RobMOOC ($3^{ième}$ édition)
Un MOOC sur la commande non-linéaire des robots

Luc Jaulin

January 7, 2018

Introduction

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

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

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

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

that rely on the utilization of the robot model. This approach will make use of the concept of *feedback linearization* which will be introduced and illustrated through numerous examples. Chapter 3 presents more pragmatic methods which do not use the state model of the robot and which will be referred to as *without model* or *mimetic*. The approach uses a more intuitive representation of the robot and is adapted to situations in which the robots are relatively simple to remotely control, such as in the case of cars, sailing boats or airplanes. Chapter 4 looks at *guidance*, which is placed at a higher level than control. In other words, it focuses on guiding and supervising the system which is already under control by the tools presented in Chapters 2 and 3. There will therefore be an emphasis on finding the instruction to give to the controller in order for the robot to accomplish its given task. The guidance will then have to take into account the knowledge of the surroundings, the presence of obstacles and the roundness of the Earth. The non-linear control and guidance methods require good knowledge of the state variables of the system, such as those which define the position of the robot.

Chapter 1

Three-dimensional modeling

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

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

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

This is only a part of this chapter. See [JAU 13] for the full text.

Chapter 2

Feedback linearization

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

2.1 Controlling an integrator chain

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

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

2.1.1 Proportional-derivative controller

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

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

where w is the wanted setpoint for y. Let us note that w may depend on time. The fact that this controller requires the differentials of y is not a problem within the frame defined by the feedback linearization. Indeed, all of these derivatives can be described as analytic functions of the state $\mathbf{x}$ of

the system and the input $\mathbf{u}$. Concerning the setpoint $w(t)$, it is chosen by the user and an analytic expression of $w(t)$ may be assumed to be known (for instance $w(t) = \sin(t)$). Thus, calculating the differentials of w is done in a formal manner and no sensitivity of the differential operator with respect to the noise has to be feared.

The feedback system is described by the differential equation:

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

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

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

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

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

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

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

Whence:

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

The controller obtained is then given by:

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

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

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

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

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

2.1.2 Proportional-integral-derivative controller

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

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

The feedback system is described by the differential equation:

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

Hence, by differentiating once:

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

The characteristic polynomial:

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

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

2.2 Introductory example

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

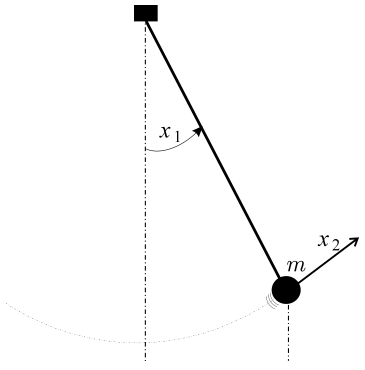

Figure 2.1: Simple pendulum with state vector $\mathbf{x} = (x_1, x_2)$

Its state representation is assumed to be:

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

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

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

Let us choose:

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

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

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

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

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

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

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

Which yields:

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

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

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

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

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

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

2.3 Principle of the method

2.3.1 Principle

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

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

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

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

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

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

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

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

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

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

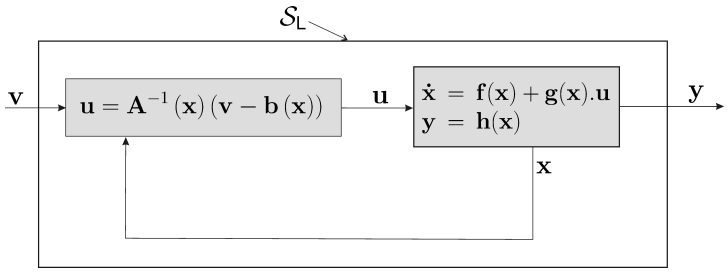

Figure 2.2: The nonlinear system, once transformed, becomes linear and decoupled ; and therefore becomes easy to control

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

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

We will therefore have:

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

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

2.3.2 Relative degree

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

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

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

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

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

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

$$\begin{cases} \dot{x} = & xu + x^3 \\ y = & 2x. \end{cases}$$

We have:

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

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

2.3.3 Differential delay matrix

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

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

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

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

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

2.3.4 Singularities

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

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

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

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

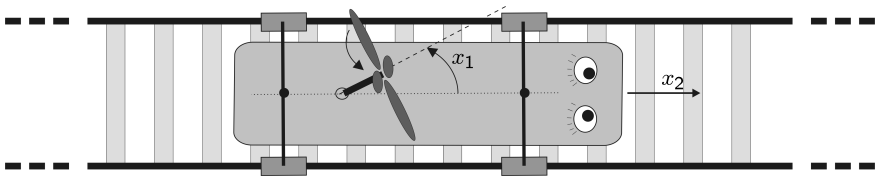

Figure 2.3: Robot cart propelled by a ventilator

The state equations that model this system are given by:

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

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

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

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

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

and therefore, the linearizing controller is given by:

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

The feedback system therefore has the following equation:

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

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

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

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

2.4 Cart

2.4.1 First model

Consider a cart described by the following state equations:

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

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

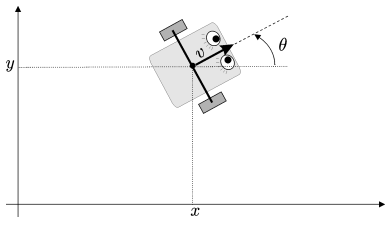

Figure 2.4: Cart (also called Dubin's car)

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

$$\left\{\begin{array}{lcl}
x_d(t) &=& R\sin(f_1 t) + R\sin(f_2 t) \\
y_d(t) &=& R\cos(f_1 t) + R\cos(f_2 t)
\end{array}\right.$$

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

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

If we take as input:

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

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

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

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

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

The transformed system then obeys the following differential equations:

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

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

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

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

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

with:

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

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

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

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

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

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

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

2.4.2 Second model

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

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

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

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

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

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

We have:

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

in other words:

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

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

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

in order to have a feedback system of the form:

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

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

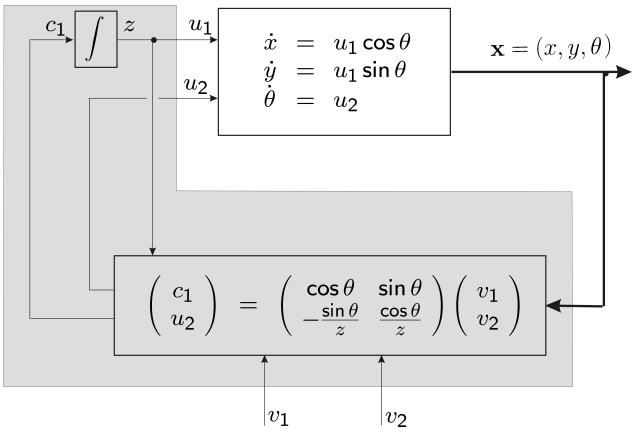

Figure 2.5: Dynamic feedback linearization

In order to have all the poles at -1, we need to take (see Equation 2.2 page 8):

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

The state equations of the controller are therefore:

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

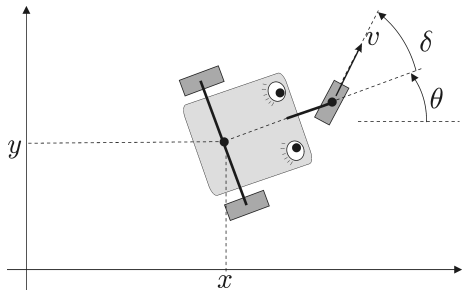

Figure 2.6: Tricycle robot to be controlled

2.5 Controlling a tricycle

2.5.1 Speed and heading control

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

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

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

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

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

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

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

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

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

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

and therefore becomes linear and decoupled. We now have two decoupled monovariate systems. The first, of order 1, can be stabilized by a proportional controller. As for the second, second-order

system, a proportional-derivative controller is best adapted. If $\mathbf{w} = (w_1,\ w_2)$ represents the setpoint for $\mathbf{y}$, this controller is expressed by:

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

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

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

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

Remark 2.3.– *Since:*

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

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

2.5.2 Position control

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

$$\begin{cases} \dot{x} &= v\cos\delta\cos\theta \\ \ddot{x} &= \dot{v}\cos\delta\cos\theta - v\dot{\delta}\sin\delta\cos\theta - v\dot{\theta}\cos\delta\sin\theta \\ &= u_1\cos\delta\cos\theta - vu_2\sin\delta\cos\theta - v^2\sin\delta\cos\delta\sin\theta \\ \dot{y} &= v\cos\delta\sin\theta \\ \ddot{y} &= \dot{v}\cos\delta\sin\theta - v\dot{\delta}\sin\delta\sin\theta + v\dot{\theta}\cos\delta\cos\theta \\ & \quad u_1\cos\delta\sin\theta - vu_2\sin\delta\sin\theta + v^2\sin\delta\cos\delta\cos\theta \end{cases}$$

Thus:

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

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

2.5.3 Choosing another output

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

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

By differentiating once, we have:

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

Differentiating again, we obtain:

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

And therefore:

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

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

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

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

2.6 Sailboat

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

state equations [JAU 05] are given by:

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

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

- position coordinates, in other words the x, y coordinates of the sailboat's center of gravity, the orientation θ, and the angles δ_s and δ_r of the sail and the rudder ;
- kinematic coordinates v and ω representing respectively the speed of the center of gravity and the angular speed of the boat.

The inputs u_1 and u_2 of the system are the differentials of the angles δ_s and δ_r. The indices «s »and «r »refer respectively to the sail and the rudder.

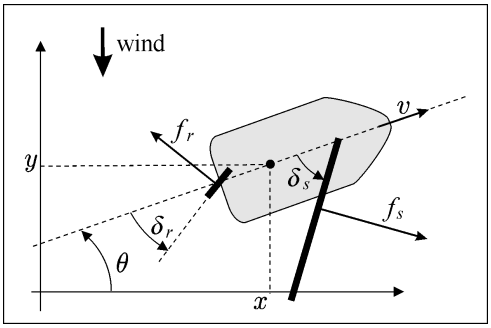

Figure 2.7: Sailing robot to be controlled

2.6.1 Polar curve

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

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

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

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

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

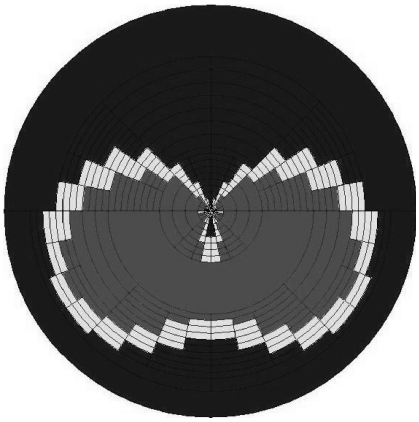

Figure 2.8: Internal frame (in light grey) and external frame (in dark grey) of the polar curve

2.6.2 Differential delay

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

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

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

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

2.6.3 The method of feedback linearization

Let us recall that the outputs (these are actually setpoint variables) chosen are the sail opening $y_1 = \delta_s$ and the heading $y_2 = \theta$. In order to apply a feedback linearization method, we first of all

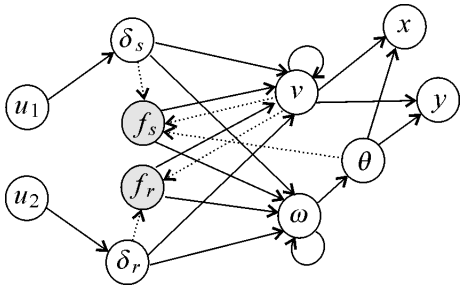

Figure 2.9: Graph of the differential delays for the sailing robot

need to differentiate the outputs as many times as the relative degree requires it, in other words three times for θ and once for δ_s. By looking at the differential dependency graph, we can observe that in order to express $\dddot{\theta}$ in function of $\mathbf{x}$ and $\mathbf{u}$, we need to do the same with $\ddot{\omega}, \dot{\omega}, \dot{\delta}_r, \dot{\delta}_s, \dot{f}_r, \dot{f}_s, \dot{v}$. This yields:

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

We have:

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

However:

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

And thus we have a relation of the form:

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

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

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

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

$$\mathcal{S}_{\mathrm{L}} : \left\{ \begin{array}{rcl} \dot{y}_1 & = & v_1 \\ \ddot{y}_2 & = & v_2 \end{array} \right. \tag{2.12}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

2.6.4 Polar curve control

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

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

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

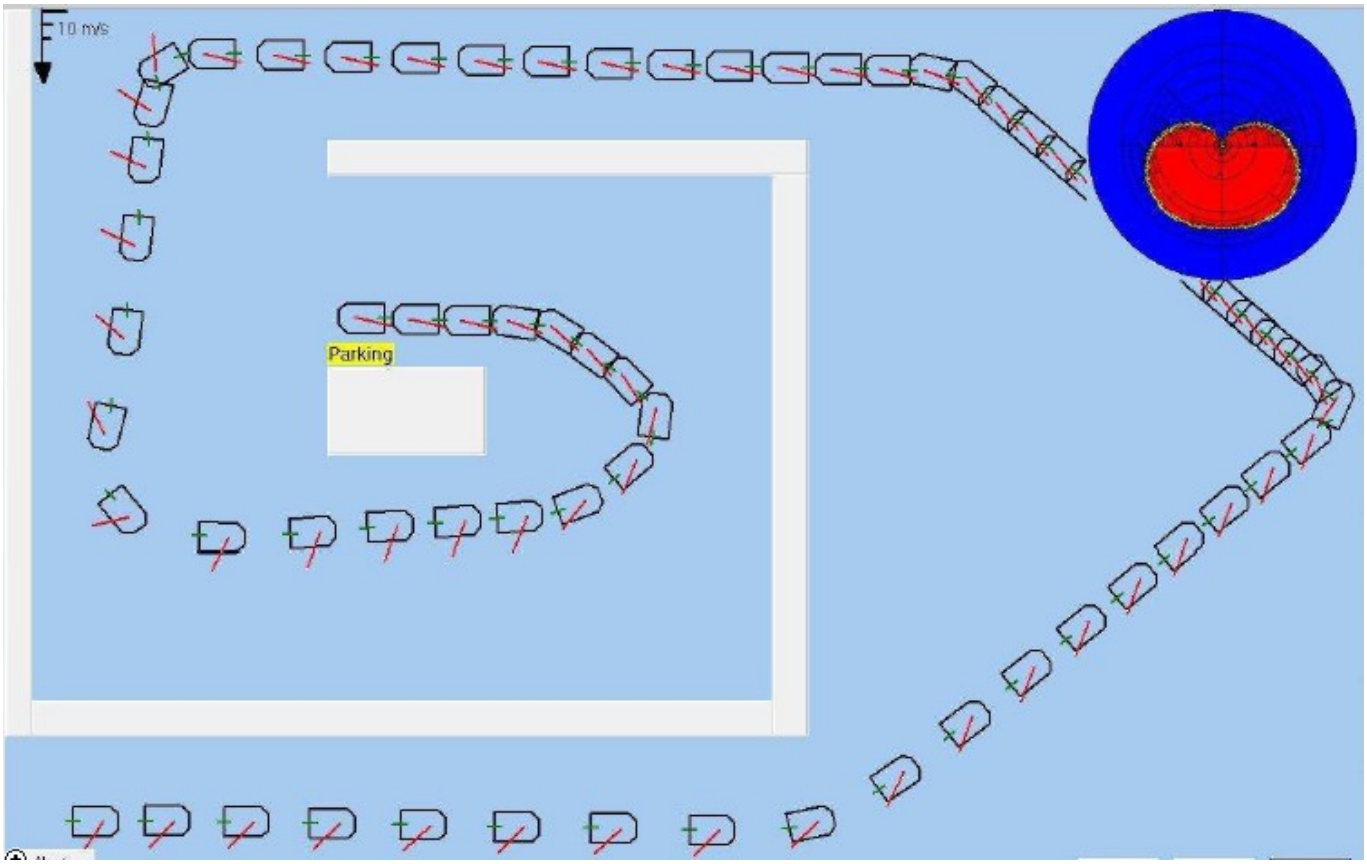

Figure 2.10: By using a linearizing controller, the robot docks in its place in the harbor ; the polar curve is represented on the top right corner

2.7 Sliding mode

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

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

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

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

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

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

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

For simplicity and without loss of generality, let us assume that we have a single input and a single output, *i.e.*, $\dim u = \dim y = 1$. We thus have

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

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

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

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

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

The corresponding feedback is

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

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

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

or equivalently

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

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

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

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

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

One important feature of sliding mode is that we do not need to insert an integral effect to compensate constant perturbations and the controller is robust with respect to many types of perturbation. Indeed, any perturbation that would make escape the system from the sliding surface yields an immediate and strong reaction from the controller that bring it back to the surface. On the other hand, a sliding mode control generates a behavior with a high-frequency and non-deterministic switching control signal that causes the system to *chatter* in a small neighborhood of the sliding surface.

2.8 Kinematic model and dynamic model

2.8.1 Principle

The dynamic models for the robots are of the form:

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

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

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

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

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

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

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

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

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

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

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

- the linearizing controller developed in this chapter requires using a reliable model such as a kinematic model. If the coefficients (which are not measured) are not well known (such as in the case of dynamic systems), the linearizing controller will not work in practice ;
- the kinematic model is easier to put into equations. It is not necessary to have a dynamic model to obtain the latter ;
- the servo-motors (see paragraph 2.8.3) are cheap and easy to find. They incorporate this high-gain controller. We may see them as mechanical operational amplifiers.

