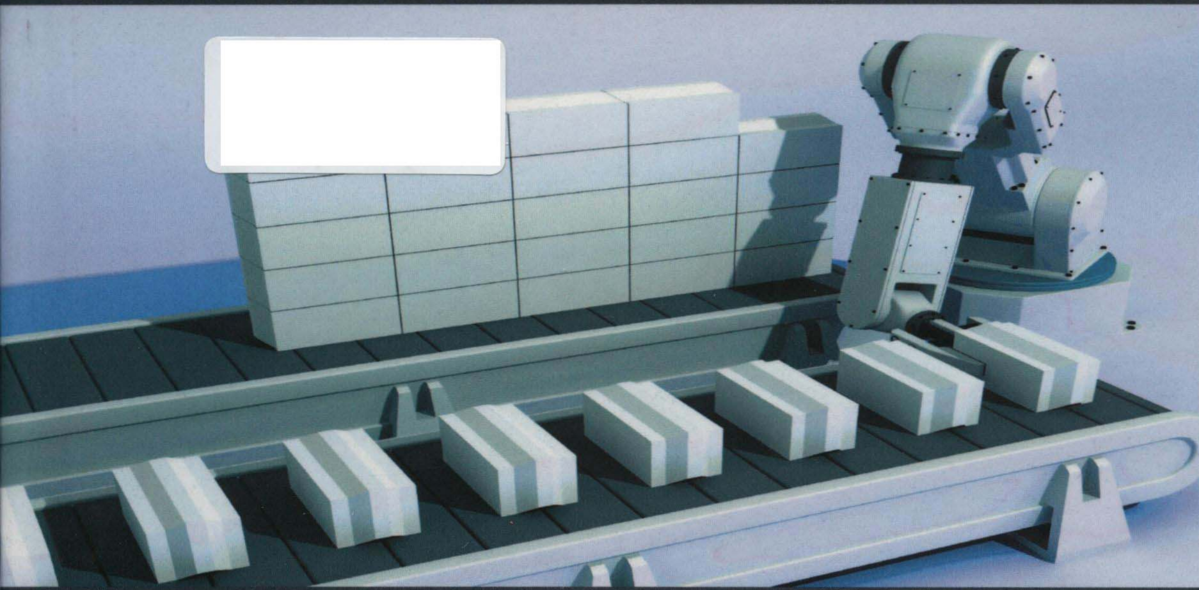
CONTROL, SYSTEMS AND INDUSTRIAL ENGINEERING SERIES



Automation for Robotics

Luc Jaulin

iSTE

WILEY

A discipline that is in full development, propelled by the rise of autonomous mobile robotics – notably drones – automation has the objective of designing controls capable of working within an existing dynamic system (automobile, airplane, economic system, etc.). The resulting controlled system is thus constructed by looping a physical system activated and equipped with sensors using smart electronics. While the initial system only obeyed the laws of physics, the evolution of the looped system also obeyed an IT program embedded in the control electronics.

In order to enable a better understanding of the key concepts of automation, this book develops the fundamental aspects of the field while also proposing numerous concrete exercises and their solutions. The theoretical approach that it presents fundamentally uses the state space and makes it possible to process general and complex systems in a simple way, involving several switches and sensors of different types. This approach requires the use of developed theoretical tools such as linear algebra, analysis and physics, generally taught in preparatory classes for specialist engineering courses.

Luc Jaulin is Professor in robotics at ENSTA-Bretagne in France. He conducts research at the Lab-STICC in the field of submarine robotics and sailing robots using set methods.

iSTE WILEY

www.iste.co.uk

9 781848 217980

Luc Jaulin

Automation for Robotics

WILEY

ISTE

Series Editor
Hisham Abou Kandil

Automation for Robotics

Luc Jaulin

iSTE

WILEY

First published 2015 in Great Britain and the United States by ISTE Ltd and John Wiley & Sons, Inc.

