MULTIVARIABLE CONTROL AND COORDINATION SYSTEMS (EE-477)

Course Notes

http://react.epfl.ch

Denis Gillet
Lausanne, EPFL, 2017

Contents

1 Representation

1.1 System

The different interactions between a dynamic system and its environment are represented in the figure below.

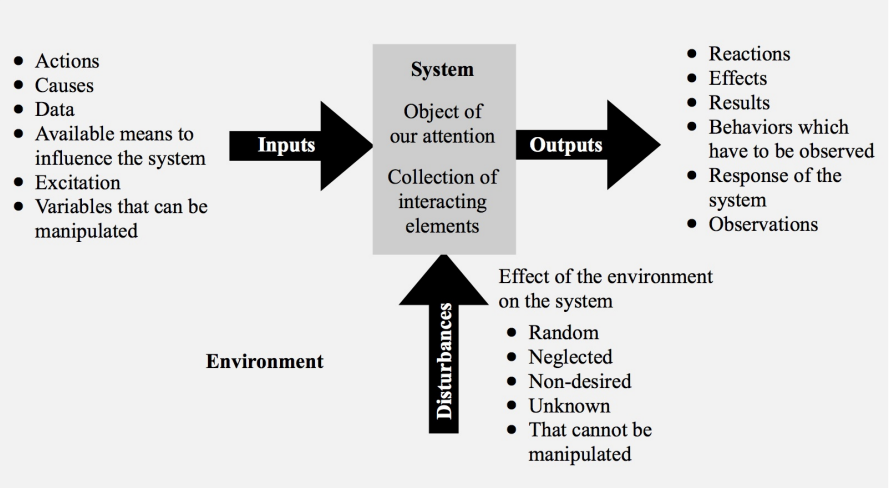


Figure 1.1 – Definitions of the quantities related to a system.

The **inputs** of a system represent the causes that act on it from the outside, for example disturbances, or standard excitations such as pulse, step and sinusoid. The outputs include the resulting effects and most often measured signals. The systems that are considered in this chapter are essentially multivariable, which means that they possess r inputs $u_1, u_2, ..., u_r$ cross-coupled with p outputs $y_1, y_2, ..., y_p$; thus, for example, the input u_1 can influence simultaneously several outputs. A schematic description of such a system is given in Figure 1.2.

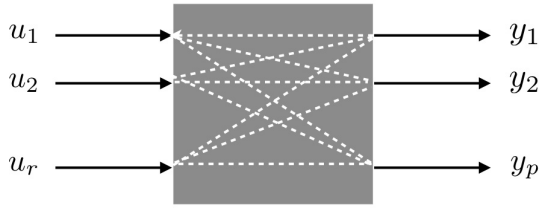


Figure 1.2 – Multivariable system with r inputs and p outputs.

Note that the inputs influence the state in which the system is and that the outputs depend on this state. Even if this state concept can be intuitively understood, it will be addressed formally in the next section. Note finally that only a mathematical representation of the mentioned interactions is able to provide the necessary information for analysis and design of control systems. Such a representation constitutes the **model** of the system.

The physical laws that govern the systems being of differential nature, the models are typically continuous. However, the behavior of the system can be defined or be of interest only at discrete instants. It is the case of models considered in control and simulation algorithms implemented on computer. An intrinsically discrete behavior arises also during sampled measures or during pulsed operations.

Multivariable systems are also referred as MIMO systems, where MIMO stands for Multiple Inputs, Multiple Outputs. Monovariable systems are a special case referred as SISO systems, where SISO stands for Single Input, Single Output.

1.2 Linearity

Any function f defined on a vectorial space V is called a functional.

A functional f is called **additive**, if $f(x+y) = f(x) + f(y)$. It is called **homogeneous**, if for any $x \in V$ and for any coefficient α we have $f(\alpha x) = \alpha f(x)$.

An additive and homogeneous functional is called a **linear** functional. An example is the defined integral which is a linear functional:

$$I(x) = \int_a^b x(t)\,dt$$

The combination of the two conditions mentioned above is known as the **principle of superposition**:

$$f(\alpha x + \beta y) = \alpha f(x) + \beta f(y)$$

A dynamic system whose model $y(t) = f[u(t), t]$ satisfies the principle of superposition is

called linear.
If the excitation $u_1(t)$ produces the output $y_1(t)$, that is to say if $y_1(t) = f[u_1(t), t]$ and if the excitation $u_2(t)$ produces the output $y_2(t)$, that is to say if $y_2(t) = f[u_2(t), t]$, then the weighted sum of the excitations leads to an output which is the weighted sum of the outputs obtained separately:

$$f[\alpha u_1(t) + \beta u_2(t), t] = \alpha f[u_1(t), t] + \beta f[u_2(t), t] = \alpha y_1(t) + \beta y_2(t)$$

We say that the system behaves in a **proportional** manner.

The system governed by the equation $\dot{y}(t) = au(t)$ is linear (Figure 1.3). Indeed:

$$a[\alpha u_1(t) + \beta u_2(t)] = \alpha a u_1(t) + \beta a u_2(t) = \alpha y_1(t) + \beta y_2(t)$$

The system which is governed by the equation $\dot{y}(t) = au(t) + b$ is not linear, in spite of the fact that this function is described by a straight line (Figure 1.4).
Indeed:

$a[\alpha u_1(t) + \beta u_2(t)] + b = \alpha a u_1(t) + \beta a u_2(t) + b$ is different from:

$$\begin{aligned} \alpha y_1(t) + \beta y_2(t) &= \alpha[a u_1(t) + b] + \beta[a u_2(t) + b] \\ &= \alpha a u_1(t) + \beta a u_2(t) + \mathbf{2}b \end{aligned}$$

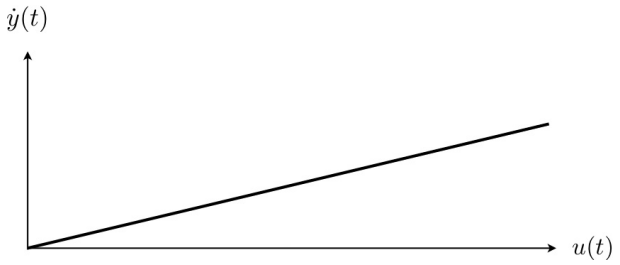


Figure 1.3 – Linear model.

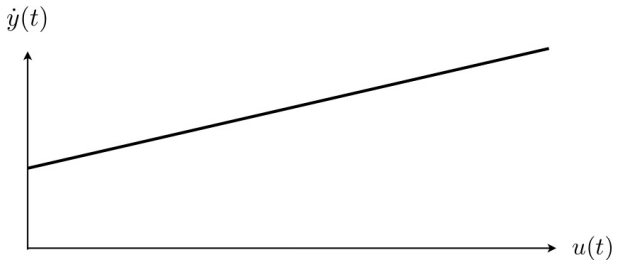


Figure 1.4 – Non-linear model.

A pure delay $y(t) = u(t-\tau)$ is a linear system as well. This can be easily demonstrated.

Consider $u_1 \rightarrow y_1$, that is to say that $y_1(t) = u_1(t-\tau)$, and consider $u_1 \rightarrow y_2$, that is to say that $y_2(t) = u_2(t-\tau)$. With $u \rightarrow y$ and with $u(t) = \alpha u_1(t) + \beta u_2(t)$, we have:

$$y(t) = u(t-\tau) = \alpha u_1(t-\tau) + \beta u_2(t-\tau) = \alpha y_1(t) + \beta y_2(t)$$

1.3 Signals

The input, output and disturbance signals of a system are either of a continuous nature, or of a discrete nature.

Consider $t \in \mathbb{R}$, a relative variable representing the elapsed time since the beginning of the observation. A real continuous signal is a continuous function $s : \{t : t \in \mathbb{R}\} \rightarrow \mathbb{R}$ of this variable. That is to say, this variable is defined for any time t.

Consider $\{t_k : k \in \mathbb{Z}\}$, which is a subset of real numbers which are called sampling instants. A real discrete signal is a function $w : \{t_k : k \in \mathbb{Z}\} \rightarrow \mathbb{R}$. This function is only defined at the sampling instants, as shown in Figure 1.5.

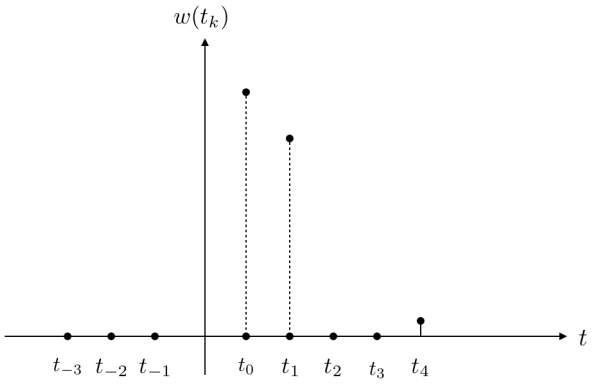


Figure 1.5 – Discrete signal.

In the present lecture notes, the sampling instants are uniformly spaced:

$$t_k = kh$$

We will say that the sampling is periodic or regular; the real number $h > 0$ is called the **sampling period** and $f_e = 1/h$ is the sampling frequency. When there is no risk of ambiguity, the notation $w(k)$ is introduced instead of $w(kh)$ to represent a discrete signal. This simplified notation does not cause any loss of generality.

1.4 Qualitative examples

1.4.1 Electrical drive

The electrical drive is a SISO and linear system. Its input is the supply voltage u and its output the angular position θ. Its dynamics can be described by a transfer function (TF) as shown in Figure 1.6.

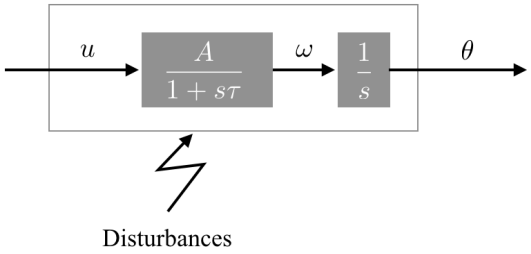


Figure 1.6 – Block diagram of an electrical drive.

A classical solution used in practice for controlling the angular position of this electrical drive is to introduce a PD controller (proportional, derivative) as shown in Figure 1.7.

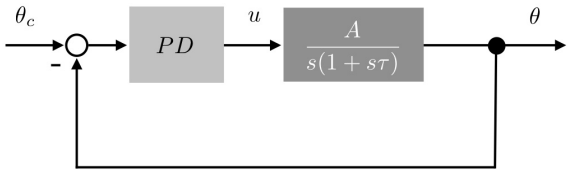


Figure 1.7 – Classical control structure for an electrical drive.

Another solution is to introduce two cascaded controllers as shown in Figure 1.8. This method presents better characteristics for disturbances rejection and it allows to limit the speed. However, it requires a speed and a position sensor.

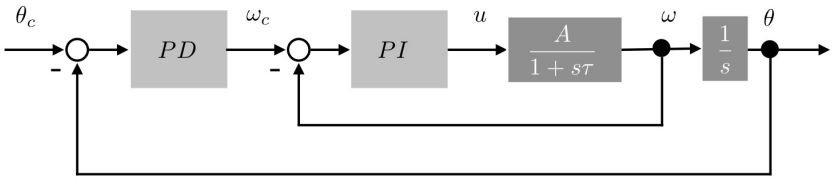


Figure 1.8 – Cascade control structure for an electrical drive.

An internal current limitation loop is sometimes also added.

The solutions described above can be too expensive when large series of drives have to be produced. In this case, it is wise to limit the number of sensors. In addition, these solutions are not able to take into account the constraints explicitly, such as the saturation of the actuator,

neither are they able to minimize the control effort (energy).

The state-space methods that we will study in this course allow to observe quantities that are unmeasurable or too expensive to be measured. It will be possible as well (with an optimal state controller) to control a MIMO system taking into account as example actuator saturations. However, the application of these techniques requires a good knowledge of the model, which is not the case, for example, during the design of a standard PID controller with the Ziegler-Nichols method.

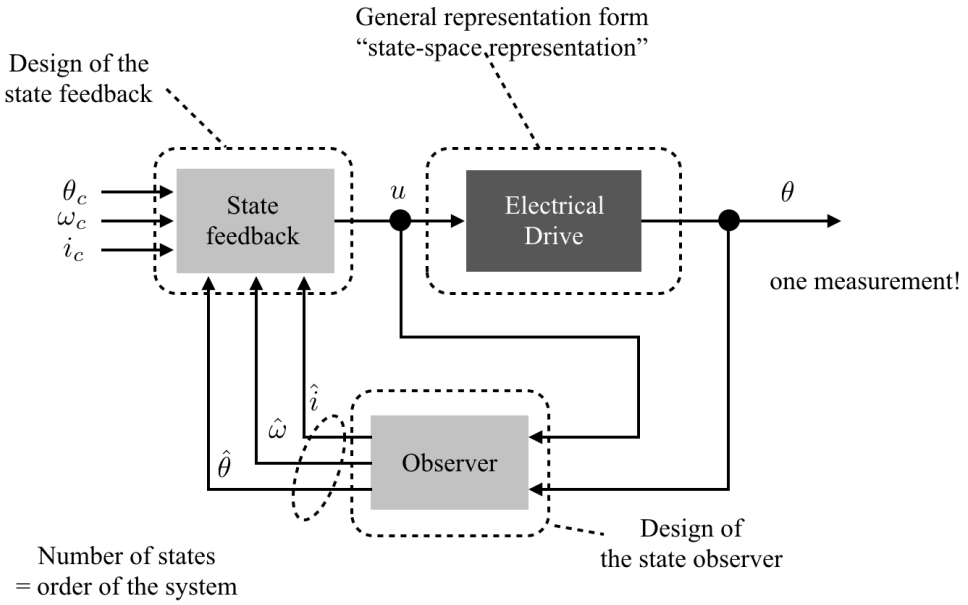


Figure 1.9 – Structure of a state feedback.

With such a control structure (Figure 1.9), and as we will see later, the whole dynamics of the system is controlled.

1.4.2 Inverted pendulum

The inverted pendulum is a simple academic system allowing the study of the dynamics of more complex unstable systems, like the dynamics of a rocket on its launch pad or like the dynamics of a boat. This is a SIMO system (single input, multiple outputs).

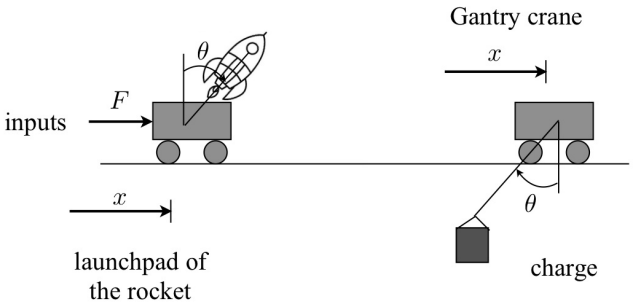


Figure 1.10 – Pendulums.

In such a system, the single input influences dynamically two outputs simultaneously (and the outputs influence themselves).

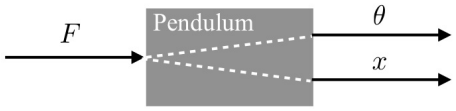


Figure 1.11 – Interdependence of the inputs-outputs of a pendulum.

A description by transfer function is not possible anymore in this case, which leads to the need for a more general representation form. We will see that a state-space representation is well adapted. The classical control solutions also do not allow to control simultaneously two quantities. We will see that the introduction of a state feedback is appropriate (Figure 1.12).

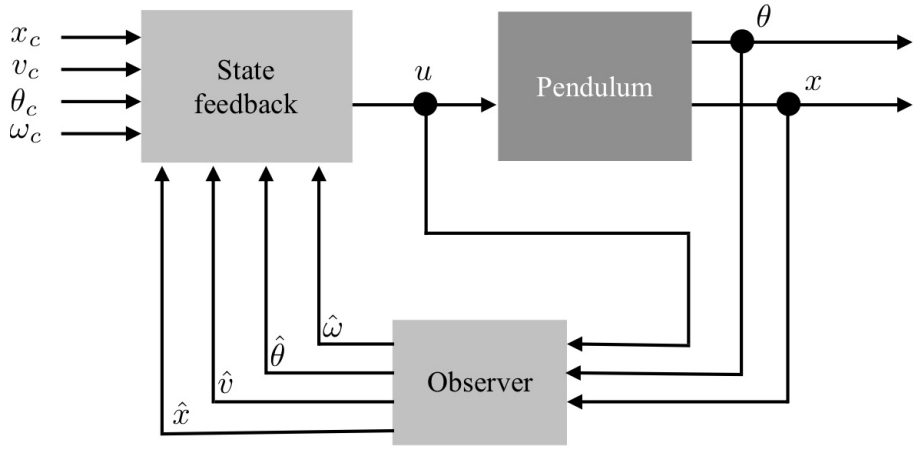


Figure 1.12 – Structure of the state feedback of a pendulum.

1.4.3 Two-degree-of-freedom robot

The robot shown below (Figure 1.13) possesses a joint allowing its arm to complete a full planar rotation under the action of a torque T. The arm itself can slide on the joint in order to move radially under the action of a force F. T and F are the inputs of this MIMO system,

respectively, its outputs are the angular position θ and the radial position r. The displacement of the arm is subject to a mechanical constraint, $r_{max} = r_0$.

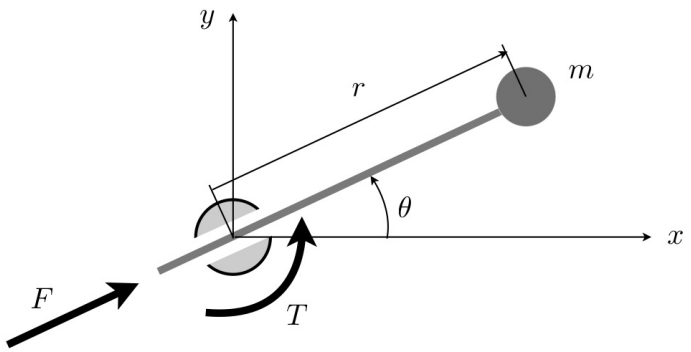


Figure 1.13 – Schematic representation of a robot.

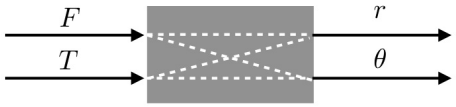


Figure 1.14 – Interdependence of the inputs-outputs of a robot.

A standard control technique for such a robot consists of introducing two controllers working separately (Figure 1.15), sometimes in an antagonist way. If the force F is modified, this modifies the arm length, therefore the inertia. The effect of T is then disturbed.

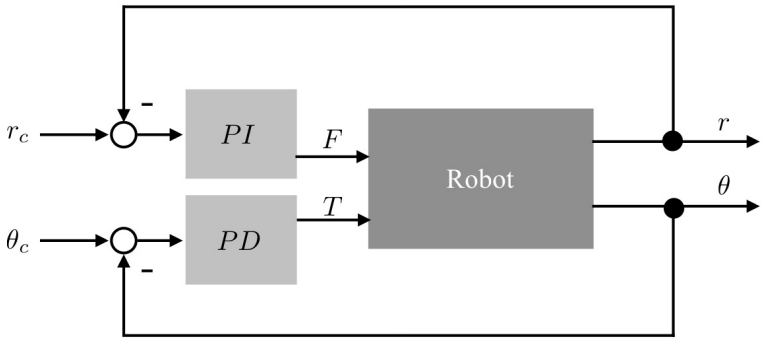


Figure 1.15 – Block diagram of the control of a robot.

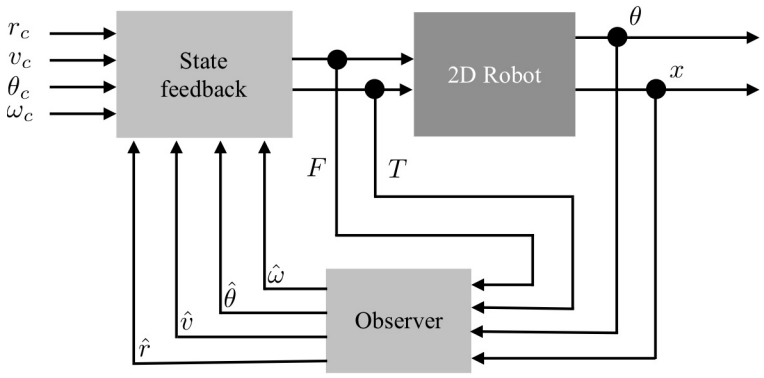


Figure 1.16 – Structure of the state feedback of a 2D Robot.

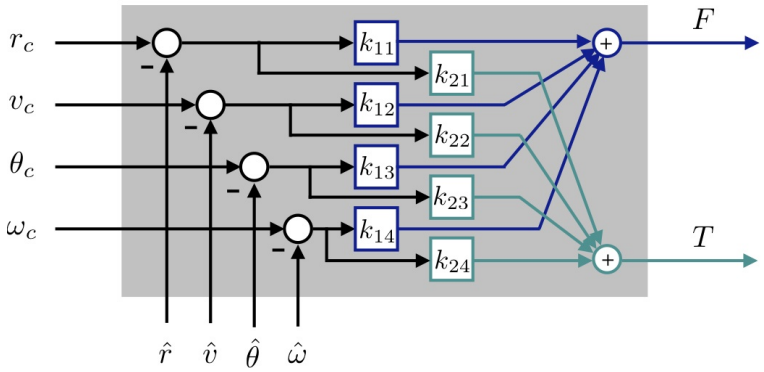


Figure 1.17 – General structure of a state feedback (weighted sum of the variation).

We will see that thanks to the state-space methods (Figures 1.16 and 1.17), it will be possible to take into account the internal coupling, or even to decouple the rotation and translation behaviors. That is to say that a modification of r_c will not induce an undesired modification of θ.

1.5 State and state variables

1.5.1 Introduction

The state of a system encompasses all its internal quantities that are subject to **change** over time, being under the effect of an addition or transfer of material, energy or information. In other words, the state corresponds to the levels of the material, energy or information reservoirs of the system. The behavior of a dynamic system depends obviously strongly on the state in which it is. There are multiple choices for the quantities which characterize the state. The level of a tank of known dimensions represents the state of the tank, but its volume can also represent the same state.

The mathematical relations which describe the dynamic interactions between the inputs and the outputs of continuous-time systems, especially electromechanical systems, have the

form of ordinary **differential** equations. In the case of discrete-time systems, the relations considered have the form of **difference** equations. The quantities which appear in the form of derivatives in differential equations, respectively in the form of differences in difference equations, play a key role in systems theory; they are called **state variables**. They carry the information about the past of the system; this information can be summed up to a precise **state**. These important quantities are now defined in a rigorous way and a general representation form, common to a large class of dynamic systems, is introduced. It is known as state-space model. Numerous system simulation and analysis methods are based on this form of representation. It is the same for the advanced tracking and estimation techniques.

1.5.2 Definitions

The **state** of a system at any time is the minimal information allowing, if the inputs are know at that time and after, the unique definition of the outputs at that time and after.

Thus, the state of a system is the information which sums up perfectly the past of the system, because it fixes any future evolution if the inputs are specified.

In this course, we only consider systems for which the number of states at a given time is a finite number n. These n states x_i are naturally gathered in a vector x of dimension n, called the **state vector**:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{1.1}$$

The coordinates $x_1, x_2, \cdots, x_n$ of the state vector are the **state variables**. The integer n is by definition the **order** of the system.

In order to simplify the notations, the inputs $u_1, u_2, \cdots, u_r$ are also gathered in a vector u of dimension r, called the **input vector**:

$$u = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_r \end{bmatrix} \tag{1.2}$$

It is the same for the outputs $y_1, y_2, \cdots, y_p$, gathered in a vector y of dimension p called the

output vector:

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_p \end{bmatrix} \tag{1.3}$$

The block diagram of figure 1.2 is often replaced by the one of figure 1.18, where the bold lines represent vector quantities.

Figure 1.18 – Block diagram of a multivariable system.

This figure highlights the fact that the state vector is an **internal** quantity of the system, often unmeasurable and therefore different from the output. Thus, any modeling will involve not only inputs and outputs, external quantities, but also internal state variables. This is the reason why such a representation is called **internal**.

1.5.3 Selection of the state variables

To extract the variables that will allow to describe the system in the chosen state-space representation, we must first make an inventory of the quantities that appear in a derivative form in the equations (but not being inputs). These quantities are noted γ_i, $i = 1, \cdots, n_\gamma$. The maximal respective derivation order of γ_i is noted ρ_i. This quantity defines the number of state variables that have to be introduced to represent a quantity γ_i. These state variables are, by successive derivations:

$$\gamma_i^{(0)}, \gamma_i^{(1)}, \cdots, \gamma_i^{(\rho_i - 1)}$$

where: $\gamma_i^{(0)} = \gamma_i$ and $\gamma_i^{(k)} = \frac{d^k \gamma_i}{dt^k}$ for $k = 1, \cdots, \rho_i - 1$

The order of the (linear or non-linear) system is equal to the sum of the ρ_i:

$$n = \sum_{i=1}^{n_\gamma} \rho_i$$

The inputs and outputs are defined by the physical system. The inputs are independent variables (typically variable controlled with actuators), and the outputs are measured quantities (observed through sensors). The quantities that are neither state variables nor inputs, nor outputs must be eliminated by substitution.

Consider for example the following system:

$$\left|\begin{array}{rl} a\ddot{w}(t)+\sin\dot{z}(t) & = u_1^2(t) \\ \sqrt{\dot{v}(t)}+\cos\dot{z}(t) & = u_2(t) \\ \dot{z}(t) & = \alpha t \end{array}\right.$$

The application of the method to this example gives:

i	**quantities** γ_i	**orders** ρ_i	**state variables**
1	v	1	$x_1 = v$
2	w	2	$x_2 = w, x_3 = \dot{w}$
3	z	1	$x_4 = z$

The order of this system is given by the expression $\rho_1 + \rho_2 + \rho_3 = 4$.

The state equation is obtained by derivation of the selected state variables. The original equations are exploited by substitution to express the derivatives of the state variables as a function of the state variables, the inputs and time.

$$\left|\begin{array}{lll} \dot{x}_1(t) = \dot{v}(t) & = (u_2(t)-\cos\dot{z}(t))^2 = (u_2(t)-\cos\alpha t)^2 \\ \dot{x}_2(t) = \dot{w}(t) & = x_3(t) \\ \dot{x}_3(t) = \ddot{w}(t) & = \frac{1}{a}(u_1(t)^2-\sin\alpha t) \\ \dot{x}_4(t) = \dot{z}(t) & = \alpha t \end{array}\right.$$

The same approach is exploited to express the outputs as a function of the state variables, the inputs and time. For example, if only the quantity w is measured, then:

$$y(t) = w(t) = x_2(t)$$

1.6 Continuous-time state-space model

1.6.1 Definitions

After having applied the physical laws (1st principles) which govern them, numerous physical systems can be described by differential and algebraic equations showing the following structure:

$$\left|\begin{array}{ll} \dot{x}_1(t) & = f_1[x_1(t),\cdots,x_n(t),u_1(t),\cdots,u_r(t),t] \\ \dot{x}_2(t) & = f_2[x_1(t),\cdots,x_n(t),u_1(t),\cdots,u_r(t),t] \\ & \vdots \\ \dot{x}_n(t) & = f_n[x_1(t),\cdots,x_n(t),u_1(t),\cdots,u_r(t),t] \end{array}\right. \tag{1.4}$$

$$\left|\begin{array}{ll} y_1(t) & = g_1[x_1(t),\cdots,x_n(t),u_1(t),\cdots,u_r(t),t] \\ y_2(t) & = g_2[x_1(t),\cdots,x_n(t),u_1(t),\cdots,u_r(t),t] \\ & \vdots \\ y_p(t) & = g_p[x_1(t),\cdots,x_n(t),u_1(t),\cdots,u_r(t),t] \end{array}\right. \tag{1.5}$$

The variable t is time. We introduce the vector notations:

$$x(t)=\begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{bmatrix}; u(t)=\begin{bmatrix} u_1(t) \\ u_2(t) \\ \vdots \\ u_r(t) \end{bmatrix} \text{ and } y(t)=\begin{bmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y_p(t) \end{bmatrix}$$

Functions f and g are given by:

$$f[x(t),u(t),t]=\begin{bmatrix} f_1[x(t),u(t),t] \\ f_2[x(t),u(t),t] \\ \vdots \\ f_n[x(t),u(t),t] \end{bmatrix}, g[x(t),u(t),t]=\begin{bmatrix} g_1[x(t),u(t),t] \\ g_2[x(t),u(t),t] \\ \vdots \\ g_p[x(t),u(t),t] \end{bmatrix}$$

By definition:

$$\dot{x}(t) = \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \vdots \\ \dot{x}_n(t) \end{bmatrix}$$

Relations (1.4) and (1.5) can then be written in a very compact way:

$$\dot{x}(t) = f[x(t, u(t), t)] \tag{1.6}$$
$$y(t) = g[x(t, u(t), t)] \tag{1.7}$$

The vector $u(t)$ is the input of the system and $y(t)$ is its output. Assume now that $x(t_0)$ and $u(t)$, $t \geqslant t_0$, are known. Equation (1.6) can conceptually be solved for $t \geqslant t_0$, assuming that the conditions of existence and of uniqueness of a solution are satisfied; relation (1.7) allows then the calculation of $y(t)$ for $t \geqslant t_0$. Thus, as the notation let it be assumed, $x(t_0)$ forms the state at time t_0.

Equation (1.6) governing the dynamic behavior of the system is called the **state equation**; relation (1.7) providing its output is the **output equation**. Note that, in practice, it is rare that $u(t)$ appears. These two equations form the **state-space model**. It is said to be continuous-time and differential.

1.6.2 State-space representation of linear systems

An extremely important particular case of state-space model is the one in which functions f and g possess the following form:

$$f[x(t), u(t), t] = A(t)x(t) + B(t)u(t)$$
$$g[x(t), u(t), t] = C(t)x(t) + D(t)u(t)$$

$A(t), B(t), C(t),$ and $D(t)$ are respectively matrices of dimensions $n \times n$, $n \times r$, $p \times n$ and $p \times r$.

The state-space representation becomes then:

$$\dot{x}(t) = A(t)x(t) + B(t)u(t) \tag{1.8}$$
$$y(t) = C(t)x(t) + D(t)u(t) \tag{1.9}$$

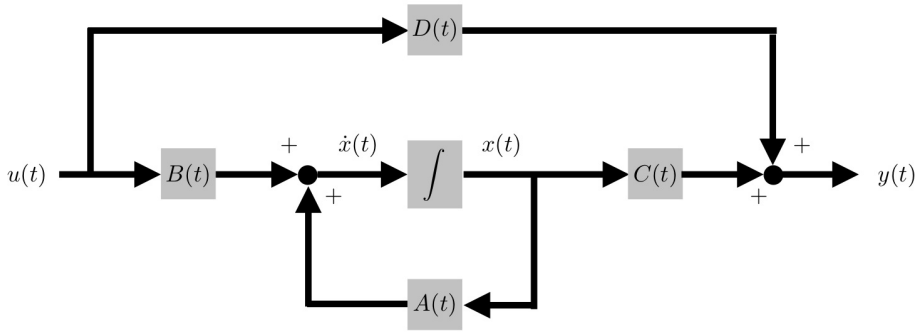


Figure 1.19 – Block diagram of a continuous-time linear system.

A continuous system described by relations (1.8) and (1.9) is called **linear**. It is **time-variant** if the matrices appearing in these equations depend on time and it is **time-invariant** in the opposite situation as shown in equations (1.10) and (1.11). Figure 1.19 provides the block diagram of such systems.

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{1.10}$$

$$y(t) = Cx(t) + Du(t) \tag{1.11}$$

Note that, in this block diagram, state variables appear at the outputs of the integrators. The diverse roles played by the matrices of the model stand out now clearly; the **input matrix** $B(t)$ sums up the link of the input vector $u(t)$ with the dynamic part characterized by the **system matrix** $A(t)$; the output matrix $C(t)$ represents the connection between this dynamic portion and the output vector $y(t)$ while the **transition matrix** $D(t)$ indicates a direct effect of $u(t)$ on $y(t)$.

The monovariable case is characterized by $r = p = 1$; the matrix $B(t)$ becomes then a column vector $b(t)$ of dimension n, the matrix $C(t)$ becomes a line vector $c^T(t)$ of dimension n, and finally the matrix $D(t)$ becomes a number $d(t)$.

$$\dot{x}(t) = A(t)x(t) + b(t)u(t)$$

$$y(t) = c^T(t)x(t) + d(t)u(t)$$

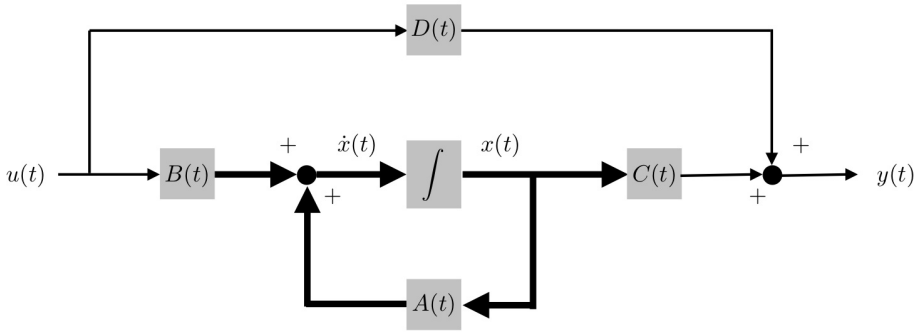


Figure 1.20 – Block diagram of a monovariable continuous-time linear system.

1.6.3 Example: electrical drive

We consider here an electrical drive composed of a DC motor (direct current motor) separately excited and coupled to a load, as illustrated in figure 1.21.

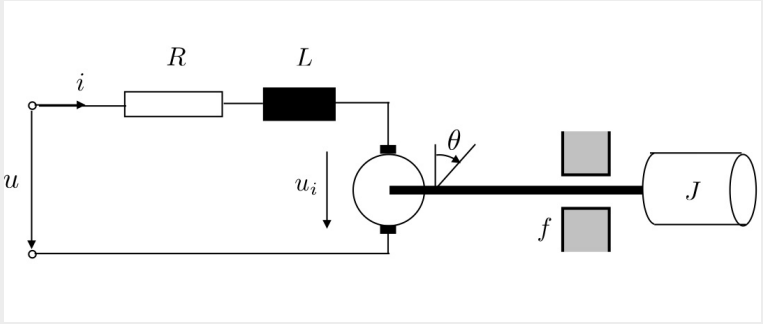


Figure 1.21 – Electrical drive.

R and L are respectively the resistance and the inductance of the armature circuit, in which the current i flows, while J and f denote the inertia and the viscous friction coefficient of the load. The angular θ position of the load is the output of the system; its input is the variable supply voltage u of the motor.