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

2.8.2 Example of the inverted rod pendulum

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

2.8.2.1 Dynamic model

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

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

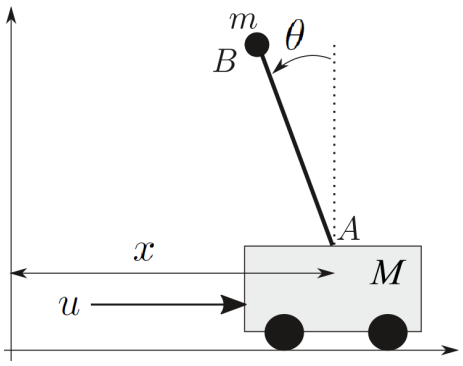

Figure 2.11: Inverted rod pendulum to be modeled and controlled

2.8.2.2 Kinematic model

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

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

Therefore:

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

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

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

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

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

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

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

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

Finally, the kinematic model is written as:

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

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

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

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

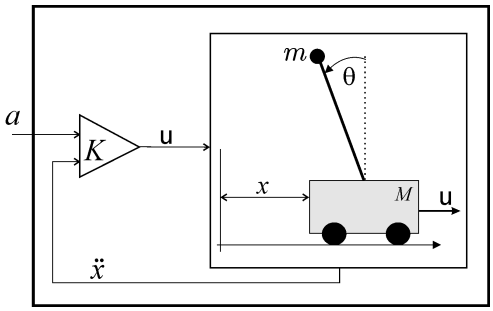

Figure 2.12: The inverted rod pendulum, looped by a high gain K, behaves like a kinematic model

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

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

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

We will therefore take:

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

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

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

And finally:

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

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

2.8.3 Servo-motors

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

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

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

Exercises

Exercise 2.1.– *Crank*

Let us consider the manipulator robot, or *crank* of Figure 2.13 (on the left). This robot is composed of two arms of length ℓ_1 and ℓ_2. Its two degrees of freedom denoted by x_1 and x_2 are represented on the figure. The inputs u_1, u_2 of the system are the angular speeds of the arms (in other words $u_1 = \dot{x}_1, u_2 = \dot{x}_2$). We will take as output the vector $\mathbf{y} = (y_1, y_2)$ corresponding to the end of the second arm.

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

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

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

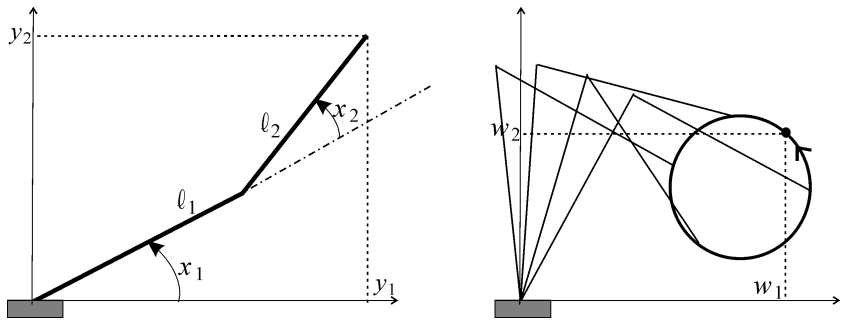

Figure 2.13: Manipulator robot whose end effector must follow a circle

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

3) Study the singularities of the control.

4) Let us consider the case $\ell_1 = \ell_2$, $\mathbf{c} = (3, 4)$ and $r = 1$. For which values of ℓ_1 are we certain to be able to move freely on the target circle, without encountering singularities ?

5) Write a program illustrating this control law.

Exercise 2.2.– *The three pools*

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

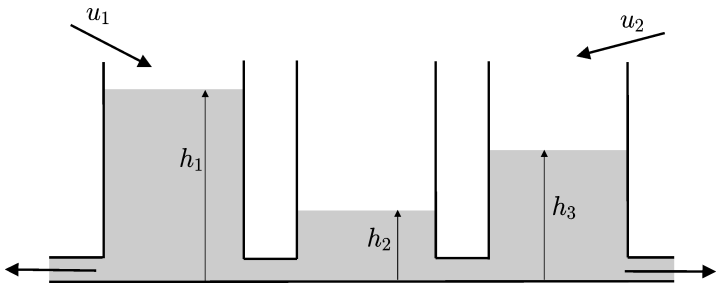

Figure 2.14: System composed of three pools containing water and connected by two channels

This system is described by the following state equations:

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

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

1) Propose a feedback that would make the system linear and decoupled.
2) Propose a proportional-integral controller for the linearized system.
3) Give the state equations of the obtained controller.
4) Write a program that simulates the system and its control law.

Exercise 2.3.– *Train robot*

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

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

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

1) Calculate $\ddot{x}_a, \ddot{y}_a$ in function of $x_a, y_a, v_a, \theta_a, u_{a1}, u_{a2}$.
2) Propose a controller that allows to follow the trajectory:

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

with $\omega = 0.1$, $L_x = 15$ and $L_y = 7$. A feedback linearization method must be used for this. Illustrate the behavior of your controller with a program.

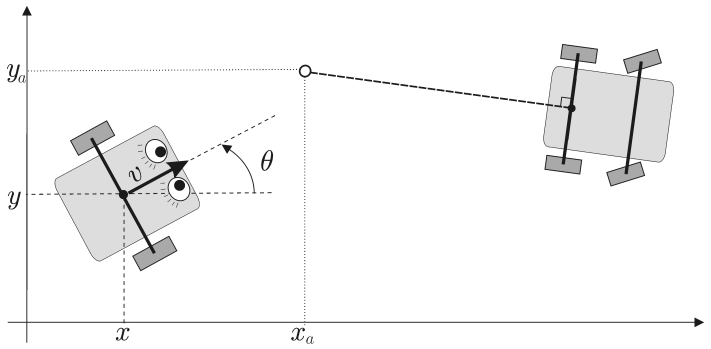

Figure 2.15: Our robot (with eyes) following a vehicle (here a car) whose state equations are unknown. This car has an imaginary attachment point (small white circle) that we must attach to

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

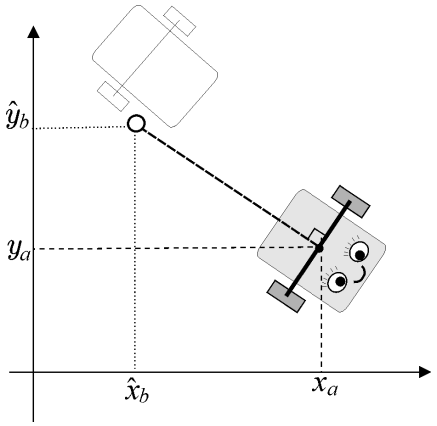

Figure 2.16: Robot B (dotted) has to follow robot A

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

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

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

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

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

Let us consider the underwater robot already discussed in Exercise **??**. This robot is described

by the following state equations:

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

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

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

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

Exercise 2.5.– *Reaction wheel pendulum*

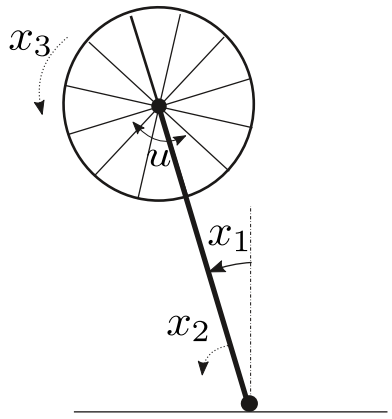

Figure 2.17: Reaction wheel pendulum

The Reaction Wheel Pendulum [SPO 01], as shown in Figure 2.17, is a physical pendulum with a disk attached to the end. The disk can rotate and is actuated by a motor. The coupling torque generated by the angular acceleration of the disk can be used to actively control the pendulum. We assume that the system can be described by the following equations

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

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

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

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

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

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

3) Taking as an output of the form

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

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

Exercise 2.6.– *Pursuit*

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

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

In this exercise, robot 1 tries to follow robot 2 (see Figure 2.18).

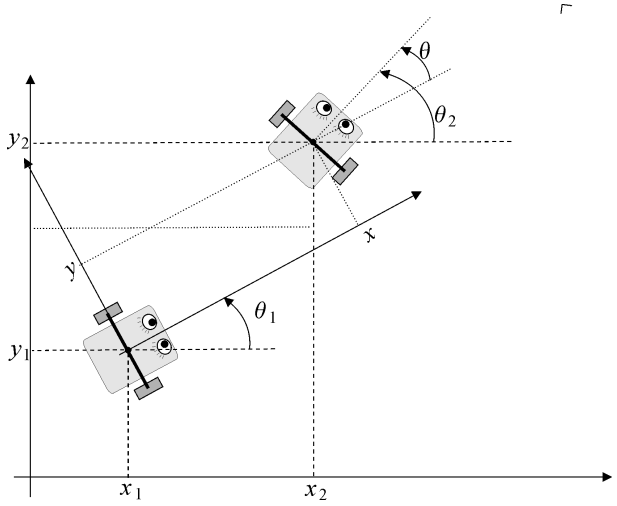

Figure 2.18: Robot 1 is in pursuit of robot 2

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

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

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

3) Study the singularities of this controller.

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

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

Consider the underwater robot represented on Figure 2.19.

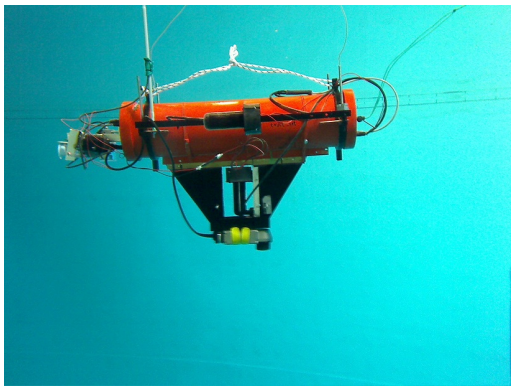

Figure 2.19: The SAUCISSE robot in a pool

This is the SAUCISSE robot, built by students of the ENSTA Bretagne for the SAUC'E competition (*Student Autonomous Underwater Challenge Europe*). It includes three propellers. Propellers 1 and 2 on the left and the right are able to act on the speed of the robot and its angular speed. Propeller 3 acts on the depth of the robot. This robot is stable in roll and pitch and we will assume that its angles of bank φ and elevation θ are always zero. The state equations of the robot are the following:

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

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

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

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

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

$$\left\{ \begin{array}{rcl} \ddot{u}_1 & = & a_1 \\ u_2 & = & a_2 \\ u_3 & = & a_3 \end{array} \right. \tag{2.21}$$

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

4) Perform a feedback linearization of the delayed system.

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

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

Consider the cart described by

$$\left\{ \begin{array}{rcl} \dot{x}_1 & = & x_4 \cos x_3 \\ \dot{x}_2 & = & x_4 \sin x_3 \\ \dot{x}_3 & = & u_1 \\ \dot{x}_4 & = & u_2 \end{array} \right.$$

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

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

$$\mathbf{y}_d(t) = 10 \cdot \left(\begin{array}{c} \cos t \\ \sin 3t \end{array} \right).$$

Illustrate the behavior of your controller.

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

EXERCISE 2.9.– *Group of robots*

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

$$\left\{ \begin{array}{rcl} \dot{x}_1 & = & x_4 \cos x_3 \\ \dot{x}_2 & = & x_4 \sin x_3 \\ \dot{x}_3 & = & u_1 \\ \dot{x}_4 & = & u_2 \end{array} \right.$$

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

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

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

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

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

Exercise 2.10.– *Convoy*

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

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

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

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

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

with $\omega = 0.1$, $L_x = 20$ and $L_y = 5$. Illustrate the behavior of the control with a sampling time $dt = 0.03\,\text{sec}$.

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

Exercise 2.11.– *Buoy*

We consider an underwater buoy corresponding to a cube of width ℓ which is immersed in the water (see Figure 2.20) of density ρ_0. The buoy at a depth d moves up and down at a speed equal to v. The total force applied to the buoy is

$$f = \underbrace{mg}_{f_g\downarrow} - \underbrace{\rho_0 g \ell^2 \cdot \max(0, \ell + \min(d, 0))}_{f_a\uparrow} - \underbrace{\frac{1}{2}\rho_0 v \cdot \mid v \mid \ell^2 c_x}_{f_d\updownarrow}$$

where

- f_g is the gravity force, m is the mass of the buoy and g the gravity.
- f_d is the drag force. For a cube we can consider that $c_x \simeq 1.05$.
- f_a is the Archimedes' force. The buoyant force is exerted upward on the body, whether fully or partially submerged. It is equal to the weight of the water that the body displaces. The volume displaced water is $\ell^2 \cdot max\left(0, \ell + min(d, 0)\right)$.

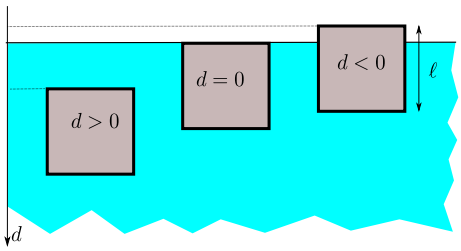

Figure 2.20: Underwater buoy controlled by a ballast

For the buoy, a ballast allows us to control the density of the buoy which is $(1 + \beta b)\,\rho_0$, where $\beta = 0.1$. The derivative u of the buoyancy variable b can be fixed using a DC-motor controlling a piston. We assume that both u and b belong to $[-1, 1]$. When $b = 0$, the density of the buoy is exactly that of the water ρ_0. When $b > 0$, the buoy sink and when $b < 0$ the buoy surfaces.

1) Write the state equations of the system $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, u)$. The state vector will be taken as $\mathbf{x} = (d, v, b)$.

2) Assume that the buoy is immerged, find a sliding mode controller so that the buoy as a depth $d\,(t)$ equal to the desired depth $d_0\,(t)$.

3) Illustrate the behavior of your controller using a simulation. First control the buoy at a depth $d_0(t) = 5m$. Then control the buoy at $d_0(t) = 3 + \sin(t/2)$.

Chapter 3

Model-free control

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

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

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

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

3.1 Model-free control of a robot cart

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

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

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

3.1.1 Proportional heading and speed controller

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

- For speed control, we take:

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

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

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

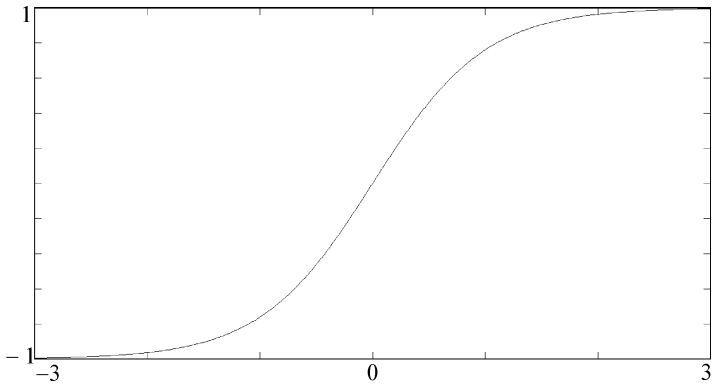

Figure 3.1: Hyperbolic tangent function used as saturation function

- For the heading control, we take:

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

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

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

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

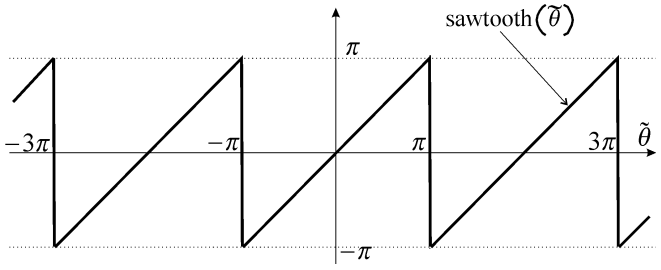

Figure 3.2: *Sawtooth* function used to avoid the jumps in the heading control

We may summarize this controller by:

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

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

3.1.2 Proportional-derivative heading controller

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

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

The quantity θ may be obtained by a compass, for example. As for $\dot{\theta}$, it is generally obtained by a gyro. Low-cost robots do not always have a gyro available and we must try to approximate $\dot{\theta}$ from measurements of θ. However, a compass might jump by 2π for small variations in heading. This is the case for instance when a compass returns an angle within the interval $[-\pi, \pi]$ and the heading varies around $(2k+1)\,\pi$. In this case, an approximation of $\dot{\theta}$ must be obtained and this approximation has to be insensitive to these jumps. Let us denote by :

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

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

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

This relation can be directly obtained by using the expression of $\mathbf{R}_t$. Thus, an Euler integration of the rotation matrix:

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

translates to:

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

Therefore:

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

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

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

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

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

which will be insensitive to jumps of 2π.