ISTE Ltd
27-37 St George's Road
London SW19 4EU
UK

www.iste.co.uk

John Wiley & Sons, Inc.
111 River Street
Hoboken, NJ 07030
USA

www.wiley.com

Library of Congress Control Number: 2014955868

British Library Cataloguing-in-Publication Data
A CIP record for this book is available from the British Library
ISBN 978-1-84821-798-0

Automation for Robotics

Introduction

I.1. State representation

Biological, economic and other mechanical systems surrounding us can often be described by a differential equation such as:

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

under the hypothesis that the time t in which the system evolves is continuous [JAU 05]. The vector $\mathbf{u}(t)$ is the *input* (or *control*) of the system. Its value may be chosen arbitrarily for all t. The vector $\mathbf{y}(t)$ is the *output* of the system and can be measured with a certain degree of accuracy. The vector $\mathbf{x}(t)$ is called the *state* of the system. It represents the memory of the system, in other words the information needed by the system in order to predict its own future, for a known input $\mathbf{u}(t)$. The first of the two equations is called the *evolution equation*. It is a differential equation that enables us to know where the state $\mathbf{x}(t)$ is headed knowing its value at the present moment t and the control $\mathbf{u}(t)$ that we are currently exerting. The second equation is called the *observation equation*. It allows us to calculate the output vector $\mathbf{y}(t)$, knowing the state and control at time t. Note, however, that, unlike the evolution equation, this equation is not a differential equation as it does not involve the derivatives of the signals. The two equations given above form the *state representation* of the system.

It is sometimes useful to consider a discrete time k, with $k \in \mathbb{Z}$, where $\mathbb{Z}$ is the set of integers. If, for instance, the universe is being considered as a computer, it is possible to consider that the time k is discrete and synchronized to the clock of the microprocessor. Discrete-time systems often respect a recurrence equation such as:

$$\begin{cases} \mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k)) \\ \mathbf{y}(k) = \mathbf{g}(\mathbf{x}(k), \mathbf{u}(k)) \end{cases}$$

The first objective of this book is to understand the concept of state representation through numerous exercises. For this, we will consider, in Chapter 1, a large number of varied exercises and show how to reach a state representation. We will then show, in Chapter 2, how to simulate a given system on a computer using its state representation.

The second objective of this book is to propose methods to *control* the systems described by state equations. In other words, we will attempt to build *automatic* machines (in which humans are practically not involved, except to give orders, or *setpoints*), called *controllers* capable of *domesticating* (changing the behavior in a desired direction) the systems being considered. For this, the controller will have to compute the inputs $\mathbf{u}(t)$ to be applied to the system from the (more or less noisy) knowledge of the outputs $\mathbf{y}(t)$ and from the setpoints $\mathbf{w}(t)$ (see Figure I.1).

From the point of view of the user, the system, referred to as a *closed-loop system*, with input $\mathbf{w}(t)$ and output $\mathbf{y}(t)$, will have a suitable behavior. We will say that we have *controlled* the system. With this objective of control, we will, in a first phase, only look at linear systems, in other words when the functions f and g are assumed linear. Thus, in the continuous-time case, the state equations of the system are written as:

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

and in the discrete-time case, they become:

$$\begin{cases} \mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k) \\ \mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{u}(k) \end{cases}$$

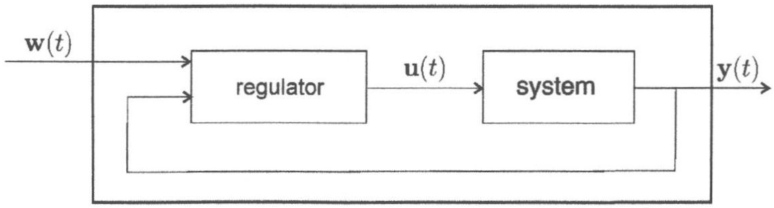


Figure I.1. *Closed loop concept illustrating the control of a system*

The matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ are called *evolution*, *control*, *observation* and *direct matrices*. A detailed analysis of these systems will be performed in Chapter 3. We will then explain, in Chapter 4, how to stabilize these systems. Finally, we will show in Chapter 5 that around certain points, called *operating points*, nonlinear systems behave like linear systems. It will then be possible to stabilize them using the same methods as those developed for the linear case.

Finally, this book is accompanied by numerous MATLAB programs available at:

http://www.ensta-bretagne.fr/jaulin/isteauto.html

I.2. Exercises

EXERCISE I.1.– Underwater robot

The underwater robot *Saucisse* of the Superior National School of Advanced Techniques (SNSAT) Bretagne [JAU 09], whose photo is given in Figure I.2, is a control system. It includes a computer, three propellers, a camera, a compass and a sonar. What does the input vector u, the output vector y, the state vector x and the setpoint w correspond to in this context? Where does the computer come in the control loop?