We know that the induced tension u_i is equal to $k\dot{\theta}$, where k is a constant and that the couple provided by the motor is ki. Thus:

$$u = Ri + L\frac{di}{dt} + k\dot{\theta}$$
$$J\ddot{\theta} = ki - f\dot{\theta}$$

Consider the state variables $x_1 = \theta, x_2 = \dot{\theta}$ and $x_3 = i$; the state-space model is easily established:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -\frac{f}{J} & \frac{k}{J} \\ 0 & -\frac{k}{L} & -\frac{R}{L} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{1}{L} \end{bmatrix} u$$
$$y = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

This description can be simplified if we neglect the inductance of the armature circuit. Indeed,

with $L = 0$, the equation is reduced to $u = Ri + k\dot{\theta}$. Hence the armature circuit:

$$i = \frac{u - k\dot{\theta}}{R}$$

The mechanical equation becomes:

$$J\ddot{\theta} = -\left(\frac{k^2}{R} + f\right)\dot{\theta} + \frac{k}{R}u$$

Only the first two state variables x_1 and x_2 previously selected are now necessary to describe the system. The simplified state-space model is thus:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & a \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ b \end{bmatrix} u, \text{ with } a = -\frac{1}{J}\left(\frac{k^2}{R} + f\right) \text{ and } b = \frac{k}{JR}$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

It is a linear and time-invariant system. These equations are indeed of the form $\dot{x}(t) = Ax(t) + Bu(t)$ and $y(t) = Cx(t) + Du(t)$, with $D = 0$ in this particular case.

1.7 Simulation of systems represented by state-space models

1.7.1 Introduction

We have examined in the previous section the state-space representation of dynamic systems. This modeling by first order differential equation is appropriate for the numerical simulation of the system.

The **simulation** of a system consists in determining the state $x(t)$ and the output $y(t)$ for $t \geqslant t_0$ when the input of the system $u(t)$ for $t \geqslant t_0$ and the initial state $x(t_0)$ are known.

There is no analytical solution to this general problem. We must use numerical integration methods to solve the system of differential equations corresponding to the state equation. These methods do not provide a function that is continuous in time as a solution, we only obtain samples of the desired function, usually uniformly spaced. The time interval between them, called the **integration step** is noted h. The instants at which the solution is known are given by:

$$t = kh, k = k_0, k_0 + 1, k_0 + 2, \cdots \in \mathbb{N} \tag{1.12}$$

The most intuitive integration method, proposed by Euler, requires first to write the state at time $t = kh$:

$$\dot{x}(kh) = f[x(kh), u(kh), kh] \tag{1.13}$$

$$y(kh) = g[x(kh), u(kh), kh] \tag{1.14}$$

then to use the first terms of the Taylor series expansion as an approximation of the derivative of the state:

$$x(kh + h) \approx x(kh) + \dot{x}(kh)h$$

so:

$$\dot{x}(kh) \approx \frac{x(kh + h) - x(kh)}{h} \tag{1.15}$$

Which ends up to:

$$x(kh + h) \approx x(kh) + hf[x(kh), u(kh), kh] \tag{1.16}$$

The state, the approximation of its derivative and the selected samples are represented in Figure 1.22:

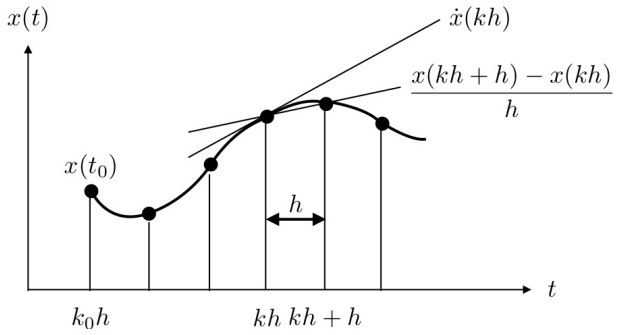


Figure 1.22 – Approximation of the state provided by a numerical integration method

After having carefully selected the integration step h, the simulation is made by exploiting iteratively equations (1.16) and (1.14):

$$
\begin{aligned}
&x(k_0h+h) \approx x(k_0h) + hf[x(k_0h), u(k_0h), k_0h] \\
&y(k_0h) = g[x(k_0h), u(k_0h), k_0h] \\
&x(k_0h+2h) \approx x(k_0h+h) + hf[x(k_0h+h), u(k_0h+h), k_0h+h] \\
&y(k_0h+h) = g[x(k_0h+h), u(k_0h+h), k_0h+h] \\
&\qquad\qquad \vdots \\
&x(kh+h) \approx x(kh) + hf[x(kh), u(kh), kh] \\
&y(kh) = g[x(kh), u(kh), kh] \\
&\qquad\qquad \vdots
\end{aligned}
$$

This method is not accurate and might not converge, but allows a quick evaluation of the solution of the state equation. A more sophisticated and commonly used method is the Runge-Kutta one. It is presented in the following paragraph.

1.7.2 Runge-Kutta method

The Runge-Kutta integration method provides a solution to the simulation problem defined above in the following form:

$$
\begin{aligned}
&x(kh+h) \approx x(kh) + \frac{1}{6}(a(kh) + 2b(kh) + 2c(kh) + d(kh)) \\
&y(kh) = g[x(kh), u(kh), kh]
\end{aligned}
$$

where $a(kh), b(kh), c(kh)$ and $d(kh)$ are vectors of dimension n given by:

$$\begin{aligned} a(kh) &= hf[x(kh), u(kh), kh] \\ b(kh) &= hf[x(kh) + \tfrac{1}{2}a(kh), u(kh + \tfrac{1}{2}h), kh + \tfrac{1}{2}h] \\ c(kh) &= hf[x(kh) + \tfrac{1}{2}b(kh), u(kh + \tfrac{1}{2}h), kh + \tfrac{1}{2}h] \\ d(kh) &= hf[x(kh) + c(kh), u(kh + h), kh + h] \end{aligned}$$

If the control signal $u(t)$ is not defined between kh and $kh + h$, the intermediate values must be computed by interpolation between $u(kh)$ and $u(kh + h)$.

This method is based on a Taylor series expansion of the function f up to the fourth order terms.

We note that the format that was introduced to represent state-space models provides, in the case of the Runge-Kutta algorithm, a very compact representation of its equations.

This method is of a satisfying precision and has in addition the advantage of diverging very quickly if the choice of the integration step h is not adequate, thus allowing to detect easily wrong results. It is available in the standard mathematical libraries and software packages such as MATLAB or Mathematica.

1.7.3 Remark

The numerical simulation of a continuous-time system offers numerous advantages. First, the implemented algorithms are perfectly known, which allows to control the precision of the results. The handling of multivariable systems does not induce any problem and the non-linear functions can be constructed easily, either by means of polynomials or transcendental functions obtained by interpolation, or by their storage in tables of values. However, we must not forget that the numerical calculation is approximative. In particular, the representation of numbers by a limited set of digits induces approximation errors that cannot be negligible depending on the precision of the calculator that is used. The truncation errors are induced by the approximation of the mathematical functions by means of numerical series from which we can only get a limited number of terms. Finally, the choice of the method and of the integration step ensuring the convergence and a good precision of the results is not always easy. The numerical stability of an algorithm is guaranteed when the integration error decreases at each iteration. This stability is often disturbed when the orders of magnitude of the numbers that are involved in the operations are very different. It is sometimes necessary to scale signals to overcome this problem. A great part of the difficulties mentioned in this section disappears if the state-space model is linear, especially if it is time-invariant. Indeed, in this particular case, the tools of linear algebra provide an **analytical** solution to the problem of the simulation.

1.8 Discrete-time state-space model

The intrinsically discrete-time systems, the discretized representations of continuous-time systems, or the numerical simulation algorithms described in the previous section are represented by difference equations and algebraic equations. Difference equations of any structure can be presented in the form of a system of first order difference equations, by an appropriate choice of the state variables. This choice is made in accordance with the technique described in paragraph 1.5.3, where the notion of derivative is replaced by the notion of shift of a sampling period. Thus, a discrete state-space model exhibits the following structure:

$$\left|\begin{array}{ll} x_1(k+1) & = f_1[x_1(k),\cdots,x_n(k),u_1(k),\cdots,u_r(k),k] \\ x_2(k+1) & = f_2[x_1(k),\cdots,x_n(k),u_1(k),\cdots,u_r(k),k] \\ & \vdots \\ x_n(k+1) & = f_n[x_1(k),\cdots,x_n(k),u_1(k),\cdots,u_r(k),k] \end{array}\right. \tag{1.17}$$

$$\left|\begin{array}{ll} y_1(k) & = g_1[x_1(k),\cdots,x_n(k),u_1(k),\cdots,u_r(k),k] \\ y_2(k) & = g_2[x_1(k),\cdots,x_n(k),u_1(k),\cdots,u_r(k),k] \\ & \vdots \\ y_p(k) & = g_p[x_1(k),\cdots,x_n(k),u_1(k),\cdots,u_r(k),k] \end{array}\right. \tag{1.18}$$

Consider the vector notations:

$$x(k)=\begin{bmatrix} x_1(k) \\ x_2(k) \\ \vdots \\ x_n(k) \end{bmatrix}; u(k)=\begin{bmatrix} u_1(k) \\ u_2(k) \\ \vdots \\ u_r(k) \end{bmatrix} \text{ and } y(k)=\begin{bmatrix} y_1(k) \\ y_2(k) \\ \vdots \\ y_p(k) \end{bmatrix},$$

and the functions f and g defined as follows:

$$f[x(k),u(k),k]=\begin{bmatrix} f_1[x(k),u(k),k] \\ f_2[x(k),u(k),k] \\ \vdots \\ f_n[x(k),u(k),k] \end{bmatrix}, g[x(k),u(k),k]=\begin{bmatrix} g_1[x(k),u(k),k] \\ g_2[x(k),u(k),k] \\ \vdots \\ g_p[x(k),u(k),k] \end{bmatrix}$$

Relations (1.17) and (1.18) become simply:

$$x(k+1)=f[x(k),u(k),k] \tag{1.19}$$

$$y(k)=g[x(k),u(k),k] \tag{1.20}$$

The vector $u(k)$ is the input of the system and $y(k)$ is its output. Assume now that $x(k_0)$ and $u(k), k \geqslant k_0$, are known. Equation (1.19) can be solved iteratively for $k \geqslant k_0$. Equation (1.20) allows the calculation of $y(k)$ for $k \geqslant k_0$. Thus, as suggested by the notation, $x(k_0)$ is the state

at time k_0.

The equation (1.19) governing the dynamic behavior of the system is called the **state equation**; relation (1.20) giving its output is the **output equation**. The **state-space model** is called discrete-time or internal, the goals of this terminology being the same as those mentioned in paragraph 1.6.1.

1.8.1 State-space representation of linear discrete-time systems

A particular case of the discrete-time state-space model is the one in which the functions f and g can be written in the following form:

$$f[x(k),u(k),k] = \Phi(k)x(k) + \Gamma(k)u(k) \tag{1.21}$$

$$g[x(k),u(k),k] = C(k)x(k) + D(k)u(k) \tag{1.22}$$

$\Phi(k), \Gamma(k), C(k),$ and $D(k)$ are respectively matrices of dimensions $n \times n$, $n \times r$, $p \times n$ and $p \times r$. Then, the state-space representation becomes:

$$x(k+1) = \Phi(k)x(k) + \Gamma(k)u(k) \tag{1.23}$$

$$y(k) = C(k)x(k) + D(k)u(k) \tag{1.24}$$

The terminology associated to the discrete-time case is identical to that adopted in paragraph 1.6.2 for the continuous-time systems; the matrices appearing in equations (1.21) and (1.22) play the same role as those in relations (1.8) and (1.9) (The reason of the notation $\Phi(k)$ for the system matrix and $\Gamma(k)$ for the input matrix will become obvious later).

Figure 1.23 provides the block diagram of such a system; q^{-1} being the delay operator.

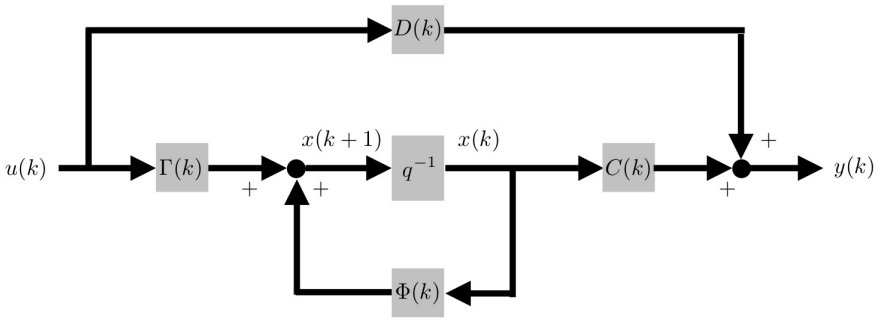


Figure 1.23 – Block diagram of a discrete-time linear system.

For monovariable systems, characterized by $r = p = 1$, the matrix $\Phi(k)$ becomes a vector $g(k)$,

the matrix $C(k)$ becomes a line vector $c^T(k)$ and finally the matrix $D(k)$ becomes a scalar $d(k)$:

$$x(k+1) = \Phi(k)x(k) + g(k)u(k) \tag{1.25}$$
$$y(k) = c^T(k)x(k) + d(k)u(k) \tag{1.26}$$

1.8.2 Example: Time delay of three sampling periods

Consider as a system a continuous time delay $y(t) = u(t-\tau)$. The delay τ is expressed as a multiple of the sampling period, in this case of three, thus $y(t) = u(t-3h)$.

The sampling of the output of this system by means of an AD converter at discrete time $t_k = kh$ provides the expression $y(kh) = u(kh-3h)$.

To simplify the notation, the sampling period h is omitted, leading to the relation $y(k) = u(k-3)$.

A positive shift of three periods gives finally the difference equation $y(k+3) = u(k)$.

Three variables have to be introduced to write this system in the form of a state-space representation, it is $x_1(k) = y(k), x_2(k) = y(k+1)$ and $x_3(k) = y(k+2)$.

Then, the state equation can be written as:

$$\left|\begin{array}{lll} x_1(k+1) & = y(k+1) & = x_2(k) \\ x_2(k+1) & = y(k+2) & = x_3(k) \\ x_3(k+1) & = y(k+3) & = u(k) \end{array}\right.$$

The output equation is simply $y(k) = x_1(k)$.

In a matrix form, this model which is linear and time-invariant can be written as follows:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u(k)$$

$$y(k) = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \end{bmatrix}$$

1.9 Exercises

1.9.1 Flexible arm

Problem

An arm secured to a turning platform can rotate around an axis A. Its displacements are limited and damped with two springs.

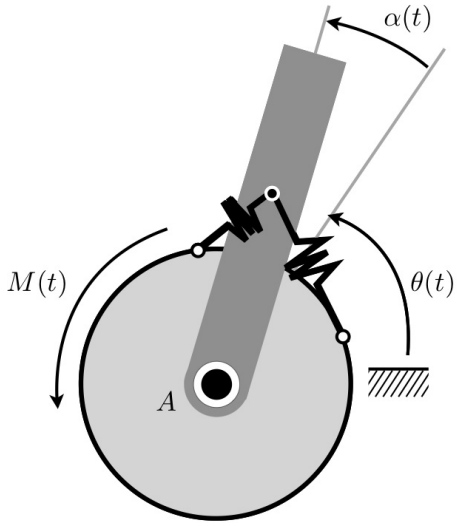


Figure 1.24 – Flexible arm

The dynamics of the system are governed in good approximation by the following differential equations:

$$\left|\begin{array}{rl} (I+I_b)\dot{\omega}(t)+I_b\ddot{\alpha}(t) & = M(t) \\ I_b\ddot{\alpha}(t)+I_b\dot{\omega}(t)+f\dot{\alpha}(t)+R\alpha(t) & = 0 \end{array}\right.$$

I and I_b are respectively the inertia of the platform and the arm. R and f are physical parameters.
$M(t)$ is the motor torque (input), $\theta(t)$ is the angular position of the platform with respect to a fixed frame of reference and $\alpha(t)$ is the relative angular position of the arm with respect to the platform.

1. Determine the state-space model that describes the dynamics of this system for the case in which the measured output is the angular position $\theta(t)$.

2. Determine the state-space model that describes the dynamics of this system for the case in which the measured output is the angular speed $\omega(t) = \dot{\theta}(t)$.

Solution

1. Determine the state-space model that describes the dynamics of this system for the case in which the measured output is the angular position $\theta(t)$.

$$\left|\begin{array}{ll} x_1(t) & = \theta(t) \\ x_2(t) & = \dot{\theta}(t) \\ x_3(t) & = \alpha(t) \\ x_4(t) & = \dot{\alpha}(t) \end{array}\right.$$

Hence,

$$\left|\begin{array}{lll} \dot{x}_1(t) & = x_2(t) & \\ \dot{x}_2(t) & = \ddot{\theta}(t) & = \frac{R}{I} x_3(t) + \frac{f}{I} x_4(t) + \frac{1}{I} u(t) \\ \dot{x}_3(t) & = x_4(t) & \\ \dot{x}_4(t) & = \ddot{\alpha}(t) & = -\frac{I+I_b}{I_b} \left[\frac{R}{I} x_3(t) + \frac{f}{I} x_4(t) \right] - \frac{1}{I} u(t) \end{array}\right.$$

$$y(t) = \theta(t) = x_1(t)$$

2. Determine the state-space model that describes the dynamics of this system for the case in which the measured output is the angular speed $\omega(t) = \dot{\theta}(t)$.

In this case, $\theta(t)$ should not be introduced. Part of the dynamics is downstream from the sensor, it is then not visible.

$$\left|\begin{array}{ll} x_1(t) & = \omega(t) \\ x_2(t) & = \alpha(t) \\ x_3(t) & = \dot{\alpha}(t) \end{array}\right.$$

Hence,

$$\left|\begin{array}{lll} \dot{x}_1(t) & = \dot{\omega}(t) & = \frac{R}{I} x_2(t) + \frac{f}{I} x_3(t) + \frac{1}{I} u(t) \\ \dot{x}_2(t) & = x_3(t) & \\ \dot{x}_3(t) & = \ddot{\alpha}(t) & = -\frac{I+I_b}{I_b} \left[\frac{R}{I} x_2(t) + \frac{f}{I} x_3(t) \right] - \frac{1}{I} u(t) \end{array}\right.$$

$$y(t) = \omega(t) = x_1(t)$$

1.9.2 Euler approximation

Problem

Use the Euler approximation introduced in 1.7.1 to write the discrete-time state-space model of the electrical drive described in paragraph 1.6.3 with $L = 0$.

Solution

Considering the continuous-time model

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & a \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ b \end{bmatrix} u(t)$$
$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$$

The discrete-time model based on the Euler approximation is

$$\begin{bmatrix} x_1(k+1) - x_1(k) \\ x_2(k+1) - x_2(k) \end{bmatrix} = h \begin{bmatrix} 0 & 1 \\ 0 & a \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + h \begin{bmatrix} 0 \\ b \end{bmatrix} u(k)$$
$$y(k) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix}$$

or, written in the standard form

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 1 & h \\ 0 & 1+ah \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ bh \end{bmatrix} u(k)$$
$$y(k) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix}$$

1.9.3 Cascade systems

Problem

Consider two dynamic systems described respectively by the discrete-time linear state-space models M_a and M_b:

$$\begin{aligned} M_a: \quad x_a(k+1) &= \Phi_a x_a(k) + \Gamma_a u_a(k) \\ y_a(k) &= C_a x_a(k) + D_a u_a(k) \end{aligned}$$

$$\begin{aligned} M_b: \quad x_b(k+1) &= \Phi_b x_b(k) + \Gamma_b u_b k \\ y_b(k) &= C_b x_b(k) + D_b u_b(k) \end{aligned}$$

1. Determine the state-space model that corresponds to the cascading of these two systems:

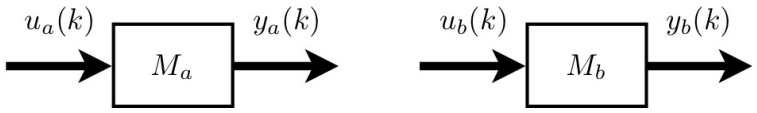


Figure 1.25 – Sub-systems

2. In which condition(s) can this cascade be realized?

Solution

$$x(k+1) = \begin{bmatrix} x_a(k+1) \\ x_b(k+1) \end{bmatrix} = \begin{bmatrix} \Phi_a & 0 \\ \Gamma_b C_a & \Phi_b \end{bmatrix} \begin{bmatrix} x_a(k) \\ x_b(k) \end{bmatrix} + \begin{bmatrix} \Gamma_a \\ \Gamma_b D_a \end{bmatrix} u_a(k)$$

$$y_b(k) = \begin{bmatrix} D_b C_a & C_b \end{bmatrix} \begin{bmatrix} x_a(k) \\ x_b(k) \end{bmatrix} + D_b D_a u_a(k)$$

To be able to cascade these two systems, the number of outputs of the first system should be the same than the number of inputs of the second one.

2 Discretization

2.1 Solution of the linear and time-invariant state equation

In the presence of a single state variable, the solution of the linear differential homogeneous equation, with constant coefficients:

$$\dot{x}(t) = ax(t)$$

is solved in the following traditional way:

$$\begin{aligned} \frac{dx}{dt} &= ax \\ \frac{dx}{x} &= a\,dt \\ \ln(x) &= at + c_1 \\ x &= c_2 e^{at} \end{aligned}$$

where c_1 and c_2 are constants. The latter value can be determined by setting as an initial condition $x = x_0$ at time $t = t_0$. It follows:

$$x = e^{a(t-t_0)} x_0$$

If necessary, the exponential function can be evaluated numerically by means of the series:

$$e^{\tau} = 1 + \tau + \frac{\tau^2}{2!} + \frac{\tau^3}{3!} + \cdots$$

To obtain the solution of the state-space equation (1.8), the latter is first solved without external excitation, with only nonzero initial conditions. It is the **homogeneous equation**:

$$\dot{x}(t) = Ax(t) \text{ with: } x(t_0) = x_0 \tag{2.1}$$

To do so, its solution is assumed sufficiently smooth so that a series expansion can be considered:

$$x(t) = A_0 + A_1(t-t_0) + A_2(t-t_0)^2 + \cdots \tag{2.2}$$

By setting $t = t_0$, we immediately find that $A_0 = x_0$. If we take the time derivative of (2.2) and substitute the expression obtained in (2.1), we get:

$$A_1 + 2A_2(t-t_0) + 3A_3(t-t_0)^2 + \cdots = Ax(t)$$

and, at $t = t_0 : A_1 = Ax_0$. By continuing to take successive time derivatives of the series and of the homogeneous equation evaluated in $t = t_0$, the following expression is obtained:

$$x(t) = \left[I + A(t-t_0) + \frac{A^2(t-t_0)^2}{2} + \frac{A^3(t-t_0)^3}{6} + \cdots \right] x_0$$

By analogy to the scalar case, the term that appears between the brackets is defined as the **matrix exponential** and is noted:

$$\begin{aligned} e^{A(t-t_0)} &= I + A(t-t_0) + \tfrac{A^2(t-t_0)^2}{2} + \tfrac{A^3(t-t_0)^3}{6} + \cdots \\ &= \textstyle\sum_{k=0}^{\infty} A^k \frac{(t-t_0)^k}{k!} \end{aligned}$$

We can show that the solution of the homogeneous equation is unique, which leads to **properties** that are very interesting for the matrix exponential. For example, considering two instants $t : t_1$ and t_2, we have:

$$x(t_1) = e^{A(t_1-t_0)}x(t_0) \text{ and } x(t_2) = e^{A(t_2-t_0)}x(t_0)$$

As the choice of t_0 is arbitrary, $x(t_2)$ can also be expressed as follows:

$$x(t_2) = e^{A(t_2-t_1)}x(t_1)$$

which gives by substitution of $x(t_1)$:

$$x(t_2) = e^{A(t_2-t_1)}e^{A(t_1-t_0)}x(t_0)$$

Now, we have two different expressions for $x(t_2)$. As the solution is unique, they must be the same. Thus, we can conclude that:

$$e^{A(t_2-t_0)} = e^{A(t_2-t_1)}e^{A(t_1-t_0)} \tag{2.3}$$

for all t_2, t_1 and t_0. In particular, if $t_2 = t_0$, then:

$$I = e^{-A(t_1-t_0)}e^{A(t_1-t_0)}$$

Thus, the inverse of e^{At} is obtained simply by changing the sign of the exponent.

The **particular solution** of the state-space equation (1.8) when the control input u is nonzero is obtained by variation of parameters. We assume that the solution has the form:

$$x(t) = e^{A(t-t_0)}v(t) \tag{2.4}$$

where $v(t)$ is a vector of variable parameters to be determined, as opposed to the constant parameters $x(t_0)$. By substituting the expression (2.4) in the state-space equation:

$$\dot{x}(t) = Ax(t) + Bu(t)$$

we get:

$$Ae^{A(t-t_0)}v(t) + e^{A(t-t_0)}\dot{v}(t) = Ae^{A(t-t_0)}v(t) + Bu(t)$$

After simplification and as the inverse of the matrix exponential is obtained by changing the sign of its exponent:

$$\dot{v}(t) = e^{-A(t-t_0)}Bu(t)$$

Assuming that the control signal is zero for all $t < t_0$, $\dot{v}$ can be integrated from t_0 to t to get:

$$v(t) = \int_{t_0}^{t} e^{-A(\tau-t_0)}Bu(\tau)d\tau$$

Consequently, the particular solution exhibits the form:

$$x(t) = e^{A(t-t_0)}\int_{t_0}^{t} e^{-A(\tau-t_0)}Bu(\tau)d\tau$$

This expression can be simplified by means of equation (2.3) to set the particular solution:

$$x(t) = \int_{t_0}^{t} e^{A(t-\tau)}Bu(\tau)d\tau$$

which is a convolution product.

The system being linear and following the principle of superposition, the **complete solution** of the state-space equation is the sum of its homogeneous solution and its particular one.

$$x(t) = e^{A(t-t_0)}x(t_0) + \int_{t_0}^{t} e^{A(t-\tau)}Bu(\tau)d\tau \tag{2.5}$$

2.2 Discretization

If the continuous-time system is controlled by a computer through AD and DA converters with a sampling period h, equation (2.5) can be written for $t = kh + h$ (AD) and $t_0 = kh$:

$$x(kh+h) = e^{Ah}x(kh) + \int_{kh}^{kh+h} e^{A(kh+h-\tau)}Bu(\tau)d\tau$$

DA converters (zero-order hold) maintain the control signal constant during a sampling period. Consequently:

$$u(\tau) = u(kh), kh \leqslant \tau < kh + h$$

By setting $\eta = kh + h - \tau$, the difference equation that is obtained becomes:

$$x(kh+h) = e^{Ah}x(kh) + \int_{0}^{h} e^{A\eta}d\eta Bu(kh)$$

If we define: $\Phi = e^{Ah}$ and $\Gamma = \int_0^h e^{A\eta}d\eta B$

then the discrete-time state-space model that corresponds to a continuous-time system controlled by a computer has the form:

$$x(kh+h) = \Phi x(kh) + \Gamma u(kh)$$
$$y(kh) = Cx(kh) + Du(kh)$$

The series defining the matrix exponential can be integrated term by term, thus providing:

$$\Gamma = \left[Ih + \frac{Ah^2}{2!} + \frac{A^2h^3}{3!} + \cdots + \frac{A^ih^{i+1}}{(i+1)!} + \cdots \right] B$$

From a computation time point of view, it is very beneficial to evaluate the two matrices Φ and Γ from a single series. To do so, the following series is introduced:

$$\psi = I + \frac{Ah}{2!} + \frac{A^2h^2}{3!} + \cdots + \frac{A^ih^i}{(i+1)!} + \cdots$$

It is easy to verify that:

$$\Phi = I + Ah\psi \text{ and } \Gamma = \psi hB$$

For practical implementation, the series ψ is evaluated in the following way:

$$\psi \widetilde{=} I + \frac{Ah}{2}\left(I + \frac{Ah}{3}\left(\cdots \frac{Ah}{N-1}\left(I + \frac{Ah}{N}\right)\right)\right)$$

N has to be chosen sufficiently large so that the contribution of new terms of the series is negligible.

The solution of the continuous-time linear state-space equation is necessary to obtain the discrete-time state-space model of a time invariant system controlled by computer.

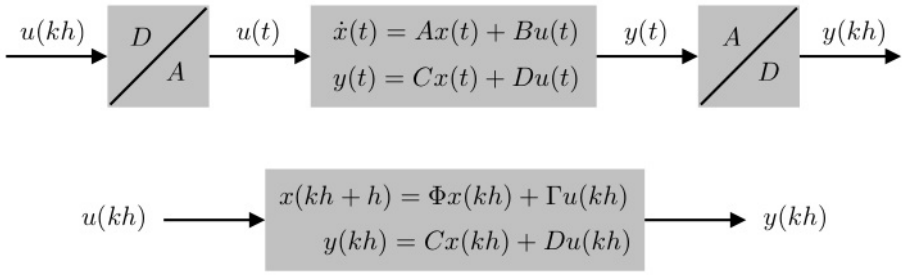
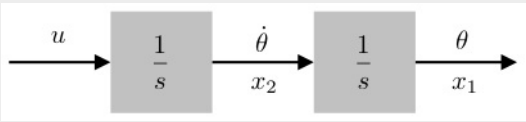


Figure 2.1 – Continuous system with AD and DA converters and its equivalent discrete representation.

with $\Phi = I + Ah\psi, \Gamma = \psi hB = A^{-1}(\Phi - I)B$ and $\psi = \sum_{i=0}^{\infty} \frac{A^i h^i}{(i+1)!}$.

2.3 Example: Double integrator

Let consider the following system:



Block diagram of a double integrator

Its state-space model is:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u = Ax + Bu$$

$$\theta = y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = Cx$$

Its transfer function is:

$$G(s) = \frac{1}{s^2}$$

When this system is observed through AD and DA converters, its model takes the form:

$$\left| \begin{array}{ll} x(k+1) & = \Phi x(k) + \Gamma u(k) \\ y(k) & = Cx(k) \end{array} \right.$$

with: $\Phi = e^{Ah} = I + Ah + \frac{A^2h^2}{2} + \frac{A^3h^3}{6} + \cdots,$

and: $\Gamma = \left[Ih + A\frac{h^2}{2} + A^2\frac{h^3}{6} + \cdots \right] B,$

being numerically: $\Phi = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} h + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \frac{h^2}{2} = \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix},$

and: $\Gamma = \int_0^h e^{A\eta} d\eta B = \int_0^h \begin{bmatrix} 1 & \eta \\ 0 & 1 \end{bmatrix} d\eta \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} h^2/2 \\ h \end{bmatrix}$

2.4 Analytic derivation of the matrix exponential

For low order systems, the exact limit of the series which corresponds to the matrix exponential can be obtained. To do so, the Laplace transform of the state-space equation is taken:

$$sX(s) - x(0) = AX(s) + BU(s)$$

From which:

$$X(s) = (sI - A)^{-1}x(0) + (sI - A)^{-1}BU(s)$$

By comparing this relation with the complete solution obtained for $t_0 = 0$, we get:

$$e^{At} = \mathscr{L}^{-1}[(sI - A)^{-1}]$$

Software packages like Mathematica or Maple can be used to analytically compute the n^2 necessary Laplace transforms.

We will see later that the matrix exponential of a system of order n can be obtained exactly by evaluating a matrix polynomial.

2.5 Leverrier algorithm

To avoid solving analytically the inverse matrix $(sI - A)$, the Leverrier algorithm can be exploited. It relies on the definition of the inverse matrix

$$(sI - A)^{-1} = \frac{\text{adj}(sI - A)}{\det(sI - A)} \tag{2.6}$$

If A_{ij} is the matrix of dimension $(n-1) \times (n-1)$ (minor) obtained by deleting row i and column j of the matrix A, the element standing at row i and column j of the matrix $adj(A)$ (adjoint) is:

$$(-1)^{i+j} det(A_{ji})$$

The determinant is written:

$$\det(sI - A) = s^n + a_1 s^{n-1} + a_2 s^{n-2} + \cdots + a_n$$

It is the characteristic polynomial of A. The values of s that cancel this characteristic polynomial are the eigenvalues of A.

The adjoint matrix can be written:

$$\text{adj}(sI - A) = H_0 s^{n-1} + H_1 s^{n-2} + H_2 s^{n-3} + \cdots + H_{n-1}$$

Matrices H_i are of dimension $n \times n$. They are computed sequentially starting from $H_0 = I$:

$$\left|\begin{array}{ll}
a_1 = -tr(AH_0) & H_1 = AH_0 + a_1 I \\
a_2 = -\frac{1}{2} tr(AH_1) & H_2 = AH_1 + a_2 I \\
a_3 = -\frac{1}{3} tr(AH_2) & H_3 = AH_2 + a_3 I \\
\vdots & \\
a_{n-1} = -\frac{1}{n-1} tr(AH_{n-2}) & H_{n-1} = AH_{n-2} + a_{n-1} I \\
a_n = -\frac{1}{n} tr(AH_{n-1}) & H_n = AH_{n-1} + a_n I = 0
\end{array}\right.$$

The trace (tr) of a matrix of dimension $n \times n$ is the sum of its diagonal elements.

2.6 Cayley-Hamilton theorem

Consider $f(A)$ any matrix function of a square matrix A of dimension n. There exists a polynomial p of degree lower than n such that:

$$f(A) = p(A) = \alpha_0 A^{n-1} + \alpha_1 A^{n-2} + \cdots + \alpha_{n-1} I$$

$$with: \quad f(\lambda_i) = p(\lambda_i) \qquad i = 1, \cdots, n$$

If the eigenvalues λ_i are distinct, the previous relation is sufficient to get the α_i. If the eigenvalues λ_i are multiple, of multiplicity m_i, supplementary conditions must be used to get the unknown coefficients:

$$\left| \begin{array}{rcl} f^{(1)}(\lambda_i) & = & p^{(1)}(\lambda_i) \\ & \vdots & \\ f^{(m_i-1)}(\lambda_i) & = & p^{(m_i-1)}(\lambda_i) \end{array} \right.$$

The exponents between brackets indicate the order of derivation with respect to λ_i. The eigenvalues are the roots of the characteristic equation of the matrix A, $\det(A - \lambda I) = 0$.