3.2 Skate car

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

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

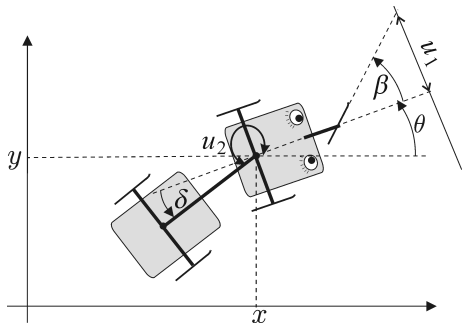

Figure 3.3: Skating robot moving like a snake

3.2.1 Model

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

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

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

$$v = v_1 \cos\beta$$

and therefore:

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

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

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

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

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

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

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

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

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

and

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

The system can be described by the following state equations:

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

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

3.2.2 Sinusoidal control

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

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

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

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

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

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

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

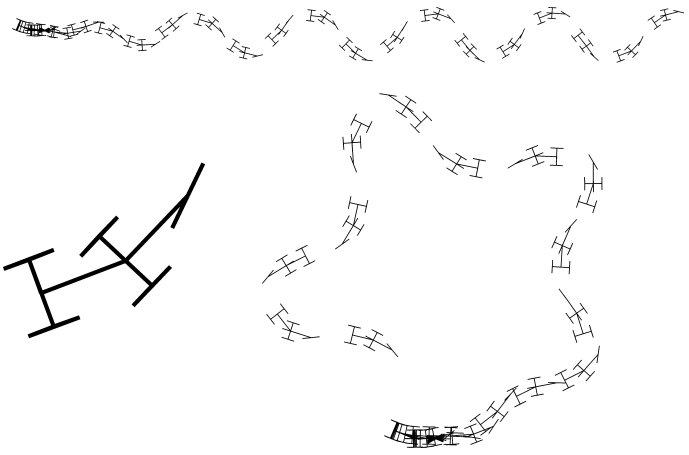

Figure 3.4: Various simulations illustrate the control law for the skating robot

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

3.2.3 Maximum thrust control

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

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

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

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

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

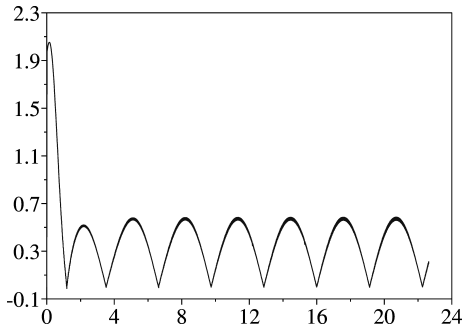

Figure 3.5: Thrust supplied by the engine u_2

(which means that δ increases to infinity). We obtain, by isolating the orientation u_1 of the front skate:

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

The maximum thrust controller is therefore given by:

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

Thus, with this controller, not only do we always thrust in the correct direction through u_2 but we can adjust the direction u_1 in order for the torque supplied by u_2 to translate into a maximum thrust $\bar{p}$. Now we only need to act on ε (which, as we recall, is a square wave equal to ± 1) and on the power $\bar{p}$. The duty cycle of the signal $\varepsilon(t)$ will allow us to direct our orientation and its frequency will give us the amplitude of the oscillations for the robot's path. As for $\bar{u}_2$, it allows us to control the average speed of the robot. In the simulation, this controller turns out indeed to be more efficient than the sinusoidal controller. Figure 3.6 shows the angle of the skate β in function of time, once the cruising regime has been reached. Let us note that the angle of the front skate $\beta = \text{atan}(u_1)$ makes discontinuities appear.

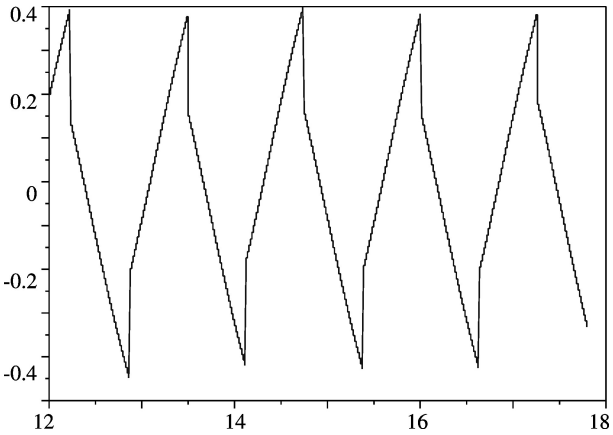

Figure 3.6: Evolution of the front skate angle β in cruising regime

3.2.4 Simplification of the fast dynamics

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

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

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

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

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

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

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

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

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

The system thus becomes:

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

which now only has two inputs: $\bar{p}$ and $\bar{q} = \frac{\bar{\varepsilon}}{\bar{u}_2}$. Let us try to control the inputs θ and v by using feedback linearization method. Note that although this system is not affine in its inputs ($\bar{p}$ and $\bar{q}$),

the method can be applied because, as we will see below, the necessary inversion is possible here. For this, let us define two new inputs v_1, v_2 such that:

$$\left\{ \begin{array}{rcl} v_1 & = & -\bar{p}\bar{q} - v\sin\delta \\ v_2 & = & \frac{\bar{p}}{v} - v \end{array} \right.$$

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

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

Thus, the feedback linearized system is:

$$\left\{ \begin{array}{rcl} \dot{\theta} & = & v_1 \\ \dot{v} & = & v_2 \end{array} \right.$$

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

$$\left\{ \begin{array}{rcl} v_1 & = & w_1 - \theta + \dot{w}_1 \\ v_2 & = & w_2 - v + \dot{w}_2 \end{array} \right.$$

Let us summarize the control law in its entirety. Its setpoints are w_1, w_2 which correspond to the desired heading and speed. It is given by the following table:

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

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

3.3 Sailboat

3.3.1 Problem

Let us recall the principles of model-free control and try to adapt it to a line-tracking controller for our sailboat. Here we will consider a sailboat whose sheet length is variable, but not directly the

angle of the sail as it was the case until now (see [2.11] on page 22). This robot has two inputs which are the angle of the rudder $u_1 = \delta_r$ and the maximum angle of the sail $u_2 = \delta_s^{\max}$ (equivalently u_2 corresponds to the length of the sheet). We will try to make the robot follow a line which passes through points **a** and **b** (see Figure 3.7).

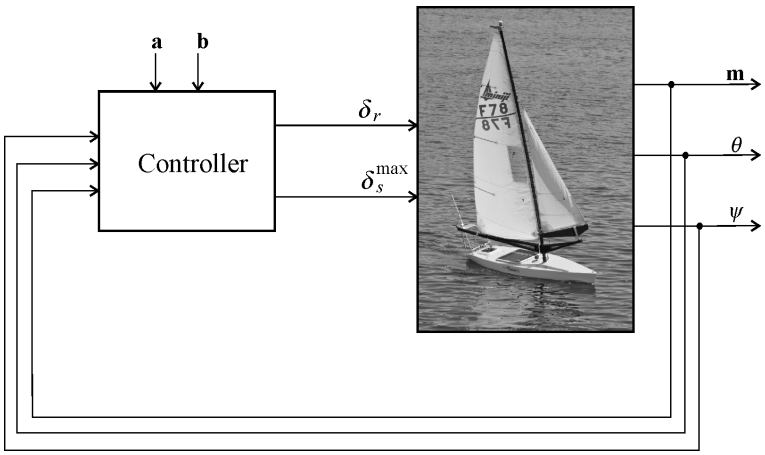

Figure 3.7: Feedback control of the sailing robot

This problem is influenced by the control strategies of the VAIMOS robot [GOR 11] of IFREMER, the sailing boat of the ERWAN naval school and the Optimousse robot from the ENSTA-Bretagne (see Figure 3.8).

As illustrated by Figure 3.9, we will denote by (x, y, θ) the posture of the boat, by v its advancing speed, by ω its angular speed, by f_s the force of the wind on the sail, by f_r the force of the water on the rudder, by δ_s the angle of the sail and by ψ the angle of the wind.

3.3.2 Controller

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

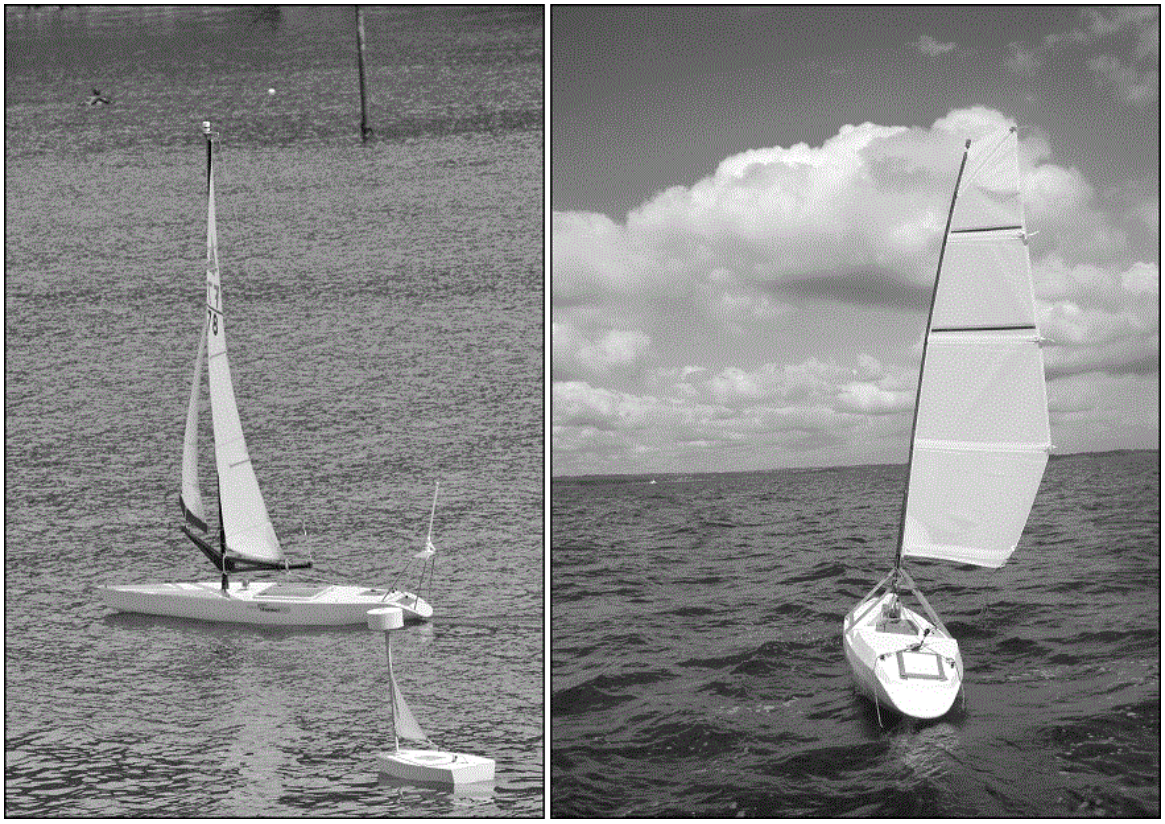

Figure 3.8: Left : the VAIMOS sailing robot of IFREMER (in the background) and the Optimousse robot from the ENSTA Bretagne ; right : robot of the Ecole Navale (Naval School) following a line

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

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

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

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

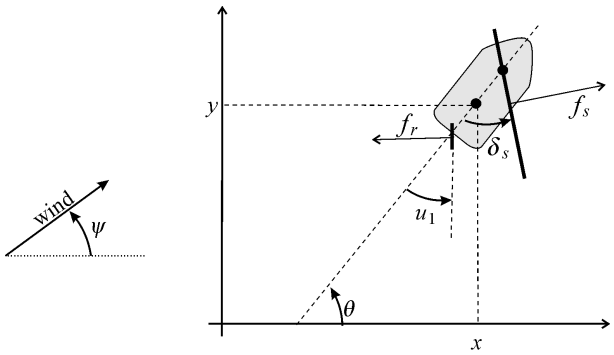

Figure 3.9: Variables used in the state equations of the robot

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

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

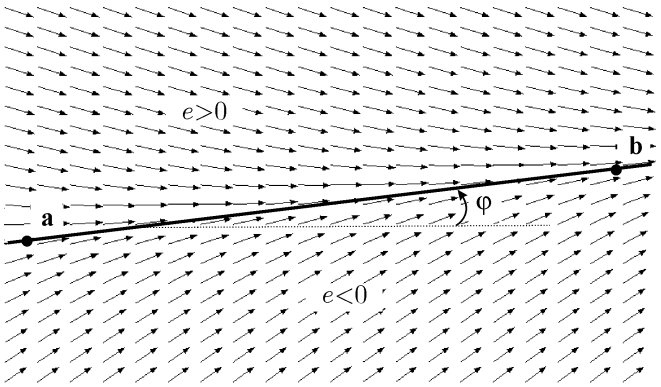

Figure 3.10: Nominal vector field that the robot tries to track, when possible

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

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

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

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

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

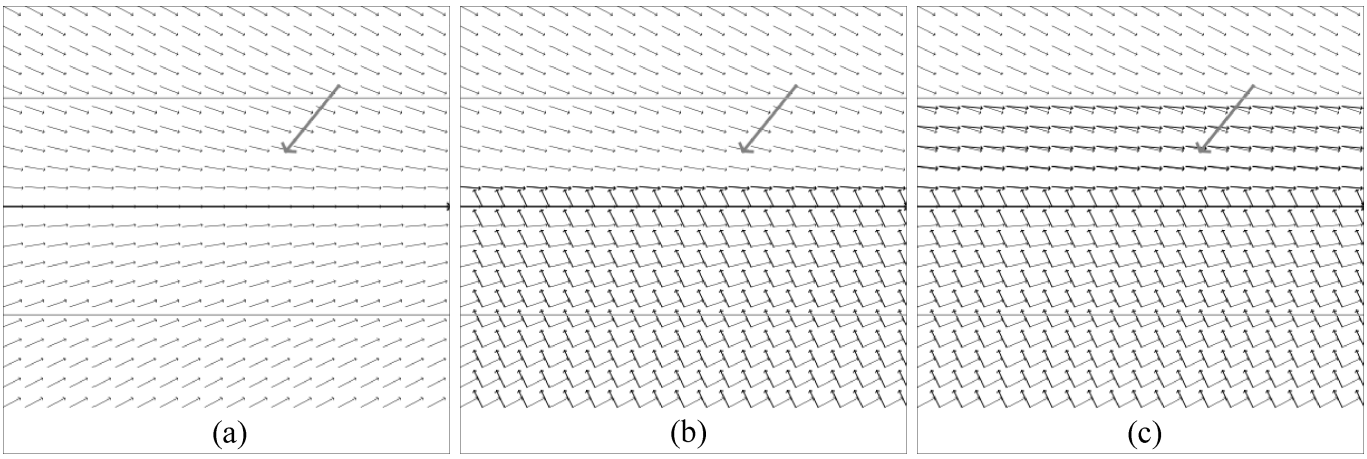

Figure 3.11: (a) The nominal vector field may be incompatible with the wind (here represented by the large arrow) ; (b) Vector field generated by the controller if we remove line 6. The thin arrows correspond to the nominal paths and the bold arrows correspond to the corrected paths ; (c) Vector field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

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

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

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

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

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

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

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

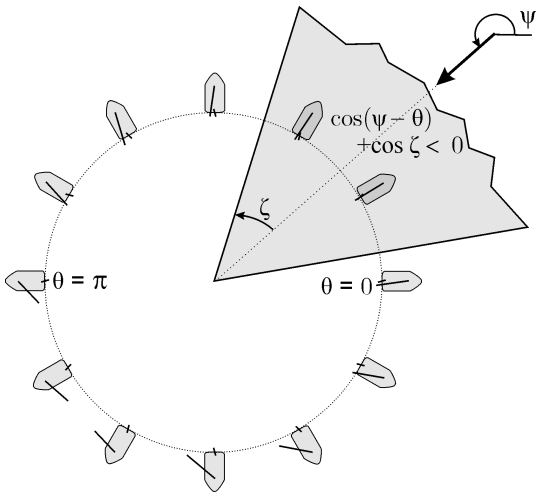

Figure 3.12: Some directions are not possible for the sailboat. These unfeasible directions form the *no-go* zone, represented in grey

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

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

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

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

i.e.:

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

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

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

3.3.3 Navigation

Once the line tracking has been correctly implemented and validated, a number of lines should be chained together with the aim of performing complex missions (such as for instance connecting two points of the globe). In such a context, a Petri net strategy is well-adapted for representing the discrete state changes [MUR 89]. Figure 3.16 illustrates a Petri net allowing to manage the mission. Before the robot is launched, it is in an initial state represented by the place p_0. The transition t_1 is crossed at the start of the mission. If everything goes well, the robot is in state p_1 and is tracking

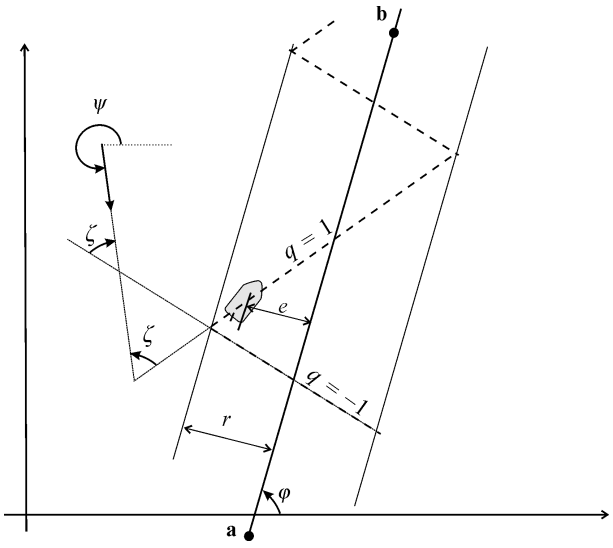

Figure 3.13: Keep close hauled strategy by remaining within the strip centered on the line **ab** with diameter r

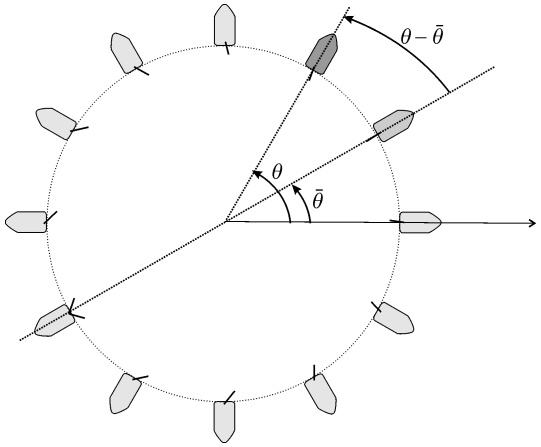

Figure 3.14: Rudder control for the sailing robot

its first line $\mathbf{a}_1\mathbf{b}_1$. The line $\mathbf{a}_j\mathbf{b}_j$ is validated as soon as the point $\mathbf{b}_j$ is surpassed, in other words if $\langle \mathbf{b}_j - \mathbf{a}_j, \mathbf{m} - \mathbf{b}_j \rangle > 0$. This stopping criterion coupled with the path can be interpreted as a sort of validation. Once this is validated, we proceed to the next line. When the list of lines to track is empty, the mission ends (place p_3).

