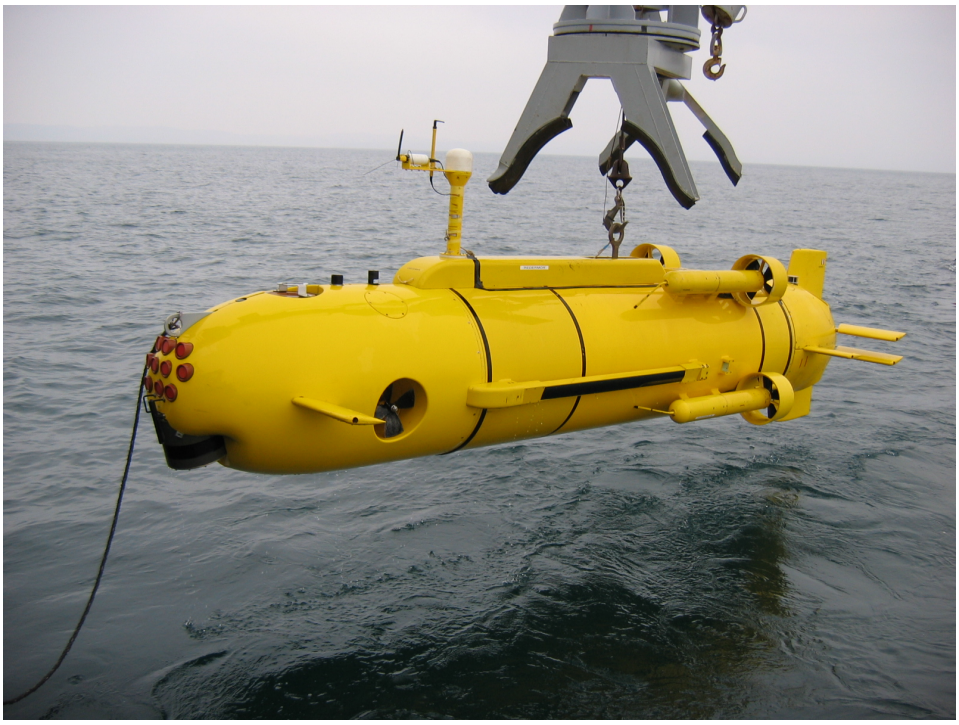

Figure 8.6: The *Redermor*, built by GESMA (Groupe d'Etude Sous-Marine de l'Atlantique), right before diving into the water

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:

$$\dot{\mathbf{p}}(t) = \mathbf{R}(\varphi(t), \theta(t), \psi(t)) \cdot \mathbf{v}_r(t)$$

with $\mathbf{R}(\varphi, \theta, \psi)$ the Euler matrix given by:

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

2) The angles φ, θ, ψ 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 seconds with a variance of $\sigma_v^2 = 1 m^2 s^{-2}$. 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 7.1 or in Exercise 11 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 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	1054	1092	1374	1748	3038	3688	4024	4817	5172	5232	5279	5688
i	1	2	1	0	1	5	4	3	3	4	5	1
$r_i(t)$	52.4	12.47	54.4	52.7	27.73	27.0	37.9	36.7	37.37	31.03	33.5	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 function $(\mathbf{y}, \mathbf{C}_k, \boldsymbol{\Gamma}_{\boldsymbol{\beta}}) = \mathbf{g}(k)$ 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 β_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 37.– *Particle filter*

See the correction video at https://youtu.be/xKaXff5yzwI

We consider a robot at position $\mathbf{x} = (x_1, x_2)$ that has to be localized. The robot is equipped with a compass, so that we can assume that its heading θ is known. We describe the evolution of the

robot by the relation

$$\mathbf{x}_{k+1} = \mathbf{x}_k + dt \cdot \begin{pmatrix} \cos\theta_k \\ \sin\theta_k \end{pmatrix} + \boldsymbol{\alpha}_k$$

where the sampling time is taken as dt, where $\boldsymbol{\alpha}_k$ is the state noise. The covariance matrix for $\boldsymbol{\alpha}_k$ is taken as

$$\boldsymbol{\Gamma}_{\boldsymbol{\alpha}} = 0.1 \cdot \begin{pmatrix} dt & 0 \\ 0 & dt \end{pmatrix}.$$