2.7 Example: Double integrator

Consider: $A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$

Eigenvalues of Ah:

$$det(Ah - \lambda I) = det \begin{bmatrix} -\lambda & h \\ 0 & -\lambda \end{bmatrix} = 0 \Rightarrow \lambda^2 = 0 \Rightarrow \lambda_1 = \lambda_2 = 0 \equiv \lambda$$

Calculation of e^{Ah} using the Cayley-Hamilton theorem:

$$e^{Ah} = \alpha_0 A + \alpha_1 I \tag{2.7}$$

$$e^{\lambda} = \alpha_0 \lambda + \alpha_1 \tag{2.8}$$

$$\frac{d}{d\lambda} e^{\lambda} = \frac{d}{d\lambda}(\alpha_0 \lambda + \alpha_1) \rightarrow e^{\lambda} = \alpha_0 \tag{2.9}$$

Hence for $\lambda = 0$ $\quad \alpha_1 = 1$

$\alpha_0 = 1$

and finally $\quad e^{Ah} = A + I$

$$e^{Ah} = \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix}$$

2.8 Example: Electrical drive

Consider an application where only the angular position $\theta(t)$ of the electrical drive is measured. Then, this position is the unique available output. The input u is the supply voltage. The state variables are $x_1(t) = \theta(t)$ and $x_2(t) = \omega(t)$. The state-space model was derived in paragraph 1.6.3. With the output chosen, it takes the form:

$$\begin{aligned} \dot{x}(t) &= \underbrace{\begin{bmatrix} 0 & 1 \\ 0 & a \end{bmatrix}}_{A} x(t) + \underbrace{\begin{bmatrix} 0 \\ b \end{bmatrix}}_{B} u(t) \\ y(t) &= \underbrace{\begin{bmatrix} 1 & 0 \end{bmatrix}}_{C} x(t) \end{aligned}$$

with:

$$a = -\frac{1}{J}\left(\frac{k^2}{R} + f\right) = -5[s^{-1}]$$
$$b = \frac{k}{JR} = 1\left[\frac{\text{rad}}{Vs^2}\right]$$

The state-space model of the drive observed through AD and DA converters is now evaluated for a sampling period $h = 25$ ms. To do so, the exponential of the matrix A must be first calculated.

$$e^{At} = \mathscr{L}^{-1}[(sI - A)^{-1}]$$
$$(sI - A) = \begin{bmatrix} s & -1 \\ 0 & s-a \end{bmatrix}$$
$$(sI - A)^{-1} = \begin{bmatrix} \frac{1}{s} & \frac{1}{s(s-a)} \\ 0 & \frac{1}{s-a} \end{bmatrix} = \frac{1}{s(s-a)} \begin{bmatrix} s-a & 1 \\ 0 & s \end{bmatrix}$$

with: $\frac{1}{s(s-a)} = \frac{1}{a}\left[\frac{1}{s-a} - \frac{1}{s}\right]$

$$\mathscr{L}^{-1}[(sI - A)^{-1}] = \begin{bmatrix} 1 & -\frac{1}{a} + \frac{1}{a}e^{at} \\ 0 & e^{at} \end{bmatrix}$$

The matrices of the discretized state-space equation are:

$$\Phi = e^{Ah} = \begin{bmatrix} 1 & \frac{1}{a}(e^{ah}-1) \\ 0 & e^{ah} \end{bmatrix} = \begin{bmatrix} 1 & 0.0235 \\ 0 & 0.8825 \end{bmatrix}$$

$$\Gamma = \int_0^h e^{A\eta} d\eta B = \begin{bmatrix} \int_0^h d\eta & \int_0^h \left(-\frac{1}{a} + \frac{1}{a} e^{a\eta}\right) d\eta \\ 0 & \int_0^h e^{a\eta} d\eta \end{bmatrix} B$$

$$\Gamma = \begin{bmatrix} h & -\frac{h}{a} + \frac{1}{a^2}(e^{ah}-1) \\ 0 & \frac{1}{a}(e^{ah}-1) \end{bmatrix} \begin{bmatrix} 0 \\ b \end{bmatrix} = \begin{bmatrix} -\frac{b}{a}\left(h - \frac{1}{a}e^{ah} + \frac{1}{a}\right) \\ \frac{b}{a}(e^{ah}-1) \end{bmatrix} = \begin{bmatrix} 0.0003 \\ 0.0235 \end{bmatrix}$$

Finally:

$$\left| \begin{array}{rcl} x(k+1) & = & \begin{bmatrix} 1 & 0.0235 \\ 0 & 0.8825 \end{bmatrix} x(k) + \begin{bmatrix} 0.0003 \\ 0.0235 \end{bmatrix} u(k) \\ \\ y(k) & = & \begin{bmatrix} 1 & 0 \end{bmatrix} x(k) \end{array} \right.$$

2.9 Exercises

2.9.1 Discrete PI controller

Problem

For an implementation on a specialised processor, a discrete PI controller has to be expressed in the general form of a state-space model. The input of the controller is the error $e(k)$ existing between the reference variable and the measurement.

The output of the controller corresponds to the control signal $c(k)$. It is the sum of the proportional and integral (u_i) terms:

$$\begin{aligned} & c(k) && = K_p e(k) + u_i(k) \\ with: \quad & u_i(k) && = u_i(k-1) + K_p \frac{h}{T_i} \left[\frac{e(k)+e(k-1)}{2} \right] \end{aligned}$$

The parameters h, K_p and T_i are respectively the sampling period, the proportional gain and the time constant of the integrator.

1. Determine the state-space model that corresponds to this dynamic system.

Solution

$$\begin{aligned} & y(k) = K_p u(k) + u_i(k) \\ & u_i(k+1) = u_i(k) + K_p \frac{h}{T_i} \left[\frac{u(k+1)+u(k)}{2} \right] \end{aligned}$$

To cancel the unwanted term $u(k+1)$, we chose as a state variable:

$$x(k) = u_i(k) + \alpha u(k) \text{(instead of } x(k) = u_i(k))$$

Thus:

$$\begin{aligned} & x(k+1) - \alpha u(k+1) = x(k) - \alpha u(k) + K_p \frac{h}{T_i} \left[\frac{u(k+1)+u(k)}{2} \right] \\ & x(k+1) = x(k) + \left[K_p \frac{h}{2T_i} - \alpha \right] u(k) + \left[K_p \frac{h}{2T_i} + \alpha \right] u(k+1) \end{aligned}$$

The unwanted term disappears for: $\alpha = -K_p \frac{h}{2T_i}$

Finally:

$$\begin{aligned} x(k+1) &= x(k) + K_p \tfrac{h}{T_i} u(k) \\ x(k) &= u_i(k) - K_p \tfrac{h}{2T_i} u(k) \\ y(k) &= K_p u(k) + u_i(k) = K_p u(k) + x(k) + K_p \tfrac{h}{2T_i} u(k) \\ &= x(k) + K_p \left[1 + \tfrac{h}{2T_i}\right] u(k) \end{aligned}$$

2.9.2 Discretization of the electrical drive with inductance

Problem

Consider the DC motor whose dynamics is described by the differential equations:

$$\left| \begin{aligned} u(t) &= k\omega(t) + Ri(t) + L\tfrac{d}{dt} i(t) \\ J\tfrac{d}{dt}\omega(t) &= ki(t) - f\omega(t) \end{aligned} \right.$$

For the selected application, the inductance L is taken into account. The input of the system is the supply voltage $u(t)$. The quantities $i(t)$ and $\omega(t)$ are respectively the armature current and the angular speed. J, R, k and f are the physical parameters.

1. Determine the state-space representation of this system seen through AD & DA converters. Only the angular speed $\omega(t)$ is measured. Compare an approximative discretization by Euler with an exact discretization (use for this the Cayley-Hamilton theorem).

Solution

$$\text{State}: \left| \begin{aligned} x_1 &= \omega \\ x_2 &= i \end{aligned} \right.$$

$$\text{Output}: \quad y = \omega$$

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -f/J & k/J \\ -k/L & -R/L \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1/L \end{bmatrix} u \text{ and}$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + [0]\, u$$

Discretization by Euler:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 1-hf/J & hk/J \\ -hk/L & 1-hR/L \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0 \\ h/L \end{bmatrix} u(k)$$

For $h = 1$ ms, $L = 0.01$, $J = 0.001$,
$k = 0.1$, $f = 0.01$ and $R = 1$:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 0.99 & 0.1 \\ -0.01 & 0.9 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0.1 \end{bmatrix} u(k)$$

3 Linearization

3.1 Nominal trajectories

The interest of linear systems lies in the possibility to apply the principle of superposition and to solve analytically differential equations that govern their dynamics. However, the majority of the systems found in practice show non-linearities leading to the general state-space representation defined previously. Significant difficulties arise unfortunately during the direct study of these systems, difficulties that can often be avoided by relying only on a linear approximation of the original model, which is much easier to analyze but valid only for small deviations around nominal trajectories. This approach called **linearization** is formalized by exploiting the concepts that where previously defined after the introduction of the necessary nomenclature.

3.1.1 Operating or nominal trajectory

A predefined profile describing the evolution of the states, the inputs and the outputs of a system is called **operating trajectory** or **nominal trajectory**. Such a trajectory is described by the upper-lined letters $\bar{x}(t)$ for the state, $\bar{u}(t)$ for the inputs and $\bar{y}(t)$ for the corresponding outputs. For example, the profile of position that has to be followed by the axis of a machine tool during production of a circular part is in the form:

$$\bar{x}(t) = A\sin(\omega t)$$

For the tool to follow such an operating trajectory, it must respect the constraints associated with its own dynamics. From a mathematical point of view, the trajectory must satisfy the differential and algebraic equations of its model. In other words and respectively in the continuous-time or discrete-time cases, the triplet $\{\bar{u}(t), \bar{x}(t), \bar{y}(t)\}$ or $\{\bar{u}(k), \bar{x}(k), \bar{y}(k)\}$ should be a solution to the system of equations:

For simplicity, we only consider here **time-invariant** systems whose functions f and g do not explicitly depend on time. The solution of the one or the other of these two systems of non-linear equations corresponds to an accessible state of the system.

Continuous case	Discrete case
$\dot{\bar{x}}(t) = f[\bar{x}(t), \bar{u}(t)]$	$\bar{x}(k+1) = f[\bar{x}(k), \bar{u}(k)]$
$\bar{y}(t) = g[\bar{x}(t), \bar{u}(t)]$	$\bar{y}(k) = g[\bar{x}(k), \bar{u}(k)]$

Table 3.1 – Specifications for an operating trajectory.

There are always $n+p$ equations to be solved, which corresponds to the number of unknowns which can be determined among the $n+p+r$ quantities that characterize the nominal trajectory. There are consequently r degrees of freedom to impose inputs, outputs, states or a combination of these quantities according to the considered application. Depending on the case, the solution can be obtained analytically or numerically with an algorithm like Newton-Raphson.

For the considered example, the control that has to be applied to the machine in order to follow the specified profile corresponds to the solution of the system of **algebraic and non-linear** equations:

$$\begin{aligned} \omega A\cos(\omega t) &= f[A\sin(\omega t), \bar{u}(t)] \\ \bar{y}(t) &= g[A\sin(\omega t), \bar{u}(t)] \end{aligned}$$

where a state derivative of the form $\dot{\bar{x}}(t) = \omega A\cos(\omega t)$ that was specified by the chosen profile has been taken into account.

3.1.2 Feedforward control

In the absence of modeling errors or disturbances on the system, the application in open loop of the inputs $\bar{u}$ generates the state $\bar{x}$ and the outputs $\bar{y}$. It is the control that is a priori necessary to follow the nominal trajectory. The vector $\bar{u}$ is called **feedforward control**, no matter if it is the continuous-time or the discrete-time case. This control can be pre-calculated or computed in real time and applied continuously to the input of the system.

3.1.3 Operating point or nominal state

If an operating trajectory does not evolve over time, it is called an **operating point, equilibrium point,** or **nominal state**. This condition is $\dot{\bar{x}}(t) = 0$ in the continuous-time case and $\bar{x}(k+1) = \bar{x}(k) \quad \forall k$ in the discrete-time case. By introducing the constant triplet $\{\bar{u}, \bar{x}, \bar{y}\}$, the operating point satisfies the following equations:

A system that is at its operating point is, by definition of the latter, in a time-invariant state. It is consequently at rest, that is to say that the effects of the initial conditions or past inputs have vanished.

Continuous case	Discrete case
$0 = f[\bar{x}, \bar{u}]$	$\bar{x} = f[\bar{x}, \bar{u}]$
$\bar{y} = g[\bar{x}, \bar{u}]$	$\bar{y} = g[\bar{x}, \bar{u}]$

Table 3.2 – Specifications for an operating point.

3.2 Small signal linearization

3.2.1 Linearization approach

To illustrate the concept of linearization, consider for **example** the case of a function f of a variable x represented in figure 3.1. This function is approximated by a straight line which passes through the operating point $\bar{x}$. This straight line is used to approximate the value of the function near $\bar{x}$, particularly at $x = \bar{x} + \tilde{x}$.

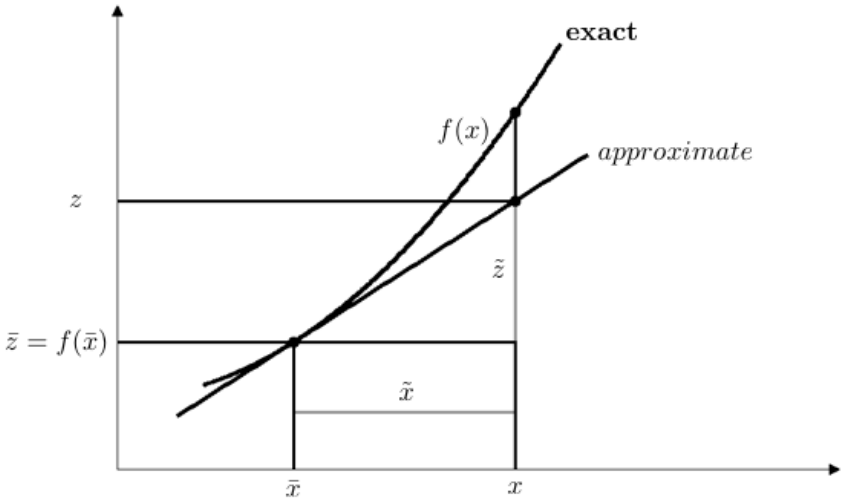


Figure 3.1 – Function of a single variable.

We have:

$$f(x) \approx z = \bar{z} + \tilde{z} = \bar{z} + \frac{d}{dx} f(\bar{x})\tilde{x}$$

If we only consider the relation between a deviation $\tilde{x}$ of x and the corresponding deviation $\tilde{z}$ of z, we get the following linear relation:

$$\tilde{z} = \frac{d}{dx} f(\bar{x})\tilde{x}$$

This relation is obviously only valid for small deviations $\tilde{x}$.

3.2.2 Linearization

The general case of the state and output equations is now discussed. The approach to set a linear representation remains the same as the one which was illustrated by the previous example. However, it relies on vector functions of several variables. Assuming that the partial derivatives:

$$\frac{\partial}{\partial x_j} f_i(x,u), \frac{\partial}{\partial u_k} f_i(x,u), \frac{\partial}{\partial x_j} g_q(x,u) \text{ and } \frac{\partial}{\partial u_k} g_q(x,u)$$

exist and are continuous for all x and u; $i, j = 1,2,\cdots,n$; $k = 1,2,\cdots,r$ and $q = 1,2,\cdots,p$; in other words, the functions f and g are continuously differentiable. $\tilde{x}$, $\tilde{u}$ and $\tilde{y}$ are deviations around the nominal trajectories $\bar{x}, \bar{u}$ and $\bar{y}$:

$$\tilde{x} = x - \bar{x} \tag{3.1}$$

$$\tilde{u} = u - \bar{u} \tag{3.2}$$

$$\tilde{y} = y - \bar{y} \tag{3.3}$$

We can write the Taylor series expansion of the function $f_i(x,u)$ around the nominal trajectories for $i = 1,2,\cdots,n$:

$$\begin{aligned} f_i(x,u) = \quad & f_i(\bar{x},\bar{u}) + \tfrac{\delta}{\delta x_1} f_i(\bar{x},\bar{u})\tilde{x}_1 + \cdots + \tfrac{\delta}{\delta x_n} f_i(\bar{x},\bar{u})\tilde{x}_n \\ & + \tfrac{\delta}{\delta u_1} f_i(\bar{x},\bar{u})\tilde{u}_1 + \cdots + \tfrac{\delta}{\delta u_r} f_i(\bar{x},\bar{u})\tilde{u}_r + p_i(\tilde{x},\tilde{u}) \end{aligned} \tag{3.4}$$

The functions p_i represent terms of higher order.

It is natural to define:

$$p(\tilde{x},\tilde{u}) = \begin{bmatrix} p_1(\tilde{x},\tilde{u}) \\ p_2(\tilde{x},\tilde{u}) \\ \vdots \\ p_n(\tilde{x},\tilde{u}) \end{bmatrix}$$

Introducing now the Jacobian matrices:

$$\frac{\partial}{\partial x} f(\bar{x},\bar{u}) = \begin{bmatrix} \frac{\partial}{\partial x_1} f_1(\bar{x},\bar{u}) & \frac{\partial}{\partial x_2} f_1(\bar{x},\bar{u}) & \cdots & \frac{\partial}{\partial x_n} f_1(\bar{x},\bar{u}) \\ \frac{\partial}{\partial x_1} f_2(\bar{x},\bar{u}) & \frac{\partial}{\partial x_2} f_2(\bar{x},\bar{u}) & \cdots & \frac{\partial}{\partial x_n} f_2(\bar{x},\bar{u}) \\ \cdots & \cdots & & \cdots \\ \frac{\partial}{\partial x_1} f_n(\bar{x},\bar{u}) & \frac{\partial}{\partial x_2} f_n(\bar{x},\bar{u}) & \cdots & \frac{\partial}{\partial x_n} f_n(\bar{x},\bar{u}) \end{bmatrix}$$

$$\frac{\partial}{\partial u} f(\bar{x},\bar{u}) = \begin{bmatrix} \frac{\partial}{\partial u_1} f_1(\bar{x},\bar{u}) & \frac{\partial}{\partial u_2} f_1(\bar{x},\bar{u}) & \cdots & \frac{\partial}{\partial u_r} f_1(\bar{x},\bar{u}) \\ \frac{\partial}{\partial u_1} f_2(\bar{x},\bar{u}) & \frac{\partial}{\partial u_2} f_2(\bar{x},\bar{u}) & \cdots & \frac{\partial}{\partial u_r} f_2(\bar{x},\bar{u}) \\ \cdots & \cdots & & \cdots \\ \frac{\partial}{\partial u_1} f_n(\bar{x},\bar{u}) & \frac{\partial}{\partial u_2} f_n(\bar{x},\bar{u}) & \cdots & \frac{\partial}{\partial u_r} f_n(\bar{x},\bar{u}) \end{bmatrix}$$

All relations (3.4) can be written in the extremely compact form:

$$f(x,u) = f(\bar{x},\bar{u}) + \frac{\partial}{\partial x} f(\bar{x},\bar{u})\tilde{x} + \frac{\partial}{\partial u} f(\bar{x},\bar{u})\tilde{u} + p(\tilde{x},\tilde{u}) \tag{3.5}$$

where the notation $\frac{\partial}{\partial x} f(\bar{x},\bar{u})$ means that first the partial derivative $\frac{\partial}{\partial x} f(x,u)$ are computed and then evaluated for $x = \bar{x}$ and $u = \bar{u}$. In the same way, $g(x,u)$ becomes:

$$g(x,u) = g(\bar{x},\bar{u}) + \frac{\partial}{\partial x} g(\bar{x},\bar{u})\tilde{x} + \frac{\partial}{\partial u} g(\bar{x},\bar{u})\tilde{u} + q(\tilde{x},\tilde{u}) \tag{3.6}$$

where the vector q which includes the higher order terms is defined in the following way:

$$q(\tilde{x},\tilde{u}) = \begin{bmatrix} q_1(\tilde{x},\tilde{u}) \\ q_2(\tilde{x},\tilde{u}) \\ \vdots \\ q_p(\tilde{x},\tilde{u}) \end{bmatrix}$$

and the Jacobian matrices by:

$$\frac{\partial}{\partial x}g(\bar{x},\bar{u}) = \begin{bmatrix} \frac{\partial}{\partial x_1}g_1(\bar{x},\bar{u}) & \frac{\partial}{\partial x_2}g_1(\bar{x},\bar{u}) & \cdots & \frac{\partial}{\partial x_n}g_1(\bar{x},\bar{u}) \\ \frac{\partial}{\partial x_1}g_2(\bar{x},\bar{u}) & \frac{\partial}{\partial x_2}g_2(\bar{x},\bar{u}) & \cdots & \frac{\partial}{\partial x_n}g_2(\bar{x},\bar{u}) \\ \cdots & \cdots & & \cdots \\ \frac{\partial}{\partial x_1}g_p(\bar{x},\bar{u}) & \frac{\partial}{\partial x_2}g_p(\bar{x},\bar{u}) & \cdots & \frac{\partial}{\partial x_n}g_p(\bar{x},\bar{u}) \end{bmatrix}$$

$$\frac{\partial}{\partial u}g(\bar{x},\bar{u}) = \begin{bmatrix} \frac{\partial}{\partial u_1}g_1(\bar{x},\bar{u}) & \frac{\partial}{\partial u_2}g_1(\bar{x},\bar{u}) & \cdots & \frac{\partial}{\partial u_r}g_1(\bar{x},\bar{u}) \\ \frac{\partial}{\partial u_1}g_2(\bar{x},\bar{u}) & \frac{\partial}{\partial u_2}g_2(\bar{x},\bar{u}) & \cdots & \frac{\partial}{\partial u_r}g_2(\bar{x},\bar{u}) \\ \cdots & \cdots & & \cdots \\ \frac{\partial}{\partial u_1}g_p(\bar{x},\bar{u}) & \frac{\partial}{\partial u_2}g_p(\bar{x},\bar{u}) & \cdots & \frac{\partial}{\partial u_r}g_p(\bar{x},\bar{u}) \end{bmatrix}$$

The **linearization** technique consists simply of neglecting the higher order terms $p(\tilde{x},\tilde{u})$ and $q(\tilde{x},\tilde{u})$. The approximation seems to be quite reasonable for small deviations $\tilde{x}$ and $\tilde{u}$ around the nominal trajectories The success of this approach remains however a surprise; a rigorous justification depends on the theory of Lyapunov, which is outside the context of this course.

The equality (3.1) provides:

$$\dot{x} = \dot{\bar{x}}(t) + \dot{\tilde{x}}(t) \tag{3.7}$$

By substituting (3.5), without the higher order terms, and also (3.7) in the original state-space equation (1.6), and by taking into account the fact that the nominal trajectories verify the latter, we get:

$$\dot{\tilde{x}}(t) = \frac{\partial}{\partial x}f[\bar{x}(t),\bar{u}(t)]\tilde{x}(t) + \frac{\partial}{\partial u}f[\bar{x}(t),\bar{u}(t)]\tilde{u}(t)$$

It is a linear state-space equation. It governs the deviations of the state or input vectors around known nominal trajectories.

The output equation (1.7) can undergo an identical procedure; the result is that the **linearized continuous-time state-space model** is defined by:

$$\dot{\tilde{x}}(t) = \frac{\partial}{\partial x}f[\bar{x}(t),\bar{u}(t)]\tilde{x}(t) + \frac{\partial}{\partial u}f[\bar{x}(t),\bar{u}(t)]\tilde{u}(t) \tag{3.8}$$

$$\tilde{y}(t) = \frac{\partial}{\partial x}g[\bar{x}(t),\bar{u}(t)]\tilde{x}(t) + \frac{\partial}{\partial u}g[\bar{x}(t),\bar{u}(t)]\tilde{u}(t) \tag{3.9}$$

The form of the functions f and g being the same in both the continuous and discrete cases,

the **linearized discrete-time state-space model** is defined by:

$$x(k+1) = \frac{\partial}{\partial x} f[\bar{x}(k), \bar{u}(k)]\tilde{x}(k) + \frac{\partial}{\partial u} f[\bar{x}(k), \bar{u}(k)]\tilde{u}(t)$$
$$\tilde{y}(k) = \frac{\partial}{\partial x} g[\bar{x}(k), \bar{u}(k)]\tilde{x}(k) + \frac{\partial}{\partial u} g[\bar{x}(k), \bar{u}(k)]\tilde{u}(k)$$

3.2.3 Intrinsically linear systems

The triplet $\{\bar{u}\,\bar{x}, \bar{y}\} = \{0,0,0\}$ is always an operating point for a system that is intrinsically linear. In this case, $\tilde{u} = u, \tilde{x} = x$ and $\tilde{y} = y$, leading to deviations representations or to identical real quantities (global variables) representations. The mentioned triplet is however not the only operating point; there can be other ones that are solutions of:

Continuous case	Discrete case
$0 = A\bar{x} + B\bar{u}$	$\bar{x} = \Phi\bar{x} + \Gamma\bar{u}$
$\bar{y} = C\bar{x} + D\bar{u}$	$\bar{y} = C\bar{x} + D\bar{u}$

Table 3.3 – Operating points for a linear system.

For example in a continuous case, if $\bar{u}$ is specified:

$$\bar{x} = -A^{-1}B\bar{u} \text{ and } \bar{y} = (D - CA^{-1}B)\bar{u}$$

3.2.4 Example: Magnetic suspension system

Consider a system consisting of a ball maintained in suspension in the vertical plane by a coil (electromagnet).

The electromagnetic force F exerted by the coil depends on the distance x that separates it from the ball and the electric current i that runs through it:

$$F(x,i) = \frac{1}{2}\frac{L}{(1+x)^2}i^2$$

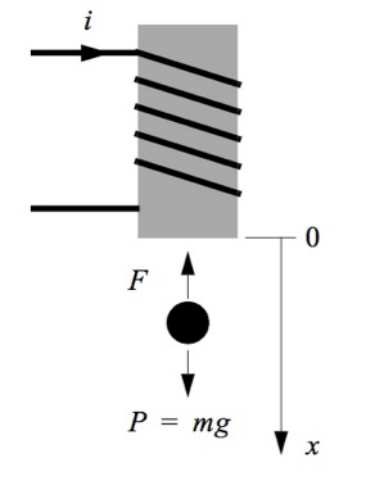


Figure 3.2 – Magnetic suspension system.

L is the difference between the inductance of the coil when the ball touches its core ($x = 0$) and the inductance without a ball.

The dynamic equation that governs the motion of the ball is obtained by applying the Newton's law:

$$m\ddot{x} = P - F(x,i)$$

P is the weight of the ball and m is its mass.

The differential equation that governs the dynamics of the magnetic suspension system is thus:

$$\ddot{x} = g - \frac{1}{2m}\frac{L}{(1+x)^2}i^2$$

By selecting respectively as state variables, input and output the quantities:

$$x_1 = x, x_2 = \dot{x}, u = i \text{ and } y = x$$

we obtain the following state-space model:

$$\dot{x}_1 = f_1(x_1, x_2, u) = x_2$$
$$\dot{x}_2 = f_2(x_1, x_2, u) = \ddot{x} = g - \frac{L}{2m(1+x_1)^2}u^2$$
$$y = g_1(x_1, x_2, u) = x_1$$

The linearized model has the form:

$$\dot{\tilde{x}} = \begin{bmatrix} 0 & 1 \\ \frac{L\bar{u}^2}{m(1+\bar{x}_1)^3} & 0 \end{bmatrix} \tilde{x} + \begin{bmatrix} 0 \\ -\frac{L\bar{u}}{m(1+\bar{x}_1)^2} \end{bmatrix} \tilde{u}$$
$$\tilde{y} = \begin{bmatrix} 1 & 0 \end{bmatrix} \tilde{x}$$

with:

$$\tilde{x} = x - \bar{x}$$
$$\tilde{u} = u - \bar{u}$$
$$\tilde{y} = y - \bar{y}$$

In this standardized state-space representation, the variable u does not represent the voltage across the coil, but the input quantity which is the current.

3.2.5 Summary

The state-space model is an appropriate way to represent the interactions that exist between the inputs and the outputs of a system. It involves internal quantities that hold all the information that is necessary to characterize the state of the system at a given time. The dynamic phenomena are described by the state-space equation (1.6) and the algebraic relations are expressed by the output equation (1.7).

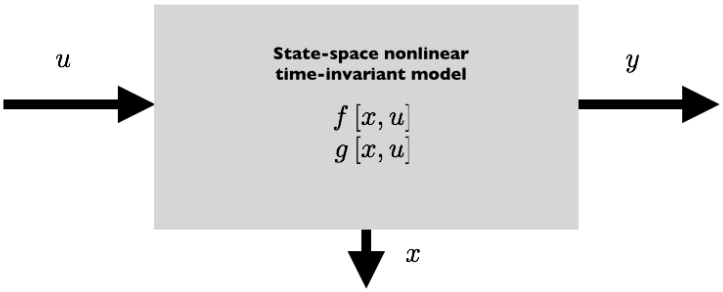


Figure 3.3 – Representation of a non-linear system by a state-space model.

To simplify the calculations during a simulation, facilitate the analysis of the dynamical behavior during the design or because the operating domain of an installation is limited, it is possible to represent the considered system by a linearized state-space model around nominal trajectories. This model behaves in the same way as the original non-linear state-space model only for **small deviations** around the specified nominal trajectories.

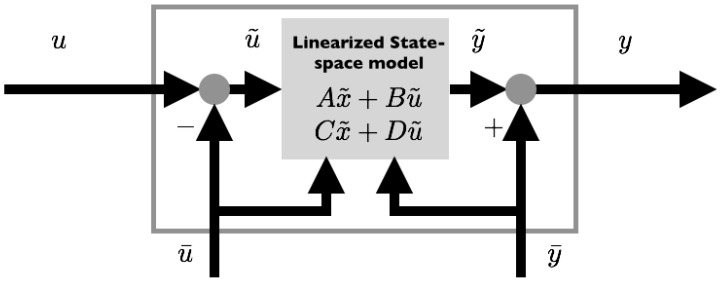


Figure 3.4 – Representation of a non-linear system by a linearized state-space model.

The simulation of the system represented in Figure 3.3 requires the use of a numerical integration algorithm, unlike the simulation of the system represented in Figure 3.4, where an analytical solution exists if the Jacobian matrices do not depend on time.

3.3 Feedback linearization

The feedback linearization is a technique that provides a global linearization. The cost of this approach is the necessity for the implementation to have access to the state of the system. Its limitation lies in the fact that not all the systems are linearizable in this way.

A rigorous study of this technique and the determination of the existence or non-existence of a solution require the introduction of advanced mathematical tools, such as Lie brackets, which are outside the context of this course. However, its principle can be easily introduced on the example of the magnetic suspension system which was introduced previously.

3.3.1 Example of the magnetic suspension system

By considering only the input-output variables, the dynamic model of the magnetic suspension system introduced in the example 3.2.4 is:

$$\ddot{y} = g - \frac{1}{2m}\frac{L}{(1+x_1)^2}u^2$$

In the context of feedback linearization, this input-output model (Fig. 3.5) of the system to be linearized is called the *direct dynamic model* and is represented by the letter M.

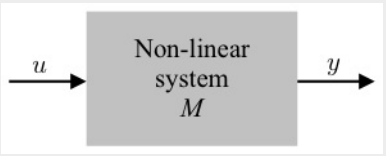


Figure 3.5 – System to be linearized.

A linearization element that uses the states x_1 and x_2 is now added upstream of the physical system. The **equivalent system** that results from the combination of this element and the physical system constitutes, from the point of view of the design of the controller, the new system to be considered (Fig. 3.6).

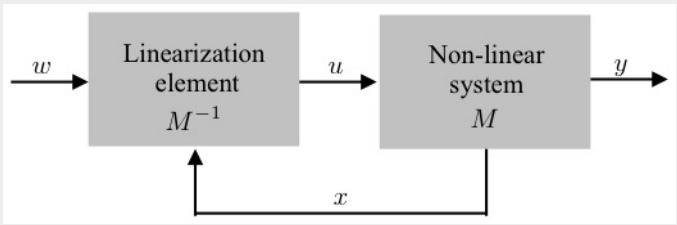


Figure 3.6 – Linearized system.

Figure 3.7 – Equivalent system.

The aim is to obtain pure integrators as an equivalent system (Fig. 3.7). To do so, w must be a time derivative of any order of y. The linearization element must exhibit a relation that links this derivative of the output with the input. In this sense, it contains the **inverse model** of the physical system noted M^{-1}. To construct this relation, the output equation of the physical system must be derived with respect of time until the input appears in the relation. This expression, which is of algebraic nature, is then inverted.

$$\begin{aligned} y &= x_1 \\ \dot{y} &= \dot{x}_1 = x_2 \\ \ddot{y} &= \dot{x}_2 = g - \tfrac{1}{2m}\tfrac{L}{(1+x_1)^2}u^2 \equiv w \end{aligned}$$

The inverted algebraic model M^{-1} that has to be implemented as a linearization element is:

$$u = \sqrt{\frac{2m(g-w)}{L}}(1+x_1) \qquad \text{(inverse static model } M^{-1}\text{)}$$

In this particular case, only the state x_1 has to be available to implement this linearization. As it is the position of the sphere and as this quantity is measured, it does not induce any implementation problem. The new system to consider for control design is a double integrator (Fig. 3.8).

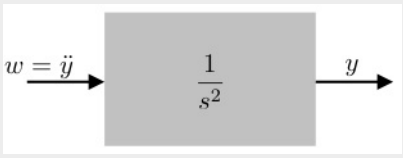


Figure 3.8 – System equivalent to the linearized suspension system.

The control input w has now the dimension of an acceleration rather than that of a current. It must be saturated so that the inverse model is always defined. In this case, $w < g$.

3.3.2 General formulation of the feedback linearization

In the MIMO case, the output equations generally have the form:

$$y_i = g_i(x)$$

To obtain the inverse model, this output equation must be derived with respect to time until one of the inputs appears in the relation:

$$y_i^{(d_i)} = g_i^*(x,u) \qquad i = 1,\cdots,p \text{ and } d_i = 0,\cdots$$

The derivative of minimum order $\bar{d}_i$ of y_i which exhibits an input is selected as the corresponding input w_i of the inverse model:

$$y_i^{(\bar{d}_i)} \equiv w_i$$

The equivalent system (inverse model and physical system) is a combination of p SISO integrators, which are decoupled and linear.

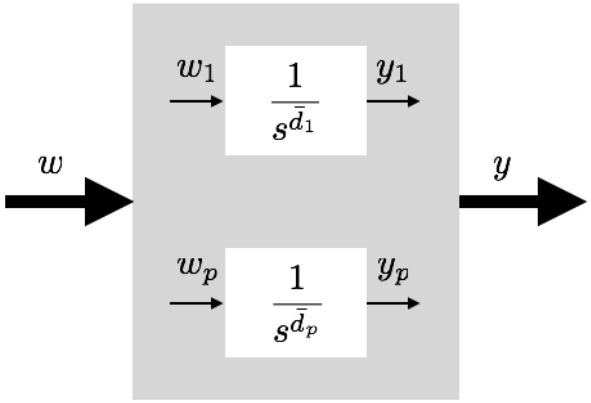


Figure 3.9 – Equivalent system.