3.3.4 Experiment

Beginning in September 2011, we carried out a series of experiments with the VAIMOS sailboat in autonomous mode. We will describe one of these experiments which is simultaneously simple and representative, which took place on Thursday 28 June 2012 in the Brest bay, close to the Moulin Blanc harbor. The trajectory performed by the robot is represented on Figure 3.17. The wind comes from South-South-East, as indicated by the arrow that represents the average wind on the robot during the entire mission, deduced from the sensor. In this zone of heavy maritime traffic, interruptions in the mission can be anticipated and a permanent wifi link is necessary between the robot and the

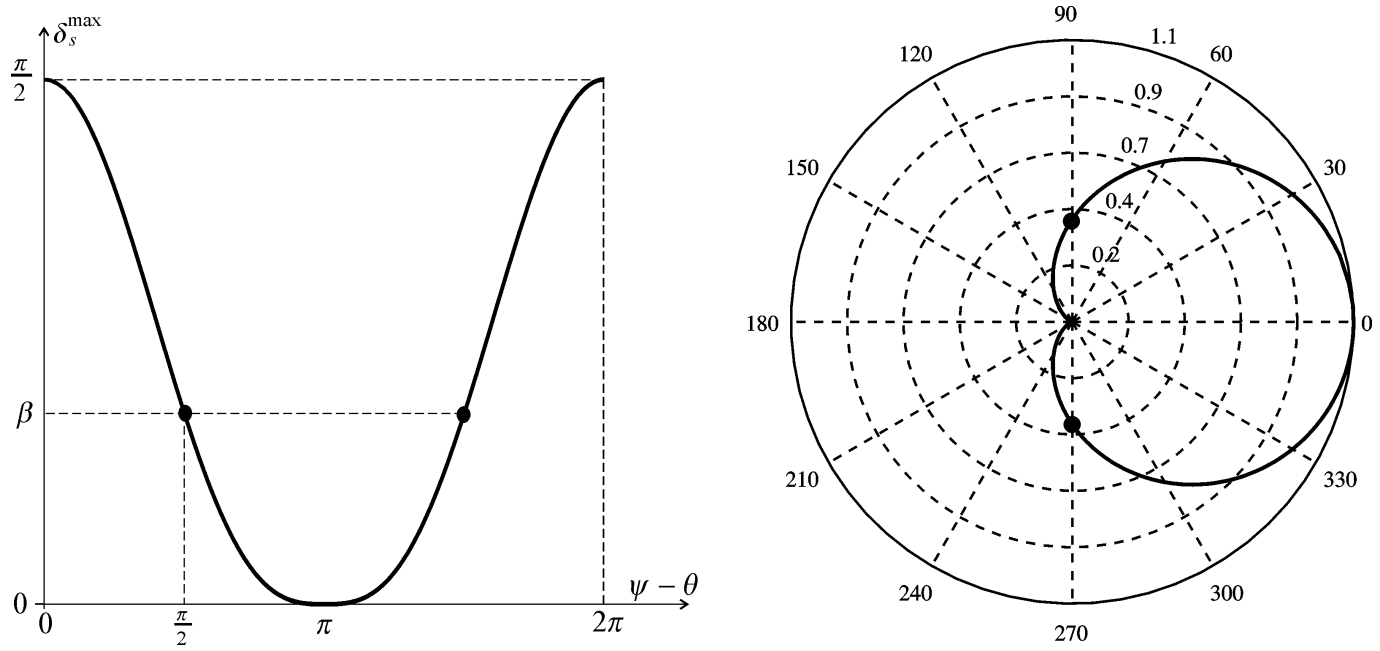

Figure 3.15: Adjusting the maximum sail angle (or the sheet length) ; left : Cartesian representation ; right : polar representation

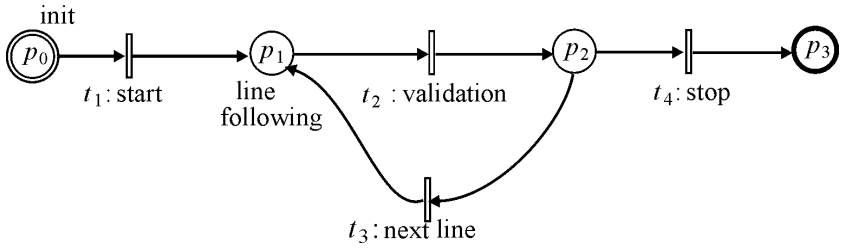

Figure 3.16: Petri net supervising the navigation of the robot

tracking boat in order to be able to cancel to mission at any time and avoid collision with other boats. Apart from these security interruptions (which by the way were not necessary during the mission), the robot is entirely autonomous. The mission is broken down into five sub-missions. First of all, the robot begins with a triangle (in the circle) in order to check whether everything is working properly. Then, it proceeds South-East against the wind by tracking the required path. It then describes a spiral. Once it is in the center of the spiral, the robot anchors virtually, in other words it maneuvers in order to remain around its attachment point. Finally, the robot returns to the harbor with the wind.

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

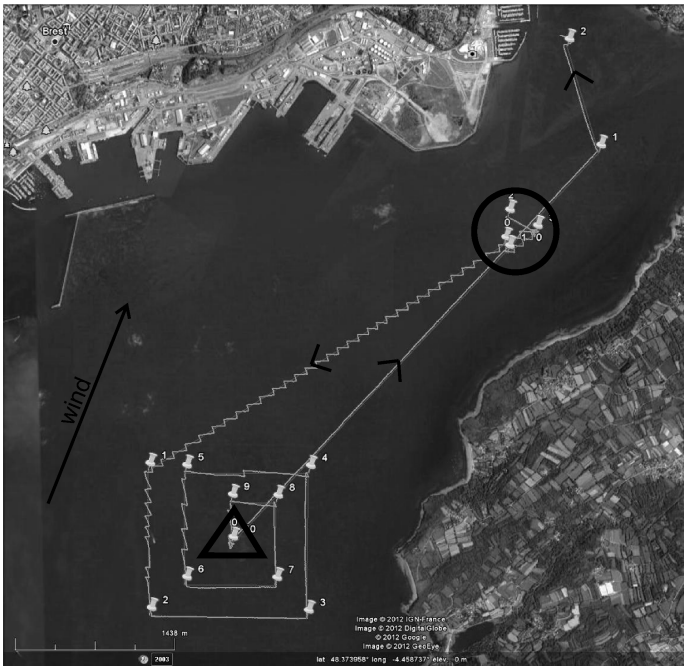

Figure 3.17: Experiment of the spiral composed of five stages : (a) the robot begins with a triangle (in the circle) ; (b) it goes upwind following a line ; (c) it described a spiral ; (d) it anchors virtually at the center of the spiral for several minutes ; (e) and goes back to the harbor.

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

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

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

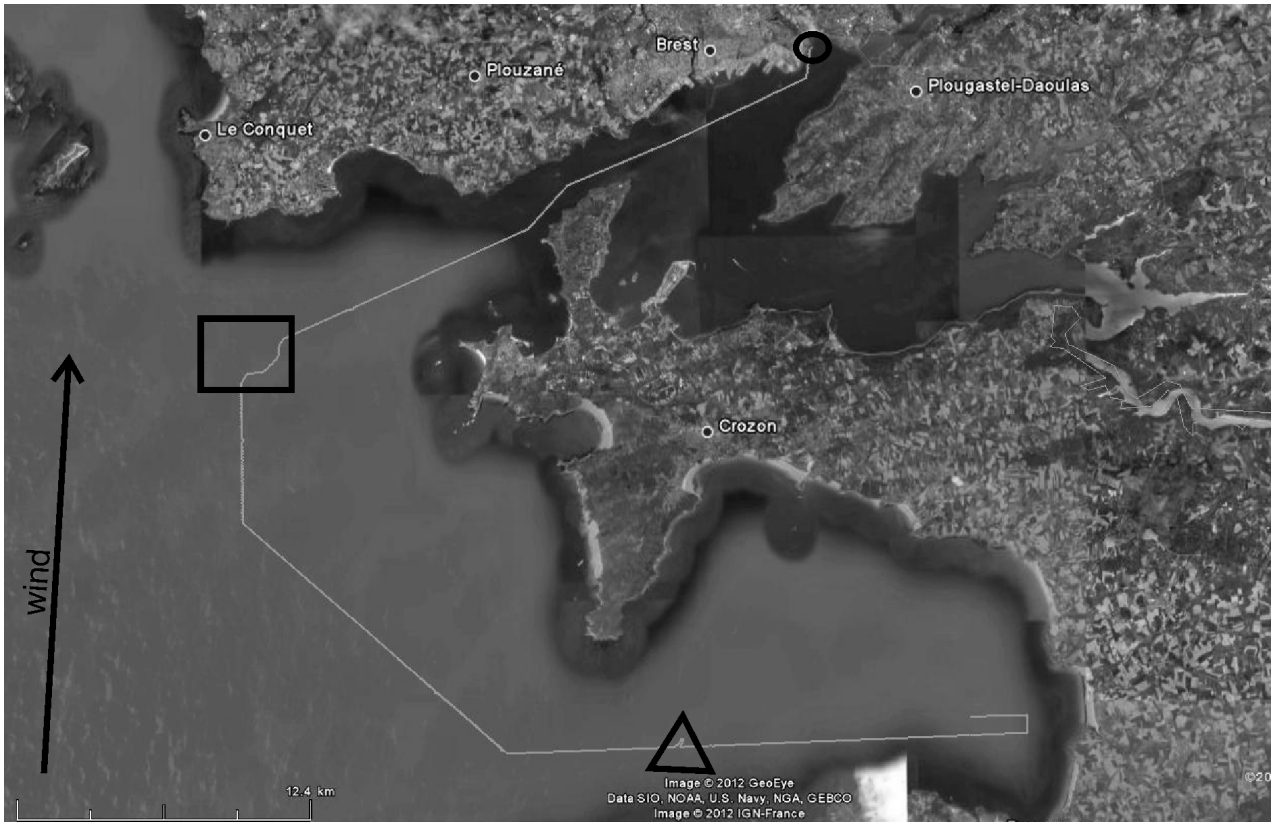

Figure 3.18: Journey from Brest to Douarnenez made by VAIMOS : (a) the robot leaves the Moulin-Blanc harbour (in the circle) ; (b) it avoids a submarine (in the square) ; (c) it avoids a cargo ship (triangle)

Exercises

EXERCISE 3.1.– *Robot tank on a line*

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

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

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

1) Simulate this system graphically with in various situations.

2) Propose a heading controller for the robot.

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

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

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

EXERCISE 3.2.– *Van der Pol car*

Consider the car represented on Figure 3.19.

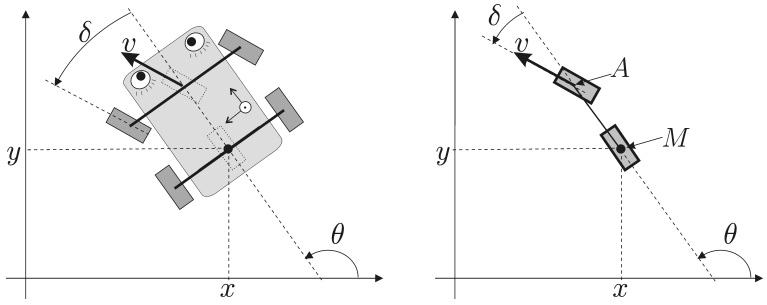

Figure 3.19: Car moving on a plane (view from above)

This car has two controls: the acceleration of the front wheels and the rotation speed of the steering wheel. The state variables of our system are composed of the position coordinates (the coordinates x, y of the center of the rear axle, the heading θ of the car and the angle δ of the front

wheels) and the speed v of the center of the front axle. The evolution equation of the car is assumed to be identical to that of a tricycle (see page 19). It is written as:

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

1) Simulate this system using an Euler's method.

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

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

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

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

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

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

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

EXERCISE 3.3.– *Anchoring*

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

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

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

1) Does this system have a point of equilibrium ?

2) Given the fact that the problem of remaining around zero admits a rotational symmetry, we suggest switching from a Cartesian representation (x, y, θ) towards a polar representation (α, d, φ), as shown on Figure 3.20. Give the state equations in the polar representation.

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

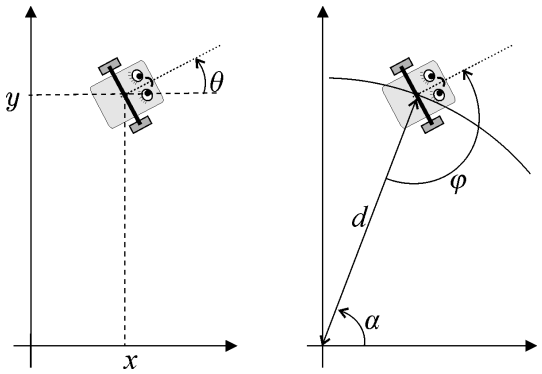

Figure 3.20: Coordinate system change allowing to take advantage of the rotational symmetry

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

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

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

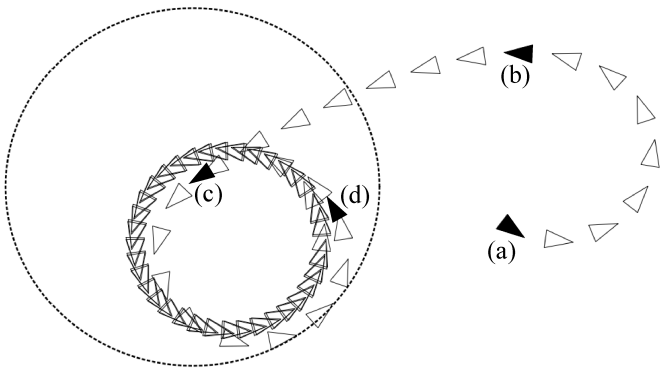

Figure 3.21: Path of the controlled system in order to remain inside the disk

5) Simulate this control law with various initial conditions.

Exercise 3.4.– *Discretization of the state space*

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

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

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

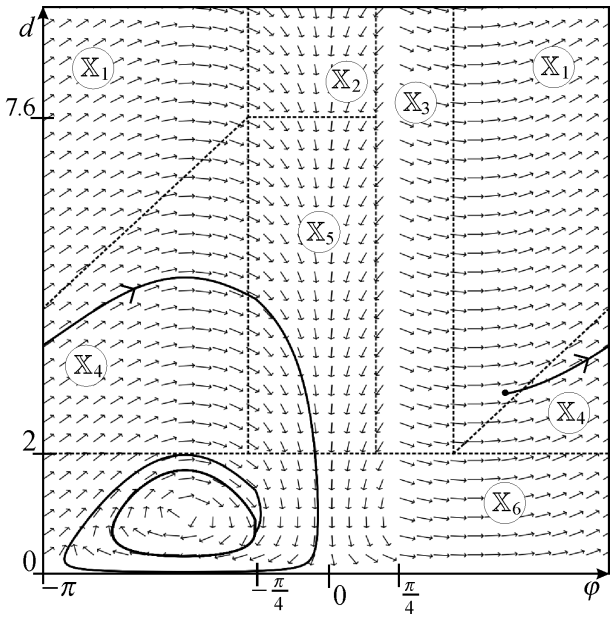

Figure 3.22: Vector field associated with our system

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

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

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

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

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

1) Draw the graph associated with this relation.

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

EXERCISE 3.5.– *Sailing robot*

Consider the sailboat described by the following state equations:

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

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

1) Simulate the boat.

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

Exercise 3.6.– *Plane*

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

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

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

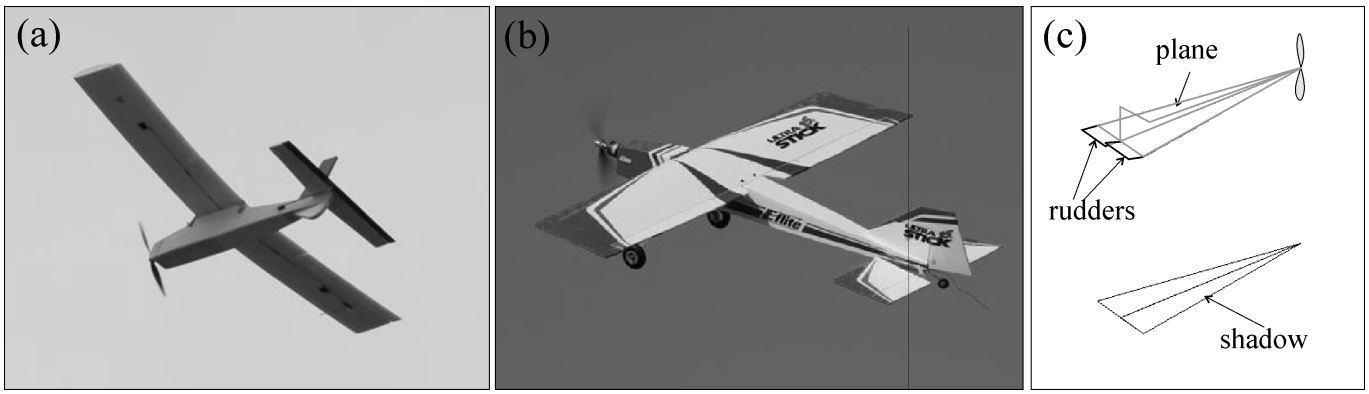

Figure 3.23: (a) μ-STIC plane made by the ENSTA Bretagne ; (b) Faser Ultra Stick plane from the University of Minnesota ; (c) graphical representation used for the simulation

with:

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

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

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

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

1) Simulate this drone. For the graphics, see Figure 3.23c.

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

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

Exercise 3.7.– *Quadrotor*

We consider the quadrotor represented on Figure 3.24.