In the environment, we have 3 landmarks at position:

j	1	2	3
$\mathbf{m}(j)$	$\begin{pmatrix} 3 \\ 8 \end{pmatrix}$	$\begin{pmatrix} 2 \\ 6 \end{pmatrix}$	$\begin{pmatrix} 4 \\ 11 \end{pmatrix}$

At each time $t_k = k \cdot dt$, the robot measures its distance to each landmark with an error. The corresponding observation equation is

$$\mathbf{y}_k = \begin{pmatrix} \|\mathbf{x} - \mathbf{m}(1)\| \\ \|\mathbf{x} - \mathbf{m}(2)\| \\ \|\mathbf{x} - \mathbf{m}(3)\| \end{pmatrix} + \boldsymbol{\beta}_k$$

where the components for $\boldsymbol{\beta}_k$ are all independent with a variance of 1. All noises $\boldsymbol{\alpha}_k$, $\boldsymbol{\beta}_k$ are white, stationary, centered and Gaussian. In this exercise, we want to estimate the probability density functions

$$\begin{array}{rcl} \text{pred}(\mathbf{x}_k) & = & \pi(\mathbf{x}_k|\mathbf{y}_{0:k-1}) \\ \text{bel}(\mathbf{x}_k) & = & \pi(\mathbf{x}_k|\mathbf{y}_{0:k}). \end{array}$$

by a weighted set of N samples, called the *cloud*,

$$(\mathcal{P}, \mathcal{W})(k) = \left\{ (w_k^{(1)}, \mathbf{x}_k^{(1)}), \ldots, (w_k^{(N)}, \mathbf{x}_k^{(N)}) \right\},$$

where $\sum_\ell w_k^{(\ell)} = 1$. The weight $w_k^{(\ell)}$ is an approximation of the probability for the state $\mathbf{x}(k)$ to be equal to $\mathbf{x}_k^{(\ell)}$. We define the set of particles $\mathcal{P}$ and the set of of weights $\mathcal{W}$ as

$$\begin{array}{rcl} \mathcal{P} & = & \left\{ \mathbf{x}_k^{(1)}, \ldots, \mathbf{x}_k^{(N)} \right\} \\ \mathcal{W} & = & \left\{ w_k^{(1)}, \ldots, w_k^{(N)} \right\}. \end{array}$$

For the simulations, we will take $\mathbf{x}(0) = (0, 0)$, $\theta(t) = 0.2 \cdot t$, $t_k = k \cdot dt$, $dt = 0.1$ and $N = 2000$.

1) Assume that at time $k = 0$, we know that $\text{pred}(\mathbf{x}_k)$ is uniformly distributed inside the box $[-15, 15] \times [-15, 15]$. Generate the corresponding cloud $(\mathcal{P}, \mathcal{W})$ and draw the particles with a width proportional to their weights.

2) At time $t = 0$, the robot at position $\mathbf{x}(0)$ measures $\mathbf{y}_0$ that you should generate from the observation equation. Using the rule

$$\text{bel}(\mathbf{x}_k) = \frac{\pi(\mathbf{y}_k|\mathbf{x}_k) \cdot \text{pred}(\mathbf{x}_k)}{\pi(\mathbf{y}_k|\mathbf{y}_{0:k-1})}$$

update the weights $\mathcal{W}$ to get an estimation of $\text{bel}(\mathbf{x}_0)$. An illustration of the cloud you should obtain is given by Figure 8.7(b).

3) Generate a *resampled* cloud $(\mathcal{P}', \mathcal{W}')$ from the previous one $(\mathcal{P}, \mathcal{W})$ which also represents $\text{bel}(\mathbf{x}_0)$. Now, with the resampled cloud, all weights should be equal to $\frac{1}{N}$, as illustrated by Figure 8.7(c).

4) Using the prediction equation

$$\text{pred}(\mathbf{x}_{k+1}) = \int \pi(\mathbf{x}_{k+1}|\mathbf{x}_k) \cdot \text{bel}(\mathbf{x}_k) \cdot d\mathbf{x}_k$$

Generate a cloud $(\mathcal{P}'', \mathcal{W}'')$ which approximates $\text{pred}(\mathbf{x}_1)$ from the resampled cloud $(\mathcal{P}', \mathcal{W}')$. See Figure 8.7(d).

5) Write a program which simulates the robot for $t \in [0, 5]$. Propose a particle filter for localizing the robot. Show the evolution of the cloud associated with $\text{pred}(\mathbf{x}_k)$.