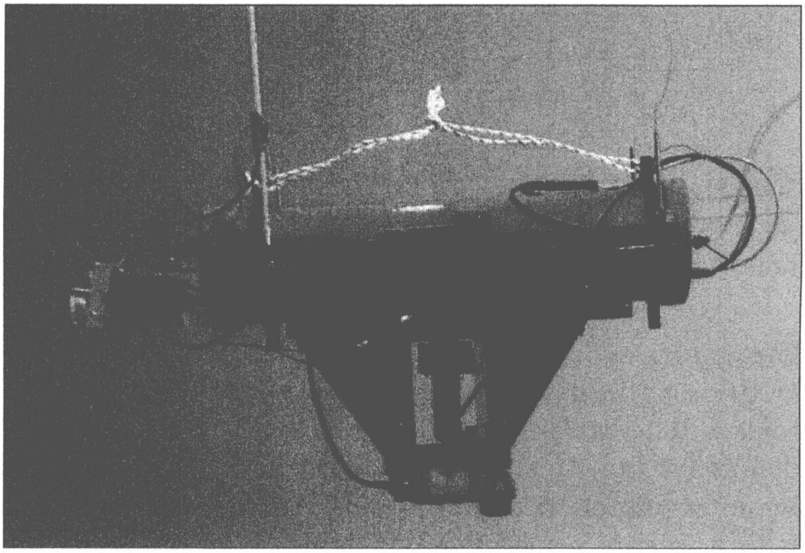
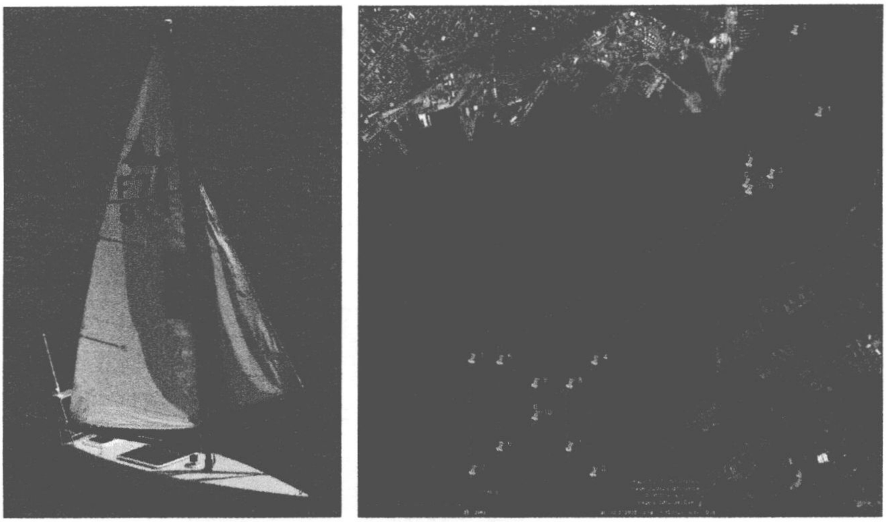


Figure I.2. *Controlled underwater robot*

EXERCISE I.2.– Sailing robot

The sailing robot *Vaimos* (French Research Institute for Exploitation of the Sea (FRIES) and SNSAT Bretagne) in Figure I.3 is also a control system [JAU 12a, JAU 12b]. It is capable of following paths by itself, such as the one drawn in Figure I.3. It has a rudder and a sail adjustable using a sheet. It also has an anemometer on top of the mast, a compass and a Global Positioning System (GPS). Describe what the input vector u, the output vector y, the state vector x and the setpoint w may correspond to.



a) b)

Figure I.3. *Sailing robot Vaimos a) and a path followed by Vaimos b). The zig-zags in the path are due to Vaimos having to tack in order to sail against the wind*

I.3. Solutions

Solution to Exercise I.1 (underwater robot)

The input vector $u \in \mathbb{R}^3$ corresponds to the electric voltage given to the three propellers and the output vector $y(t)$ includes the compass, the sonar data and the images taken by the cameras. The state vector x corresponds to the position, orientation and speeds of the robot. The setpoint w is requested by the supervisor. For instance, if we want to perform a course control, the setpoint w will be the desired speed and course for the robot. The controller is a pogram executed by the computer.