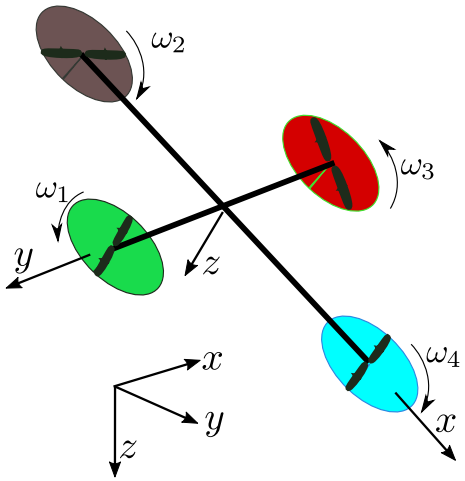

Figure 3.24: Quadrotor to be controlled

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

The state variables are the position $\mathbf{p}$ of the robot, (φ, θ, ψ) the Euler angles, the speed $\mathbf{v}_r$ expressed in the robot frame and the rotation vector $\boldsymbol{\omega}_r$ also expressed in the robot frame. The inertia matrix will be taken as

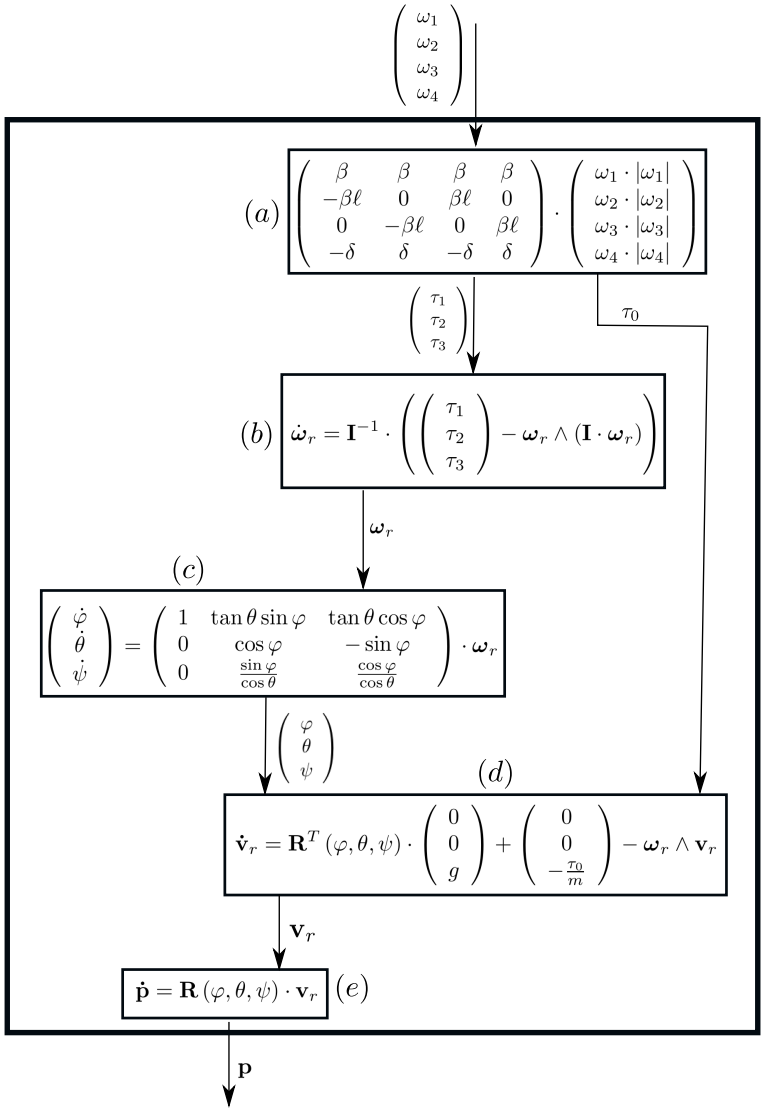

Figure 3.25: Dynamic of the quadrotor represented by a causal chain which links 5 blocks *(a)*, *(b)*, *(c)*, *(d)*, *(e)*

$$\mathbf{I} = \begin{pmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 20 \end{pmatrix}.$$

Moreover, we will take $m = 10$, $g = 9.81$, $\beta = 2$, $d = 1$, and $\ell = 1$, all expressed in the international metric system.

To control this quadrotor, a feedback linearization method could be done, but the computation of the controller is not easy and assumes that the model corresponds to the true system. Here, we want to use a more pragmatic method, based on a backstepping technique [KHA 02]. It corresponds to a sequence of loops, each of which inverting one block in the causal chain. This will make the controller easier to develop and to debug.

1) Propose a feed forward controller with the new input $\left(\tau_0^d, \tau_1^d, \tau_2^d, \tau_2^d\right)$ with eliminates the effect

of Block *(a)*. Here, the supscript d means 'desired' since it is what we want for the τ_i.

2) Using a feedback linearization approach, build a controller with a desired input $\boldsymbol{\omega}_r^d$ and an output $\boldsymbol{\tau}_{1:3}^d = \left(\tau_1^d, \tau_2^d, \tau_2^d\right)$, such that the vector $\boldsymbol{\omega}_r$ converges to $\boldsymbol{\omega}_r^d$. This loop will make us possible to eliminate Block *(b)*.

3) Build a controller with a desired input $\left(\varphi^d, \theta^d, \psi^d\right)$ which and an output $\boldsymbol{\omega}_r^d$, such that the vector (φ, θ, ψ) converges to $\left(\varphi^d, \theta^d, \psi^d\right)$.

4) Add another control loop with an output $\left(\varphi^d, \theta^d, \psi^d, \tau_0^d\right)$, based on a vector field approach, so that the quadrotor follows a path that obeys the Van der Pol equation:

$$\begin{cases} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\left(0.001\ x_1^2 - 1\right) x_2 - x_1 \end{cases}$$

The quadrotor should also stay at an altitude of 15m and a speed of $v_d = 10\text{ms}^{-1}$. Moreover, we want the front of the robot points forward. An illustration of what should be obtained is depicted on Figure 3.26. The Van der Pol cycle is painted green.

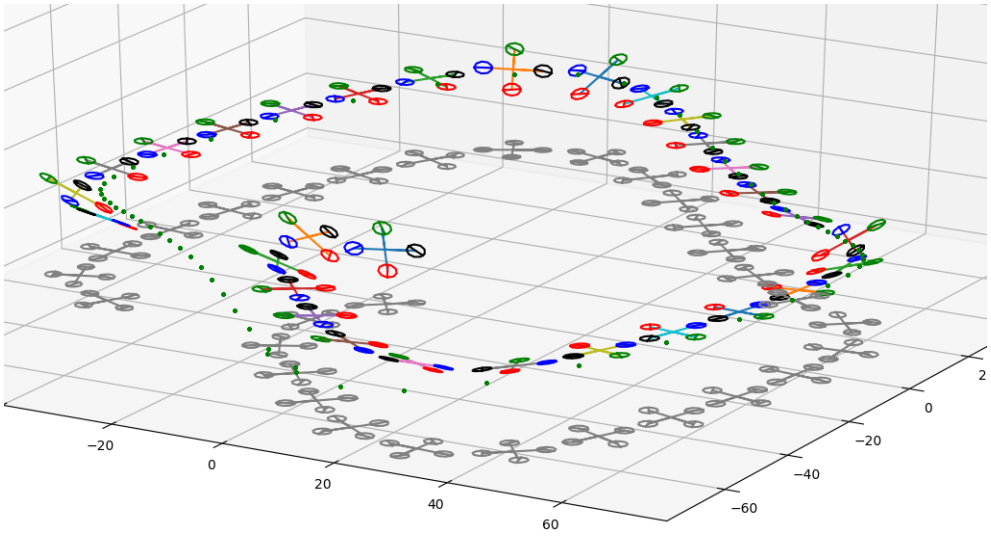

Figure 3.26: Simulation of the quadrotor which follows the Van der Pol cycle

---

Exercise 3.8.– *Isobath*

An isobath is imaginary curve that connects all points having the same depth below a water surface, *i.e.*, an underwater level curve. Consider an underwater robot described by the state equation:

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

For the initial condition, we take $x = 2, y = -1, z = -2, \psi = 0$. The robot is able to measure its altitude y_1 (distance to the seafloor) and the angle y_2 of the gradient of h in its own frame using

a sonar. Moreoever it is able to know its depth z_3 using a pressure sensor. Thus its observation function is

$$\begin{cases} y_1 &= z - h(x,y) \\ y_2 &= angle(\nabla h(x,y)) - \psi \\ y_3 &= -z \end{cases}$$

where $\nabla h(x,y)$ the gradient of h. For instance, if $\mathbf{y} = \left(-7, \frac{\pi}{2}, 2\right)$, the robot knows that it is following an isobath corresponding to $-7-2=-9$m, at a depth of 2m.

1) Propose a controller of the form $\mathbf{u} = \mathbf{r}(\mathbf{y})$, which makes the robot follows an isobath corresponding to $h_0 = -9$m at a depth $\overline{y}_3 = 2$m.

2) Validate your controller using a simulation. An illustration of what should be obtained is depicted on Figure 3.27. For the simulation, we will consider a seafloor described by

$$h(x,y) = 2 \cdot e^{-\frac{(x+2)^2+(y+2)^2}{10}} + 2 \cdot e^{-\frac{(x-2)^2+(y-2)^2}{10}} - 10.$$

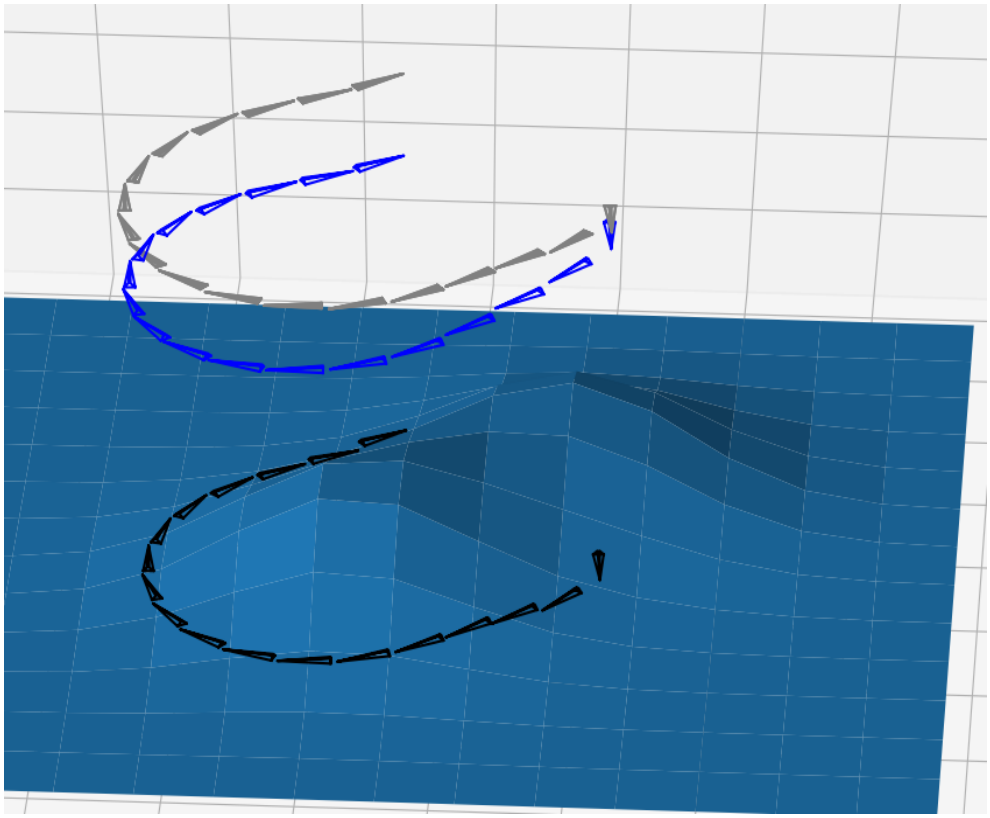

Figure 3.27: Simulation of underwater robot (blue) following an isobath. The surface shadow (gray) and the seafloor shadow are also painted.

Chapter 4

Guidance

In previous chapters, we have studied how to build a control law for a robot described by nonlinear state equations (see Chapter 2) or when the robot's behavior is known (see Chapter 3). *Guidance* is performed on a higher level and focuses on the setpoint to give the controller in order for the robot to be able to accomplish its assigned mission. It will therefore have to take into account the knowledge of its surroundings, the presence of obstacles, the roundness of the Earth, and so forth. Conventionally, guidance is applied in four different environments: terrestrial, marine, aerial and spatial. Given the fields of application covered in this book, we will not study the spatial environment.

4.1 Guidance on a sphere

For longer paths over the surface of the Earth, the Cartesian coordinate system, which assumes a flat Earth, can no longer be considered. We then have to rethink our control laws by navigating relative to a spherical coordinate system (also referred to as *geographical coordinates*), which rotates together with the Earth. Let us denote by ℓ_x the longitude and by ℓ_y the latitude of the point being considered. The transformation in the geographical coordinate system is written as:

$$\mathcal{T} : \begin{pmatrix} \ell_x \\ \ell_y \\ \rho \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \rho \cos \ell_y \cos \ell_x \\ \rho \cos \ell_y \sin \ell_x \\ \rho \sin \ell_y \end{pmatrix} \tag{4.1}$$

When $\rho = 6\ 370$ km, we are on the surface of the Earth, which we will assume to be spherical (see Figure 4.1a).

Let us consider two points $\mathbf{a}$, $\mathbf{m}$ on the surface of the Earth, as illustrated on Figure 4.1b, located by their geographical coordinates. Here, $\mathbf{a}$ for instance represents a reference point to reach and $\mathbf{m}$ is the center of our robot. By assuming that the two points $\mathbf{a}$, $\mathbf{m}$ are not too far apart (by no more than 100 km), we may consider being in the plane and use a local map, as shown on Figure 4.2a.

Let us differentiate Relation (4.1). We obtain:

$$\begin{pmatrix} dx \\ dy \\ dz \end{pmatrix} = \underbrace{\begin{pmatrix} -\rho \cos \ell_y \sin \ell_x & -\rho \sin \ell_y \cos \ell_x & \cos \ell_y \cos \ell_x \\ \rho \cos \ell_y \cos \ell_x & -\rho \sin \ell_y \sin \ell_x & \cos \ell_y \sin \ell_x \\ 0 & \rho \cos \ell_y & \sin \ell_y \end{pmatrix}}_{=\mathbf{J}} \cdot \begin{pmatrix} d\ell_x \\ d\ell_y \\ d\rho \end{pmatrix}$$

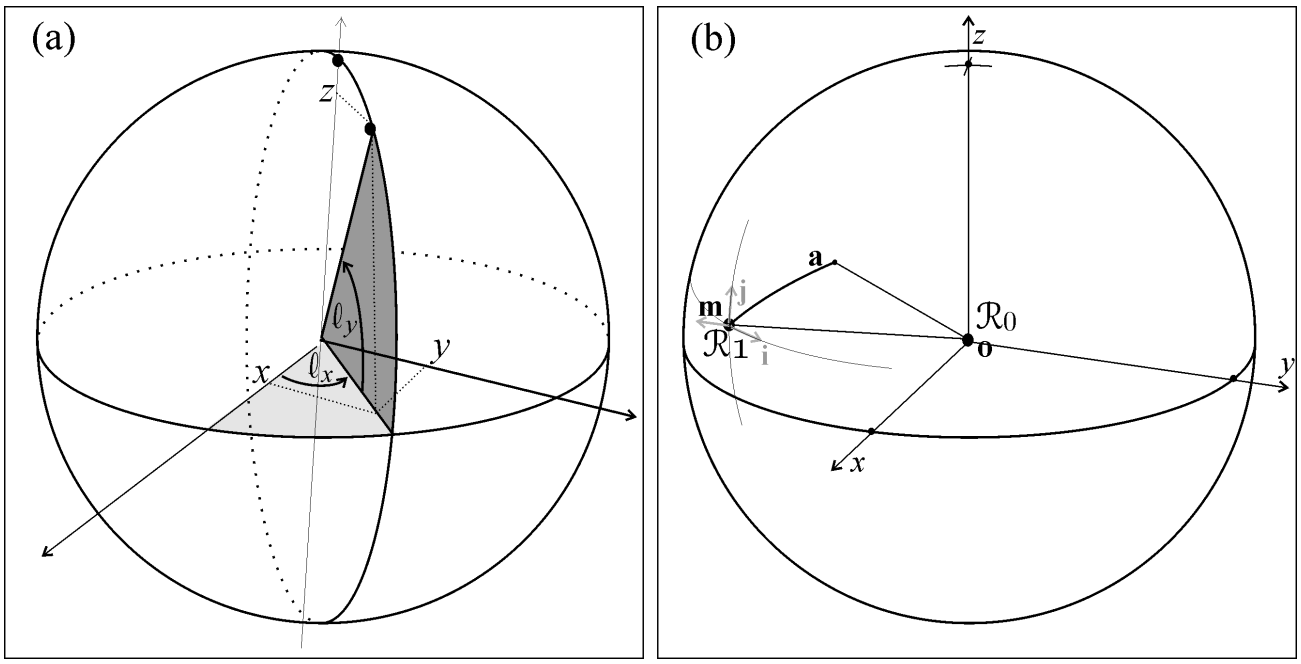

Figure 4.1: (a) Geographical coordinate system ; (b) we would like to express **a** in the local map $\mathcal{R}_1$

This formula can be used to find the geographical coordinates of the cardinal directions, which change depending on the location of the robot **m**. For instance, the vector corresponding to *East* is found in the first column of the matrix **J**, North in the second column and the altitude in the third column. We will therefore be able to build a coordinate system (East-North-Altitude) $\mathcal{R}_1$ centered on the robot (in grey in Figure 4.2a) that corresponds to the local map. The corresponding rotation matrix is obtained by normalizing each column of the Jacobian matrix **J**. We obtain:

$$\mathbf{R} = \begin{pmatrix} -\sin \ell_x & -\sin \ell_y \cos \ell_x & \cos \ell_y \cos \ell_x \\ \cos \ell_x & -\sin \ell_y \sin \ell_x & \cos \ell_y \sin \ell_x \\ 0 & \cos \ell_y & \sin \ell_y \end{pmatrix} \tag{4.2}$$

The transformation that allows switching from the geographic coordinate system $\mathcal{R}_0$ to the local map $\mathcal{R}_1$ is:

$$\mathbf{v}_{|\mathcal{R}_1} = \mathbf{R}^{\mathrm{T}} \cdot \mathbf{v}_{|\mathcal{R}_0} \tag{4.3}$$

This coordinate system change relation can be applied in different contexts such as for instance when establishing coherence in the data collected by two different robots.