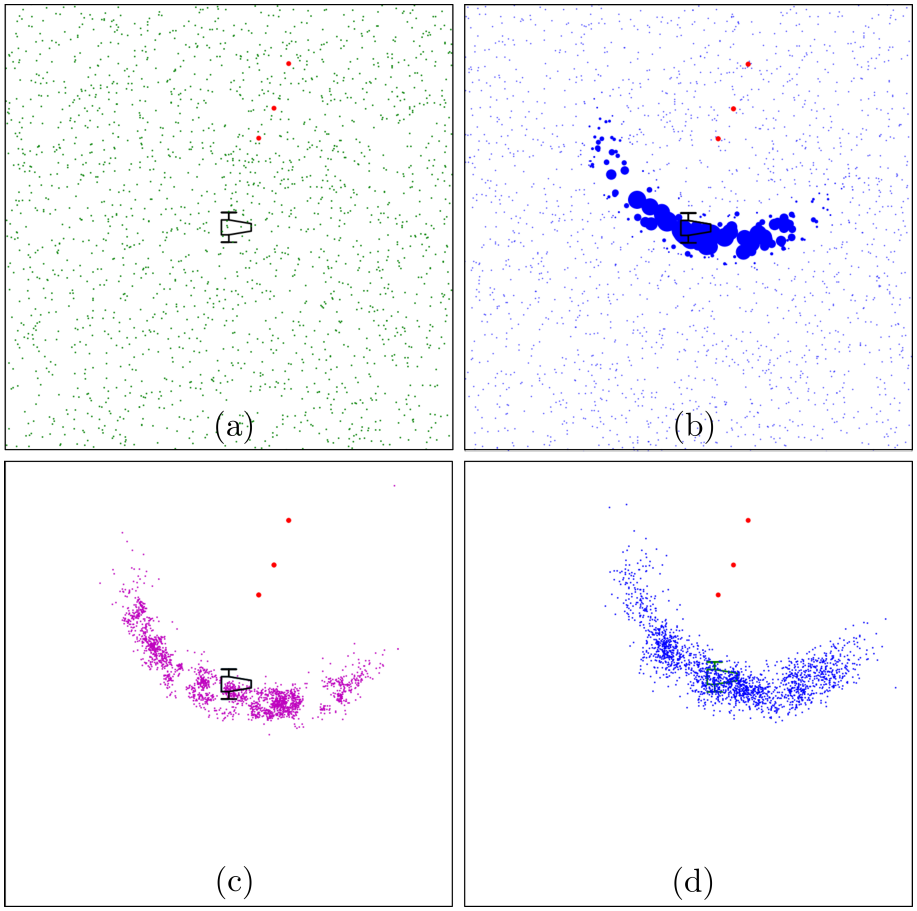

Figure 8.7: (a) pred($\mathbf{x}_0$); (b) bel($\mathbf{x}_0$): the particles have different weights; (c) bel($\mathbf{x}_0$) after resampling: all particles have the same weight; (d) pred($\mathbf{x}_1$) : the robot has moved forward

Bibliography

[1] V. Creuze. Robots marins et sous-marins ; perception, modélisation, commande. *Techniques de l'ingénieur*, 2014.

[2] L. Jaulin. *Représentation d'état pour la modélisation et la commande des systèmes (Coll. Automatique de base)*. Hermès, London, 2005.

[3] L. Jaulin. *Automation for Robotics*. ISTE editions, 2015.

[4] L. Jaulin, M. Kieffer, E. Walter, and D. Meizel. Guaranteed robust nonlinear estimation with application to robot localization. *IEEE Transactions on systems, man and cybernetics; Part C Applications and Reviews*, 32(4):374–382, 2002.

[5] E. Walter. *Numerical Methods and Optimization ; a Consumer Guide*. Springer, London, 2014.

[6] R. E. Kalman. Contributions to the theory of optimal control. *Bol. Soc. Mat. Mex.*, 5:102–119, 1960.

[7] P. de Larminat. *Automatique, commande des systèmes linéaires*. Hermès, Paris, France, 1993.

[8] V. Drevelle. *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.

[9] S. Thrun, W. Bugard, and D. Fox. *Probabilistic Robotics*. MIT Press, Cambridge, M.A., 2005.