3.3.3 Discrete implementation

For the design of a controller or an observer to be implemented on computer or microcontroller, it is the block diagram given in figure 3.10 that is usually considered.

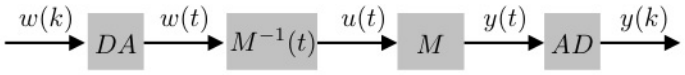


Figure 3.10 – Equivalent system seen through AD and DA converters.

In reality, the inverse model is not implemented continuously but numerically. However, the corresponding modification of the diagram does not induce any problems. As the in-

verse model is static, we can indeed move the D/A converters without loss of generality nor approximation (it means writing the inverse model with $t = kh$).

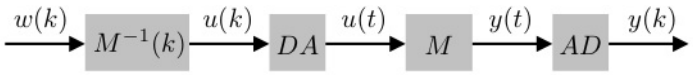


Figure 3.11 – Equivalent block diagram corresponding to the real implementation.

Hence, there is no need to modify the inverse model for a discrete implementation on a computer system.

3.4 Non-linear decoupling

The technique of **feedback linearization** introduced in section 3.3 provides simultaneously a global linearization and a dynamic decoupling. However, it only applies to a limited class of systems that possess the same number of inputs as outputs ($r = p$). The implementation of this technique matches a pure integrator of the order $\bar{d}_i$ to each input-output pair (u_i, y_i). A non-linear MIMO system transforms thus into p linear decoupled SISO systems whose respective controllers can be designed only by SISO techniques.

Non-linear decoupling requires the knowledge of the state of the system.

The modeling of the robot described in paragraph 1.4.3 is undertaken assuming that its mass is negligible with respect to the mass m of the load. Note that this robot moves in the vertical plane.

The dynamic model is obtained by the method of Lagrange with r and θ as generalized coordinates. For $r_0 = r_{maximum}$, the Lagrangian is the following:

$$L = \frac{m}{2}(\dot{r}^2 + r^2\dot{\theta}^2) - mg(r\sin\theta + r_0)$$

Thus:

$$\left|\begin{array}{ll} \frac{d}{dt}\frac{\delta L}{\delta \dot{r}} - \frac{\delta L}{\delta r} & = F \\ \frac{d}{dt}\frac{\delta L}{\delta \dot{\theta}} - \frac{\delta L}{\delta \theta} & = T \end{array}\right. \text{ give } \left|\begin{array}{ll} \ddot{r} - r\dot{\theta}^2 + g\sin\theta & = \frac{F}{m} \\ r^2\ddot{\theta} + 2r\dot{r}\dot{\theta} + gr\cos\theta & = \frac{T}{m} \end{array}\right.$$

The state model is constructed by defining the inputs $u_1 = F$ and $u_2 = T$, by selecting the state variables $x_1 = r, x_2 = \dot{r}, x_3 = \theta$ and $x_4 = \dot{\theta}$, and by defining the outputs $y_1 = \dot{r}$ and $y_2 = \theta$, which correspond to the available sensors.

Consider the direct model M of the system introduced in paragraph 1.4.3:

$$\dot{x}_1 = \dot{r} = x_2 \tag{3.10}$$

$$\dot{x}_2 = \ddot{r} = x_1 x_4^2 - g \sin x_3 + \frac{u_1}{m} \tag{3.11}$$

$$\dot{x}_3 = \dot{\theta} = x_4 \tag{3.12}$$

$$\dot{x}_4 = \ddot{\theta} = \frac{1}{x_1^2}\left[-2x_1 x_2 x_4 - g x_1 \cos x_3 + \frac{u_2}{m}\right] \tag{3.13}$$

The output equations are:

$$y_1 = x_2 \tag{3.14}$$

$$y_2 = x_3 \tag{3.15}$$

The successive derivation of these outputs provides the artificial inputs w.

$i = 1$	$d_1 = 0$	$y_1 = x_2$
	$d_1 = 1$	$\dot{y}_1 = \dot{x}_2 = x_1 x_4^2 - g \sin x_3 + \frac{u_1}{m}$ One of the inputs appears (u_1), we can stop
	$\bar{d_1} = 1$	$\dot{y}_1 \equiv w_1$

$i = 2$	$d_2 = 0$	$y_2 = x_3$
	$d_2 = 1$	$\dot{y}_2 = \dot{x}_3 = x_4$
	$d_2 = 2$	$\ddot{y}_2 = \dot{x}_4 = \frac{1}{x_1^2}\left[-2x_1 x_2 x_4 - g x_1 \cos x_3 + \frac{u_2}{M}\right]$
		One of the inputs appears (u_2), we can stop
$\bar{d_2}$	$\bar{d_2} = 2$	$\ddot{y}_2 \equiv w_2$

The choice of $\dot{y}_1 = w_1$ and $\ddot{y}_2 = w_2$ obtained above gives the following equations:

$$w_1 = x_1 x_4^2 - g \sin x_3 + \frac{u_1}{m}$$
$$w_2 = \frac{1}{x_1^2}\left[-2x_1 x_2 x_4 - g x_1 \cos x_3 + \frac{u_2}{m}\right]$$

that allow to obtain the inverse model M^{-1}:

$$u_1 = m(w_1 - x_1 x_4^2 + g \sin x_3) \tag{3.16}$$
$$u_2 = m(w_2 x_1^2 + 2x_1 x_2 x_4 + g x_1 \cos x_3) \tag{3.17}$$

The complete diagram of the system linearized by feedback linearization is represented in Figure 3.12.

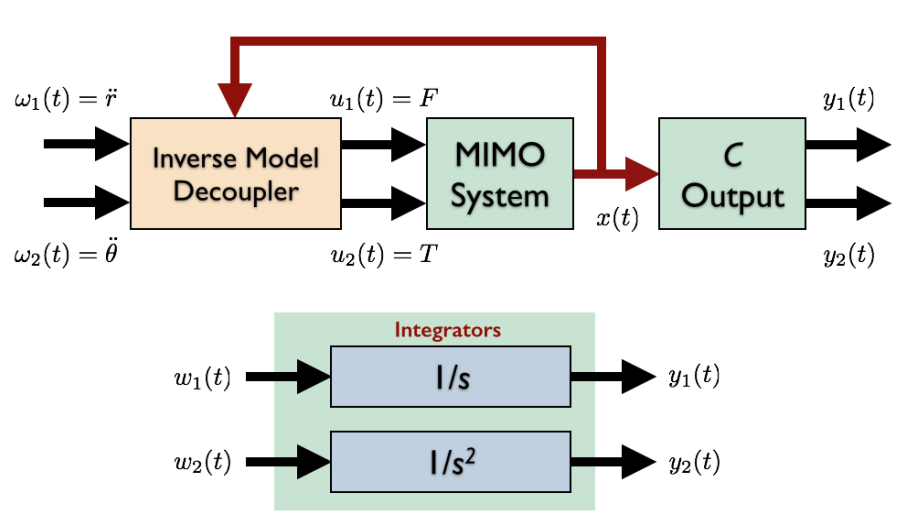


Figure 3.12 – Combination of the robot and its inverse model.

3.5 Exercises

3.5.1 Quadrotor

Problem

One way for controlling a quadrotor is by decoupling the altitude and the attitude sub-systems and designing separate controllers for these two sub-systems. The sub-set of equations that describes the dynamics of the attitude is the following:

$$\ddot{\phi} = I_1\dot{\theta}\dot{\psi} - b_1\dot{\theta} + u_1$$
$$\ddot{\theta} = I_2\dot{\phi}\dot{\psi} - b_2\dot{\phi} + u_2$$
$$\ddot{\psi} = I_3\dot{\theta}\dot{\phi} + u_3$$

1. Write the state space representation of this sub-system; consider the attitude of the quadrotor as output, i.e. the triplet ϕ, θ, ψ.

2. Linearize by feedback this sub-system, i.e. provide the inverse and the resulting linearized models.

3. Compare this method with the small signal linearization method.

Solution

1.

$x_1 = \phi$	$\dot{x}_1 = x_2$
$x_2 = \dot{\phi}$	$\dot{x}_2 = I_1 x_4 x_6 - b_1 x_4 + u_1$
$x_3 = \theta$	$\dot{x}_3 = x_4$
$x_4 = \dot{\theta}$	$\dot{x}_4 = I_2 x_2 x_6 - b_2 x_2 + u_2$
$x_5 = \psi$	$\dot{x}_5 = x_6$
$x_6 = \dot{\psi}$	$\dot{x}_6 = I_3 x_2 x_4 + u_3$

$$y_1 = \phi$$
$$y_2 = \theta$$
$$y_3 = \phi$$

2.

$$
\left|\begin{array}{l}
\dot{y}_1 = x_2 \\
\ddot{y}_1 = I_1 x_4 x_6 - b_1 x_4 + u_1 \equiv w_1 \\
\dot{y}_2 = x_4 \\
\ddot{y}_2 = I_2 x_2 x_6 - b_2 x_2 + u_2 \equiv w_2 \\
\dot{y}_3 = x_6 \\
\ddot{y}_3 = I_3 x_2 x_4 + u_3 \equiv w_3
\end{array}\right.
\quad
\left|\begin{array}{l}
w_1 = I_1 x_4 x_6 - b_1 x_4 + u_1 \\
w_2 = I_2 x_2 x_6 - b_2 x_2 + u_2 \\
w_3 = I_3 x_2 x_4 + u_3
\end{array}\right.
\quad
\left|\begin{array}{l}
\ddot{y}_1 = w_1 \\
\ddot{y}_2 = w_2 \\
\ddot{y}_3 = w_3
\end{array}\right.
$$

$$
\left|\begin{array}{l}
u_1 = w_1 - I_1 x_4 x_6 + b_1 x_4 \\
u_2 = w_2 - I_2 x_2 x_6 + b_2 x_2 \\
u_3 = w_3 - I_3 x_2 x_4
\end{array}\right.
$$

3. Small signal linearization only works very near to operating point but give the opportunity to design a MIMO optimal controller. With the feedback linearisation, we need an observer to estimate all the states required to compute the inverse model, but we get three decoupled simple linear systems.

3.5.2 Heat exchanger

Problem

The cross-section view of a heat exchanger is represented below:

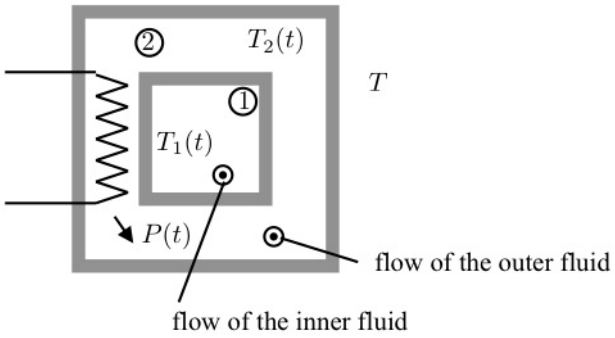


Figure 3.13 – Schematic representation of the heat exchanger

A fluid at temperature $T_1(t)$ flows inside the inner conduit ①. A fluid at temperature $T_2(t)$ flows inside the outer conduit ②. The external temperature T is constant. Parameters $m_1 c_1 = 0.5, m_2 c_2 = 2, k_{12} = 1$ and $k_{20} = 0.5$ have an impact on the heat exchange. It is:

m_i, c_i: specific mass and heat of the fluid, $i = 1,2$

k_{12}: thermal conductance between environments ① and ②

k_{20}: thermal conductance between fluid ② and the ambient environment

The dynamics of this system is governed by the following differential equations:

$$\left|\begin{array}{ll} m_1 c_1 \dot{T}_1(t) & = k_{12}[T_2(t) - T_1(t)] \\ m_2 c_2 \dot{T}_2(t) & = P(t) - k_{12}[T_2(t) - T_1(t)] - k_{20}[T_2(t) - T] \end{array}\right.$$

1. Knowing that it is the temperature of the internal fluid $T_1(t)$ which interests the designer and that the input is the power supply $P(t)$, describe the dynamics of this system with a state-space model.

2. Determine the state at the equilibrium specified by a power of 20 W and an ambient temperature of 20°.

3. Determine the state-space model that is valid for the evolutions around the operating point determined at point 2.

Solution

State-space model with $x_1 = T_1, x_2 = T_2, u = P$ and $y = T_1$

Operating point:

$0 = k_{12}(\bar{x}_2 - \bar{x}_1)$

$0 = \bar{u} - k_{12}(\bar{x}_2 - \bar{x}_1) - k_{20}(\bar{x}_2 - T)$

For $T = 20$ and $\bar{u} = 20$, we get $\bar{x}_2 = \bar{x}_1 = \bar{x}$ and $\bar{u} = k_{20}(\bar{x} - T)$.

Thus,

$$\bar{x} = \frac{\bar{u}}{k_{20}} + T = \frac{20}{0.5} + 20 = 60^\circ$$

Model to be linearized:

$$\left|\begin{array}{ll} f_1 & = -\frac{k_{12}}{m_1 c_1} x_1 + \frac{k_{12}}{m_1 c_1} x_2 \\ f_2 & = \frac{k_{12}}{m_2 c_2} x_1 - \frac{(k_{12}+k_{20})}{m_2 c_2} x_2 + \frac{1}{m_2 c_2} u + \frac{k_{20}}{m_2 c_2} T \end{array}\right.$$

Linearized model:

$$\dot{\tilde{x}} = \begin{bmatrix} -2 & 2 \\ 0.5 & -0.75 \end{bmatrix} \tilde{x} + \begin{bmatrix} 0 \\ 0.5 \end{bmatrix} \tilde{u}$$
$$\tilde{y} = \begin{bmatrix} 1 & 0 \end{bmatrix} \tilde{x}$$

3.5.3 Blood glucose concentration

Problem

The blood glucose concentration of people with diabetes can be Describe using the following nonlinear model:

$$\dot{G}(t) = -I(t)G(t) - ZG(t) + E$$

where $G(t)$ is the blood glucose concentration, $I(t)$ is the insulin action, Z and E are constants representing respectively the independent glucose consumption and the endogenous glucose production. Insulin action can be linked to insulin delivery $U(t)$ by the following equation:

$$\ddot{I}(t) = -2a\dot{I}(t) - a^2 I(t) + a^2 FU(t)$$

where a is the inverse of the time constant and F is a constant representing the sensitivity to insulin. $U(t)$ is consider as the input of the system.

1. Describe the model of this system using the state-space representation.

2. Determine the nominal states and the nominal input of this system for $G(t) = \overline{G}$.

3. Provide a linearized model of this system at the nominal point sets above (matrix form).

4. If the output is G, is it possible to linearize this system by feedback linearization (justify your answer)?

Solution

Selection of state variables:

$$\left| \begin{array}{l|l} x_1 = I & \dot{x}_1 = \dot{I} = x_2 \\ x_2 = \dot{I} & \dot{x}_2 = \ddot{I} = -2ax_2 - a^2 x_1 + a^2 Fu(t) \\ x_3 = G & \dot{x}_3 = \dot{G} = -x_1 x_3 - Zx_3 + E \end{array} \right.$$

$$\left| y = G = x_3 \right.$$

Nominal values:

$$\left|\begin{array}{l} 0 = \bar{x}_2 \\ 0 = -2a\bar{x}_2 - a^2\bar{x}_1 + a^2 F\bar{u} \\ 0 = -\bar{x}_1\bar{x}_3 - Z\bar{x}_3 + E \end{array}\right| \quad \left|\begin{array}{l} 0 = \bar{x}_2 \\ \bar{x}_1 = F\bar{u} \\ 0 = -\bar{x}_1\bar{G} - Z\bar{G} + E \end{array}\right|$$

$$\left|\begin{array}{l} \bar{x}_1 = \frac{E}{\bar{G}} - Z = \frac{E - Z\bar{G}}{\bar{G}} \\ \bar{x}_2 = 0 \\ \bar{x}_3 = \bar{G} \\ \bar{u} = \frac{\bar{x}}{F} = \frac{E - Z\bar{G}}{\bar{G}} \\ \bar{y} = \bar{G} \end{array}\right|$$

Linearized Model:

$$\dot{\tilde{x}}(t) = \begin{bmatrix} 0 & 1 & 0 \\ -a^2 & -2a & 0 \\ -\bar{x}_3 & 0 & -Z - \bar{x}_1 \end{bmatrix} \tilde{x}(t) + \begin{bmatrix} 0 \\ a^2 F \\ 0 \end{bmatrix} \tilde{u}(t)$$

$$\tilde{y}(t) = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \tilde{x}(t)$$

Feedback linearization:

$$\begin{aligned} y &= x_3 \\ \dot{y} &= \dot{x}_3 = -x_1 x_3 - Z x_3 + E \\ \ddot{y} &= -\dot{x}_1 x_3 - x_1 \dot{x}_3 - Z\dot{x}_3 = -x_2 x_3 - (x_1 + Z)(-x_1 x_3 - Z x_3 + E) \\ \dddot{y} &= - \underbrace{\dot{x}_2}_{\text{u will appear here}} x_3 - x_2 \dot{x}_3 - \cdots \equiv w \end{aligned}$$

4 Analysis

4.1 Solution of the discrete-time state equation and stability

The analysis and the controller design for a multivariable system are supported by powerful mathematical tools, as long as the considered system is represented by a linear and time-invariant model. In addition, the control solution is generally implemented on a micro-controller. Thanks to the successive linearization and discretization operations, the following adequate form is obtained:

$$x(k+1) = \Phi x(k) + \Gamma u(k) \tag{4.1}$$

$$y(k) = Cx(k) + Du(k) \tag{4.2}$$

The solution of this system of difference equations is obtained iteratively:

$$\begin{aligned}
x(k_0+1) = &\quad \Phi x(k_0) + \Gamma u(k_0) \\
x(k_0+2) = &\quad \Phi x(k_0+1) + \Gamma u(k_0+1) = \\
&\quad \Phi[\Phi x(k_0) + \Gamma u(k_0)] + \Gamma u(k_0+1) \\
x(k_0+2) = &\quad \Phi^2 x(k_0) + \Phi\Gamma u(k_0) + \Gamma u(k_0+1) \\
x(k_0+3) = &\quad \Phi x(k_0+2) + \Gamma u(k_0+2) = \Phi\{\Phi^2 x(k_0) \\
&\quad + \Phi\Gamma u(k_0) + \Gamma u(k_0+1)\} + \Gamma u(k_0+2) \\
x_0(k_0+3) = &\quad \Phi^3 x(k_0) + \Phi^2\Gamma u(k_0) + \Phi\Gamma u(k_0+1) + \Gamma u(k_0+2)
\end{aligned}$$

and so on, so that we obtain:

$$x(k) = \Phi^{k-k_0} x(k_0) + \sum_{i=k_0}^{k-1} \Phi^{k-i-1} \Gamma u(i) \tag{4.3}$$

4.1.1 Discrete transfer matrix

The application of the z-transform on equations (4.1) and (4.2) gives, for zero initial conditions:

$$zX(z) - \Phi X(z) = \Gamma U(z)$$
$$Y(z) = CX(z) + DU(z)$$

or, in a more compact form:

$$\left| \begin{array}{rl} (zI - \Phi)X(z) & = \Gamma U(z) \\ Y(z) & = CX(z) + DU(z) \end{array} \right.$$

$$\begin{array}{lrl} \text{thus:} & X(z) & = (zI - \Phi)^{-1} \Gamma U(z) \\ \text{and:} & Y(z) & = \{C(zI - \Phi)^{-1} \Gamma + D\} U(z). \end{array}$$

Finally $H(z) = C(zI - \Phi)^{-1}\Gamma + D$ is the discrete transfer matrix which is of dimension $p \times r$.

The discrete Leverrier algorithm allows to compute easily the inverse of the matrix $(zI - \Phi)$.

4.1.2 Discrete Leverrier algorithm

The Leverrier algorithm described in paragraph 2.5 can simply be written by replacing the argument $sI - A$ by $zI - \Phi$. Thus:

$$(zI - \Phi)^{-1} = \frac{H_0 z^{n-1} + H_1 z^{n-2} + H_2 z^{n-3} + \cdots + H_{n-1}}{\det(zI - \Phi)} \tag{4.4}$$

The determinant $\det(zI - \Phi) = z^n + a_1 z^{n-1} + a_2 z^{n-2} + \cdots + a_n$ is the characteristic polynomial of Φ. The values of z that cancel this characteristic polynomial are the eigenvalues of Φ. By the structure of equation (4.4), we note that these eigenvalues play the same role in the transfer matrix as the poles in a transfer function. The matrices H_i are of dimension $n \times n$. They are

sequentially calculated from $H_0 = I$:

$$\left|\begin{array}{ll} a_1 = -tr(\Phi H_0) & H_1 = \Phi H_0 + a_1 I \\ a_2 = -\frac{1}{2} tr(\Phi H_1) & H_2 = \Phi H_1 + a_2 I \\ a_3 = -\frac{1}{3} tr(\Phi H_2) & H_3 = \Phi H_2 + a_3 I \\ & \vdots \\ a_{n-1} = -\frac{1}{n-1} tr(\Phi H_{n-2}) & H_{n-1} = \Phi H_{n-2} + a_{n-1} I \\ a_n = -\frac{1}{n} tr(\Phi H_{n-1}) & H_n = \Phi H_{n-1} + a_n I = 0 \end{array}\right.$$

4.1.3 Stability

As seen previously, the output of a multivariable system can be expressed in the form:

$$Y(z) = H(z)U(z) \tag{4.5}$$

with:

$$H(z) = C(zI - \Phi)^{-1}\Gamma + D$$

or, thanks to Leverrier:

$$H(z) = \frac{C\text{adj}(zI - \Phi)\Gamma}{\det(zI - \Phi)} + D \tag{4.6}$$

Equation (4.5) represents, in a compact form, the relation:

$$\begin{bmatrix} Y_1(z) \\ Y_2(z) \\ \vdots \\ Y_p(z) \end{bmatrix} = \begin{bmatrix} H_{11}(z) & H_{12}(z) & \cdots & H_{1r}(z) \\ H_{21}(z) & H_{22}(z) & \cdots & H_{2r}(z) \\ \vdots & \vdots & \ddots & \vdots \\ H_{p1}(z) & H_{p2}(z) & \cdots & H_{pr}(z) \end{bmatrix} \begin{bmatrix} U_1(z) \\ U_2(z) \\ \vdots \\ U_r(z) \end{bmatrix}$$

According to (4.6), each transfer function H_{ij} has the same poles that correspond to the eigenvalues of Φ. H_{ij} is the transfer function of the system between the input j and the output i when all other inputs are set to zero. The system is stable if all these transfer functions are stable, that is to say if all the eigenvalues of Φ are in the unit circle. It is a sufficient condition of stability, but not a necessary one.

4.1.4 Example: Double integrator

The transfer matrix of the double integrator introduced in paragraph 2.3 is obtained in the following way:

$$(zI-\Phi)^{-1} = \frac{\text{adj}(zI-\Phi)}{\det(zI-\Phi)} = \frac{zI+H_1}{z^2+a_1 z+a_2}$$

By Leverrier:

$$a_1 = tr\Phi = -2 \qquad H_1 = \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix} - 2\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & h \\ 0 & -1 \end{bmatrix}$$

$$a_2 = -\frac{1}{2}tr(\Phi H_1) = -\frac{1}{2}tr\left\{\begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix}\begin{bmatrix} -1 & h \\ 0 & -1 \end{bmatrix}\right\} = -\frac{1}{2}tr\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} = 1$$

$$(zI-\Phi)^{-1} = \frac{1}{z^2-2z+1}\begin{bmatrix} z-1 & h \\ 0 & z-1 \end{bmatrix}$$

$$H(z) = C[zI-\Phi]^{-1}\Gamma + D = \begin{bmatrix} 1 & 0 \end{bmatrix}\begin{bmatrix} z-1 & h \\ 0 & z-1 \end{bmatrix}\begin{bmatrix} h^2/2 \\ h \end{bmatrix}\frac{1}{(z-1)^2}$$

$$H(z) = \begin{bmatrix} z-1 & h \end{bmatrix}\begin{bmatrix} h^2/2 \\ h \end{bmatrix}\frac{1}{(z-1)^2} = \frac{(z-1)h^2/2+h^2}{(z-1)^2} = \frac{h^2}{2}\frac{z+1}{(z-1)^2}$$

the same result as that obtained by applying the well known relation from the SISO case:

$$(1-z^{-1})\mathcal{Z}\left\{\mathcal{L}^{-1}\left[\frac{G(s)}{s}\right]\right\}$$

4.2 State-space model of a system defined by a transfer function

Consider the discrete transfer function:

$$H(z) = \frac{Y(z)}{U(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_n z^{-n}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_n z^{-n}} \tag{4.7}$$

By performing the polynomial division, we get:

$$H(z) = b_0 + \frac{(b_1 - a_1 b_0)z^{-1} + (b_2 - a_2 b_0)z^{-2} + \cdots + (b_n - a_n b_0)z^{-n}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_n z^{-n}}$$

Consequently,

$$Y(z) = b_0 U(z) + \tilde{Y}(z) \tag{4.8}$$

with:

$$\tilde{Y}(z) = \frac{(b_1 - a_1 b_0) z^{-1} + (b_2 - a_2 b_0) z^{-2} + \cdots + (b_n - a_n b_0) z^{-n}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_n z^{-n}} U(z)$$

This equation can also be written:

$$\frac{\tilde{Y}(z)}{(b_1 - a_1 b_0) z^{-1} + (b_2 - a_2 b_0) z^{-2} + \cdots + (b_n - a_n b_0) z^{-n}}$$

$$= \frac{U(z)}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_n z^{-n}} \equiv Q(z)$$

Two equations can be obtained from the last expression:

$$Q(z) = -a_1 z^{-1} Q(z) - a_2 z^{-2} Q(z) - \cdots - a_n z^{-n} Q(z) + U(z) \tag{4.9}$$

$$\tilde{Y}(z) = (b_1 - a_1 b_0) z^{-1} Q(z) + (b_2 - a_2 b_0) z^{-2} Q(z) + \cdots$$
$$\cdots + (b_n - a_n b_0) z^{-n} Q(z) \tag{4.10}$$

Now, let's make the following selection of state variables:

$$\left| \begin{array}{lcl} W_1(z) & = & z^{-1} Q(z) \\ W_2(z) & = & z^{-2} Q(z) \\ & \vdots & \\ W_n(z) & = & z^{-n} Q(z) \end{array} \right.$$

which can also be written:

$$\left|\begin{array}{rcl} zW_1(z) & = & Q(z) \\ zW_2(z) & = & W_1(z) \\ zW_3(z) & = & W_2(z) \\ & \vdots & \\ zW_n(z) & = & W_{n-1}(z) \end{array}\right.$$

According to relation (4.8) and using the defined state variables:

$$zW_1(z) = -a_1W_1(z) - a_2W_2(z) - \cdots - a_nW_n(z) + U(z)$$

hence the discrete-time state-space representation is:

$$\left|\begin{array}{rcl} w_1(k+1) & = & -a_1w_1(k) - a_2w_2(k) - \cdots - a_nw_n(k) + u(k) \\ w_2(k+1) & = & w_1(k) \\ w_3(k+1) & = & w_2(k) \\ & \vdots & \\ w_n(k+1) & = & w_{n-1}(k) \end{array}\right.$$

The letter w is introduced instead of x to emphasize that the state variables are artificial, that is to say that they do not necessarily have a physical meaning. Their selection simply derives from the considered mathematical construction.

According to (4.8) and (4.10), the output equation becomes:

$$Y(z) = (b_1 - a_1b_0)W_1(z) + (b_2 - a_2b_0)W_2(z) + \cdots + (b_n - a_nb_0)W_n(z) + b_0U(z)$$

In matrix form, we get:

$$w(k+1) = \underbrace{\begin{bmatrix} -a_1 & -a_2 & \cdots & -a_{n-1} & -a_n \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & 0 \\ 0 & \cdots & 0 & 1 & 0 \end{bmatrix}}_{\Phi_w} w(k) + \underbrace{\begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}}_{g_w} u(k) \tag{4.11}$$

$$y(k) = \underbrace{\left[b_1 - a_1 b_0 | b_2 - a_2 b_0 | \cdots | b_n - a_n b_0 \right]}_{c_w^T} w(k) + b_0 u(k) \tag{4.12}$$

Consequently, to construct the state-space representation of a SISO system described by a transfer function, just introduce the opposite of the coefficients of the denominator in the first row of the matrix Φ_w. The column vector g_w is independent of the system. The vector c_w^T is simply derived from the coefficients of the numerator of the transfer function.

4.2.1 Example: Discrete-time model of an electrical drive

The electrical drive described in paragraph 1.6.3 can be represented by the following transfer function:

$$G(s) = \frac{\theta(s)}{U(s)} = \frac{A}{s(1+s\tau)}$$

The conversion between the continuous-time state-space model and the transfer function exhibits a time constant $\tau = -1/a$ and a static gain for the velocity $A = -b/a$. The model of this linear and time-invariant system observed through AD and DA converters is obtained, in this well known SISO case, by the relation:

$$H(z) = (1-z^{-1})\mathcal{Z}\left\{\mathcal{L}^{-1}\left[\frac{G(s)}{s}\right]\right\}$$

$$H(z) = \frac{\theta(s)}{U(s)} = \frac{b_1 z^{-1} + b_2 z^{-2}}{(1-z^{-1})(1+\alpha z^{-1})} = \frac{b_1 z^{-1} + b_2 z^{-2}}{1+(\alpha-1)z^{-1} - \alpha z^{-2}}$$

where:

$b_1 = A[h - \tau(1 - e^{-h/\tau})]$,
$b_2 = A[\tau(1 - e^{-h/\tau}) - he^{-h/\tau}]$ and
$\alpha = -e^{-h/\tau}$

An equivalent discrete-time state-space model can be constructed by exploiting the approach which was introduced in the previous paragraph. Thus:

$$\begin{bmatrix} w_1(k+1) \\ w_2(k+2) \end{bmatrix} = \begin{bmatrix} -(\alpha-1) & \alpha \\ 1 & 0 \end{bmatrix} \begin{bmatrix} w_1(k) \\ w_2(k) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(k)$$

$$y(k) = \theta(k) = \begin{bmatrix} b_1 & b_2 \end{bmatrix} \begin{bmatrix} w_1(k) \\ w_2(k) \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} u(k)$$

4.3 Controllability canonical form

4.3.1 State-space model of a SISO system represented by a transfer function

As introduced in paragraph 4.2, the discrete transfer function:

$$H(z) = \frac{Y(z)}{U(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_n z^{-n}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_n z^{-n}} \tag{4.13}$$

can be represented by a state-space model that relies on the artificial state variables w:

$$w(k+1) = \underbrace{\begin{bmatrix} -a_1 & -a_2 & \cdots & -a_{n-1} & -a_n \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & 0 \\ 0 & \cdots & 0 & 1 & 0 \end{bmatrix}}_{\Phi_w} w(k) + \underbrace{\begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}}_{g_w} u(k) \tag{4.14}$$

$$y(k) = \underbrace{[b_1 - a_1 b_0 | b_2 - a_2 b_0 | \cdots | b_n - a_n b_0]}_{c_w^T} w(k) + b_0 u(k) \tag{4.15}$$

It is trivial to show that the characteristic equation of the matrix Φ_w is the following:

$$\det(\lambda I - \Phi_w) = \lambda^n + a_1 \lambda^{n-1} + a_2 \lambda^{n-2} + \cdots + a_{n-1}\lambda + a_n = 0$$

The eigenvalues of Φ_w are equal to the poles of the transfer function $H(z)$.

4.3.2 Transfer function of a SISO system represented by a state-space model

Consider the state-space model of a discrete SISO system:

$$\begin{aligned} x(k+1) &= \Phi x(k) + g u(k) \\ y(k) &= c^T x(k) + d u(k) \end{aligned}$$

We will substitute state variables in order to find, as a representation, the **controllability canonical form**. The corresponding transfer function is obtained immediately thanks to the results of paragraph 4.3.1.

The state-space representation being not unique, any bijective linear application can be chosen to substitute the state vector x by a new state vector w, for example $w = Px$. The matrix P, of dimension $n \times n$, must be regular (invertible), so that the inverse transformation $x = P^{-1}w$ is possible. The new state-space model, expressed with respect to the artificial state variable w, is:

$$\left| \begin{array}{rl} w(k+1) & = P\Phi P^{-1} w(k) + Pg\,u(k) \\ y(k) & = c^T P^{-1} w(k) + d\,u(k) \end{array} \right.$$

The aim is now to find a regular matrix P, such that the state-space model is in the desired canonical form.