EXAMPLE 4.1.– *A robot situated at* $\mathbf{m} : \left(\ell_x^{\mathbf{m}}, \ell_y^{\mathbf{m}}\right)$ *is moving with a speed vector* $\mathbf{v}^{\mathbf{m}}$*, relative to a fixed ground. This vector is expressed in the local map of the robot. We want to find the speed with which this vector* $\mathbf{v}^{\mathbf{m}}$ *is perceived in the local map of an observer situated at* $\mathbf{a} : \left(\ell_x^{\mathbf{a}}, \ell_y^{\mathbf{a}}\right)$*. Following [4.3], we have:*

$$\begin{cases} \mathbf{v}^{\mathbf{m}} = \mathbf{R}^T \left(\ell_x^{\mathbf{m}}, \ell_y^{\mathbf{m}}\right) \cdot \mathbf{v}_{|\mathcal{R}_0} \\ \mathbf{v}^{\mathbf{a}} = \mathbf{R}^T \left(\ell_x^{\mathbf{a}}, \ell_y^{\mathbf{a}}\right) \cdot \mathbf{v}_{|\mathcal{R}_0} \end{cases}$$

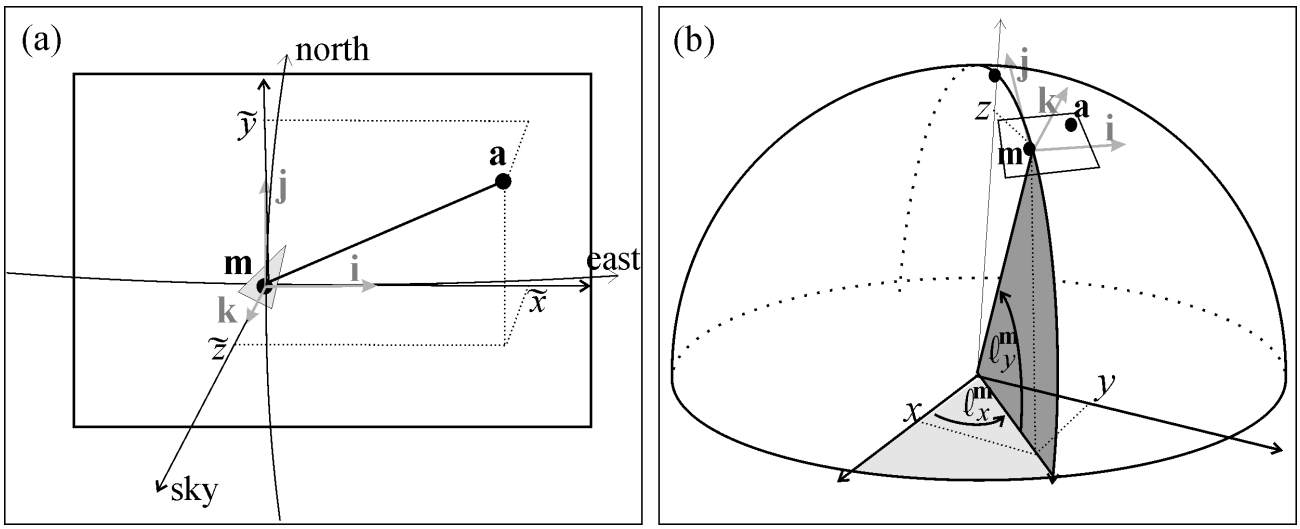

Figure 4.2: (a) The map gives a local Cartesian viewpoint around the robot ; (b) a slight shift $d\ell_x, d\ell_y, d\rho$ creates a displacement dx, dy, dz in the local map

Therefore:

$$\mathbf{v^a} = \mathbf{R}^T\left(\ell_x^{\mathbf{a}}, \ell_y^{\mathbf{a}}\right) \cdot \mathbf{R}\left(\ell_x^{\mathbf{m}}, \ell_y^{\mathbf{m}}\right) \cdot \mathbf{v^m}.$$

This kind of calculation is useful when, for instance, two robots are trying to meet.

Switching to a local map. Let us take a local map $\mathcal{R}_{\mathbf{m}}$ centered on a point $\mathbf{m}$, (in other words a coordinate system located on the surface of the Earth whose origin is $\mathbf{m}$ and whose directions are East-North-upwards. Let us now try to express in $\mathcal{R}_{\mathbf{m}}$ the coordinates $(\widetilde{x}, \widetilde{y}, \widetilde{z})$ of a point $\mathbf{a}$ located by its GPS coordinates (ℓ_x, ℓ_y, ρ). We can switch from the geographical coordinates to the local coordinates with the following relation:

$$\underbrace{\begin{pmatrix} \widetilde{x} \\ \widetilde{y} \\ \widetilde{z} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \rho^{\mathbf{m}} \end{pmatrix}}_{\mathbf{oa}_{|\mathcal{R}_{\mathbf{m}}}} \overset{[4.3]}{=} \underbrace{\begin{pmatrix} -\sin\ell_x^{\mathbf{m}} & \cos\ell_x^{\mathbf{m}} & 0 \\ -\cos\ell_x^{\mathbf{m}}\sin\ell_y^{\mathbf{m}} & -\sin\ell_x^{\mathbf{m}}\sin\ell_y^{\mathbf{m}} & \cos\ell_y^{\mathbf{m}} \\ \cos\ell_x^{\mathbf{m}}\cos\ell_y^{\mathbf{m}} & \cos\ell_y^{\mathbf{m}}\sin\ell_x^{\mathbf{m}} & \sin\ell_y^{\mathbf{m}} \end{pmatrix}}_{\mathbf{R}^{\mathrm{T}}\left(\ell_x^{\mathbf{m}}, \ell_y^{\mathbf{m}}\right)} \cdot \underbrace{\begin{pmatrix} \rho\cos\ell_y\cos\ell_x \\ \rho\cos\ell_y\sin\ell_x \\ \rho\sin\ell_y \end{pmatrix}}_{\mathbf{oa}_{|\mathcal{R}_0}}$$

where $\mathbf{o}$ is the origin corresponding to the center of the Earth. When $\ell_x^{\mathbf{m}} \simeq \ell_x$ and $\ell_y^{\mathbf{m}} \simeq \ell_y$, a first-order approximation is directly obtained with the aid of Figure 4.3.

$$\begin{pmatrix} \widetilde{x} \\ \widetilde{y} \\ \widetilde{z} \end{pmatrix} \simeq \begin{pmatrix} \rho \cdot \cos\ell_y \cdot (\ell_x - \ell_x^{\mathbf{m}}) \\ \rho \cdot \left(\ell_y - \ell_y^{\mathbf{m}}\right) \\ \rho - \rho^{\mathbf{m}} \end{pmatrix}. \tag{4.4}$$

Note that the circle of latitude has a radius of $\rho\cos\ell_y$.

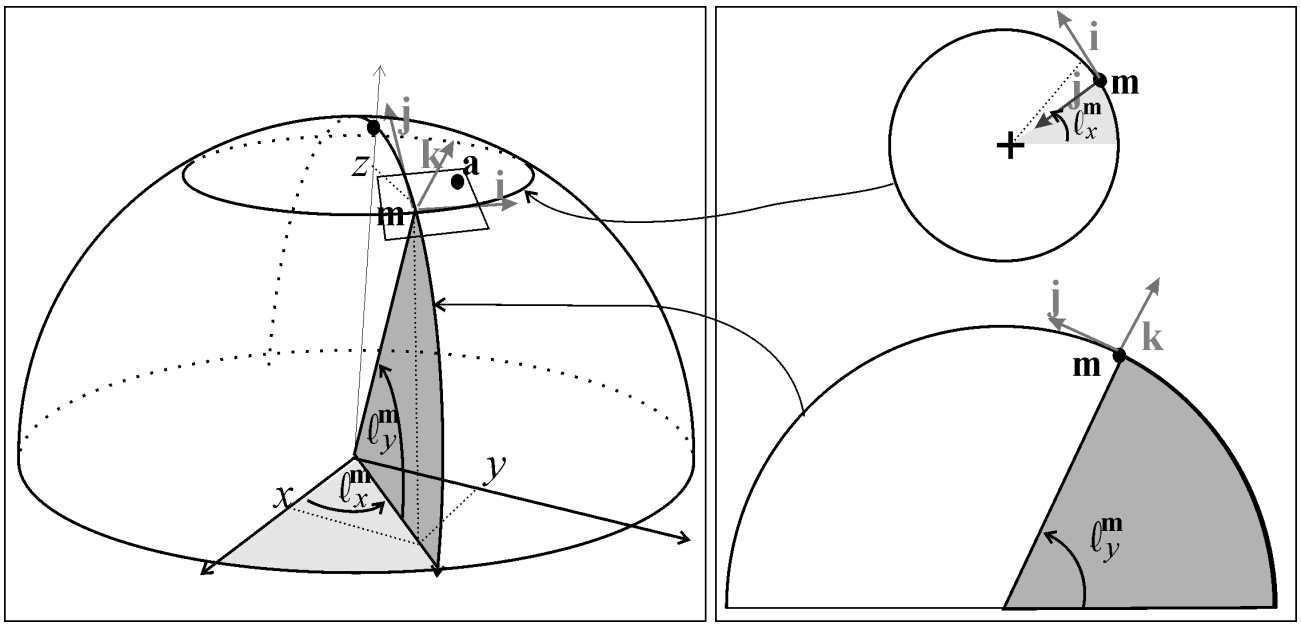

Figure 4.3: Getting Cartesian coordinates in a local frame centred in $\mathbf{m}$

We could also have formally obtained these results using trigonometric relations:

$$
\begin{aligned}
\begin{pmatrix} \widetilde{x} \\ \widetilde{y} \\ \widetilde{z} \end{pmatrix} &= \begin{pmatrix} 0 \\ 0 \\ -\rho^{\mathbf{m}} \end{pmatrix} + \rho \begin{pmatrix} -\sin\ell_x^{\mathbf{m}}\cos\ell_y\cos\ell_x & + & \cos\ell_x^{\mathbf{m}}\cos\ell_y\sin\ell_x \\ -\cos\ell_x^{\mathbf{m}}\sin\ell_y^{\mathbf{m}}\cos\ell_y\cos\ell_x & - & \sin\ell_x^{\mathbf{m}}\sin\ell_y^{\mathbf{m}}\cos\ell_y\sin\ell_x \quad + \quad \cos\ell_y^{\mathbf{m}}\sin\ell_y \\ \cos\ell_x^{\mathbf{m}}\cos\ell_y^{\mathbf{m}}\cos\ell_y\cos\ell_x & + & \cos\ell_y^{\mathbf{m}}\sin\ell_x^{\mathbf{m}}\cos\ell_y\sin\ell_x \quad + \quad \sin\ell_y^{\mathbf{m}}\sin\ell_y \end{pmatrix} \\
&= \begin{pmatrix} 0 \\ 0 \\ -\rho^{\mathbf{m}} \end{pmatrix} + \rho \begin{pmatrix} \cos\ell_y\left(\cos\ell_x^{\mathbf{m}}\sin\ell_x - \sin\ell_x^{\mathbf{m}}\cos\ell_x\right) \\ -\sin\ell_y^{\mathbf{m}}\cos\ell_y\left(\cos\ell_x^{\mathbf{m}}\cos\ell_x + \sin\ell_x^{\mathbf{m}}\sin\ell_x\right) + \cos\ell_y^{\mathbf{m}}\sin\ell_y \\ \cos\ell_y^{\mathbf{m}}\cos\ell_y\left(\cos\ell_x^{\mathbf{m}}\cos\ell_x + \sin\ell_x^{\mathbf{m}}\sin\ell_x\right) + \sin\ell_y^{\mathbf{m}}\sin\ell_y \end{pmatrix} \\
&= \begin{pmatrix} 0 \\ 0 \\ -\rho^{\mathbf{m}} \end{pmatrix} + \rho \begin{pmatrix} \cos\ell_y\sin\left(\ell_x - \ell_x^{\mathbf{m}}\right) \\ -\sin\ell_y^{\mathbf{m}}\cos\ell_y\cos\left(\ell_x^{\mathbf{m}} - \ell_x\right) + \cos\ell_y^{\mathbf{m}}\sin\ell_y \\ \cos\ell_y^{\mathbf{m}}\cos\ell_y\cos\left(\ell_x^{\mathbf{m}} - \ell_x\right) + \sin\ell_y^{\mathbf{m}}\sin\ell_y \end{pmatrix} \\
&\simeq \begin{pmatrix} 0 \\ 0 \\ -\rho^{\mathbf{m}} \end{pmatrix} + \rho \begin{pmatrix} \cos\ell_y\cdot\left(\ell_x - \ell_x^{\mathbf{m}}\right) \\ -\sin\ell_y^{\mathbf{m}}\cos\ell_y + \cos\ell_y^{\mathbf{m}}\sin\ell_y \\ \cos\ell_y^{\mathbf{m}}\cos\ell_y + \sin\ell_y^{\mathbf{m}}\sin\ell_y \end{pmatrix} \\
&\simeq \begin{pmatrix} 0 \\ 0 \\ -\rho^{\mathbf{m}} \end{pmatrix} + \rho \begin{pmatrix} \cos\ell_y\cdot\left(\ell_x - \ell_x^{\mathbf{m}}\right) \\ \sin\left(\ell_y - \ell_y^{\mathbf{m}}\right) \\ \cos\left(\ell_y - \ell_y^{\mathbf{m}}\right) \end{pmatrix} \simeq \begin{pmatrix} \rho\cdot\cos\ell_y\cdot\left(\ell_x - \ell_x^{\mathbf{m}}\right) \\ \rho\cdot\left(\ell_y - \ell_y^{\mathbf{m}}\right) \\ \rho - \rho^{\mathbf{m}} \end{pmatrix}.
\end{aligned}
$$

Let us note that when the robot is moving in a small-diameter area, we sometimes choose a reference point other than the center $\mathbf{m}$ of the robot, such as for instance its launch position.

4.2 Path planning

When the robot is completely autonomous, the desired path must be planned out [LaV 06]. Very often, these paths are polynomials, for two reasons. First of all, the space of polynomials has a vector space structure and can therefore utilize the power of linear algebra. Second, they are easier

to differentiate, which is useful for feedback linearization, since it requires the successive derivatives of the setpoints.

4.2.1 Simple example

Let us illustrate how such a planning is performed on the example of a robot tank. Assume that at the initial moment $t = 0$, the robot is located at the point (x_0, y_0) and that we would like to reach the point (x_1, y_1) at time t_1 with a speed equal to (v_x^1, v_y^1). We suggest a polynomial path of the form:

$$\begin{aligned} x_d &= a_x t^2 + b_x t + c_x \\ y_d &= a_y t^2 + b_y t + c_y \end{aligned}$$

We need to solve the system of equations:

$$\begin{array}{ll} c_x = x_0, & c_y = y_0 \\ a_x t_1^2 + b_x t_1 + c_x = x_1 & a_y t_1^2 + b_y t_1 + c_y = y_1, \\ 2a_x t_1 + b_x = v_x^1, & 2a_y t_1 + b_y = v_y^1 \end{array}$$

which is linear. We easily obtain:

$$\begin{pmatrix} a_x \\ a_y \\ b_x \\ b_y \\ c_x \\ c_y \end{pmatrix} = \begin{pmatrix} \frac{1}{t_1^2} x_0 - \frac{1}{t_1^2} x_1 + \frac{1}{t_1} v_x^1 \\ \frac{1}{t_1^2} y_0 - \frac{1}{t_1^2} y_1 + \frac{1}{t_1} v_y^1 \\ -v_x^1 - \frac{2}{t_1} x_0 + \frac{2}{t_1} x_1 \\ -v_y^1 - \frac{2}{t_1} y_0 + \frac{2}{t_1} y_1 \\ x_0 \\ y_0 \end{pmatrix}$$

We therefore have:

$$\begin{array}{ll} \dot{x}_d = 2a_x t + b_x & \dot{y}_d = 2a_y t + b_y \\ \ddot{x}_d = 2a_x, & \ddot{y}_d = 2a_y \end{array}$$

By inserting these quantities into a control law obtained using linearizing feedback (as is the case for instance with Equation [2.9]), we obtain a controller that meets our objectives.