Solution to Exercise I.2 (sailing robot)

The input vector u $\in \mathbb{R}^2$ corresponds to the length of the sail sheet $\delta_v^{\max}$ and to the angle of the rudder δ_g. The output vector y $\in \mathbb{R}^4$ includes the GPS data m, the ultrasound anemometer (weather vane on top of the mast) ψ and the compass θ. The setpoint w indicates here the segment ab to follow. Figure I.4 illustrates this control loop. A supervisor, not represented on the figure, takes care of sequencing the segments to follow in such a way that the robot follows the desired path (here 12 segments forming a square box followed by a return to port).

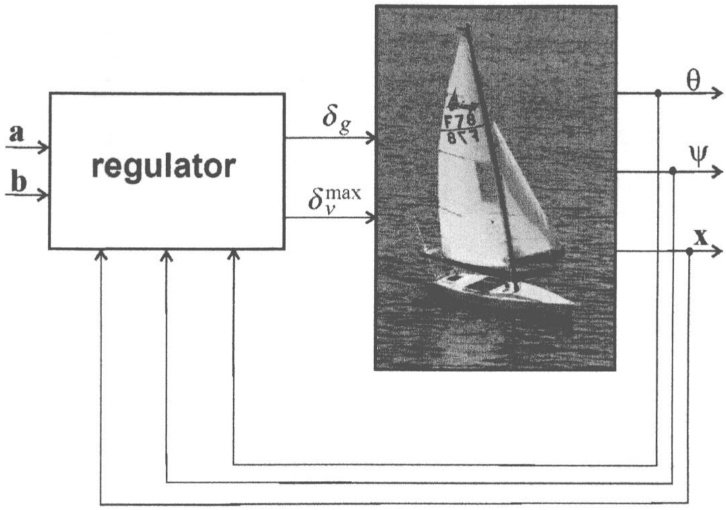


Figure I.4. *Control loop of the sailing robot*

Contents

1

Modeling

We will call *modeling* the step that consists of finding a more or less accurate state representation of the system we are looking at. In general, constant parameters appear in the state equations (such as the mass or the inertial moment of a body, the coefficient of viscous friction, the capacitance of a capacitor, etc.). In these cases, an identification step may prove to be necessary. In this book, we will assume that all the parameters are known, otherwise we invite the reader to consult Eric Walter's book [WAL 14] for a broad range of identification methods. Of course, no systematic methodology exists that can be used to model a system. The goal of this chapter and of the following exercises is to present, using several varied examples, how to obtain a state representation.

1.1. Linear systems

In the continuous-time case, linear systems can be described by the following state equations:

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

Linear systems are rather rare in nature. However, they are relatively easy to manipulate using linear algebra techniques and often approximate in an acceptable manner the nonlinear systems around their operating point.

1.2. Mechanical systems

The fundamental principle of dynamics allows us to easily find the state equations of mechanical systems (such as robots). The resulting calculations are relatively complicated for complex systems and the use of computer algebra systems may prove to be useful. In order to obtain the state equations of a mechanical system composed of several subsystems $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_m$, assumed to be rigid, we follow three steps:

1) *Obtaining the differential equations*. For each subsystem $\mathcal{S}_k$, with mass m and inertial matrix $\mathbf{J}$, the following relations must be applied:

$$\begin{aligned} \textstyle\sum_i \mathbf{f}_i &= m\mathbf{a} \\ \textstyle\sum_i \mathcal{M}_{\mathbf{f}_i} &= \mathbf{J}\dot{\omega} \end{aligned}$$

where the f_i are the forces acting on the subsystem $\mathcal{S}_k$, $\mathcal{M}_{\mathbf{f}_i}$ represents the torque created by the force f_i on $\mathcal{S}_k$, with respect to its center. The vector a represents the tangential acceleration of $\mathcal{S}_k$ and the vector $\dot{\omega}$ represents the angular acceleration of $\mathcal{S}_k$. After decomposing these $2m$ vectorial equations according to their components, we obtain $6m$ scalar differential equations such that some of them might be degenerate.

2) *Removing the components of the internal forces*. In differential equations there are the so-called *bonding* forces, which are internal to the whole mechanical system, even though they are external to each subsystem composing it. They represent the action of a subsystem $\mathcal{S}_k$ on another subsystem $\mathcal{S}_\ell$. Following the action–reaction principle, the