First, choose a particular matrix G:

$$G = \begin{bmatrix} Ig & \Phi g & \cdots & \Phi^{n-1} g \end{bmatrix} \tag{4.16}$$

constructed from the elements of the original state-space model. This matrix is named after the controllability matrix. This denomination will be justified later. The rows of the inverse of the matrix G can be spotted arbitrarily in the following way:

$$G^{-1} = \begin{bmatrix} e_1^T \\ \vdots \\ e_n^T \end{bmatrix}$$

By definition of the inverse of a matrix, we have: $I = G^{-1}G$, that is to say:

$$\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix} = \begin{bmatrix} e_1^T Ig & e_1^T \Phi g & \cdots & e_1^T \Phi^{n-1} g \\ e_2^T Ig & e_2^T \Phi g & \cdots & e_2^T \Phi^{n-1} g \\ \vdots & \vdots & & \vdots \\ e_n^T Ig & e_n^T \Phi g & \cdots & e_n^T \Phi^{n-1} g \end{bmatrix}$$

By identifying the last row of the two members of this equation, we get:

$$e_n^T Ig = e_n^T \Phi g = \cdots = e_n^T \Phi^{n-2} g = 0 \tag{4.17}$$

$$e_n^T \Phi^{n-1} g = 1 \tag{4.18}$$

The line vectors $e_n^T I, e_n^T \Phi, \cdots, e_n^T \Phi^{n-1}$ are linearly independent. Indeed, suppose the existence of the constants $\alpha_0, \alpha_1, \cdots, \alpha_{n-1}$ such that:

$$\alpha_0 e_n^T I + \alpha_1 e_n^T \Phi + \cdots + \alpha_{n-1} e_n^T \Phi^{n-1} = 0 \tag{4.19}$$

The multiplication of (4.19) by the vector g gives:

$$\alpha_0 e_n^T Ig + \alpha_1 e_n^T \Phi g + \cdots + \alpha_{n-1} e_n^T \Phi^{n-1} g = 0$$

By equations (4.17) and (4.18), we get: $\alpha_{n-1} = 0$. The multiplication of (4.19) by Φg gives:

$$\alpha_0 e_n^T \Phi g + \alpha_1 e_n^T \Phi^2 g + \cdots + \alpha_{n-2} e_n^T \Phi^{n-1} g = 0$$

By equations (4.17) and (4.18), we get: $\alpha_{n-2} = 0$. By repeating this operation, we end up to:

$$\alpha_0 = \alpha_1 = \cdots = \alpha_{n-1} = 0$$

This shows that the vectors $e_n^T I, e_n^T \Phi, \cdots, e_n^T \Phi^{n-1}$ are linearly independent. They can be used to construct the matrix P, which will be regular.

$$P = \begin{bmatrix} e_n^T \Phi^{n-1} \\ e_n^T \Phi^{n-2} \\ \vdots \\ e_n^T I \end{bmatrix}$$

Consequently, the matrix Pg which is part of the new state-space model is:

$$Pg = \begin{bmatrix} e_n^T \Phi^{n-1} g \\ e_n^T \Phi^{n-2} g \\ \vdots \\ e_n^T I g \end{bmatrix}$$

A simple inspection shows that it is in fact the transpose of the last row of the matrix $G^{-1}G$. Thus:

$$Pg = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

The product $P\Phi P^{-1}$, also part of the new state-space equation, can be expressed in the following form:

$$P(\Phi P^{-1}) = \begin{bmatrix} e_n^T \Phi^n P^{-1} \\ e_n^T \Phi^{n-1} P^{-1} \\ \vdots \\ e_n^T \Phi P^{-1} \end{bmatrix} \tag{4.20}$$

The first row of this matrix is unknown. We set arbitrarily:

$$e_n^T \Phi^n P^{-1} = \begin{bmatrix} -a_1 & -a_2 & \cdots & -a_{n-1} & -a_n \end{bmatrix}$$

The other rows are obtained by comparison, knowing that $PP^{-1} = I$:

$$PP^{-1} = \begin{bmatrix} e_n^T \Phi^{n-1} P^{-1} \\ e_n^T \Phi^{n-2} P^{-1} \\ \vdots \\ e_n^T \Phi P^{-1} \\ e_n^T I P^{-1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix}$$

Thus:

$$P\Phi P^{-1} = \begin{bmatrix} -a_1 & -a_2 & \cdots & -a_{n-1} & -a_n \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & 0 \\ 0 & \cdots & 0 & 1 & 0 \end{bmatrix}$$

Finally, the state-space model expressed as a function of w by the transformation is similar to the controllability canonical form:

$$w(k+1) = \underbrace{\begin{bmatrix} -a_1 & -a_2 & \cdots & -a_{n-1} & -a_n \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & 0 \\ 0 & \cdots & 0 & 1 & 0 \end{bmatrix}}_{P\Phi P^{-1} = \Phi_w} w(k) + \underbrace{\begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}}_{Pg = g_w} u(k) \tag{4.21}$$

$$y(k) = \underbrace{[b_1 - a_1 b_0 | b_2 - a_2 b_0 | \cdots | b_n - a_n b_0]}_{c^T P^{-1} = c_w^T} w(k) + b_0 u(k) \tag{4.22}$$

The factor $c^T P^{-1}$ must be calculated from case to case.

The coefficients of the characteristic polynomial of $P\Phi P^{-1}$ are the same as those of the characteristic polynomial of the matrix Φ and as those of the denominator of $H(z)$. Indeed, the eigenvalues and the poles constitute a characteristic that is intrinsic to a system, in relation with its stability, and must be independent of the representation form.

4.3.3 Example: Double integrator

Consider the discrete transfer function of the double integrator obtained in paragraph 4.1.4.

$$H(z) = \frac{h^2}{2}\frac{(z+1)}{(z-1)^2} = \frac{\frac{h^2}{2}z + \frac{h^2}{2}}{z^2 - 2z + 1} = \frac{\frac{h^2}{2}z^{-1} + \frac{h^2}{2}z^{-2}}{1 - 2z^{-1} + z^{-2}}$$
$$b_0 = 0; b_1 = b_2 = \frac{h^2}{2}; a_1 = -2; a_2 = 1$$

The results of paragraph 4.3.1 allow to construct directly a state-space representation that however does not possess any physical meaning.

$$w(k+1) = \begin{bmatrix} 2 & -1 \\ 1 & 0 \end{bmatrix} w(k) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(k)$$

$$y(k) = \begin{bmatrix} \frac{h^2}{2} & \frac{h^2}{2} \end{bmatrix} w(k)$$

The state-space representation, equivalent but possessing a physical meaning, was also obtained in paragraph 4.1.4.

$$x(k+1) = \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix} x(k) + \begin{bmatrix} h^2/2 \\ h \end{bmatrix} u(k)$$
$$y(k) = \begin{bmatrix} 1 & 0 \end{bmatrix} x(k)$$

The results of paragraph 4.3.2 allow to come back to a transfer function. The coefficients a_1 and a_2 of the denominator of $H(z)$ are the coefficients of the characteristic polynomial of Φ:

$$\det(\lambda I - \Phi) = \begin{vmatrix} \lambda - 1 & -h \\ 0 & \lambda - 1 \end{vmatrix} = 0$$
$$\lambda^2 - 2\lambda + 1 = z^2 + a_1 z + a_2 = 0$$

Note that the undertaken transformation does not require the calculation of the roots of this polynomial.

The coefficients b_1 and b_2 of the numerator of the transfer function come from the output equation: $y(k) = c^T P^{-1} w(k)$.

$$G = \begin{bmatrix} g & \Phi g \end{bmatrix} = \begin{bmatrix} h^2/2 & 3h^2/2 \\ h & h \end{bmatrix} \tag{4.23}$$

$$G^{-1} = -\frac{1}{h^3}\begin{bmatrix} h & -3h^2/2 \\ -h & h^2/2 \end{bmatrix} \rightarrow e_n^T = -\frac{1}{h^3}\begin{bmatrix} -h & h^2/2 \end{bmatrix}$$

$$P = \begin{bmatrix} e_n^T\Phi \\ e_n^T \end{bmatrix} = -\frac{1}{h^3}\begin{bmatrix} h & -h^2/2 \\ -h & h^2/2 \end{bmatrix}$$

$$P^{-1} = -\frac{1}{h^3}(-h^3)\begin{bmatrix} h^2/2 & h^2/2 \\ h & -h \end{bmatrix}$$

$$c^T P^{-1} = \begin{bmatrix} 1 & 0 \end{bmatrix}\begin{bmatrix} h^2/2 & h^2/2 \\ h & -h \end{bmatrix} = \begin{bmatrix} h^2/2 & h^2/2 \end{bmatrix}$$

As $d = 0$, we know that $b_0 = 0$ so $c^T P^{-1} = \begin{bmatrix} b_1 & b_2 \end{bmatrix}$.

We have directly:

$$b_1 = \frac{h^2}{2} \text{ and } b_2 = \frac{h^2}{2}$$

4.4 Observability canonical form

4.4.1 State model of a SISO system represented by a transfer function

The system described by the discrete transfer function (4.13):

$$H(z) = \frac{Y(z)}{U(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_n z^{-n}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_n z^{-n}}$$

can be described by the following state-space model:

$$v(k+1) = \underbrace{\begin{bmatrix} -a_1 & 1 & 0 & \cdots & 0 \\ -a_2 & 0 & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ -a_{n-1} & 0 & \cdots & 0 & 1 \\ -a_n & 0 & \cdots & 0 & 0 \end{bmatrix}}_{\Phi_v} v(k) + g_v u(k) \tag{4.24}$$

with: $$g_v = \begin{bmatrix} b_1 - a_1 b_0 \\ b_2 - a_2 b_0 \\ \vdots \\ b_{n-1} - a_{n-1} b_0 \\ b_n - a_n b_o \end{bmatrix}$$

$$y(k) = \underbrace{\begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \end{bmatrix}}_{c_V^T} v(k) + b_0 u(k) \tag{4.25}$$

It is the **observability canonical form**. To demonstrate this backwards representation, we only have to take the z-transform of equations (4.24) and (4.25), and remove the state variables. An approach similar to that applied in paragraph 4.2 can also be chosen to derive it. Equation (4.13) can indeed be written in the form:

$$Y(z)(1 + a_1 z^{-1} + \cdots + a_n z^{-n}) = U(z)(b_0 + b_1 z^{-1} + \cdots + b_n z^{-n})$$

or:

$$[Y(z) - b_0 U(z)] + z^{-1}[a_1 Y(z) - b_1 U(z)] + \cdots$$
$$\cdots + z^{-n}[a_n Y(z) - b_n U(z)] = 0$$

This expression ca be set in a coupled form:

$$\begin{aligned} Y(z) &= b_0 U(z) + z^{-1}(b_1 U(z) - a_1 Y(z) + z^{-1}\{b_2 U(z) - a_2 Y(z) \\ &+ z^{-1}[b_3 U(z) - a_3 Y(z) + z^{-1}(\cdots)]\}) \end{aligned} \tag{4.26}$$

This equation suggests, for the sake of simplicity, the following choice of the artificial state variables v_i:

$$\left| \begin{array}{rcl} V_1(z) &=& z^{-1}[b_1 U(z) - a_1 Y(z) + V_2(z)] \\ V_2(z) &=& z^{-1}[b_2 U(z) - a_2 Y(z) + V_3(z)] \\ &\vdots& \\ V_{n-1}(z) &=& z^{-1}[b_{n-1} U(z) - a_{n-1} Y(z) + V_n(z)] \\ V_n(z) &=& z^{-1}[b_n U(z) - a_n Y(z)] \end{array} \right. \tag{4.27}$$

The inverse z-transform of (4.26) and (4.27) provides respectively the expected output and state equations:

$$
\begin{aligned}
y(k) &= b_0 u(k) + v_1(k) \\
\left|\begin{array}{rcl}
v_1(k+1) &=& b_1 u(k) - a_1 \overbrace{[b_0 u(k) + v_1(k)]}^{y(k)} + v_2(k) \\
v_2(k+1) &=& b_2 u(k) - a_2 [b_0 u(k) + v_1(k)] + v_3(k) \\
&\vdots& \\
v_{n-1}(k+1) &=& b_{n-1} u(k) - a_{n-1} [b_0 u(k) + v_1(k)] + v_n(k) \\
v_n(k+1) &=& b_n u(k) - a_n Y(z)
\end{array}\right.
\end{aligned}
$$

4.4.2 Transfer function of a SISO system represented by a state-space model

Consider the state-space model of a discrete SISO system:

$$
\left|\begin{array}{rcl}
x(k+1) &=& \Phi x(k) + g u(k) \\
y(k) &=& c^T x(k) + d u(k)
\end{array}\right.
$$

We will use another set of state variables to find, as a representation, the **observability canonical form**. The corresponding transfer function is obtained immediately thanks to the results of paragraph 4.4.1.

Because the state-space representation is not unique, any bijective linear transformation can be chosen to switch from the state-space vector x to a new state-space vector v, for example $x = Rv$. The matrix R, of dimension $n \times n$, must be regular (invertible), so that the inverse transformation $v = R^{-1}x$ is possible. The new state-space model, expressed with respect to the new artificial state variable v, is then:

$$
\left|\begin{array}{rl}
v(k+1) &= R^{-1}\Phi R v(k) + R^{-1} g u(k) \\
y(k) &= c^T R v(k) + d u(k)
\end{array}\right.
$$

The aim is now to find a regular matrix R, such that the state-space model is in the desired canonical form.

Choose first a particular matrix Q:

$$Q = \begin{bmatrix} c^T I \\ c^T \Phi \\ \vdots \\ c^T \Phi^{n-1} \end{bmatrix} \tag{4.28}$$

constructed from the elements of the original state-space model. This matrix is named after the **observability matrix**. This denomination will be justified later. The columns of the inverse of the matrix Q can be spotted arbitrarily in the following way:

$$Q^{-1} = \begin{bmatrix} e_1 & e_2 & \cdots & e_n \end{bmatrix}$$

The last column of this matrix is used to construct the regular matrix R (the demonstration of its regularity is similar to that elaborated to show the regularity of P in paragraph 4.3.2.):

$$R = \begin{bmatrix} \Phi^{n-1} e_n & \cdots & \Phi e_n & I e_n \end{bmatrix}$$

By this transformation, the product $c^T R$ present in the new output equation is in the form:

$$c^T R = \begin{bmatrix} c^T \Phi^{n-1} e_n & \cdots & c^T \Phi e_n & c^T I e_n \end{bmatrix} \tag{4.29}$$

and by definition of the inverse of a matrix $I = QQ^{-1}$ we get:

$$\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix} = \begin{bmatrix} c^T I e_1 & c^T I e_2 & \cdots & c^T I e_n \\ c^T \Phi e_1 & c^T \Phi e_2 & \cdots & c^T \Phi e_n \\ \vdots & \vdots & & \vdots \\ c^T \Phi^{n-1} e_1 & c^T \Phi^{n-1} e_2 & \cdots & c^T \Phi^{n-1} e_n \end{bmatrix} \tag{4.30}$$

then the comparison of vector 4.29 with the last transposed column of 4.30 leads to an output equation similar to that exhibited by the considered canonical form:

$$c^T R = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix} = c_v^T$$

The complete state-space representation obtained by the substitution $x = R\nu$ is thus similar to the observability canonical form.

The product $R^{-1}\Phi R$ is written:

$$(R^{-1}\Phi)R = R^{-1}\Phi\begin{bmatrix}\Phi^{n-1}e_n & \cdots & \Phi e_n & Ie_n\end{bmatrix}$$

$$R^{-1}\Phi R = \begin{bmatrix}R^{-1}\Phi^n e_n & \cdots & R^{-1}\Phi^2 e_n & R^{-1}\Phi e_n\end{bmatrix} \tag{4.31}$$

in addition:

$$I = R^{-1}R = \begin{bmatrix}R^{-1}\Phi^{n-1}e_n & \cdots & R^{-1}\Phi e_n & R^{-1}e_n\end{bmatrix} \tag{4.32}$$

The comparison of (4.31) with (4.32) gives:

$$R^{-1}\Phi^{n-1}e_n = \begin{bmatrix}1\\0\\\vdots\\0\end{bmatrix}, \cdots, R^{-1}\Phi e_n = \begin{bmatrix}0\\\vdots\\1\\0\end{bmatrix}, R^{-1}e_n = \begin{bmatrix}0\\\vdots\\1\\0\end{bmatrix}$$

Only the term $R^{-1}\Phi^n e_n$ is unknown and it is set arbitrarily:

$$R^{-1}\Phi^n e_n = \begin{bmatrix}-a_1\\-a_2\\\vdots\\-a_{n-1}\\-a_n\end{bmatrix}$$

Thus,

$$R^{-1}\Phi R = \begin{bmatrix}-a_1 & 1 & 0 & \cdots & 0\\-a_2 & 0 & 1 & \ddots & \vdots\\\vdots & \vdots & \ddots & \ddots & 0\\-a_{n-1} & 0 & \cdots & 0 & 1\\-a_n & 0 & \cdots & 0 & 0\end{bmatrix} = \Phi_v$$

The factor $R^{-1}g = g_v$ have meanwhile to be determined from case to case.

The coefficients of the characteristic polynomial of $R^{-1}\Phi R$ are the same as those of the char-

acteristic polynomial of Φ and of the denominator of $H(z)$:

$$\det(\lambda I - \Phi_v) = \lambda^n + a_1\lambda^{n-1} + \cdots + a_{n-1}\lambda + a_n = 0$$

4.5 Controllability

One of the aims of an automatic control is often to transfer the initial state of a process to another state. Before tackling the synthesis of a controller operating this transition, it is primordial to know if it is physically possible or not. If the answer is no, major difficulties will probably arise. It is an analysis problem concerning a structural property of the system, named controllability.

We consider the stationary linear system of the order n, described by its state equation:

$$x(k+1) = \Phi x(k) + \Gamma u(k) \tag{4.33}$$

System (4.33) or the ordered pair (Φ, Γ) is controllable if, for any initial state $x(0)$ and for any given state x_d, there exists a finite time N and input vectors $u(0), u(1), ..., u(N-1)$ performing the transfer from $x(0)$ to $x(N) = x_d$.

System (4.33) is controllable if and only if the rank of the controllability matrix

$$G = \begin{bmatrix} \Gamma & \Phi\Gamma & \cdots & \Phi^{n-1}\Gamma \end{bmatrix} \in \mathbb{R}^{n \times nr} \tag{4.34}$$

has the value of n.

This result can be derived by writing:

$$x_d = x(N) = \Phi^N x(0) + \sum_{i=0}^{N-1} \Phi^{N-1-i}\Gamma u(i)$$

This equation can also be written:

$$\begin{aligned} & x_d - \Phi^N x(0) \\ & \quad = \Phi^{N-1}\Gamma u(0) + \Phi^{N-2}\Gamma u(1) + ... + \Phi\Gamma u(N-2) + \Gamma u(N-1) \\ & \quad = \begin{bmatrix} \Gamma & \Phi\Gamma & ... & \Phi^{N-2}\Gamma & \Phi^{N-1}\Gamma \end{bmatrix} \begin{bmatrix} u(N-1) \\ u(N-2) \\ \vdots \\ u(1) \\ u(0) \end{bmatrix} \end{aligned} \tag{4.35}$$

It is a system composed of n linear equations whose unknowns are the vectors $u(0)$, $u(1)$, ..., $u(N-2)$, $u(N-1)$; besides, because $x(0)$ and x_d are given, $x_d - \Phi^N x(0) \in \mathbb{R}^n$ is known.

Assume that the rank of the controllability matrix is n. By setting $N = n$ in (4.35):

$$G \begin{bmatrix} u(n-1) \\ u(n-2) \\ \vdots \\ u(1) \\ u(0) \end{bmatrix} = x_d - \Phi^n x(0) \tag{4.36}$$

G possesses n linearly independent columns; the addition of column $x_d - \Phi^n x(0) \in \mathbb{R}^n$ to G produces an augmented matrix $\begin{bmatrix} G & x_d - \Phi^n x(0) \end{bmatrix}$ whose rank is also n. We have $\text{rk}(G) = \text{rk}\begin{bmatrix} G & x_d - \Phi^n x(0) \end{bmatrix}$ and the system of equations (4.36) possesses a solution $u(0)$, $u(1)$, ..., $u(n-2)$, $u(n-1)$ operating the transition from $x(0)$ to x_d, proving the controllability of the process.

Assume now that the system is controllable. The system of equations (4.35) possesses, for an integer N, one solution and the rank of matrix $\begin{bmatrix} \Gamma & \Phi\Gamma & \cdots & \Phi^{N-2}\Gamma & \Phi^{N-1}\Gamma \end{bmatrix}$ is equal to the rank of augmented matrix $\begin{bmatrix} \Gamma & \Phi\Gamma & \cdots & \Phi^{N-2}\Gamma & \Phi^{N-1}\Gamma & x_d - \Phi^N x(0) \end{bmatrix}$. As vector $x_d - \Phi^N x(0) \in \mathbb{R}^n$ is known, this rank has the value of n. Thus:

$$\text{rk}\begin{bmatrix} \Gamma & \Phi\Gamma & \cdots & \Phi^{N-2}\Gamma & \Phi^{N-1}\Gamma \end{bmatrix} = n$$

If $N = n$, $\text{rk}(G) = n$. When $N < n$, the addition of supplementary columns to matrix

$$\begin{bmatrix} \Gamma & \Phi\Gamma & \cdots & \Phi^{N-2}\Gamma & \Phi^{N-1}\Gamma \end{bmatrix} \in \mathbb{R}^{n \times Nr}$$

to make matrix

$$\begin{bmatrix} \Gamma & \Phi\Gamma & \cdots & \Phi^{N-1}\Gamma & \Phi^{N}\Gamma & \cdots & \Phi^{n-1}\Gamma \end{bmatrix} = G \in \mathbb{R}^{n \times nr}$$

does not alter the rank; consequently, $\text{rk}(G) = n$. When $N > n$, consider a matrix $\Phi^M\Gamma$ appearing in $\begin{bmatrix} \Gamma & \Phi\Gamma & \cdots & \Phi^{n-1}\Gamma & \cdots & \Phi^{M}\Gamma & \cdots & \Phi^{N-1}\Gamma \end{bmatrix}$, with $n-1 < M < N-1$. It follows from the Cayley-Hamilton theorem that Φ^M is a linear combination of matrixes $\Phi^{n-1}, \Phi^{n-2}, ..., I$:

$$\Phi^M = r_0\Phi^{n-1} + r_1\Phi^{n-2} + ... + r_{n-1}I \qquad\qquad r_0, r_1, ..., r_{n-1} \in \mathbb{R}$$

Hence:

$$\Phi^M\Gamma = r_0\Phi^{n-1}\Gamma + r_1\Phi^{n-2}\Gamma + ... + r_{n-1}\Gamma$$

Each column of $\Phi^M B$ is a linear combination of the columns of $G = \begin{bmatrix} \Gamma & \Phi\Gamma & \cdots & \Phi^{n-1}\Gamma \end{bmatrix}$, knowing that the coefficients of these linear combinations are $r_0, r_1, ..., r_{n-1}$. We conclude that:

$$\text{rk}\begin{bmatrix} \Gamma & \Phi\Gamma & \cdots & \Phi^{n-1}\Gamma & \cdots & \Phi^{M}\Gamma & \cdots & \Phi^{N-1}\Gamma \end{bmatrix} = \text{rk}(G) = n$$

The demonstration of this theorem reveals that, if the controllability is acquired, the transition from the initial vector $x(0)$ to vector x_d can be completed within n sampling periods at most.

4.5.1 Example: Second order system

Consider the system represented by the state equation:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} a_1 & 0 \\ 1 & a_2 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k)$$

The controllability matrix is:

$$G = \begin{bmatrix} 0 & 0 \\ 1 & a_2 \end{bmatrix}$$

Its rank is 1 and the system is not controllable. The block diagram shown in Figure 4.1 reveals that the state variable $x_1(k)$ is not affected by the input $u(k)$, meaning that the system is not fully controllable.

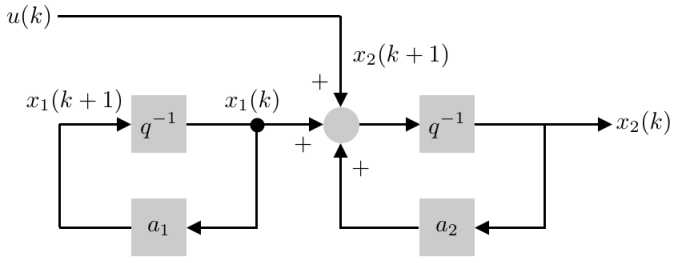


Figure 4.1 – Non-controllable system.

The system is modified by using another actuator influencing directly the first state variable (Fig. 4.2).

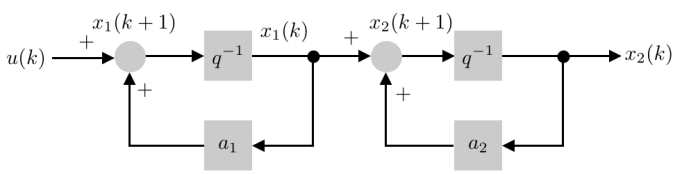


Figure 4.2 – Controllable system.

The corresponding state equation is:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} a_1 & 0 \\ 1 & a_2 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(k)$$

Hence the controllability matrix is:

$$G = \begin{bmatrix} 1 & a_1 \\ 0 & 1 \end{bmatrix}$$

Its rank is equal to 2: the system is controllable, which is suggested besides by a quick review of the block diagram from Figure 4.2.

4.5.2 Example: Electrical drive

The sampled state-space equation of the electrical drive is:

$$x(k+1) = \begin{bmatrix} 1 & 0.024 \\ 0 & 0.95 \end{bmatrix} x(k) + \begin{bmatrix} 1.23 \cdot 10^{-3} \\ 0.0926 \end{bmatrix} u(k)$$

The rank of the controllability matrix, written below, is 2, which shows that the system is controllable:

$$G = \begin{bmatrix} 1.23 \cdot 10^{-3} & 3.57 \cdot 10^{-3} \\ 0.0975 & 0.0926 \end{bmatrix}$$

4.6 Observability

The state variables of a system, which are internal quantities, are rarely all accessible by measurement. Only input $u(k)$ and output $y(k)$ are available. Hence the fundamental question is: is it possible to reconstruct, from the signals $u(k)$ and $y(k)$, the state $x(k)$? This is an analysis problem, linked to a structural feature of the system, which is named *observability*.

We assume that the system, of the order n, is represented by the state-space model:

$$x(k+1) = \Phi x(k) + \Gamma u(k) \tag{4.37}$$

$$y(k) = Cx(k) + Du(k) \tag{4.38}$$

The system described by (4.37) and (4.38) or the ordered pair (Φ, C) is *observable* if, for any initial state $x(0)$, there exists an integer N such that the knowledge of the input vectors $u(0), u(1), ..., u(N-1)$ and of the output vectors $y(0), y(1), ..., y(N-1)$ allows to determine in a unique way the initial state $x(0)$.

The system represented by (4.37) and (4.38) is observable if and only if the rank of the following *observability matrix* is n:

$$Q = \begin{bmatrix} C \\ C\Phi \\ \vdots \\ C\Phi^{n-1} \end{bmatrix} \in \mathbb{R}^{pn \times n}$$

This result can be derived by using the solution of the discrete state-space model given in

section 4.3:

$$y(k) = C\Phi^k x(0) + C\sum_{i=0}^{k-1} \Phi^{k-1-i}\Gamma u(i) + Du(k) \qquad k \geqslant 0$$

This equation can also be written:

$$y(k) - C\sum_{i=0}^{k-1} \Phi^{k-1-i}\Gamma u(i) - Du(k) = C\Phi^k x(0)$$

By defining:

$$\tilde{y}(k) = y(k) - C\sum_{i=0}^{k-1} \Phi^{k-1-i}\Gamma u(i) - Du(k)$$

we have:

$$\tilde{y}(k) = C\Phi^k x(0)$$

This equality provides, with $k = 0, 1, ..., N-1$:

$$\begin{bmatrix} \tilde{y}(0) \\ \tilde{y}(1) \\ \vdots \\ \tilde{y}(N-1) \end{bmatrix} = \begin{bmatrix} C \\ C\Phi \\ \vdots \\ C\Phi^{N-1} \end{bmatrix} x(0) \tag{4.39}$$

It is a system of pN linear equations whose unknown is vector $x(0)$.

Assume that the rank of the observability matrix is n. By setting $N = n$ in (4.39):

$$Qx(0) = \begin{bmatrix} \tilde{y}(0) \\ \tilde{y}(1) \\ \vdots \\ \tilde{y}(n-1) \end{bmatrix} \tag{4.40}$$

As $\text{rk}(Q) = n$, matrix Q contains n linearly independent lines; these lines provide n equations allowing to determine in a unique way the initial state $x(0)$: the observability of the process is proven.

Assume now that the system is observable. The system of equations (4.39) possesses, for a

certain integer N, a unique solution and:

$$\text{rk}\begin{bmatrix} C \\ C\Phi \\ \vdots \\ C\Phi^{N-1} \end{bmatrix} = \text{rk}\begin{bmatrix} C & \tilde{y}(0) \\ C\Phi & \tilde{y}(1) \\ \vdots & \vdots \\ C\Phi^{N-1} & \tilde{y}(N-1) \end{bmatrix} = n$$

If $N = n$, $\text{rk}(Q) = n$. When $N < n$, we must, in order to get the observability matrix $Q \in \mathbb{R}^{np \times n}$, add lines to the matrix:

$$\begin{bmatrix} C \\ C\Phi \\ \vdots \\ C\Phi^{N-1} \end{bmatrix} \in \mathbb{R}^{pN \times n}$$

This operation does not alter the rank and $\text{rk}(Q) = n$. When $N > n$, consider a matrix $C\Phi^M$ appearing in:

$$\begin{bmatrix} C \\ C\Phi \\ \vdots \\ C\Phi^{n-1} \\ \vdots \\ C\Phi^{M} \\ \vdots \\ C\Phi^{N-1} \end{bmatrix} \qquad n-1 < M < N-1 \tag{4.41}$$

But:

$$\Phi^M = r_0\Phi^{n-1} + r_1\Phi^{n-2} + ... + r_{n-1}I \qquad r_0, r_1, ..., r_{n-1} \in \mathbb{R}$$

Hence:

$$C\Phi^M = r_0C\Phi^{n-1} + r_1C\Phi^{n-2} + ... + r_{n-1}C$$

Thus, each line of $C\Phi^M$ is a linear combination of the lines of Q, the coefficients of these linear combinations being $r_0, r_1, ..., r_{n-1}$. Consequently, the rank of matrix (4.41), which is n, is equal to the rank of Q.

The derivation of this result shows that, if the observability is acquired, the determination of $x(0)$ requires the knowledge of the input and the output during at most n sampling periods.

4.6.1 Example: Second order system

Consider the system described by the state-space model:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} a_1 & 1 \\ 0 & a_2 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix}$$
$$y(k) = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix}$$

The observability matrix is:

$$Q = \begin{bmatrix} 0 & 1 \\ 0 & a_2 \end{bmatrix}$$

Its rank is 1 and the system is not observable. A block diagram is shown in Figure 4.3, and it reveals that the state variable $x_1(k)$ does not affect the output.

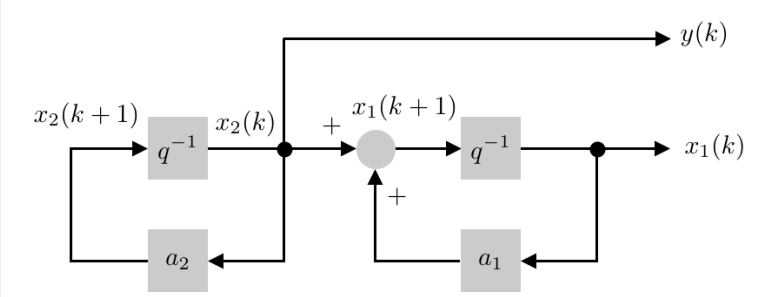


Figure 4.3 – Non-observable system.

This block diagram is modified by measuring x_1 instead of x_2 (Fig. 4.4).

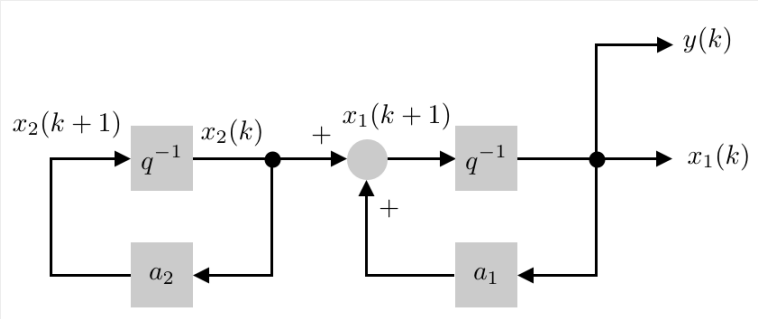


Figure 4.4 – Observable system.

The corresponding output equation is:

$$y(k) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix}$$

Hence the observability matrix is:

$$Q = \begin{bmatrix} 1 & 0 \\ a_1 & 1 \end{bmatrix}$$

Its rank is 2: the system is observable, which is suggested besides by a quick review of block diagram from Figure 4.4.

4.6.2 Example: Electrical drive

The sampled state-space model of the electrical drive, is:

$$x(k+1) = \begin{bmatrix} 1 & 0.024 \\ 0 & 0.95 \end{bmatrix} x(k) + \begin{bmatrix} 1.23 \cdot 10^{-3} \\ 0.0975 \end{bmatrix} u(k)$$
$$y(k) = \begin{bmatrix} 1 & 0 \end{bmatrix} x(k)$$

The rank of the observability matrix, calculated below, is 2, indicating that the system is observable:

$$Q = \begin{bmatrix} 1 & 0 \\ 1 & 0.024 \end{bmatrix}$$

4.7 Exercises

4.7.1 Transfer matrix

Problem

The discrete state-space model of a dynamic system is the following:

$$\begin{aligned} x(k+1) &= \begin{bmatrix} 0 & 1 \\ & \\ 1 & 0 \end{bmatrix} x(k) + \begin{bmatrix} 0 & 1 \\ & \\ 1 & 0 \end{bmatrix} u(k) \\ y(k) &= x(k) \end{aligned}$$

1. Determine the transfer matrix of this system.

Solution

The discrete-time state-space model of a dynamic system is the following:

$$\begin{aligned} x(k+1) &= \begin{bmatrix} 0 & 1 \\ & \\ 1 & 0 \end{bmatrix} x(k) + \begin{bmatrix} 0 & 1 \\ & \\ 1 & 0 \end{bmatrix} u(k) \\ y(k) &= \begin{bmatrix} 1 & 0 \\ & \\ 0 & 1 \end{bmatrix} x(k) \end{aligned}$$

1. The transfer matrix is given by:

$$H(z) = C(zI - \phi)^{-1}\Gamma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \frac{1}{z^2-1} \begin{bmatrix} z & 1 \\ 1 & z \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \frac{1}{z^2-1} \begin{bmatrix} 1 & z \\ z & 1 \end{bmatrix}$$

4.7.2 Characteristic polynomial

Problem

1. Show that

$$\det(\lambda I - \Phi) = \det(\lambda I - \Phi_w)$$

Solution

$$\det(\lambda I - \Phi_w) = \det\left(\begin{bmatrix} \lambda & 0 & \cdots & 0 \\ 0 & \lambda & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \lambda \end{bmatrix} - \begin{bmatrix} -a_1 & -a_2 & \cdots & -a_n \\ 1 & 0 & \cdots & 0 \\ & \ddots & & \vdots \\ 0 & & 1 & 0 \end{bmatrix}\right) = \begin{vmatrix} \lambda + a_1 & a_2 & \cdots & a_n \\ -1 & \lambda & & \vdots \\ & \ddots & \ddots & 0 \\ 0 & & -1 & \lambda \end{vmatrix}$$

$$\det(\lambda I - \Phi_w) = (\lambda + a_1)\underbrace{\begin{vmatrix} \lambda & 0 & \cdots & 0 \\ -1 & \lambda & & \vdots \\ & \ddots & \ddots & 0 \\ 0 & & -1 & \lambda \end{vmatrix}}_{\lambda^{n-1}} + \underbrace{\begin{vmatrix} a_2 & a_3 & \cdots & a_n \\ -1 & \lambda & & \vdots \\ & \ddots & \ddots & 0 \\ 0 & & -1 & \lambda \end{vmatrix}}_{}$$

$$a_2\underbrace{\begin{vmatrix} \lambda & 0 & \cdots & 0 \\ -1 & \lambda & & \vdots \\ & \ddots & \ddots & 0 \\ 0 & & -1 & \lambda \end{vmatrix}}_{\lambda^{n-2}} + \underbrace{\begin{vmatrix} a_3 & a_4 & \cdots & a_n \\ -1 & \lambda & & \vdots \\ & \ddots & \ddots & 0 \\ 0 & & -1 & \lambda \end{vmatrix}}_{\cdots}$$

$$\det(\lambda I - \Phi_w) = (\lambda + a_1)\lambda^{n-1} + a_2\lambda^{n-2} + a_3\lambda^{n-3} + \ldots + a_n$$

$$\det(\lambda I - \Phi_w) = \lambda^n + a_1\lambda^{n-1} + a_2\lambda^{n-2} + a_3\lambda^{n-3} + \ldots + a_n$$

4.7.3 Observability

Problem

The discrete-time state-space model of a dynamic system is the following:

$$x(k+1) = \begin{bmatrix} -1 & 0 \\ 1 & 0 \end{bmatrix} x(k) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(k)$$
$$y(k) = \begin{bmatrix} 1 & \alpha \end{bmatrix} x(k)$$

1. Determine all the values of α for which the system is not observable.

2. Explain what happens in the different cases where the observability is lost. If necessary, represent graphically the dynamic couplings within the system to support your claim.