4.2.2 Bézier polynomials

Here we will look at generalizing the approach presented in the previous section. Given the control points $\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n$, we can generate a polynomial $\mathbf{f}(t)$ such that $\mathbf{f}(0) = \mathbf{p}_0$, $\mathbf{f}(1) = \mathbf{p}_n$ and such that for $t \in [0, 1]$, the polynomial $\mathbf{f}(t)$ is successively attracted by the $\mathbf{p}_i$ with $i \in \{0, \ldots, n\}$. In order to correctly understand the method of building Bézier polynomials, let us examine various cases:

- case $n = 1$. We take the standard linear interpolation:

  $$\mathbf{f}(t) = (1 - t)\,\mathbf{p}_0 + t\mathbf{p}_1.$$

  The point $\mathbf{f}(t)$ corresponds to a barycenter between the control points $\mathbf{p}_0$ and $\mathbf{p}_1$, and the weights assigned to these two points change with time:

- case $n = 2$. We now have three control points $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2$. We create an auxiliary control point $\mathbf{p}_{01}$ which moves on the segment $[\mathbf{p}_0, \mathbf{p}_1]$ and another $\mathbf{p}_{12}$ which is associated with the segment $[\mathbf{p}_1, \mathbf{p}_2]$. We take:

$$\begin{aligned}\mathbf{f}(t) &= (1-t)\,\mathbf{p}_{01} + t\mathbf{p}_{12} \\ &= (1-t)\,\underbrace{((1-t)\,\mathbf{p}_0 + t\mathbf{p}_1)}_{\mathbf{p}_{01}} + t\underbrace{((1-t)\,\mathbf{p}_1 + t\mathbf{p}_2)}_{\mathbf{p}_{12}} \\ &= (1-t)^2\,\mathbf{p}_0 + 2\,(1-t)\,t\mathbf{p}_1 + t^2\mathbf{p}_2.\end{aligned}$$

  We thus obtain a second-order polynomial ;

- case $n = 3$. We apply the previous method for four control points. We obtain:

$$\begin{aligned}\mathbf{f}(t) &= (1-t)\,\mathbf{p}_{012} + t\mathbf{p}_{123} \\ &= (1-t)\,\underbrace{((1-t)\,\mathbf{p}_{01} + t\mathbf{p}_{12})}_{\mathbf{p}_{012}} + t\underbrace{((1-t)\,\mathbf{p}_{12} + t\mathbf{p}_{23})}_{\mathbf{p}_{123}} \\ &= (1-t)^3\,\mathbf{p}_0 + 3\,(1-t)^2\,t\mathbf{p}_1 + 3\,(1-t)\,t^2\mathbf{p}_2 + t^3\mathbf{p}_3\end{aligned}$$

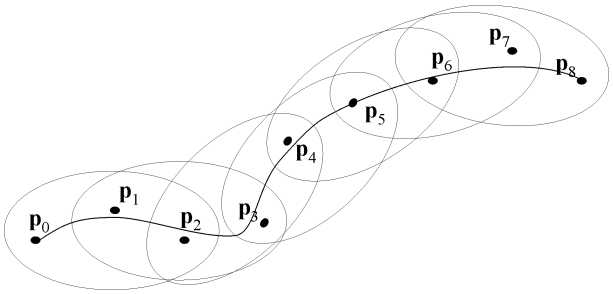

Figure 4.4: Illustration of second-order B-splines

- for a given n, we obtain:

$$\mathbf{f}(t) = \sum_{i=0}^{n} \underbrace{\frac{n!}{i!\,(n-i)!}\,(1-t)^{n-i}\,t^i}_{b_{i,n}(t)}\,\mathbf{p}_i.$$

The polynomials $b_{i,n}(t)$, called *Bernstein polynomials*, form a basis of the space of n-degree polynomials. When we increase the degree (in other words the number of control points), numerical instability and oscillations appear. This is called *Runge's phenomenon*. For complex curves with hundreds of control points, it is preferable to use B-splines corresponding to a concatenation of Bézier curves of limited order. Figure 4.4 illustrates such a concatenation in which, for each group of three points, we can calculate a second-order Bézier polynomial.

4.3 Voronoi diagram

Let us consider n points $\mathbf{p}_1, \ldots, \mathbf{p}_n$. Contrarily to the previous sections, the $\mathbf{p}_i$ here do not correspond to control points, but to point obstacles that we try to avoid. To each of these points,

we associate the set:

$$\mathbb{P}_i = \left\{ \mathbf{x} \in \mathbb{R}^d, \forall j, \|\mathbf{x} - \mathbf{p}_i\| \leq \|\mathbf{x} - \mathbf{p}_j\| \right\}.$$

For all i, this set is a polygon. The collection of these $\mathbb{P}_i$ is called a *Voronoi diagram*. Figure 4.5 represents a set of points with the associated Voronoi diagram. If an environment contains obstacles, the robot will have to plan a path that remains on the borders of the $\mathbb{P}_i$.

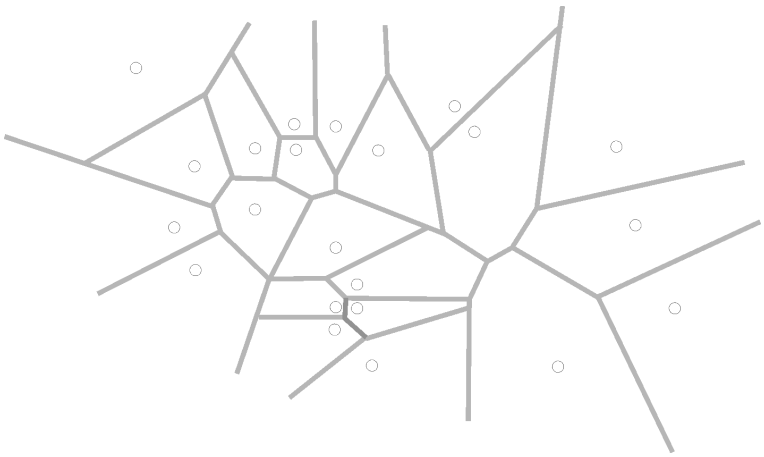

Figure 4.5: Voronoi diagram

Delaunay triangulation. Given n points in space, we can use the Voronoi diagrams to perform a triangulation of the space. This corresponding triangulation, referred to as *Delaunay triangulation*, allows maximizing the quantity of acute angles and thus avoid elongated triangles. It is obtained by connecting the neighboring points of the corresponding regions with an edge in the Voronoi diagram. In a Delaunay triangulation none of the triangles contains another point within its circumscribed circle. Figure 4.6 represents the Delaunay triangulation associated with the Voronoi diagram in Figure 4.5. A Delaunay triangulation is often used in robotics to represent space such as for example the area already explored, restricted areas, lakes, etc. We often associate a color with each triangle following the characteristics of the space the triangle belongs to (water, land, road, etc.).

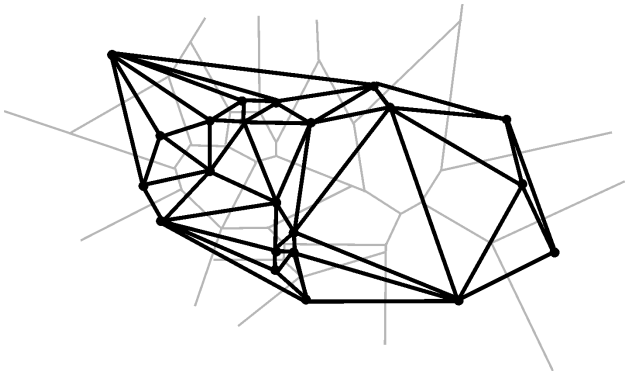

Figure 4.6: Delaunay triangulation

4.4 Artificial potential field method

A mobile robot has to move in a congested environment that contains mobile and stationary obstacles. The artificial potential field method [LAT 91] consists of imagining that the robot can behave like an electric particle that can be attracted or repelled by other objects following the sign of their electric charge. This is a reactive approach to guidance in which the path is not planned in advance. In physics, we have the following relation:

$$\mathbf{f} = -\text{grad}V(\mathbf{p}),$$

where $\mathbf{p}$ is the position of point particle in space, V is the potential and $\mathbf{f}$ the force applied on the particle. We will have the same relation in mobile robotics, but with $\mathbf{p}$ the position of the center of the robot, V a potential imagined by the robot and $\mathbf{f}$ the speed vector to follow. The potential fields will help us express a desired behavior for a robot. The obstacles will be represented by potentials exerting a repulsive force on the robot while the objective to follow will exert an attractive force. In a situation where several robots need to remain grouped while avoiding collisions, we can use a near-field repulsive potential and a far-field attractive potential. More generally, the vector fields used might not derive from a potential, as is the case if we would like the robot to have a cyclic behavior. The following table gives several types of potential that can be used:

Potential	$V(\mathbf{p})$	$-\text{grad}(V(\mathbf{p}))$
attractive conical	$\|\mathbf{p}-\hat{\mathbf{p}}\|$	$-\frac{\mathbf{p}-\hat{\mathbf{p}}}{\|\mathbf{p}-\hat{\mathbf{p}}\|}$
attractive quadratic	$\|\mathbf{p}-\hat{\mathbf{p}}\|^2$	$-2(\mathbf{p}-\hat{\mathbf{p}})$
attractive plane or line	$(\mathbf{p}-\hat{\mathbf{p}})^{\mathrm{T}} \cdot \hat{\mathbf{n}}\,\hat{\mathbf{n}}^{\mathrm{T}} \cdot (\mathbf{p}-\hat{\mathbf{p}})$	$-2\,\hat{\mathbf{n}}\,\hat{\mathbf{n}}^{\mathrm{T}}(\mathbf{p}-\hat{\mathbf{p}})$
repulsive	$\frac{1}{\|\mathbf{p}-\hat{\mathbf{q}}\|}$	$\frac{(\mathbf{p}-\hat{\mathbf{q}})}{\|\mathbf{p}-\hat{\mathbf{q}}\|^3}$
uniform	$-\hat{\mathbf{v}}^{\mathrm{T}} \cdot \mathbf{p}$	$\hat{\mathbf{v}}$

In this table, $\hat{\mathbf{p}}$ represents an attractive point, $\hat{\mathbf{q}}$ a repulsive point and $\hat{\mathbf{v}}$ a desired speed for the robot. In the case of the attractive plane, $\hat{\mathbf{p}}$ is a point of the plane and $\hat{\mathbf{n}}$ is a vector orthonormal to the plane. By adding several potentials, we can ask the robot (which is supposed to follow the direction that tends to decrease the potential) to accomplish its objectives while moving away from the obstacles. Figure 4.7 represents three vector fields derived from artificial potentials. The one on the left corresponds to a uniform field, the one in the middle to a repulsive potential that is added to a uniform field and the one on the right to the sum of an attractive potential and a repulsive potential.

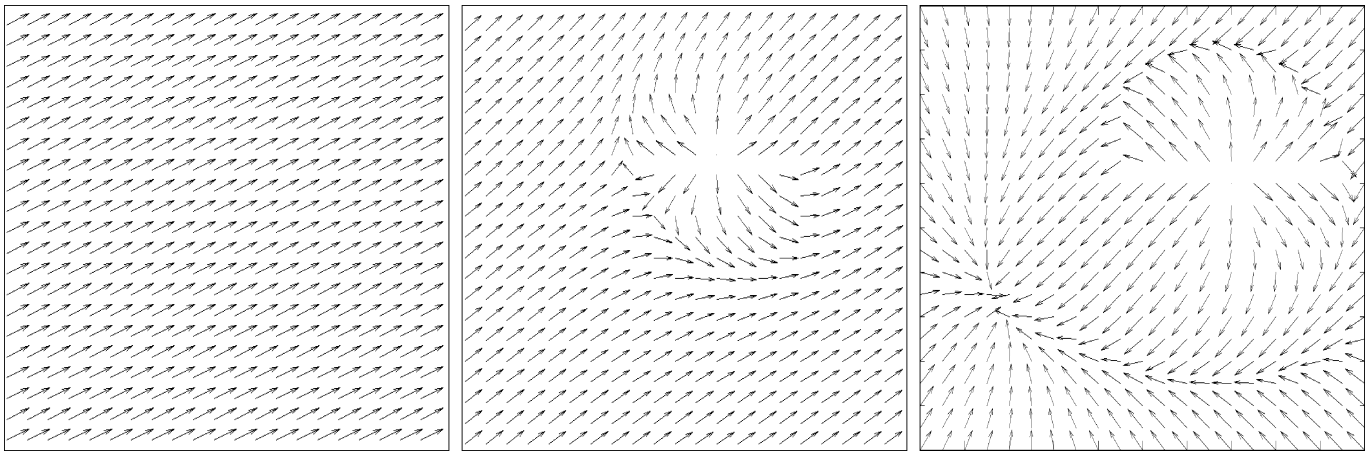

Figure 4.7: Artificial potential fields

Exercises

EXERCISE 4.1.– *Pursuit on a sphere*

Consider a robot $\mathcal{R}$ moving on the surface of a sphere similar to that of the Earth, with a radius of $\rho = 30$ m. This robot is located by its longitude ℓ_x, its latitude ℓ_y and its heading ψ, relative to the East. In a local coordinate system, the state equations of the robot are of the type:

$$\left\{ \begin{array}{rcl} \dot{x} & = & \cos\psi \\ \dot{y} & = & \sin\psi \\ \dot{\psi} & = & u \end{array} \right.$$

1) Give the state equations in the case in which the state vector is (ℓ_x, ℓ_y, ψ).

2) Simulate this evolving system graphically in 3D.

3) A second robot $\mathcal{R}_a$, described by the same equations, is moving randomly on the sphere (see Figure 4.8). Suggest a control law that allows the robot $\mathcal{R}$ to meet the robot $\mathcal{R}_a$.

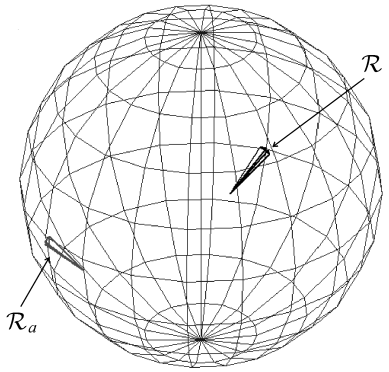

Figure 4.8: On the sphere, the robot $\mathcal{R}$ follows the robot $\mathcal{R}_a$

EXERCISE 4.2.– *Planning a path*

Let us consider a scene with two triangles as shown on Figure 4.9, and a robot described by the state equations:

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

with initial state $(x, y, \theta, v) = (0, 0, 0, 1)$. This robot has to reach the point with coordinates $(8, 8)$.

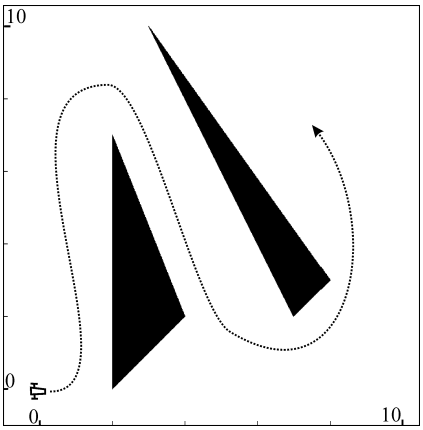

Figure 4.9: The robot has to follow a path without hitting the obstacles

1)Write a program in order to find the control points for a Bézier polynomial that connects the initial position to the desired position as shown on the Figure.

2) Using feedback linearization, deduce the control law that allows to reach the objective in 50 sec.

Exercise 4.3.– *Drawing a Voronoi diagram*

Let us consider the ten points in Figure 4.10. Draw the associated Voronoi diagram on a piece of paper as well as the corresponding Delaunay triangulation.

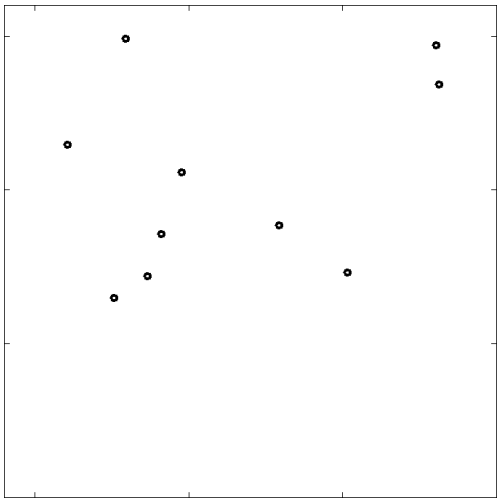

Figure 4.10: Ten points for which we want to build a Voronoi diagram

---

EXERCISE 4.4.– *Calculating a Voronoi diagram*

1) Show that if $\mathbf{x}$ and $\mathbf{y}$ are two vectors of $\mathbb{R}^n$, we have the so-called *polarization* equations:

$$\left\{ \|\mathbf{x}-\mathbf{y}\|^2 = \|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 - 2\langle \mathbf{x}, \mathbf{y}\rangle \right.$$

For this, develop the expression of the scalar product $\langle \mathbf{x}-\mathbf{y}, \mathbf{x}-\mathbf{y}\rangle$.

2) Let us consider $n+1$ points $\mathbf{a}^1, \ldots, \mathbf{a}^{n+1}$ and their circumscribed sphere denoted by $\mathcal{S}$. With the aid of the previous question, give an expression, in function of the $\mathbf{a}^i$, of the center $\mathbf{c}$ of $\mathcal{S}$ and of its radius r.

3) Let us now consider three points in the plane $\mathbf{a}^1, \mathbf{a}^2, \mathbf{a}^3$. Which conditions must be verified for $\mathbf{m}$ to be within the circle circumscribed to the triangle $(\mathbf{a}^1, \mathbf{a}^2, \mathbf{a}^3)$?

4) Consider m points in the plane $\mathbf{p}^1, \ldots, \mathbf{p}^m$, a Delaunay triangulation is a partition of the space into triangles $\mathcal{T}(k) = (\mathbf{a}^1(k), \mathbf{a}^2(k), \mathbf{a}^3(k))$, whose vertices are taken from the $\mathbf{p}^i$ such that each circle $\mathcal{C}(k)$ circumscribed to this triangle $\mathcal{T}(k)$ contains none of the $\mathbf{p}^i$. Write a program that takes $m = 10$ random points of the plane and draws a Delaunay triangulation. What is the complexity of the algorithm ?