Solution

The observability matrix is:

$$Q = \begin{bmatrix} c^T \\ c^T\Phi \end{bmatrix} = \begin{bmatrix} 1 & \alpha \\ \alpha - 1 & 0 \end{bmatrix}$$

The rank of the matrix Q is not full if the determinant is zero: $\alpha(\alpha - 1) = 0$

The system is not observable for $\alpha = 0$ and $\alpha = 1$.

For $\alpha = 0$, x_2 does not influence the output. This state is not observable.

For $\alpha = 1$, the initial values of x_1 and x_2 cannot be differentiated.

$$\begin{aligned} x_1(k+1) &= -x_1(k) + u(k) \\ x_2(k+1) &= x_1(k) \\ y(k) &= x_1(k) + x_2(k) \end{aligned}$$

The first equation can be written:

$$x_1(k+1) = -x_2(k+1) + u(k)$$

which leads to:

$$\begin{aligned} u(k) &= x_1(k+1) + x_2(k+1) \\ y(k) &= x_1(k) + x_2(k) \end{aligned}$$

As the observer is using $u(k)$ and $y(k)$, the above equations show that x_1 and x_2 cannot be differentiated.

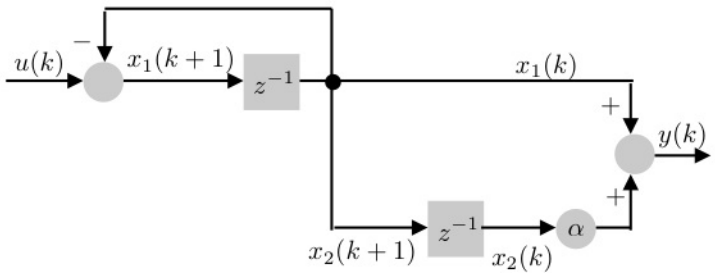


Figure 4.5 – Unobservable system

By using the controllability canonical form:

$$x(k+1) = \begin{bmatrix} -a_1 & -a_2 \\ 1 & 0 \end{bmatrix} x(k) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(k)$$
$$y(k) = \begin{bmatrix} b_1 & b_2 \end{bmatrix} x(k)$$

we get the equivalent transfer function:

$$\frac{Y(z)}{U(z)} = \frac{z+\alpha}{z(z+1)}$$

We note that in the two mentioned cases, there is a zero-pole cancellation.

$$\text{For } \alpha = 0 : \frac{Y(z)}{U(z)} = \frac{1}{(z+1)} \text{ and for } \alpha = 1 : \frac{Y(z)}{U(z)} = \frac{1}{z}.$$

5 State feedback

5.1 Feedback principle

In this chapter, we will only consider the case of time-invariant systems that only possess one input (SISO or SIMO). Generally, the dynamics is described around an operating point. If necessary, it is possible to go back to the case of dynamics expressed according to real quantities x, u and y by setting the nominal quantities $\bar{x}, \bar{u}$ and $\bar{y}$ to zero.

$$\left|\begin{array}{rl} \tilde{x}(k+1) & = \Phi\tilde{x}(k) + g\tilde{u}(k) \\ \tilde{y}(k) & = C\tilde{x}(k) + D\tilde{u}(k) \end{array}\right.$$

The feedback studied in this chapter is chosen as being proportional to the state deviation with respect to the nominal state. The control signal corresponds to a weighted sum of the deviations of the state by the control gains K_i:

$$\tilde{u} = -K\tilde{x} = -\begin{bmatrix} K_1 & K_2 & \cdots & K_n \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \\ \vdots \\ \tilde{x}_n \end{bmatrix}$$

A state feedback acts somehow like a generalized "PD" controller.

The case of the control of an electrical drive, where the two state variables are $\tilde{x}_1 = \tilde{\theta}$ and $\tilde{x}_2 = \dot{\tilde{\theta}}$, illustrates this assertion. Indeed, the proposed feedback leads to a control similar to that coming from a standard PD controller:

$$\tilde{u}(k) = -\begin{bmatrix} K_1 & K_2 \end{bmatrix} \begin{bmatrix} \tilde{\theta} \\ \dot{\tilde{\theta}} \end{bmatrix} = K_1 e(k) + K_2 \dot{e}(k) = K_p \left[e(k) + \frac{1}{Td} \dot{e}(k) \right]$$

with: $e(k) = \theta_r - \theta = \bar{\theta} - \theta = -\tilde{\theta}$

5.1.1 Closed-loop dynamics

The addition of the state feedback in the description of the dynamics of the system to be controlled gives a new state-space model, governing the dynamics of the closed-loop system. *The order of this system is the same as that of the system to be considered.*

$$\begin{aligned} \tilde{x}(k+1) &= \Phi \tilde{x}(k) + g[-K\tilde{x}(k)] \\ \tilde{x}(k+1) &= \underbrace{[\Phi - gK]}_{\Phi_{cl}} \tilde{x}(k) \end{aligned}$$

The dynamics of this closed-loop system is governed by the eigenvalues of the matrix Φ_{cl}. These eigenvalues are the solutions of the characteristic equation $\alpha_c(\lambda)$:

$$\alpha_c(\lambda) = \det[\lambda I - \Phi_{cl}] = \det[\lambda I - \Phi + gK] = 0$$

The eigenvalues of Φ depend on the physical constitution of the system to be controlled. They cannot be changed.

The eigenvalues of Φ_{cl} depend on the system to be controlled **and** on the controller. As Φ_{cl} possesses n eigenvalues and the controller consists of n gains, all its eigenvalues (λ_i for $i = 1, \cdots, n$) can be chosen freely, provided the system is controllable (see section 4.5). A given

dynamic behavior is then imposed in closed loop. Thus:

$$\begin{aligned}\alpha_c(\lambda) &= \det[\lambda I - \Phi + g \underbrace{K}_{unknown}] = \underbrace{(\lambda-\lambda_1)(\lambda-\lambda_2)\cdots(\lambda-\lambda_n)}_{chosen} = 0\\ \alpha_c(\lambda) &= \lambda^n + \alpha_1\lambda^{n-1} + \cdots + \alpha_n = 0\end{aligned}$$

The gain K is obtained by identifying term by term the successive powers of λ in the two members of the last equation. This comparison provides n equations that allow to determine the n unknown K_i, for $i = 1,\ldots,n$.

If the closed-loop system deviates from its nominal state (because of non-zero initial conditions or under the effect of a disturbance) and the eigenvalues of Φ_{cl} have been chosen in the unit circle, the system converges asymptotically to the nominal state, that is to say in $\tilde{x}(k) = 0$, with the following dynamics:

$$z[\tilde{X}(z) - \tilde{x}(0)] = \Phi_{cl}\tilde{X}(z)$$

$$\tilde{X}(z) = (zI - \Phi_{cl})^{-1}\tilde{x}(0)z$$

The inverse matrix that appears in the last expression is obtained by means of the Leverrier algorithm. It is a matrix of rational fractions in z, whose denominators are all equal and have the value of $\det(zI - \Phi_{cl})$, so $\alpha_c(z)$. The eigenvalues of Φ_{cl} govern the dynamics of the closed loop system.

5.1.2 Example: Double integrator

The discrete state-space model of the double integrator was obtained in Section 2.3. It is characterized by the matrices:

$$\Phi = \begin{bmatrix}1 & h\\ 0 & 1\end{bmatrix}; \Gamma = \begin{bmatrix}h^2/2\\ h\end{bmatrix}$$

As the system is of second order, it is possible to impose in closed-loop an oscillatory behavior described by a transfer function of the same order:

$$K\frac{\omega_0^2}{s^2 + 2\zeta\omega_0 s + \omega_0^2} \quad \text{(here, } K \text{ is the static gain)}$$

The poles of this continuous-time system are $s_{1,2} = -\omega_0(\zeta \pm \sqrt{\zeta^2 - 1})$, and their discrete-time equivalent $z_{1,2} = e^{hs_{1,2}}$, where h is the sampling period.

At the design stage, some criteria must be specified. Particularly, considerations related to the targeted application can require a limitation of the amplitude of the 1^{st} overshoot. In this example, we would like that this value does not exceed 12.3%. To satisfy this specification, $\zeta = 0.5$ has to be chosen. Usually, the application also defined how fast the response should be. In this example, the time of the first maximum is set to one second ($t_p = 1$), thus $\omega_0 = 3.6$. Finally, the corresponding continuous-time poles are $s_{1,2} = -1.8 \pm 3.12\,j$ and their discrete-time equivalent $z_{1,2} = 0.8 \pm 0.25\,j$, for a sampling period $h = 0.1$ second.

In this monovariable case, the notions of eigenvalues and poles are equivalent. Consequently, the imposed characteristic equation is the following:

$$\det[zI - \Phi + gK] = (z - 0.8 - 0.25\,j)(z - 0.8 + 0.25\,j) = z^2 - 1.6z + 0.7 = 0$$

$$\det\left\{ z \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} h^2/2 \\ h \end{bmatrix} \begin{bmatrix} K_1 & K_2 \end{bmatrix} \right\}$$

$$det \begin{bmatrix} z - 1 + K_1 h^2/2 & -h + K_2 h^2/2 \\ hK_1 & z - 1 + hK_2 \end{bmatrix}$$

$$\left(z - 1 + \frac{h^2}{2} K_1\right)(z - 1 + hK_2) - hK_1 \left(\frac{h^2}{2} K_2 - h\right) = 0$$

$$z^2 - z + hK_2 z - z + 1 + hK_2 + \frac{h^2}{2} K_1 z - \frac{h^2}{2} K_1 + \frac{h^3}{2} K_1 K_2$$

$$-hK_1 \frac{h^2}{2} K_2 + h^2 K = 0$$

$$z^2 + \left[hK_2 + \frac{h^2}{2} K_1 - 2\right] z + \left[\frac{h^2}{2} K_1 - hK_2 + 1\right] = 0$$

Thus:
$$\left| \begin{array}{ll} hK_2 + \frac{h^2}{2} K_1 - 2 & = -1.6 \\ \frac{h^2}{2} K_1 - hK_2 + 1 & = 0.7 \end{array} \right.$$

and finally:
$$\left| \begin{array}{lll} K_1 & = \frac{0.1}{h^2} & = 10 \\ K_2 & = \frac{0.35}{h} & = 3.5 \end{array} \right.$$

5.2 Constructive design

The previous example highlighted the problems linked to the determination of the state feedback gain. It is indeed necessary to develop the determinant algebraically, to group the terms related to the different powers of z and finally to identify them. This method is impracticable for systems of an order that is superior to two. However, there is an interesting alternative, based on the exploitation of the controllability canonical form (section 4.3). By

this technique, the design of the controller is carried out by relying on an artificial state-space representation of the system to be controlled obtained by the following transformation:

$$\tilde{w} = P\tilde{x} \qquad \text{with:} \qquad P = \begin{bmatrix} e_n^T \Phi^{n-1} \\ e_n^T \Phi^{n-2} \\ \vdots \\ e_n^T I \end{bmatrix}$$

where e_n^T is the last row of the inverse of the controllability matrix G (4.16):

$$G = \begin{bmatrix} I\Gamma & \Phi\Gamma & \cdots & \Phi^{n-1}\Gamma \end{bmatrix}$$

Thus: $\tilde{w}(k+1) = \Phi_w \tilde{w}(k) + \Gamma_w \tilde{u}(k)$

with: $\Phi_w = P\Phi P^{-1} = \begin{bmatrix} -a_1 & -a_2 & \cdots & -a_n \\ 1 & 0 & \cdots & 0 \\ \vdots & & & \vdots \\ 0 & \cdots & 1 & 0 \end{bmatrix}$; $\Gamma_w = P\Gamma = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$

The a_i are the coefficients of the characteristic polynomial of the system to be controlled.

We choose: $\tilde{u} = -K'\tilde{w}$ $\qquad \tilde{w}(k+1) = (\Phi_w - \Gamma_w K')\tilde{w}(k)$

$$\Phi_{wcl} = \Phi_w - \Gamma_w K' = \begin{bmatrix} -a_1 - K_1' & -a_2 - K_2' & \cdots & -a_n - K_n' \\ 1 & 0 & \cdots & 0 \\ \vdots & & & \vdots \\ 0 & \cdots & 1 & 0 \end{bmatrix}$$

The characteristic polynomial of Φ_{wcl} that governs the dynamics of the closed loop system is:

$$\alpha_c(\lambda) = \lambda^n + (a_1 + K_1')\lambda^{n-1} + \cdots + (a_n + K_n') = 0$$

The choice of the desired eigenvalues in closed loop leads to the characteristic polynomial:

$$\alpha_c(\lambda) = \lambda^n + \alpha_1\lambda^{n-1} + \alpha_2\lambda^{n-2} + \cdots + \alpha_n = 0$$

The feedback gains of the artificial states are finally given by the simple expressions:

$$\left|\begin{array}{rcl} K'_1 & = & \alpha_1 - a_1 \\ K'_2 & = & \alpha_2 - a_2 \\ & \vdots & \\ K'_n & = & \alpha_n - a_n \end{array}\right.$$

As the real implementation of the control is based on the physical state variables, the gains have to be converted by an inverse transformation:

$$\tilde{u} = -K'\tilde{w} = -\underbrace{K'P}_{K}\tilde{x} = -K\tilde{x}$$

5.2.1 Summary of the method

- Compute numerically the coefficients a_i of the characteristic polynomial of the system according to the expression $\det(zI - \Phi) = 0$.
- Compute numerically the coefficients α_i of the characteristic polynomial as a function of the eigenvalues chosen in closed loop according to the expression $(\lambda - \lambda_1)(\lambda - \lambda_2)\cdots(\lambda - \lambda_n) = 0$.
- The difference between the coefficients of the two characteristic polynomials (open loop and closed loop) gives the gains K'.
- Bring these gains back to the real work domain by multiplying them by the matrix P which is constructed using the inverse of the controllability matrix.

5.3 Ackermann formula

The method described previously to determine the gain of the state feedback can be reduced to one single relation, called the Ackermann formula. Indeed,

$$K = K'P = \begin{bmatrix} \alpha_1 - a_1 & \alpha_2 - a_2 & \cdots & \alpha_n - a_n \end{bmatrix} P \tag{5.1}$$

The coefficients a_i are those of the characteristic polynomial of the system to be controlled $\alpha(\lambda)$. The α_i are those of the characteristic polynomial of the complete system in closed loop $\alpha_c(\lambda)$.

The right hand side of equation (5.1) can be split into two parts:

$$K = \begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_n \end{bmatrix} P + \underbrace{\begin{bmatrix} -a_1 & -a_2 & \cdots & -a_n \end{bmatrix}}_{e_n^T \Phi^n P^{-1}} P$$

The term above the curlybracket is the first row of the matrix $P\Phi P^{-1}$ defined in paragraph 4.3.2 and equal to $e_n^T \Phi^n P^{-1}$. The second member of this equation is then just $e_n^T \Phi^n$. The gain vector can then be obtained by:

$$K = \begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_n \end{bmatrix} \begin{bmatrix} e_n^T \Phi^{n-1} \\ e_n^T \Phi^{n-2} \\ \vdots \\ e_n^T I \end{bmatrix} + e_n^T \Phi^n$$

$$= e_n^T \Phi^n + \alpha_1 e_n^T \Phi^{n-1} + \alpha_2 e_n^T \Phi^{n-2} + \cdots + \alpha_n e_n^T I$$

$$= \underbrace{e_n^T}_{①} \left[\underbrace{\Phi^n + \alpha_1 \Phi^{n-1} + \alpha_2 \Phi^{n-2} + \cdots + \alpha_n I}_{\alpha_c(\Phi)} \right]$$

It is the product of the last row of the inverse of the controllability matrix ① and of a matrix polynomial whose coefficients are those of the characteristic polynomial chosen in closed loop, so $\alpha_c(\Phi)$. Thus:

$$K = \begin{bmatrix} 0 & \cdots & 0 & 1 \end{bmatrix} G^{-1} \alpha_c(\Phi) \tag{5.2}$$

$$K = \begin{bmatrix} 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} g & \Phi g & \cdots & \Phi^{n-1} g \end{bmatrix}^{-1} \alpha_c(\Phi) \tag{5.3}$$

If G is invertible, the system is controllable and the gain K can be computed.

5.3.1 Example: Double integrator

We would like to impose in closed loop the following dynamics:

$$\alpha_c(\lambda) = \det[\lambda I - \Phi_{cl}] = \lambda^2 - 1.6\lambda + 0.7 = 0$$

So with $\alpha_1 = -1.6$ and $\alpha_2 = 0.7$, we get:

$$G = \begin{bmatrix} g & \Phi g \end{bmatrix}, \text{ so: } G = \begin{bmatrix} h^2/2 & 3h^2/2 \\ h & h \end{bmatrix}$$

$$G^{-1} = -\frac{1}{h^3}\begin{bmatrix} h & -3h^2/2 \\ -h & h^2/2 \end{bmatrix} = \frac{1}{h^2}\begin{bmatrix} -1 & 3h/2 \\ 1 & -h/2 \end{bmatrix}$$

$$K = \begin{bmatrix} K_1 & K_2 \end{bmatrix} = \frac{1}{h^2}\begin{bmatrix} 0 & 1 \end{bmatrix}\begin{bmatrix} -1 & 3h/2 \\ 1 & -h/2 \end{bmatrix}\alpha_c(\Phi)$$

$$\alpha_c(\Phi) = \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix}^2 - 1.6\begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix} + 0.7\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 2h \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} -1.6 & -1.6h \\ 0 & -1.6 \end{bmatrix} + \begin{bmatrix} 0.7 & 0 \\ 0 & 0.7 \end{bmatrix}$$

$$= \begin{bmatrix} 0.1 & 0.4h \\ 0 & 0.1 \end{bmatrix}$$

$$K = \frac{1}{h^2}\begin{bmatrix} 1 & -\frac{h}{2} \end{bmatrix}\begin{bmatrix} 0.1 & 0.4h \\ 0 & 0.1 \end{bmatrix} = \frac{1}{h^2}\begin{bmatrix} 0.1 & 0.4h - \frac{0.1}{2}h \end{bmatrix} = \begin{bmatrix} \frac{0.1}{h^2} & \frac{0.35}{h} \end{bmatrix} = \begin{bmatrix} 10 & 3.5 \end{bmatrix}$$

5.4 Imposed behaviors in closed loop

5.4.1 Imposition of a dead-beat response

Search the gain K that allows to bring a disturbed system back to its equilibrium point in n sampling periods.

Disturbance:	$x(0) = x_0$	$(\tilde{x}(0) = x_0)$
Bring back to:	$x(n) = 0$	$(\tilde{x}(n) = 0 \rightarrow x(n) = \bar{x})$

The sequence of necessary control samples is given by (4.3):

$$\begin{bmatrix} u(n-1) \\ u(n-2) \\ \vdots \\ u(0) \end{bmatrix} = \underbrace{\begin{bmatrix} \Gamma & \Phi\Gamma & \cdots & \Phi^{n-1}\Gamma \end{bmatrix}^{-1}}_{G^{-1}} \underbrace{\begin{bmatrix} x(n) - \Phi^n x_0 \end{bmatrix}}_{-\Phi^n x_0}$$

In our case:

By multiplying both sides by $\begin{bmatrix} 0 & \cdots & 0 & 1 \end{bmatrix}$, we obtain $u(0)$:

$$u(0) = \begin{bmatrix} 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} u(n-1) \\ u(n-2) \\ \vdots \\ u(0) \end{bmatrix} = - \begin{bmatrix} 0 & \cdots & 0 & 1 \end{bmatrix} G^{-1} \Phi^n x_0$$

In addition, we have chosen: $u = -Kx$ so $u(0) = -Kx(0)$

$$-Kx(0) = - \begin{bmatrix} 0 & \cdots & 0 & 1 \end{bmatrix} G^{-1} \Phi^n x_0$$

from which:

$$K = \begin{bmatrix} 0 & \cdots & 0 & 1 \end{bmatrix} G^{-1} \Phi^n$$

According to Ackermann, this corresponds to $\alpha_c(\Phi) = \Phi^n$. Consequently, $\alpha_c(\lambda) = \lambda^n = 0$. All the poles are in the center of the unit circle. However, one should underline that the resulting control amplitude to get such a quick convergence may have a high amplitude.

5.4.2 Imposition of a dominant pole

The step response of a first-order system is chosen as closed-loop response.

$$G(s) = \frac{K}{1+s\tau}$$

The dominant pole $s_1 = -1/\tau$ is selected. The other poles are placed in the center of the unit circle. The corresponding discrete poles are given by:

$$z_i = e^{s_i h}$$

The corresponding time response is illustrated in Fig. 5.1 and corresponds to $y(t) = U[1-e^{-t/\tau}]$

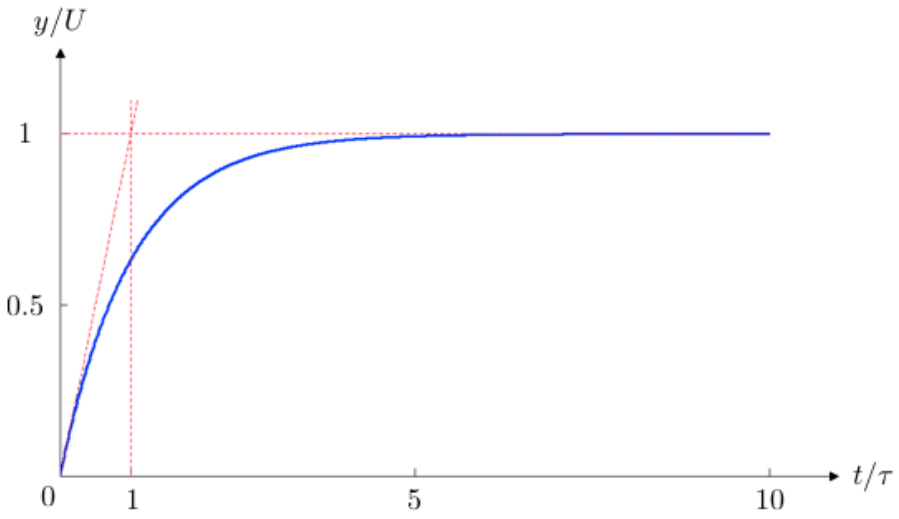


Figure 5.1 – Step response of a first-order system.

5.4.3 Imposition of a pair of dominant complex conjugate poles

The step response of a second-order system is chosen as closed-loop response.

$$G(s) = \frac{K}{\tau^2 s^2 + 2\zeta\tau s + 1} \quad \text{with} \quad \omega_0 = \frac{1}{\tau} \quad \text{and} \quad s_{1,2} = -\frac{1}{\tau}(\zeta \pm \sqrt{\zeta^2 - 1})$$

The two poles $s_{1,2}$ are imposed. The other poles are placed at the center of the unit circle.

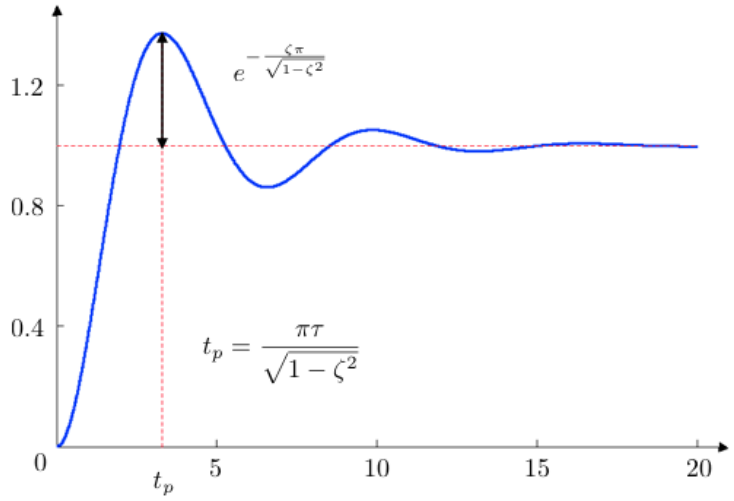


Figure 5.2 – Step response of a second-order system.

5.4.4 Bessel filter

The Bessel filter is characterized by a small phase shift, inducing limited signal distortion. The cutoff frequency ω_n is the closed loop bandwidth.

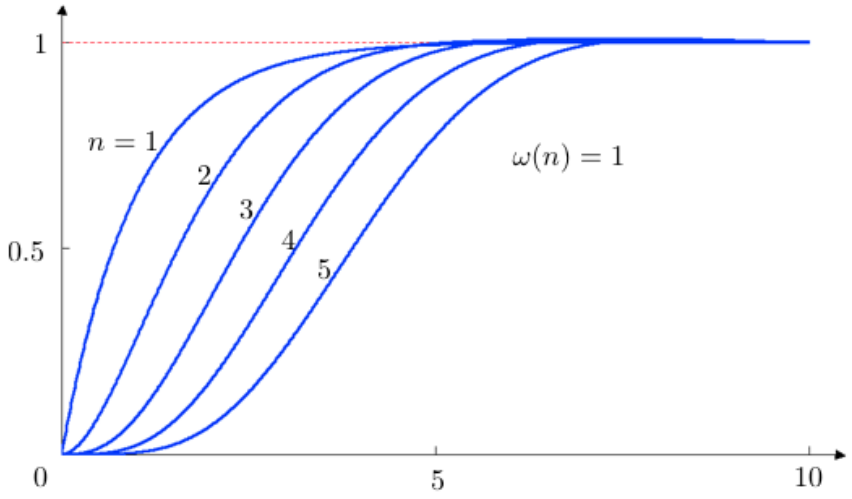


Figure 5.3 – Bessel filter.

$$
\begin{aligned}
H_N(s) &= \frac{K}{\prod(s-s_i)} = \frac{K}{B_N(s)} B_N(0) \\
B_0(s) &= 1 \\
B_1(s) &= s+1 \\
B_N(s) &= (2N-1)B_{N-1}(s) + s^2 B_{N-2}(s)
\end{aligned}
$$

n	K	poles $s_i; z_i = e^{s_i h}$
1	1	-1
2	1	$-0.866 \pm 0.5i$
3	1	$-0.7456 \pm 0.7114i, -0.9416$
4	1	$-0.9048 \pm 0.2709i, -0.6572 \pm 0.8302i$
5	1	$-0.5906 \pm 0.9072i, -0.8516 \pm 0.4437i, -0.9264$

5.5 Implementation of the state feedback

Unlike the design of a classic PID control by the Ziegler-Nichols method, the design of a state feedback requires the knowledge of the model of the system to be controlled. Once this model is available, it can be exploited in several ways to improve the dynamic behavior of the installation. It is for example natural to use the model in order to elaborate a feedforward control, able to bring in open loop the system to its operating point. Thereby, the addition of the control is only useful to compensate the effect of the expected disturbances, or to the correction of the imprecision of the feedforward control, consecutive to the inevitable inaccuracy of the model (structure or value of the parameters). Thus, the traditional block diagram of a state feedback is usually represented as in Figure 5.4.

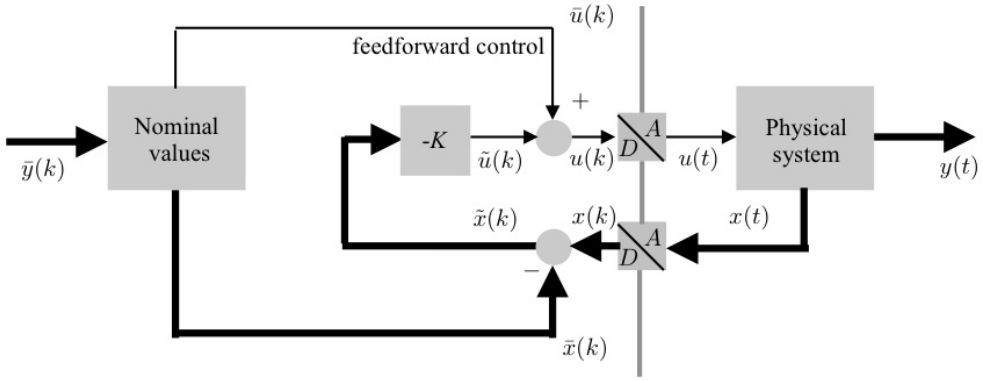


Figure 5.4 – Block diagram of a state control.

With such a structure, the disturbance rejection and the trajectory tracking are simultaneously enforced. Indeed, the feedback maintains the state around its nominal value, while the nominal values are generated according to the desired outputs.

If the physical system is nonlinear, the dynamical model exploited for the design of the control $\tilde{u}(k) = -K\tilde{x}(k)$ is:

$$\left|\begin{array}{rl} \tilde{x}(k+1) & = \Phi\tilde{x}(k) + g\tilde{u}(k) \\ \tilde{y}(k) & = C\tilde{x}(k) + D\tilde{u}(k) \end{array}\right.$$

In this nonlinear case, the nominal values have to be constant so that the matrices Φ, g, C and D are constant too. It is a tracking around an operating point and not the tracking of nominal trajectories. In the latter case where the model is evolutive, the stability is not necessarily guaranteed by a choice of eigenvalues in the unit circle. It is guaranteed, however, if the operating point changes slowly enough.

The whole part of the diagram situated on the left of the AD & DA converters is implemented as numerical algorithm on a micro-controller. Note that the outputs are controlled in open loop, only the states are controlled in closed loop. This can induce problems if the matrices C and D are poorly known. However, in practice, D is generally zero and the state variables can often be chosen so that the matrix C shows only a few elements equal to one on the diagonal (the outputs are thus a subset of the state variables). In addition, the state is often reconstructed by an observer from the outputs and inputs, as it will be shown in the next chapter.

With such a control structure, selecting a setpoint for the output quantities is done by choosing their nominal values. The controller brings thus the state of the system back to its nominal state, that is to say to its setpoint. The previous block diagram can be rearranged in a more traditional form to highlight the setpoint.

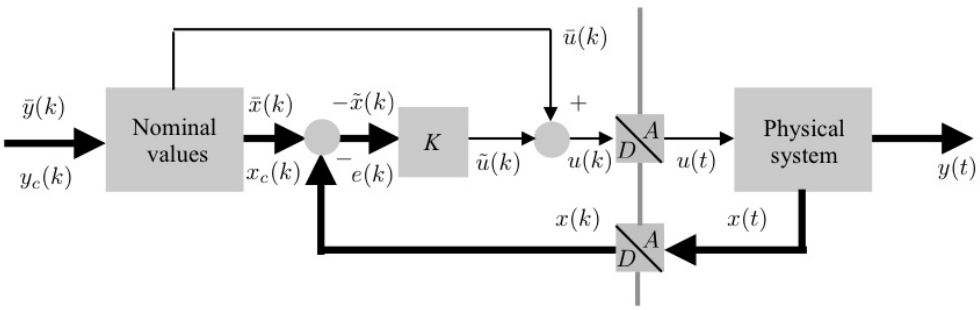


Figure 5.5 – Block diagram of a state feedback in traditional form.

5.6 Example: Heating chamber

Consider the heating chamber represented below:

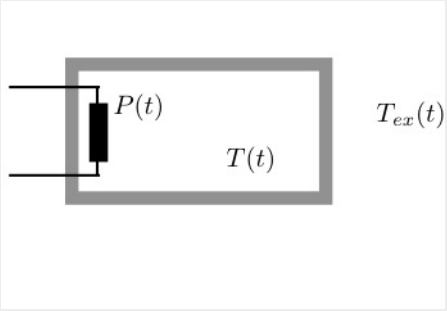


Figure 5.6 – Schematic representation of the heating chamber.

The input quantity is the heating power $u(t) = P(t)$. The output is the temperature in the chamber $y(t) = T(t)$. The external temperature $q(t) = T_{ex}(t)$ that varies slowly is considered as a disturbance.

5.6.1 Modeling

The continuous model is: $mc\dot{T}(t) = P(t) - R[T(t) - T_{ex}(t)]$

For $mc = 1$ and $x(t) = T(t)$, we get:

$$\left|\begin{array}{ll} \dot{x}(t) & = -Rx(t) + u(t) + Rq(t) \\ y(t) & = x(t) \end{array}\right.$$

5.6.2 Solution by defining an operating point

Consider an operating point such that $\bar{y} = y_c$ and suppose that the external temperature q is constant. By setting $\dot{\bar{x}} = 0$ in the state model, we get: $\bar{x} = \bar{y} = y_c$ and $\bar{u} = R(y_c - q)$. The model expressed in deviations around this operating point is:

$$\left|\begin{array}{l} \dot{\tilde{x}}(t) = -R\tilde{x}(t) + \tilde{u}(t) \\ \tilde{y}(t) = \tilde{x}(t) \end{array}\right.$$

The matrices of the discrete model are $\Phi = e^{Ah} = e^{-Rh} \equiv \varphi$ and:

$$\Gamma = \left\{\int_0^h e^{-R\eta} d\eta\right\} B = -\frac{1}{R}(e^{-Rh} - 1) = \frac{1}{R}(1 - p) = \gamma$$

Finally, the discrete state model expressed in deviations is:

$$\left|\begin{array}{ll} \tilde{x}(k+1) & = \varphi\tilde{x}(k) + \gamma\tilde{u}(k) \\ \tilde{y}(k) & = \tilde{x}(k) \end{array}\right.$$

By choosing one single eigenvalue in closed loop: $\alpha_c(\lambda) = \lambda + \alpha_1 = 0$, the feedback gain is a scalar:

$$K = [1][\gamma]^{-1}\alpha_c(\varphi) = \frac{\varphi + \alpha_1}{\gamma}$$

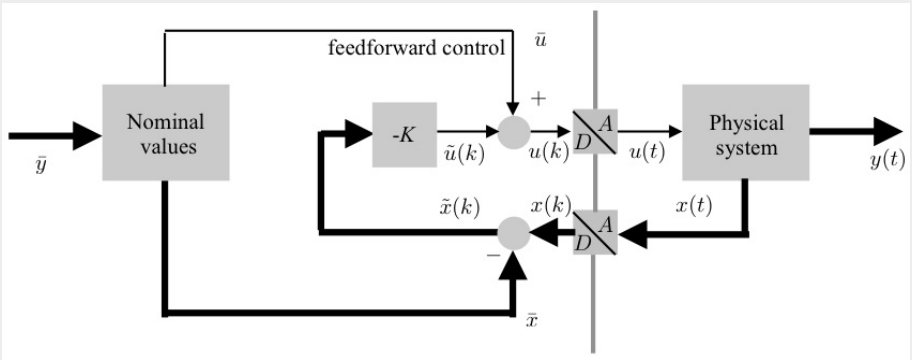


Figure 5.7 – Block diagram.

5.7 Exercises

5.7.1 State feedback

Problem

Consider the system represented by the transfer function:

$$G(s) = \frac{10}{s^2 + 0.18s + 9}$$

1. Determine the discrete state-space model of this system in presence of AD and DA converters with $h = 0.1$ s.

2. Determine the state feedback K so as to have in closed loop a dominant behavior characterized by $\zeta = 0.5$ and $\omega_0 = 8$ rad/s.

3. Repeat the design of the controller, but place all the eingenvalues at the center of the unit circle. It is a deadbeat controller. The latter brings indeed the system back to the operating point in a finished number of sampling periods. How many?

Solution

The differential equation that corresponds to the given transfer function $G(s) = Y(s)/U(s)$ is the following: $\ddot{y}(t) + 0.18\dot{y}(t) + 9y(t) = 10u(t)$.

By choosing as a state variable $x_1 = y$ and $x_2 = \dot{y}$, this equation can be arranged in the form of a state-space model.

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -9 & -0.18 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 10 \end{bmatrix} u(t)$$
$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$$

The discrete-time state-space model can be obtained by using the Matlab c2d function:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 0.9556 & 0.0976 \\ -0.8786 & 0.9380 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0.0493 \\ 0.9763 \end{bmatrix} u(k)$$
$$y(k) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix}$$

State feedback:

The specifications $\zeta = 0.5$ and $\omega_0 = 8$ rad/s correspond to the following continuous poles:

$$s_{1,2} = -\omega_0 \left(\zeta + \sqrt{\zeta^2 - 1} \right) = -4 \pm 6.93\, j$$

In the discrete-time domain, these poles are: $z_{1,2} = e^{hs_{1,2}} = 0.5158 \pm 0.4281\, j$. Ackermann gives us the corresponding gain for the state feedback: $K = \begin{bmatrix} 3.35 & 0.714 \end{bmatrix}$.

The discrete-time responses obtained with these gains and from the initial conditions $x_1(0) = x_2(0) = 0.5$ are the following:

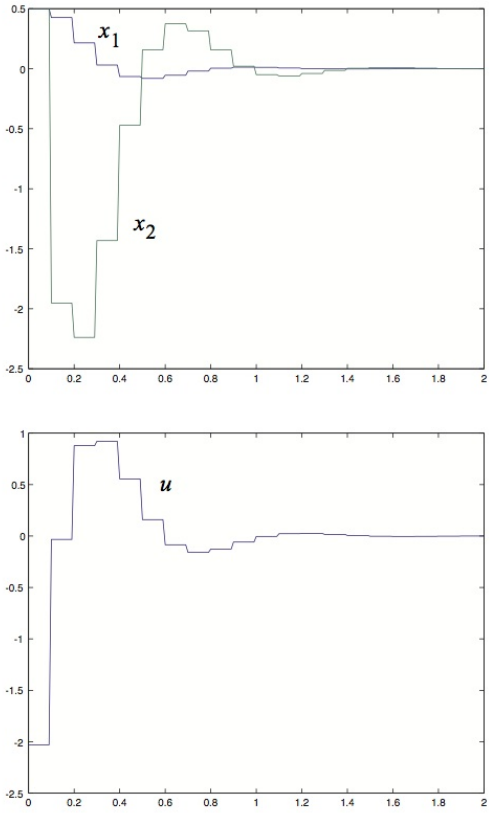


Figure 5.8 – State and control variables.

In this case, several sampling periods are required to stabilize the system.

By fixing the discrete poles (eigenvalues) at the center of the unit circle, Ackermann gives: $K = \begin{bmatrix} 9.27 & 1.47 \end{bmatrix}$.

The discrete-time responses obtained with these gains and from the initial conditions $x_1(0) = x_2(0) = 0.5$ are the following:

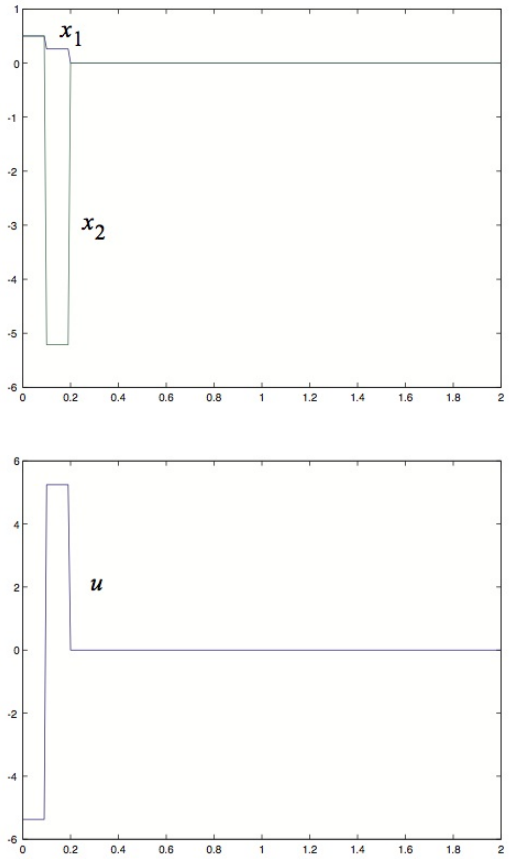


Figure 5.9 – State and input variables.

In this case, two sampling periods are required, which is the minimum achievable number corresponding to the order of the system.

6 State-space observer

6.1 Observer

It appears from the previous chapter that the full state of the system to be controlled has to be available to implement a state feedback. However, some variables are sometimes unmeasurable, or not measured because of the costs of the sensors.

6.1.1 Observation approach

To overcome the unavailability of some state variables, a technique able to observe the missing quantities by means of the available information has to be elaborated. This information is the input and output signals, and the model of the system. To make the presentation more simple, only the systems with a single output will be considered here (SISO or MISO). In addition, we will assume that the input does not effect directly the output, which is always the case for sampled continuous-time systems. Thus, the model is in the form:

$$\left|\begin{array}{rcl} x(k+1) & = & \Phi x(k) + \Gamma u(k) \\ y(k) & = & c^T x(k) \end{array}\right.$$

The basic idea for the observation (or the estimation) of the missing states lies in the exploitation of the available information. The computer or the micro-controller exploited as an observer will perform the simulation of the discrete-time model of the physical system using the known inputs. The simulation consists of a simple successive evaluation of the state-space equation from the initial conditions. The model being an approximation of the reality, the result of the simulation is just a simple estimation of the real state $x(k)$ of the system. To highlight this difference, the estimation is spotted by a hat, $\hat{x}(k)$.

This estimation can then be exploited in open loop in a context of measurement, or by a controller to elaborate a state feedback (Fig. 6.1).

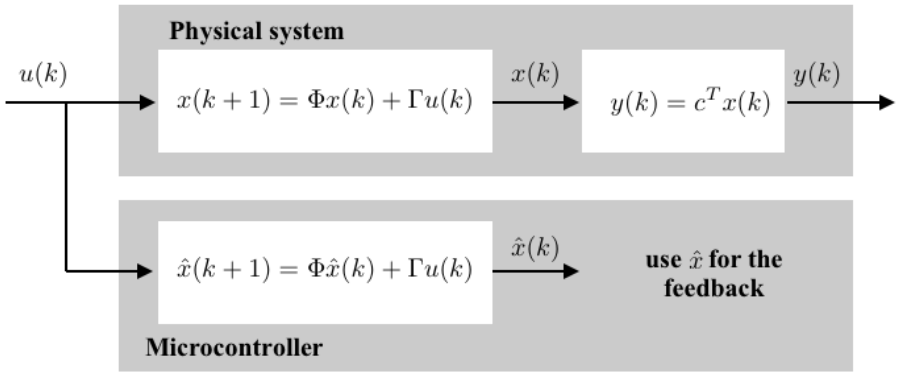


Figure 6.1 – Principle of state estimation.

Although this approach is tempting from a conceptual point of view, it suffers from two major problems. First, the initial conditions $\hat{x}(0)$ of the state are generally unknown; secondly, the measurements $y(k)$ which constitute a rich information source are not exploited. It is an open-loop estimation.

An analyzis of the estimation error $\delta(k) = \hat{x}(k) - x(k)$ allows to highlight an additional problem. Indeed, the error at time $k+1$ has the value of:

$$\delta(k+1) = \underbrace{[\Phi\hat{x}(k) + \Gamma u(k)]}_{\hat{x}(k+1)} - \underbrace{[\Phi x(k) + \Gamma u(k)]}_{x(k+1)} = \Phi[\hat{x}(k) - x(k)]$$

In other words:

$$\delta(k+1) = \Phi\delta(k)$$

This error equation is in the form of a linear discrete state-space model. The considerations obtained previously on the stability of such systems can then be exploited to characterize the convergence of the estimation error. It is interesting to note that the dynamics of this error is the same as the one of the observed system, that is to say that it is conditioned by the eigenvalues of the matrix Φ. The error diminishes with the same dynamics as the system evolutes. If the system is unstable, the estimation diverges.

6.1.2 Leuenberger observer

The solution to the problems highlighted in the previous paragraph comes from the exploitation of the outputs of the real system. The idea is to compare the available measurements and the outputs provided by the estimator. The resulting error corresponds to a measurement of the quality of the estimation. This error can be exploited to correct the state estimates. The block diagram associated with the proposed technique is represented in Figure 6.2. The estimation of the output comes from the original model of the system and is given by the

expression $\hat{y}(k) = c^T \hat{x}(k)$.

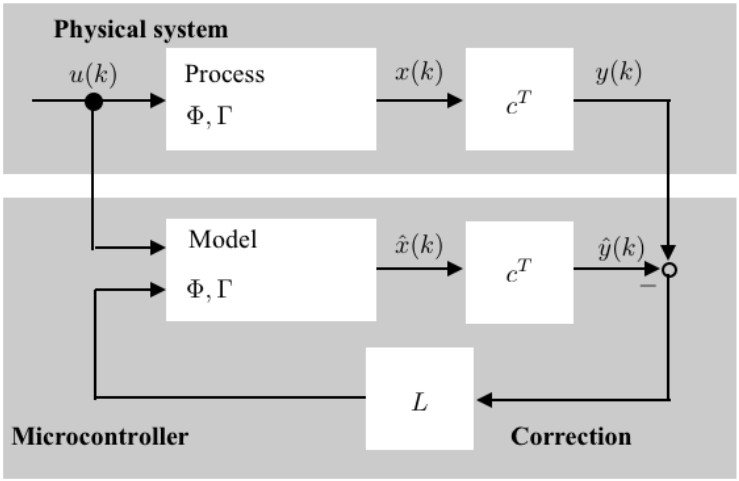


Figure 6.2 – Structure of the Leuenberger state observer.

A feedback of the estimation error is introduced. From an analytical point of view, the dynamic behavior of the observer results in the difference equation:

$$\hat{x}(k+1) = \underbrace{\Phi\hat{x}(k) + \Gamma u(k)}_{\substack{\text{Dynamics assumed} \\ \text{to be known}}} + \underbrace{L[y(k) - \hat{y}(k)]}_{\substack{\text{Correction as a function} \\ \text{of the estimation error}}}$$

This equation of the observer can be written:

$$\hat{x}(k+1) = \Phi\hat{x}(k) + \Gamma u(k) + L[y(k) - c^T \hat{x}(k)] \tag{6.1}$$

L is a column vector of dimension n. Since there is only one measured output $y(k)$, the correction term is in the form:

$$\underbrace{\begin{bmatrix} L_1 \\ \vdots \\ L_n \end{bmatrix}}_{n\times 1} \underbrace{\left[y(k) - \hat{y}(k) \right]}_{1\times 1}$$

The estimation error of the state can now be evaluated, by a similar approach to the one

adopted in the previous paragraph:

$$\delta(k+1) = \hat{x}(k+1) - x(k+1)$$
$$\delta(k+1) = \Phi[\hat{x}(k) - x(k)] + L[\underbrace{c^T x(k)}_{y(k)} - c^T \hat{x}(k)]$$
$$\delta(k+1) = \underbrace{(\Phi - Lc^T)}_{\Phi_e}\delta(k)$$

It is a homogeneous state-space equation (without excitation). Its solution converges to zero if the eigenvalues of Φ_e are inside the unit circle.

There are n estimation errors of the n states that can be dynamically controlled by the n gains L_i of the observer. Consequently, the dynamic convergence of the estimation can be chosen freely, provided the system is observable (see section 4.6). If the eigenvalues β_i of Φ_e are all imposed in the unit circle, the estimation error $\delta(k)$ converges to zero, regardless of the initial conditions $\hat{x}(0)$. The characteristic equation of Φ_e is:

$$\alpha_e(\lambda) = \det(\lambda I - \Phi_e) = (\lambda - \beta_1)(\lambda - \beta_2)\cdots(\lambda - \beta_n) = 0$$

Note that one single measurement $y(k)$ allows to estimate n states. In addition, the dynamics of the observer being that of an algorithm, there is no physical limitation to the chosen speed of convergence, nor to the amplitude of the signals in the estimator. These are only numbers stored in memory. The eigenvalues are consequently chosen in order to obtain the fastest dynamics possible, or at least a dynamics quicker than that of the controller. We will see later that this objective is reached if all the eigenvalues are placed at the center of the unit circle. Too fast attempted convergence can however make the system sensitive to noise, or lead to bad transients.

As in the case of the state feedback, the quality of the state-space estimation depends on the quality of the available model and on the precision of the sensors.

6.1.3 Example: Double integrator

Consider the discrete-time model of the double integrator:

$$x(k+1) = \underbrace{\begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix}}_{\Phi} x(k) + \begin{bmatrix} h^2/2 \\ h \end{bmatrix} u(k)$$
$$y(k) = \underbrace{\begin{bmatrix} 1 & 0 \end{bmatrix}}_{c^T} x(k)$$

We impose $\alpha_e(\lambda) = (\lambda - \beta_1)(\lambda - \beta_2)$, with $\beta_{1,2} = 0.4 \pm 0.4\,j$. Thus:

$$\alpha_e(\lambda) = \lambda^2 - 0.8\lambda + 0.32 = 0$$
$$\det[\lambda I - \Phi_e] = \det[\lambda I - \Phi + Lc^T] = 0$$
$$\det\left\{\begin{bmatrix}\lambda & 0\\ 0 & \lambda\end{bmatrix} - \begin{bmatrix}1 & h\\ 0 & 1\end{bmatrix} + \begin{bmatrix}L_1\\ L_2\end{bmatrix}\begin{bmatrix}1 & 0\end{bmatrix}\right\}$$
$$\det\left\{\begin{bmatrix}\lambda - 1 & -h\\ 0 & \lambda - 1\end{bmatrix} + \begin{bmatrix}L_1 & 0\\ L_2 & 0\end{bmatrix}\right\} = \det\begin{bmatrix}\lambda - 1 + L_1 & -h\\ L_2 & \lambda - 1\end{bmatrix}$$
$$(\lambda - 1 + L_1)(\lambda - 1) + L_2 h = \lambda^2 - \lambda - \lambda + 1 + L_1\lambda - L_1 + L_2 h = 0$$

For $h = 0.1$ s, we get: $\lambda^2 + \underbrace{(L_1 - 2)}_{-0.8}\lambda + \underbrace{(L_2 h - L_1 + 1)}_{0.32} = 0$

The gains of the observer are finally:

$$L_1 = 1.2; L_2 = \frac{0.52}{h} = 5.2$$

6.2 Constructive design

The previous example highlighted the problems linked to the evaluation of the gain of the state-space observer. It is indeed necessary to develop the determinant in an algebraic manner, and to regroup the terms that are relative to the powers of λ, and finally to identify them. This method is not applicable for systems of an order that is higher than two. However, there is an interesting alternative, based on the exploitation of the observability canonical form (4.4.2). By this technique, the design of the observer is carried out on the basis of an artificial state-space representation of the system to be observed:

$$\left|\begin{array}{rcl} x(k+1) & = & \Phi x(k) + \Gamma u(k)\\ y(k) & = & c^T x(k)\end{array}\right.$$

The switch to the artificial representation is obtained by the transformation:

$$x = Rv \qquad \text{with:} \qquad R = \begin{bmatrix}\Phi^{n-1}e_n & \cdots & \Phi e_n & Ie_n\end{bmatrix}$$

where e_n is the last column of the inverse of the observability matrix Q (4.4.2).

Thus: $$v(k+1) = \Phi_v v(k) + \Gamma_v u(k)$$
$$y(k) = \underbrace{c^T R}_{c_v^T} v(k) = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix} v(k)$$

with: $$\Phi_v = R^{-1}\Phi R = \begin{bmatrix} -a_1 & 1 & 0 & \cdots & 0 \\ -a_2 & 0 & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ -a_{n-1} & 0 & \cdots & 0 & 1 \\ -a_n & 0 & \cdots & 0 & 0 \end{bmatrix}$$

The a_i are the coefficients of the characteristic polynomial of the system to be observed.

Rather than constructing an observer of the physical state x of the system, an observer of the artificial state v is implemented. Its dynamics is governed by the following difference equation:

$$\hat{v}(k+1) = \Phi_v \hat{v}(k) + \Gamma_v u(k) + L'[y(k) - \hat{y}(k)]$$

The estimation error of v is thereby governed by:

$$\delta_v(k+1) = (\Phi_v - \underbrace{L' c_v^T})\delta_v(k)$$
$$\begin{bmatrix} L'_1 \\ \vdots \\ L'_n \end{bmatrix} \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix} = \left[\begin{array}{c|c} L'_1 & \\ \vdots & \mathbf{0} \\ L'_n & \end{array}\right]$$

The dynamics of the error depends on the eigenvalues of:

$$\Phi_v - L' c_v^T = \begin{bmatrix} -a_1 - L'_1 & 1 & 0 & \cdots & 0 & 0 \\ -a_2 - L'_2 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & & & \vdots & \vdots \\ -a_{n-1} - L'_{n-1} & 0 & & & 0 & 1 \\ -a_n - L'_n & 0 & \cdots & \cdots & 0 & 0 \end{bmatrix} \equiv \Phi_{ve}$$

This matrix corresponds to the observability canonical form. Hence, the characteristic polynomial of $\Phi_v - L' c_v^T$ is written by analogy:

$$\lambda^n + (a_1 + L'_1)\lambda^{n-1} + \cdots + (a_n + L'_n) = 0 \tag{6.2}$$

The relation between the state variables v and x being static (algebraic linear transformation), the dynamics of the estimation error of v will be the same as that of x. Consequently, the eigenvalues of Φ_{ve} and Φe are the same. The matrices Φ_{ve} and Φ_e are called similar.

The characteristic polynomial (6.2) can be compared to the one chosen to govern the dynamics of the estimation error of x:

$$\alpha_e(\lambda) = \lambda^n + \alpha_1 \lambda^{n-1} + \alpha_2 \lambda^{n-2} + \cdots + \alpha_n = 0$$

The gains will be given by the relations:

$$\left|\begin{array}{rcl} L'_1 & = & \alpha_1 - a_1 \\ L'_2 & = & \alpha_2 - a_2 \\ & \vdots & \\ L'_n & = & \alpha_n - a_n \end{array}\right.$$

To go back to the physical quantities, the inverse state-space transformation is applied in the observer equation:

$$\begin{aligned} \hat{v}(k+1) &= \Phi_v \hat{v}(k) + \Gamma_v u(k) + L'[y(k) - \hat{y}(k)] \\ R^{-1}\hat{x}(k+1) &= \Phi_v R^{-1}\hat{x}(k) + \Gamma_v u(k) + L'[y(k) - \hat{y}(k)] \\ \hat{x}(k+1) &= \underbrace{R\Phi_v R^{-1}}_{\Phi}\hat{x}(k) + \underbrace{R\Gamma_v}_{\Gamma} u(k) + \underbrace{RL'}_{L}[y(k) - \hat{y}(k)] \end{aligned}$$

Thus: $L = RL'$

Only numerical operations are required to obtain the gains of the observer by this methodology based on the observability canonical form.

6.2.1 Ackermann formula for the observer

The procedure described previously to obtain the gain L of the observer can be reduced to one single relation, called the Ackermann formula for the observer. From the previous paragraph,

the gain can be written:

$$L = RL' = R\begin{bmatrix}\alpha_1 - a_1\\ \alpha_2 - a_2\\ \vdots\\ \alpha_n - a_n\end{bmatrix} = R\begin{bmatrix}\alpha_1\\ \alpha_2\\ \vdots\\ \alpha_n\end{bmatrix} + \underbrace{R\begin{bmatrix}-a_1\\ -a_2\\ \vdots\\ -a_n\end{bmatrix}}_{R^{-1}\Phi^n e_n}$$

The coefficients a_i are those of the characteristic polynomial of the system to be observed $\alpha(\lambda)$. The α_i are those of the characteristic polynomial $\alpha_e(\lambda)$ of the matrix Φ_e which governs the estimation error. The term under the curly braces is the first column of the matrix $R^{-1}\Phi R$ (4.4.2). The second member of this equation is consequently reduced to the factor $\Phi^n e_n$.
The gain vector can be obtained by:

$$\begin{aligned} L &= \begin{bmatrix}\Phi^{n-1}e_n & \cdots & \Phi e_n & Ie_n\end{bmatrix}\begin{bmatrix}\alpha_1\\ \alpha_2\\ \vdots\\ \alpha_n\end{bmatrix} + \Phi^n e_n \\ &= \Phi^n e_n + \alpha_1\Phi^{n-1}e_n + \alpha_2\Phi^{n-2}e_n + \cdots + \alpha_n Ie_n \\ &= \underbrace{(\Phi^n + \alpha_1\Phi^{n-1} + \alpha_2\Phi^{n-2} + \cdots + \alpha_n I)}_{\alpha_e(\Phi)} e_n \end{aligned}$$

The vector e_n is the last column of the inverse of the observability matrix Q. It can be expressed in the following way:

$$e_n = Q^{-1}\begin{bmatrix}0\\ \vdots\\ 0\\ 1\end{bmatrix} = \begin{bmatrix}c^T\\ c^T\Phi\\ \vdots\\ c^T\Phi^{n-1}\end{bmatrix}^{-1}\begin{bmatrix}0\\ \vdots\\ 0\\ 1\end{bmatrix}$$

Finally:

$$L = \alpha_e(\Phi)\begin{bmatrix}c^T\\ c^T\Phi\\ \vdots\\ c^T\Phi^{n-1}\end{bmatrix}^{-1}\begin{bmatrix}0\\ \vdots\\ 0\\ 1\end{bmatrix}$$

It is the Ackermann formula for the state observer. It only exploits elements of the model of

the system to be observed and no longer includes transformation matrix.

In MATLAB, the observability matrix is obtained by means of the command Q = obsv(Φ, c^T).

6.3 Dead-beat observer

Consider the discrete-time system described by:

$$\left|\begin{array}{rcl} x(k+1) & = & \Phi x(k) + \Gamma u(k) \\ y(k) & = & c^T x(k) \end{array}\right.$$

and the associated observer:

$$\hat{x}(k+1) = \Phi\hat{x}(k) + \Gamma u(k) + L[y(k) - c^T\hat{x}(k)]$$

The equation that governs the dynamics of the estimation error $\delta(k) = \hat{x}(k) - x(k)$ is:

$$\delta(k+1) = (\Phi - Lc^T)\delta(k) \tag{6.3}$$

The initial estimation is generally chosen as zero ($\hat{x}(0) = 0$), thus $\delta(0) = -x_0$.

The observer is called a dead-beat observer if the estimation error is canceled after n sampling periods:

$$\delta(n) = (\Phi - Lc^T)^n\delta(0) = 0$$

If the characteristic polynomial $\alpha_e(\lambda)$ of $(\Phi - Lc^T)$ in equation (6.3) is chosen equal to λ^n, i.e. $(\alpha_1, \alpha_2, \cdots, \alpha_n = 0)$, the gains of the observer of the system represented in the observability canonical form are $L'_i = -a_i$. Thus, the matrix that describes the dynamics of the estimation error of the artificial system is:

$$\Phi_v - L'c_v^T = \begin{bmatrix} -a_1 - L'_1 & 1 & \cdots & 0 \\ -a_2 - L'_2 & 0 & & \vdots \\ \vdots & \vdots & & 1 \\ -a_n - L'_n & 0 & \cdots & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ 0 & 0 & & \vdots \\ \vdots & \vdots & & 1 \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

It is a matrix called “nilpotent" whose property is a shift of the diagonal at each multiplication by itself.

For example:

$$M_{nil} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}; M_{nil}^2 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}; M_{nil}^3 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

and $M_{nil}^4 = 0$.

Generally, for a nilpotent matrix of dimension n:

$$M_{nil}^n = 0.$$

Thus:

$$\begin{aligned} &\delta(n) = (\Phi - Lc^T)^n \delta(0) = (R\Phi_v R^{-1} - RL'c_v^T R^{-1})^n \delta(0) \\ &= [R(\Phi_v - L'c_v^T)R^{-1}]^n \delta(0) \\ &= R^n \underbrace{(\Phi_v - L'c_v^T)^n}_{0} (R^{-1})^n \delta(0) \end{aligned}$$

Hence, the choice of $\alpha_e(\lambda) = \lambda^n$ leads to the cancelling of the estimation error in n sampling periods.

6.4 Full state feedback structure

To implement a state feedback, it is necessary to know the state of the system to be controlled. When the measurement of the state variables is too expensive, or even impossible, a state-space observer has to be introduced. The combination and the dynamic interactions of the feedback loop and of the observer are studied in this section, for the general case of a non-linear continuous-time system. The model used for the design of the full digital control is in the form of a linearized and discretized model (light-grey box of figure 6.3).

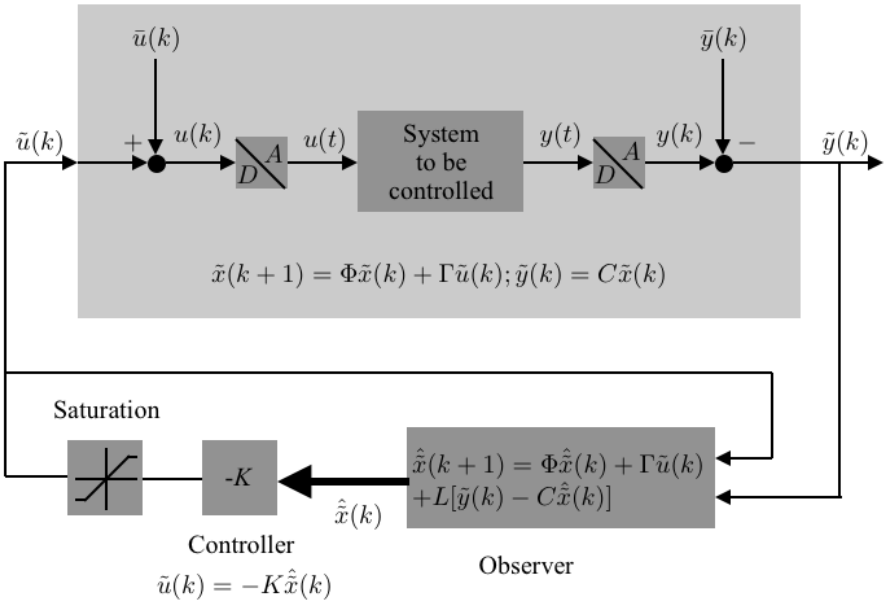


Figure 6.3 – Full state feedback.

The estimation of the output is not computed explicitly in the observer, however, it appears in the form of the expression $C\hat{\tilde{x}}(k)$.

If the control signal is saturated under the effect of the actuator before being applied to the physical system, the control input exploited by the observer has to be saturated too. For this reason, a block of saturation having the same static characteristic as the actuator is inserted in the feedback loop.

By omitting the tilde for readability, the complete system represented previously is described by:

$$\hat{x}(k+1) = \Phi\hat{x}(k) + \Gamma u(k) + L[y(k) - C\hat{x}(k)] \quad \text{Observer}$$

$$x(k+1) = \Phi x(k) + \Gamma u(k) \quad \text{State equation of the system}$$

$$u(k) = -K\hat{x}(k) \quad \text{State feedback}$$

$$y(k) = Cx(k) \quad \text{output equation of the system}$$

The substitution of $u(k)$ which is neither a state variable, nor an input or an output of the

complete system, leads to:

$$\hat{x}(k+1) = \Phi\hat{x}(k) - \Gamma K\hat{x}(k) + LC[x(k) - \hat{x}(k)] \tag{6.4}$$
$$x(k+1) = \Phi x(k) - \Gamma K\hat{x}(k) \tag{6.5}$$
$$y(k) = Cx(k) \tag{6.6}$$

By introducing the estimation error $\delta(k) = \hat{x}(k) - x(k)$, we get:

$$\delta(k+1) = \Phi\delta(k) - LC\delta(k) \qquad \text{Equations (6.4)-(6.5)}$$
$$x(k+1) = \Phi x(k) - \Gamma K[\delta(k) + x(k)] \qquad \text{Equation (6.5)}$$
$$y(k) = Cx(k) \qquad \text{Equation (6.6)}$$

By choosing as state variables of the full system described before the quantities $x(k)$ and $\delta(k)$, the model can be arranged as follows:

$$\begin{bmatrix} \delta(k+1) \\ x(k+1) \end{bmatrix} = \underbrace{\begin{bmatrix} \Phi - LC & 0 \\ -\Gamma K & \Phi - \Gamma K \end{bmatrix}}_{\Phi_t} \begin{bmatrix} \delta(k) \\ x(k) \end{bmatrix}$$
$$y(k) = \begin{bmatrix} 0 & C \end{bmatrix} \begin{bmatrix} \delta(k) \\ x(k) \end{bmatrix}$$

It is the state-space model of the full system that is of the order $2n$. Its dynamics is governed by the eigenvalues of the matrix Φ_t, that is to say by the zeros of the characteristic equation:

$$\det(\lambda I_{2n} - \Phi_t) = \det\left\{\lambda I_{2n} - \begin{bmatrix} \Phi - LC & 0 \\ -\Gamma K & \Phi - \Gamma K \end{bmatrix}\right\} = 0$$

$$\det\begin{bmatrix} \lambda I_n - \Phi + LC & 0 \\ \Gamma K & \lambda I_n - \Phi + \Gamma K \end{bmatrix} = 0$$

In these expressions, the index of the identity square matrices I indicates their dimension. The calculation rules of the determinant of a matrix subdivided in blocks give:

$$\underbrace{\det[\lambda I - \Phi + LC]}_{\alpha_e(\lambda)} \underbrace{\det[\lambda I - \Phi + \Gamma K]}_{\alpha_c(\lambda)} - \underbrace{\det[\Gamma K]\det[0]}_{0} = 0$$

Thus, the eigenvalues of the full system are the zeros of the equation $\alpha_e(\lambda)\alpha_c(\lambda) = 0$ which corresponds to the product of the characteristic equation of the estimator with that of the state

feedback. Consequently, the set of the eigenvalues of the complete system is a combination of the eigenvalues chosen to govern the estimation error of the state and those chosen to govern the closed-loop dynamics. In other words, the eigenvalues imposed to the system by the feedback are not modified by the addition of an observer. Thus, there is no degradation of the dynamic performances.

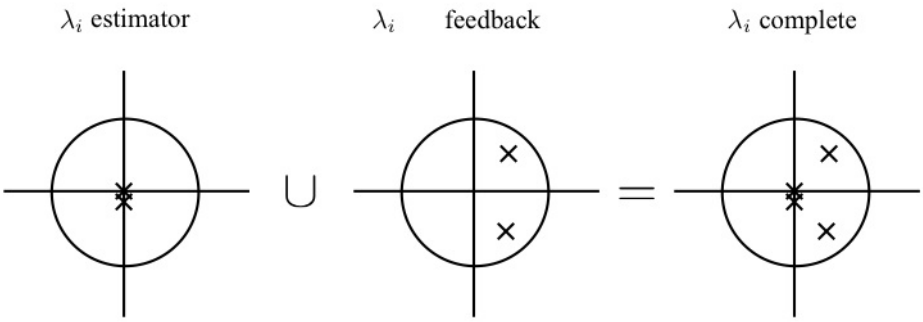


Figure 6.4 – Combination of the eigenvalues.

This property of independence of the dynamics of the observer and of the dynamics of the state feedback is known as the separation principle.

6.5 Example: Ball and beam

6.5.1 System description

The system represented in figure 6.5 is made of a ball that is moving on a beam whose tilt can be modified with the help of an electrical drive.

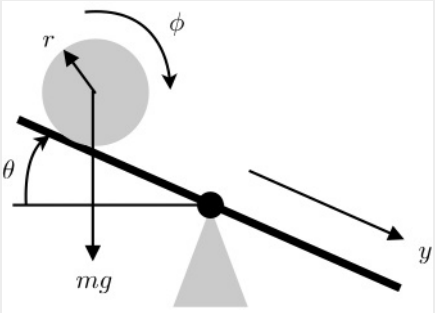


Figure 6.5 – Schematic diagram of the ball and beam system.

ϕ is the angular position of the ball on its rotation axis, r its radius, m its mass, J its moment of inertia and y the position on the beam. θ is the tilt angle of the beam with respect to the horizontal plane and L is its length.

6.5.2 Modeling

By neglecting the dynamics of the motor, that is to say that we consider a control applied directly on the angle: $u(t) = \theta(t)$, the ball and beam can be described by the following nonlinear model M with $x_1 = \phi$ and $x_2 = \dot{\phi}$:

$$\left|\begin{array}{rl} \dot{x}_1(t) & = x_2(t) \\ \dot{x}_2(t) & = \frac{mgr}{J} \sin u(t) \\ y(t) & = r x_1(t) \end{array}\right.$$

6.5.3 Decoupler

The decoupler that allows to linearize this system by feedback is the inverse model M^{-1} of the original system (Fig. 6.6).

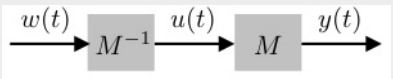


Figure 6.6 – Decoupler of the ball and beam.