5) Given the triangulation established in the previous question, build a Voronoi diagram associated with the points $\mathbf{a}^1, \ldots, \mathbf{a}^{n+1}$.

EXERCISE 4.5.– *Heading control of a Dubins car*

The results of this exercise will be used in Exercise 4.6 for calculating Dubins paths. A Dubins car is described by the state equations:

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

θ is the robot's heading and (x, y) the coordinates of its center. This robot has to be aligned with a heading setpoint $\bar{\theta}$.

1) We assume that the input $u \in [-1, 1]$. Give an analytic expression of the error angle $\delta \in [-\pi, \pi]$ in function of θ and $\bar{\theta}$ that indicates the angle the robot has to turn by in order to reach its setpoint as fast as possible. Taken into account that the expression of δ has to be periodic relative to θ and $\bar{\theta}$. Indeed, the angles of $-\pi, \pi$ or 3π have to be considered as equivalent. Give the associated control law and simulate it.

2) Same as above, with the exception that the robot can only turn left (in the direct trigonometric sense), in other words $u \in [0, 1]$. Let us note that now, $\delta \in [0, 2\pi]$.

3) Same as above, but the robot can now only turn right, i.e. $u \in [-1, 0]$.

EXERCISE 4.6.– *Dubins paths*

As in the previous exercise, let us consider a robot moving on a plane, described by:

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

where θ is the robot's heading and (x, y) the coordinates of its center. Its state vector is given by $\mathbf{x} = (x, y, \theta)$ and its input u must remain within the interval $[-u_{\max}, u_{\max}]$. This is the Dubins car [DUB 57] corresponding to the simplest possible nonholonomic mobile vehicle. Despite its simplicity, it illustrates many difficulties that may appear within the context of nonholonomic robots.

1) Calculate the maximum radius of curvature r that can be executed by the path of the robot.

2) Dubins showed that in order to switch from a configuration $\mathbf{a} = (x_a, y_a, \theta_a)$ to a configuration $\mathbf{b} = (x_b, y_b, \theta_b)$ that aren't too close together (in other words separated by a distance superior to $4r$), the minimum time strategy always consists of (1) turning to the maximum in one direction (in other words $u = \pm u_{\max}$) ; (2) moving straight ahead ; (3) then turning to the maximum again. The path corresponding to such a maneuver is called a *Dubins path*, and is thus composed of a starting arc, a segment and a termination arc. There are four ways to construct a Dubins path: LSL, LSR, RSL, RSR, where L means left, R means right and S stands for straight ahead. Give a configuration for **a** and **b** such that none of the four paths corresponds to an optimal strategy (and therefore **a** and **b** are quite close together). A situation in which the optimal strategy is RLR will be chosen.

3) In the case of an RSL strategy as illustrated on Figure 4.11, calculate the length L of the Dubins path in function of **a** and **b**.

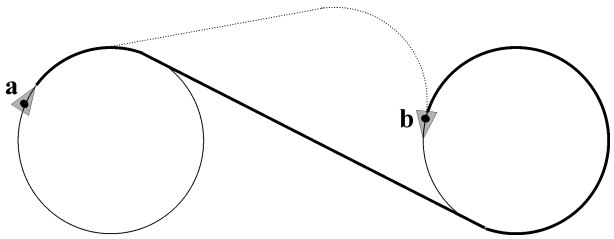

Figure 4.11: In bold : an RSL-type (*Right-Straight-Left*) Dubins path leading from **a** to **b** ; in dotted : an RLR-type Dubins path

4) In the case of an LSL strategy, calculate the length L of the Dubins path in function of **a** and **b**.

5) By using the reflection symmetry of the problem, deduce L in the case of RSR and LSR strategies, then write a function that calculates L in all situations. For this, the two Booleans ε_a ,ε_b will be used, which are equal to 1 if the corresponding arc is in the forward direction (i.e. to the left) and -1 otherwise.

6) Use the previous questions to write program that calculates the minimum length path for a Dubins car.

EXERCISE 4.7.– *Artificial potentials*

A robot situated at $\mathbf{p} = (x, y)$ must reach a target of unknown movement whose position $\hat{\mathbf{p}}$ and speed $\hat{\mathbf{v}}$ are known at the present time. This pair $(\hat{\mathbf{p}}, \hat{\mathbf{v}})$ might for instance correspond to a setpoint given by a human operator. A fixed obstacle located at position $\hat{\mathbf{q}}$ must be avoided. We model the desired behavior of our robot by the potential:

$$V(\mathbf{p}) = -\hat{\mathbf{v}}^{\mathrm{T}} \cdot \mathbf{p} + \|\mathbf{p} - \hat{\mathbf{p}}\|^2 + \frac{1}{\|\mathbf{p} - \hat{\mathbf{q}}\|}$$

where the potential $-\hat{\mathbf{v}}^{\mathrm{T}} \cdot \mathbf{p}$ represents the speed setpoint, the potential $\|\mathbf{p} - \hat{\mathbf{p}}\|^2$ makes the target position $\hat{\mathbf{p}}$ attractive and the potential $\frac{1}{\|\mathbf{p}-\hat{\mathbf{q}}\|}$ makes the obstacle $\hat{\mathbf{q}}$ repulsive.

1) Calculate the gradient of the potential $V(\mathbf{p})$ and deduce the speed vector setpoint $\mathbf{w}(\mathbf{p}, t)$ to apply to our robot so that it responds correctly to this potential.

2) We assume that our robot obeys the following state equations:

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

Give the control law that corresponds to the desired potential field. We will use the same principle as the one shown in Figure 4.12. First of all, disassemble the robot (in grey on the figure) into a chain made of two blocks. The first block forms the speeds from the actuators and the second builds the position vector $\mathbf{p} = (x, y)$. Then calculate the left inverse of the first block in order to end up with a system of the type:

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

Use a simple proportional control to perform this approximate inversion. Then, generate the new input $\left(\bar{v}, \bar{\theta}\right)$ using the potential to be satisfied. Illustrate the behavior of the robot with a target $\hat{\mathbf{p}} = (t, t)$ and a fixed obstacle placed at $\hat{\mathbf{q}} = (4, 5)$.

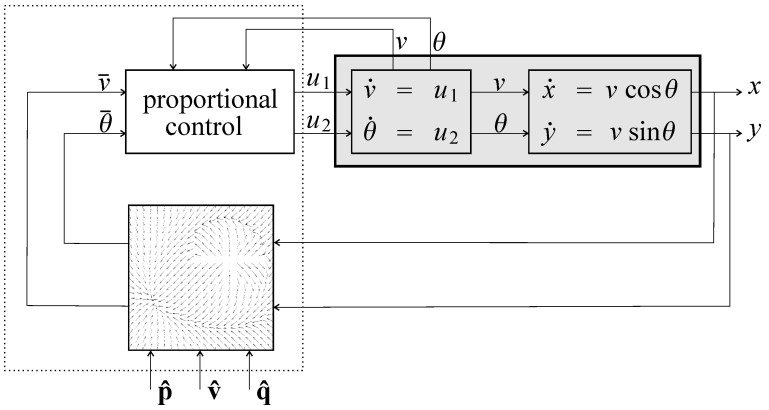

Figure 4.12: Controller (dotted) obtained by the potential method

3) We would now like to follow the target $\hat{\mathbf{p}}$ with a mobile obstacle at $\hat{\mathbf{q}}$ with:

$$\hat{\mathbf{p}} = \begin{pmatrix} \cos\frac{t}{10} \\ 2\sin\frac{t}{10} \end{pmatrix} \text{ and } \hat{\mathbf{q}} = \begin{pmatrix} 2\cos\frac{t}{5} \\ 2\sin\frac{t}{5} \end{pmatrix}.$$

Adjust the parameters of the potential in order to follow the target without hitting the obstacle. Illustrate the behavior of the controlled robot.

EXERCISE 4.8.– *Flocking*

We consider $m = 20$ robots described by the following state equations:

$$\begin{cases} \dot{x}_i &= \cos\theta_i \\ \dot{y}_i &= \sin\theta_i \\ \dot{\theta}_i &= u_i \end{cases}$$

The state vector is $\mathbf{x}(i) = (x_i, y_i, \theta_i)$. These robots can see all other robots, but are not able to communicate with them. We want that there robots behave as a flock as illustrated by Figure 4.13. Basic models of flocking behavior are controlled by three rules of Reynolds: the *separation* (short range repulsion) the *alignment* and the *cohesion* (long range attraction). Using a potential based method, find a controller for each robot to obtain a flock.

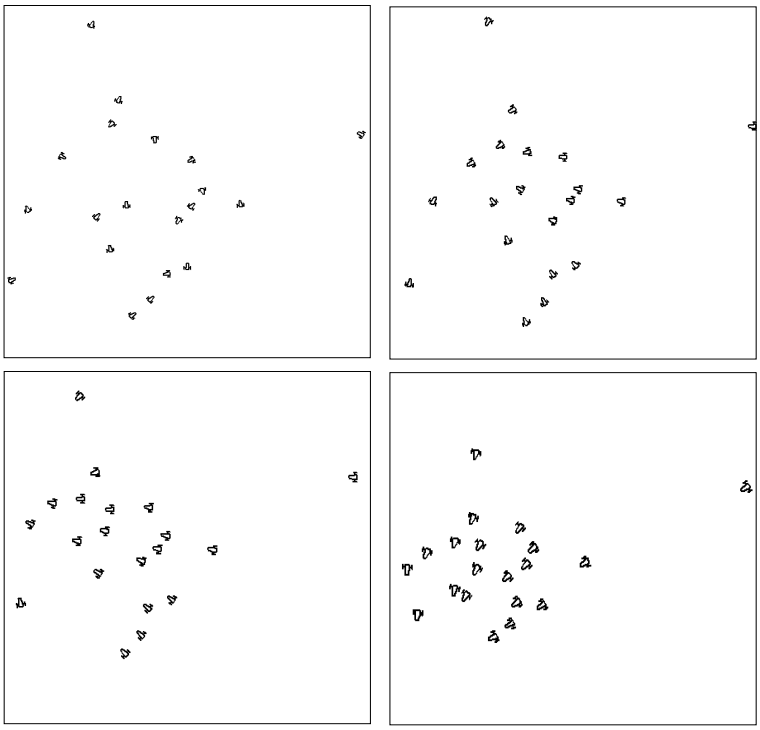

Figure 4.13: Illustration of the flocking behavior from a random initialization

---

EXERCISE 4.9.– *Traffic jam*

We consider $m = 10$ robots turning on a circular road of length radious $L = 100$m and with

radius $r = \frac{L}{2\pi}$. Each robot $\mathcal{R}_i$ satisfies the following state equations

$$\left\{ \begin{array}{rcl} \dot{a}_i & = & v_i \\ \dot{v}_i & = & u_i \end{array} \right.$$

The state vector is $\mathbf{x}(i) = (a_i, v_i)$ where a_i corresponds to the position of the robot and v_i to its speed. Each robot $\mathcal{R}_i$ is equipped with a radar which returns the distance d_i to the previous robot $\mathcal{R}_{i-1}$ and its derivative $\dot{d}_i$, as illustrated by Figure 4.14.

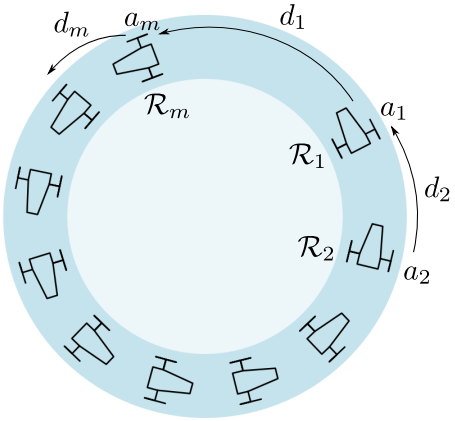

Figure 4.14: Traffic jam on the circle

1) Write the expression of the observation function $\mathbf{g}\left(a_{i^-}, a_i, v_{i^-}, v_i\right)$ which returns the vector $\mathbf{y}(i) = \left(d_i, \dot{d}_i\right)$. In this formula, $i^- = i - 1$ if $i > 0$ and $i^- = m$ if $i = 0$. Indeed, since the road is circular, the robot $\mathcal{R}_1$ follows the robot $\mathcal{R}_m$.

2) Propose a proportional and derivative control so that the robots will get a uniform distribution and go at a speed equal to $v_0 = 10ms^{-1}$. Check with a simulation.

3) In case of stability, prove theoretically that when the steady behavior is reached, all robots have a speed equal to v_0 and they are uniformly distributed.

4) Prove the stability of the system for 4 robots.

Bibliography

[BAZ 12] Bazeille S., Quidu I, Jaulin L., *Color-based underwater object recognition using water light attenuation*, Journal of Intelligent Service Robotics, vol. 5, **2**, 2012.

[BEA 12] Beard R., McLain T., *Small Unmanned Aircraft, Theory and Practice*, Princeton University Press, 2012.

[BOY 06] Boyer F., Alamir M., Chablat D., Khalil W., Leroyer A., Lemoine P., Robot anguille sous-marin en 3d, *Techniques de l'Ingénieur*, 2006.

[CHE 07] Chevallereau C., Bessonnet G., Abba G., Aoustin Y., *Les robots marcheurs bipèdes ; Modélisation, conception, synthèse de la marche, commande*, Hermès-Lavoisier, Paris, 2007.

[COR 11] Corke P., *Robotics, Vision and Control*, Springer, Berlin Heidelberg, 2011.

[CRE 14] Creuze V., Robots marins et sous-marins ; perception, modélisation, commande, *Techniques de l'ingénieur*, 2014.

[DEL 93] de Larminat P., *Automatique, commande des systèmes linéaires*, Hermès, Paris, France, 1993.

[DRE 11] Drevelle V., *Etude de méthodes ensemblistes robustes pour une localisation multisensorielle intègre, Application à la navigation des véhicules en milieu urbain*, PhD dissertation, Université de Technologie de Compiègne, Compiègne, France, 2011.

[DUB 57] Dubins L.E., On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents, *American Journal of Mathematics*, vol. 79, **3**, p. 497-516, 1957.

[FAN 01] Fantoni I., Lozano R., *Non-linear control for underactuated mechanical systems*, Springer-Verlag, 2001.

[FLI 13] Fliess M., Join C., Model-free control, *International Journal of Control*, vol. 86, **12**, p. 2228-2252, 2013.

[FOS 2002] Fossen T., *Marine Control Systems: Guidance, Navigation and Control of Ships, Rigs and Underwater Vehicles*, Marine Cybernetics, 2002.

[KAI 80] Kailath T., *Linear Systems*, Prentice Hall, Englewood Cliffs, 1980.

[GOR 11] Gorgues T., Ménage O., Terre T., Gaillard F., An innovative approach of the surface layer sampling, *Journal des Sciences Halieutique et Aquatique*, vol. 4, p. 105-109, 2011.

[MUR 89] Murata T., *Petri nets : properties, analysis and applications*. Proceedings of the IEEE, vol. 77, **4**, 1989.

[HER 10] Herrero P., Jaulin L., Vehi J., Sainz M.A., Guaranteed set-point computation with application to the control of a sailboat, *International Journal of Control Automation and Systems*, vol. 8, **1**, p. 1-7, 2010.

[JAU 05] Jaulin L., *Représentation d'état pour la modélisation et la commande des systèmes (Coll. Automatique de base)*, Hermès, London, 2005.

[JAU 10] Jaulin L., Commande d'un skate-car par biomimétisme, *CIFA 2010*, Nancy, France, 2010.

[JAU 12] Jaulin L., Le Bars F., An Interval Approach for Stability Analysis; Application to Sailboat Robotics, *IEEE Transaction on Robotics*, vol. 27, **5**, 2012.

[JAU 13] Jaulin L., *Automation for Robotics*, ISTE editions, 2016.

[JAU 13] Jaulin L., Mobile robotics, ISTE editions, 2015.

[JAU 02] Jaulin L., Kieffer M., Walter E., Meizel D., Guaranteed Robust Nonlinear Estimation with Application to Robot Localization, *IEEE Transactions on systems, man and cybernetics; Part C Applications and Reviews*, vol. 32, 2002.

[KAL 60] Kalman E.R. Contributions to the theory of optimal control. *Bol. Soc. Mat. Mex.*, vol. 5, p. 102-119, 1960.

[KHA 02] Khalil H.K., *Nonlinear Systems*, Third Edition, Prentice Hall, 2002

[KLE 06] Klein E.M.V., *Aircraft System Identification: Theory And Practice*, American Institute of Aeronautics and Astronautics, 2006.

[LAR 03] Laroche B., Martin P., Petit N., *Commande par platitude, Equations différentielles ordinaires et aux dérivées partielles*, Available at http://cas.ensmp.fr/∼petit/ensta/main.pdf, 2003.

[LAT 91] Latombe J., *Robot Motion Planning*, Kluwer Academic Publishers, Boston, MA, 1991.

[LAU 01] Laumond J., *La robotique mobile*, Hermès, Paris, France, 2001.

[LaV 06] LaValle S., *Planning algorithm*, Cambridge University Press, 2006.

[ROM 12] Romero-Ramirez M., *Contribution à la commande de voiliers robotisés*, PhD dissertation, Université Pierre et Marie Curie, France, 2012.

[SLO 91] Slotine J.J., Li W., *Applied nonlinear control*, Prentice Hall, 1991.

[SPO 01] Spong M., Corke P., Lozano R. *Nonlinear control of the Reaction Wheel Pendulum*, *Automatica*, vol. 37, p. 1845-1851, 2001.

[THR 05] Thrun S., Bugard W. and Fox D., *Probabilistic Robotics*, MIT Press, Cambridge, 2005.

[WAL 14] Walter E., *Numerical Methods and Optimization ; a Consumer Guide*, Springer, London, 2014.

Index