With $b = mgr/J$, we get:

$$\begin{array}{rlll} y(t) & = r x_1(t) & & \\ \dot{y}(t) & = r \dot{x}_1(t) & = r x_2(t) & \\ \ddot{y}(t) & = r \dot{x}_2(t) & = r b \sin u(t) & \equiv w(t) \end{array}$$

$w(t)$ is the input of the decoupler. The static inverse model M^{-1} which does not require any measurement of the state in this particular case is then:

$$u(t) = \text{asin}\frac{w(t)}{rb}$$

The model of the equivalent system is $\ddot{y}(t) = w(t)$ (Fig. 6.7). This system has the same states as the original system. The advantage is that this equivalent model is linear. It is possible to discretize it and to design an observer of its two states $y(t)$ and $\dot{y}(t)$.

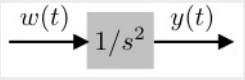


Figure 6.7 – Equivalent system.

To guarantee the linearization for all operating conditions, the control $w(t)$ has to be limited by a saturation $|w(t)| \leqslant \pm rb$.

6.5.4 Discretization

The new equivalent state-space model to be exploited for the design of the observer and of the controller is:

$$\left|\begin{array}{rl} \dot{x}_1(t) & = x_2(t) \\ \dot{x}_2(t) & = w(t) \\ y(t) & = x_1(t) \end{array}\right.$$

The discrete model of this double integrator was already obtained in the example 2.3. It is:

$$x(k+1) = \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix} x(k) + \begin{bmatrix} h^2/2 \\ h \end{bmatrix} w(k)$$
$$y(k) = \begin{bmatrix} 1 & 0 \end{bmatrix} x(k)$$

6.5.5 Dead-beat observer

The dead-beat observer for the discrete double integrator was also obtained in the example 6.1.3 with $\alpha_e(\lambda) = \lambda^2$.

$$L = \alpha_e(\Phi) \begin{bmatrix} c^T \\ c^T\Phi \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \Phi^2 \begin{bmatrix} 1 & 0 \\ 1 & h \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 1/ & h \end{bmatrix}$$

6.5.6 Controller design by imposing the eigenvalues

A state feedback for the discrete double integrator was designed in example 5.3.1 with $\alpha_c(\lambda) = \lambda^2 - 1.6\lambda + 0.7 = 0$.

$$K = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} g & \Phi g \end{bmatrix}^{-1} \begin{bmatrix} \frac{0.1}{h^2} & \frac{0.35}{h} \end{bmatrix}$$

6.5.7 Implementation

The complete implementation diagram is given in figure 6.8.

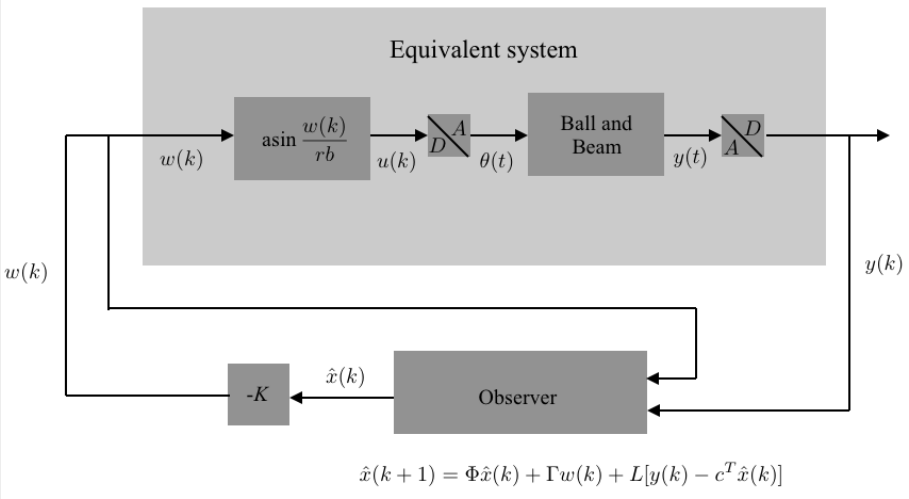


Figure 6.8 – Full control of the ball and beam with the decoupler, the observer and the state feedback.

The observed state corresponds to the position and the angular speed of the ball, $\hat{x}_1(k) = \hat{y}(k) = r\hat{\phi}$ and $\hat{x}_2(k) = \dot{\hat{y}}(k) = r\dot{\hat{\phi}}$. It is an acceleration control $w(k) = \ddot{\phi}(k)$.

6.6 Exercises

6.6.1 State-space observer of the electrical drive

Problem

Consider the discretized version of the model of an electrical drive derived in paragraph 2.8:

$$\left|\begin{array}{rcl} x(k+1) & = & \underbrace{\begin{bmatrix} 1 & 0.0235 \\ 0 & 0.8825 \end{bmatrix}}_{\Phi} x(k) + \underbrace{\begin{bmatrix} 0.0003 \\ 0.0235 \end{bmatrix}}_{\Gamma} u(k) \\ y(k) & = & \underbrace{\begin{bmatrix} 1 & 0 \end{bmatrix}}_{c^T} x(k) \end{array}\right.$$

1. Verify this discrete model with the aid of the adequate Matlab command. The sampling period is 25 ms.

2. Determine the gain of a state-space observer by specifying all the eigenvalues associated to the center of the unit circle.

3. Determine in open loop the response of the drive and that of the observer by means of Simulink (chose different initial conditions for those two elements).

4. Chose the dynamics to design a state feedback and determine the corresponding gain K.

5. Simulate the full system in Simulink (observer and controller).

Solution

1. Consider:

$$A = \begin{bmatrix} 0 & 1 \\ 0 & a \end{bmatrix}, b = \begin{bmatrix} 0 \\ b \end{bmatrix} \text{ and } c^T = \begin{bmatrix} 1 & 0 \end{bmatrix}, \text{ with: } a = -5 \text{ and } b = 1$$

   The corresponding discrete-time model is obtained in Matlab using the command [phi, Gamma]=c2d(A,B,h).

2. The dynamics imposed for the observer is:

$$L = \alpha_e(\Phi) \begin{bmatrix} c^T \\ c^T\Phi \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \text{ with } \alpha_e(\Phi) = \Phi^2$$

We get:

$$L = \begin{bmatrix} 1.883 \\ 33.14 \end{bmatrix}$$

3. To simulate the observer using Simulink, its model has to be arranged in the form of a standard state-space model.

$$\begin{aligned} \hat{x}(k+1) &= \Phi\hat{x}(k) + \Gamma u(k) + L[y(k) - c^T\hat{x}(k)] \\ &= (\Phi - Lc^T)\hat{x}(k) + \begin{bmatrix} \Gamma & L \end{bmatrix} \begin{bmatrix} u(k) \\ y(k) \end{bmatrix} \end{aligned}$$

The inputs of the observer are the input and the output of the system to be observed.

The simulation diagram is given below.

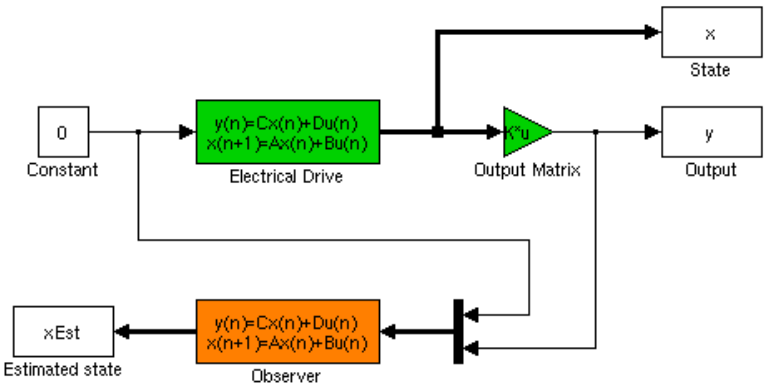


Figure 6.9 – Simulink diagram of the observer in open loop.

The time responses are reported in the following figure. We note that the state-space estimations reach the real values of the state after two sampling periods.

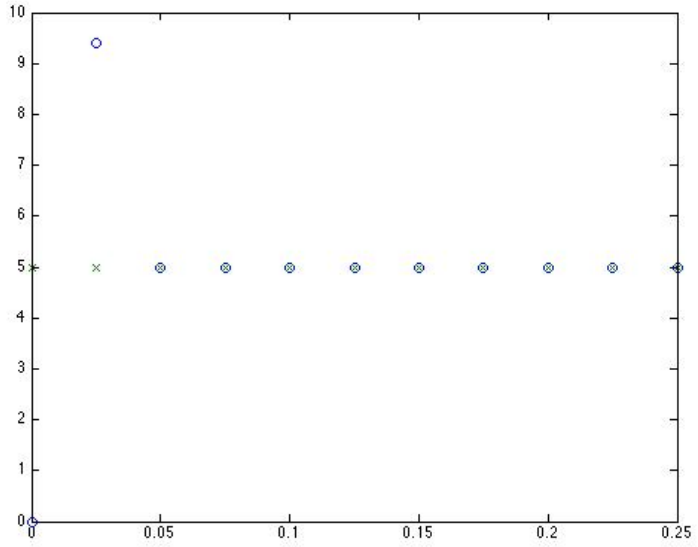


Figure 6.10 – Actual (x) and estimated (o) position.

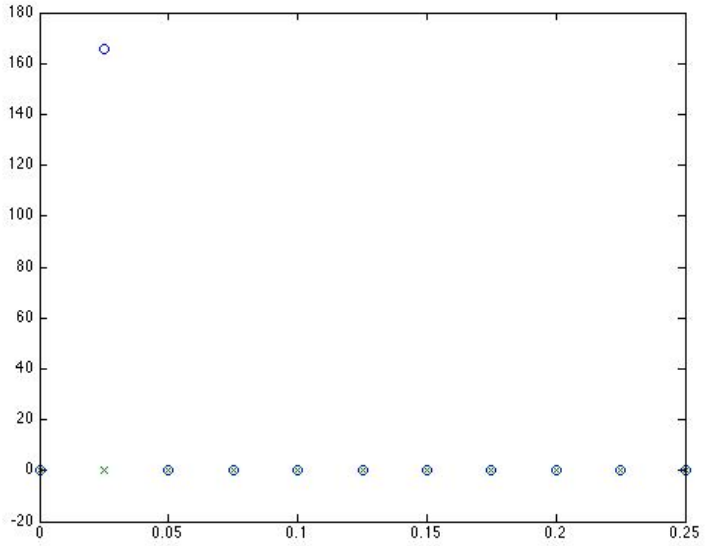


Figure 6.11 – Actual (x) and estimated (o) velocity.

4. The control is chosen to impose in closed loop the following eigenvalues:

$$\lambda_{1,2} = 0.8 \pm 0.25\,j$$

The corresponding feedback gain is obtained by the Ackermann formula, whose Matlab control is:

$$K = \text{acker}(\Phi, \Gamma, \begin{bmatrix} \lambda_1 & \lambda_2 \end{bmatrix}) = \begin{bmatrix} 174.5 & 9.794 \end{bmatrix}$$

5. The simulation diagram of the complete system (observer and feedback) is given below. The block definitions are the same as previously.

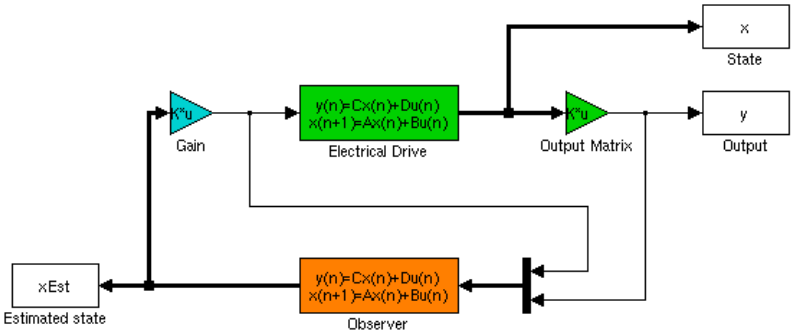


Figure 6.12 – Simulink diagram of the observer and the state feedback in closed loop.

The initial position is 5 rad and the initial velocity is zero. The controller brings these two quantities to zero as shown in the figures below.

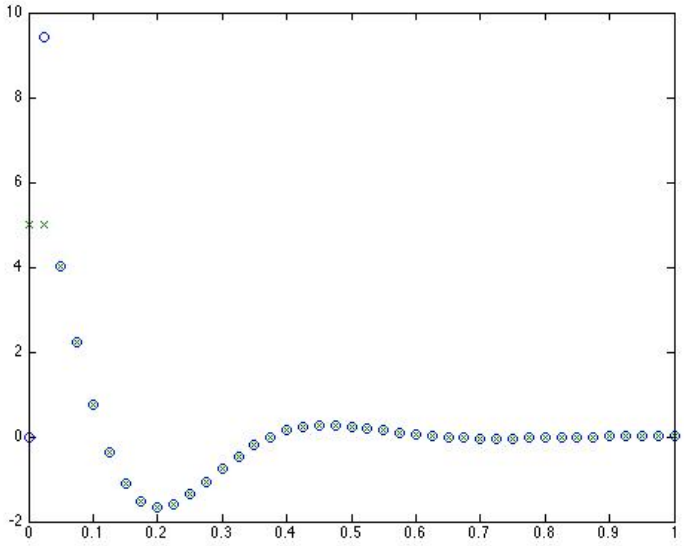


Figure 6.13 – Actual (x) and estimated (o) controlled position.

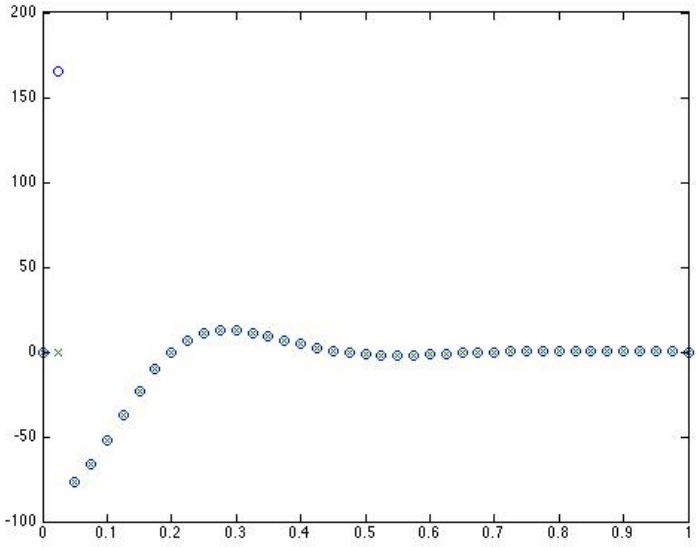


Figure 6.14 – Actual (x) and estimated (o) controlled velocity.

6.6.2 State observer of a dual stage actuator

Problem

A dual stage actuator (Figure 6.15) combining a sliding table (coarse stage) and a piezoelectric stack (fine stage) for nano positioning can be represented by the following simplified model:

$$m\ddot{\delta}_1(t) = F(t) - k_1\delta_1(t)$$
$$\dot{\delta}_2(t) = -\frac{1}{\tau}\delta_2(t) + \frac{K}{\tau}U(t)$$

The inputs are the force F applied on the coarse stage and the voltage supply U of the fine stage. The measured output is the sum of the displacement of each stage ($y = \delta_1 + \delta_2$).

1. Give the state-space model of this systems with $m = 1$, $k_1 = 1$, $\tau = 10^{-3}$, and $K = 10^{-3}$.

2. Express the model in terms of deviations around an operation point corresponding to $\bar{\delta}_1 = r$ and $\bar{\delta}_2 = 0$, and determine the corresponding feedforward control.

3. Provide the corresponding discrete-time state-space representation of this system including AD and DA converters using the Euler approximation for $h = 10^{-3}$.

4. Design a dead-beat state observer (no need to compute inverse matrices) and draw the corresponding block diagram (system, AD & DA converters, observer).

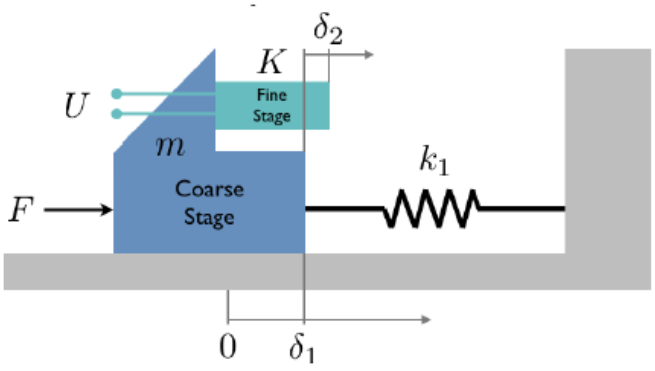


Figure 6.15 – Dual stage actuator for nano positioning

Solution

1. State-space model (2 inputs)

$$\left|\begin{array}{lcl} x_1 &=& \delta_1 \\ x_2 &=& \dot{\delta}_1 \\ x_3 &=& \delta_2 \end{array}\right| \qquad \left|\begin{array}{lcl} \dot{x}_1 &=& \dot{\delta}_1 = x_2 \\ \dot{x}_2 &=& \ddot{\delta}_1 = \frac{1}{m}u_1 - \frac{k_1}{m}x_1 = u_1 - x_1 \\ \dot{x}_3 &=& \dot{\delta}_2 = -\frac{1}{\tau}x_3 + \frac{K}{\tau}u_2 = -1000x_3 + u_2 \end{array}\right|$$

$$\left|\begin{array}{lcl} u_1 &=& F \\ u_2 &=& U \end{array}\right| \qquad \left|\begin{array}{lcl} y &=& x_1 + x_3 \end{array}\right|$$

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & -1000 \end{bmatrix} x + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} u \qquad y = [\,1 \;\; 0 \;\; 1\,]\,x$$

2. Operating point

$$\left|\begin{array}{lll} 0 = \bar{x}_2 & \bar{x}_1 = r & \bar{u}_1 = \bar{x}_1 = r \\ 0 = \bar{u}_1 - \bar{x}_1 & \bar{x}_2 = 0 & \bar{u}_2 = 1000\bar{x}_3 = 0 \\ 0 = -1000\bar{x}_3 + \bar{u}_2 & \bar{x}_3 = 0 & \bar{y} = r \end{array}\right|$$

$$\left|\begin{array}{l} \bar{y} = \bar{x}_1 + \bar{x}_3 \end{array}\right|$$

3. Discrete-time model

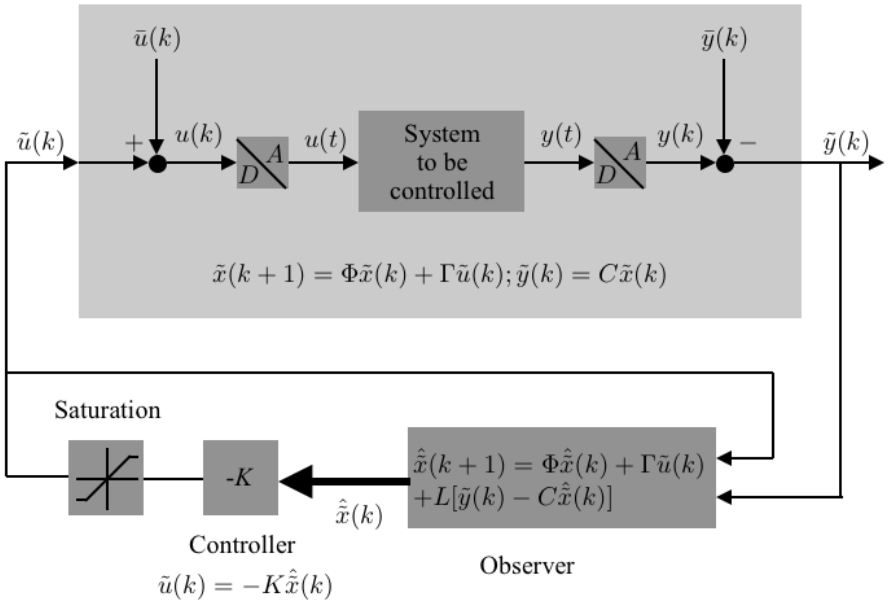
$$x(k+1) - x(k) = hAx + hBu$$
$$x(k+1) = (I + Ah)x + Bhu$$
$$y(k) = Cx$$

4. Dead-beat observer

$$\Phi = \begin{bmatrix} 1 & h & 0 \\ -h & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \Gamma = \begin{bmatrix} 0 & 0 \\ h & 0 \\ 0 & h \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix} \quad D = \begin{bmatrix} 0 & 0 \end{bmatrix}$$

$$Q = \begin{bmatrix} C \\ C\Phi \\ C\Phi^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & h & 0 \\ 1-h^2 & 2h & 0 \end{bmatrix} \qquad \Phi^2 = \begin{bmatrix} 1-h^2 & 2h & 0 \\ -2h & 1-h^2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$L = \Phi^3 Q^{-1} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1-h^2 & 3h-h^3 & 0 \\ h^3-h & 1-2h-h^2 & 0 \\ 0 & 0 & 0 \end{bmatrix} Q^{-1} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$



6.6.3 Controller and observer

Problem

Consider the system Describe by the following transfer function:

$$H(z) = \frac{z^{-1} + 2z^{-2}}{1 + 2z^{-1} + z^{-2}}$$

1. Design a dead-beat observer and a dead-beat state feedback controller.
2. Represent the block diagram of the closed-loop system, including the observer.
3. How many sampling periods are necessary to bring back the state to zero?

Solution

State space model:

$$w(k+1) = \begin{bmatrix} -2 & -1 \\ 1 & 0 \end{bmatrix} w(k) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(k)$$

$$y(k) = \begin{bmatrix} 1 & 2 \end{bmatrix} w(k)$$

Dead-beat observer:

$$Q = \begin{bmatrix} CI \\ C\Phi \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 0 & -1 \end{bmatrix}$$

$$L = \alpha_e(\Phi) Q^{-1} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \Phi^2 \frac{1}{-1} \begin{bmatrix} -1 & -2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 & 2 \\ -2 & -1 \end{bmatrix} \begin{bmatrix} 2 \\ -1 \end{bmatrix} = \begin{bmatrix} 4 \\ -3 \end{bmatrix}$$

Dead-beat controller:

$$K = \begin{bmatrix} 0 & 1 \end{bmatrix} G^{-1} \alpha_c(\Phi) = \begin{bmatrix} -2 & -1 \end{bmatrix}$$

7 Optimal control

7.1 Optimal control

7.1.1 Standard state feedback structure

In the case of systems having only one input, the control structure illustrated in Figure 7.1 is generally exploited.

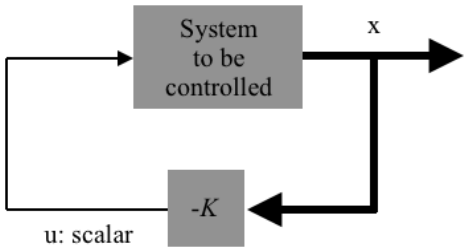


Figure 7.1 – Standard state control structure

The design of the state controller is easily done by choosing the closed-loop eigenvalues. In this situation, the difficulty of the design lies in the choice of the eigenvalues. This choice is not trivial. To obtain a quick response, the specification of a dead-beat can be considered. However, this leads generally to amplitudes of the control signal that the actuators cannot provide. In fact, the technique for placement of the eigenvalues only gives an explicit control on the stability, the transient on the signals are only indirectly restrained. In addition, in the MIMO case, the Ackermann formula is not exploitable anymore and the use of a term-by-term identification of the coefficients of the characteristic polynomial is the only possible solution. This more complex approach requires analytical developments and shows additional degrees of freedom to be wisely exploited.

To overcome these difficulties, an optimal control approach is introduced. This approach also enables to define criteria on the input signals, allowing in such a way to control as example the energy for actuation.

7.1.2 Optimal control approach

The optimal control approach lies in the explicit selection of a sequence of inputs $u(0)$, $u(1)$, $\ldots$, $u(N-1)$ that minimizes a cost function predefined on an optimization horizon N, rather than in the choice of the location of the eigenvalues.

In this approach, the handling of the SISO and MIMO cases is similar.

The cost function to minimize is the following:

$$J = \frac{1}{2}\sum_{k=0}^{N}[x^T(k)Q_1x(k) + u^T(k)Q_2u(k)]$$

The parameters to be adjusted by the designer are the weighting matrices Q_1 and Q_2 which will allow to make a trade-off between closed-loop dynamics and actuation efforts. These matrices must be symmetric and positive-semidefinite, that is to say that $x^TQx \leqslant 0$. Generally, they are chosen diagonal, with all the elements positive or zero. The respective values of the elements of the diagonal of Q_1 allow to weight the importance of the states among themselves. The respective values of the elements of the diagonal of Q_2 allow to weight the importance of the inputs among themselves. A zero value amounts not to take into account the corresponding term. The opposite sign values are avoided so that the terms of the sum do not compensate each other.

The term $x^T(k)Q_1x(k)$ allows to limit the weighted sum of the $x_i^2(k)$ and to bring back the state to zero or to its nominal value (if the deviation variables are introduced in the cost function). The term $u^T(k)Q_2u(k)$ allows to limit the weighted sum of the $u_i^2(k)$ and to avoid in such a way too strong variations of the control signals.

The couples $\langle x(k), u(k)\rangle$ that appear in the cost function must be compatible with the dynamics of the system to be controlled; the minimization has consequently to take into account the constraint $x(k+1) = \Phi x(k) + \Gamma u(k)$.

The speed of the correction is adjusted by means of the optimization horizon. The sequence to be obtained is such that the samples $u(N+i)$ are zero for $i = 0, 1, \cdots$. That is to say that the cost function has to be minimized by exploiting the control inputs on less than N sampling periods.

7.1.3 Summary of the optimization problem

Consider as known the initial state $x(0) = x_0$, the matrices Q_1 and Q_2 and the final condition $u(N) = 0$, find the states $x(k)$, where $k = 1, \cdots, N$ and the control signals $u(k)$, where $k = 0, \cdots, N-1$ that minimize the cost function:

$$J = \frac{1}{2}\sum_{k=0}^{N}[x^T(k)Q_1x(k) + u^T(k)Q_2u(k)]$$

under the constraint: $-x(k+1) + \Phi x(k) + \Gamma u(k) = 0$, where $k = 0, 1, \cdots, N$.

7.1.4 Solution of the optimization problem

To find the minimum of a function, its partial derivatives have to be cancelled. If there are equality constraints, the method of the Lagrange multipliers has to be used, that is to say that the expanded cost function J' has to be minimized:

$$\begin{aligned} J' &= \sum_{k=0}^{N}[\tfrac{1}{2}x^T(k)Q_1x(k) + \tfrac{1}{2}u^T(k)Q_2u(k)] + \\ &\quad \sum_{k=0}^{N}[\lambda^T(k+1)\{-x(k+1) + \Phi x(k) + \Gamma u(k)\}] \end{aligned}$$

It is the combination of the initial cost function J and the equality constraint. A Lagrange multiplier $\lambda(k+1)$ is introduced for each value of k (vector of n elements). The addition of these Nn unknown variables is the price to pay for the simplification of the problem.

7.1.5 Summary of the expanded optimization problem without constraint

Consider the given matrices Q_1 and Q_2, find the states $x(k)$, where $k = 1, \cdots, N$, the controls $u(k)$, where $k = 0, \cdots, N-1$ and the multipliers $\lambda(k)$, where $k = 0, \cdots, N-1$, which minimize the cost function J'.

In the minimization process, the states are obtained by means of the model and from the initial condition $x(0) = x_0$ (Figure 7.2):

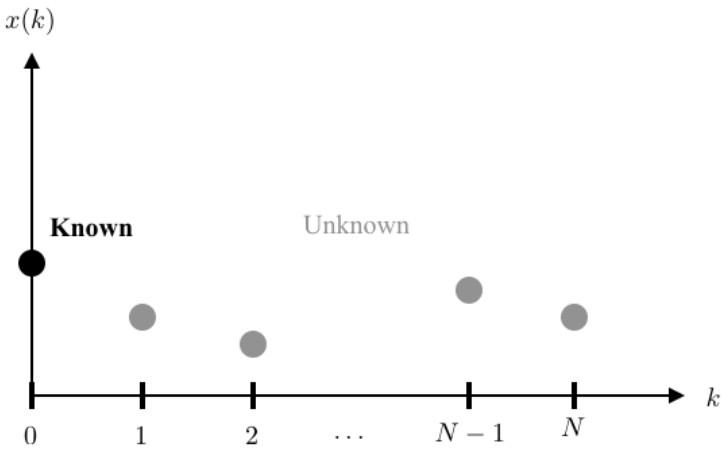


Figure 7.2 – Unknown states to be computed by minimization.

The control inputs are computed in order to satisfy the final condition $u(N) = 0$ (Figure 7.3).

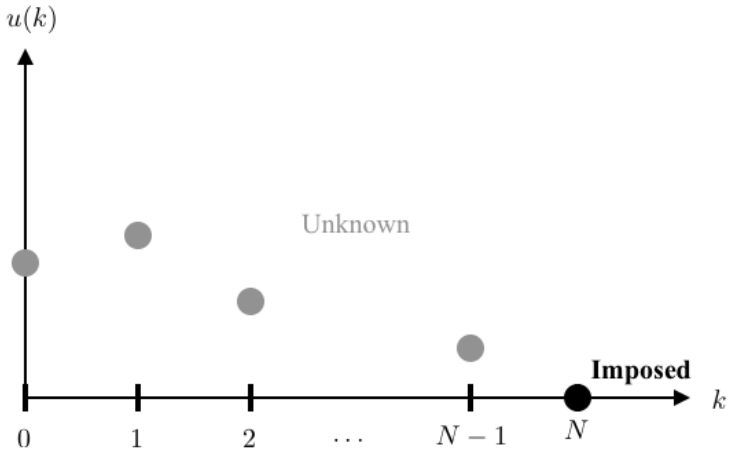


Figure 7.3 – Unknown control inputs to be computed by minimization.

7.1.6 Solution of the expanded optimization problem without constraint

The cancellation of the partial derivatives with respect to the control inputs provides, for any values of k:

$$\begin{bmatrix} \frac{\partial J'}{\partial u_1(k)} \\ \vdots \\ \frac{\partial J'}{\partial u_r(k)} \end{bmatrix} = \frac{\partial J'}{\partial u(k)} = Q_2 u(k) + \Gamma^T \lambda(k+1) = 0 \tag{7.1}$$

Thus, for $k = N$, the following additional final condition is obtained:

$Q_2 \underbrace{u(N)}_{0} + \Gamma^T \lambda(N+1) = 0$, from which:

$$\lambda(N+1) = 0 \tag{7.2}$$

The cancellation of the partial derivatives with respect to the multipliers provides, for all values of k:

$$\begin{bmatrix} \frac{\partial J'}{\partial \lambda_1(k+1)} \\ \vdots \\ \frac{\partial J'}{\partial \lambda_n(k+1)} \end{bmatrix} = \frac{\partial J'}{\partial \lambda(k+1)} = -x(k+1) + \Phi x(k) + \Gamma u(k) = 0 \tag{7.3}$$

This expression is nothing else than the constraint that reappears thanks to the Lagrangian formulation of the optimization problem.

The obtention of the partial derivatives with respect to the states requires to take into account two consecutive terms of the sum J' where $x(k)$ appears.

$$\begin{aligned} J' &= \cdots + \tfrac{1}{2}x^T(k-1)Q_1x(k-1) + \tfrac{1}{2}u^T(k-1)Q_2u(k-1) \\ &+\lambda^T(k)\{-x(k) + \Phi x(k-1) + \Gamma u(k-1)\} \\ &+\tfrac{1}{2}x^T(k)Q_1x(k) + \tfrac{1}{2}u^T(k)Q_2u(k) \\ &+\lambda^T(k+1)\{-x(k+1) + \Phi x(k) + \Gamma u(k)\} + \cdots \end{aligned}$$

Thus, for all values of k:

$$\begin{bmatrix} \frac{\partial J'}{\partial x_1(k)} \\ \vdots \\ \frac{\partial J'}{\partial x_n(k)} \end{bmatrix} = \frac{\partial J'}{\partial x(k)} = -\lambda(k) + Q_1x(k) + \Phi^T\lambda(k+1) = 0 \tag{7.4}$$

For $k = N$ and by exploiting the relations (7.2) and (7.4), the following additional final condition is obtained:

$-\lambda(N) + Q_1x(N) + \Phi^T\underbrace{\lambda(N+1)}_{0} = 0$, from which:

$$\lambda(N) = Q_1x(N) \tag{7.5}$$

The solution of the system made of the equations (7.1), (7.3) and (7.4) is not trivial. Indeed, the state equation (7.3) has to be evaluated iteratively (forward) from the initial condition $x(0) = x_0$:

$$x(0) \rightarrow x(1) \rightarrow \cdots \rightarrow x(k+1) = \Phi x(k) + \Gamma u(k) \rightarrow \cdots \rightarrow x(N)$$

The control $u(k)$ which appears in this expression is obtained by means of relation (7.1):

$$u(k) = -Q_2^{-1}\Gamma^T\lambda(k+1)$$

This evaluation requires the knowledge of the multiplier $\lambda(k+1)$, which is determined iteratively (backward) by (7.4) from the final condition $\lambda(N) = Q_1x(N)$. Thus:

$$\lambda(N) = Q_1x(N) \rightarrow \cdots \rightarrow \lambda(k) = Q_1x(k) + \Phi^T\lambda(k+1) \rightarrow \cdots \rightarrow \lambda(0)$$

It is a coupled problem since the state should be known to determine the multipliers and the multipliers should be known to determine the state. Riccati found the solution to this problem by introducing an auxiliary variable in the form of a symmetric square matrix $S(k)$ such that:

$$\lambda(k) = S(k)x(k) \tag{7.6}$$

The final value of $S(k)$ is obtained by setting $k = N$. It appears that:

$\underbrace{\lambda(N)}_{Q_1 x(N)} = S(N)x(N)$, from which:

$$S(N) = Q_1 \tag{7.7}$$

The matrix $S(k)$ is used to substitute $\lambda(k)$ and $\lambda(k+1)$ in the equations (7.1), (7.3) and (7.4). We get:

$$S(k) = \Phi^T\{S(k+1) - S(k+1)\Gamma[Q_2 + \Gamma^T S(k+1)\Gamma]^{-1}\Gamma^T S(k+1)\}\Phi + Q_1$$

It is a discrete Riccati equation in which only the variable $S(k)$ appears. This equation can be solved backwards from the final value $S(N) = Q_1$. Once the matrix $S(k)$ is determined for each sampling instant, the following factorization can be highlighted from the equations (7.1) and (7.6):

$$\begin{aligned}
&Q_2 u(k) = -\Gamma^T \underbrace{\lambda(k+1)}_{S(k+1)x(k+1)} \\
&Q_2 u(k) = -\Gamma^T S(k+1) \underbrace{x(k+1)}_{[\Phi x(k)+\Gamma u(k)]} \\
&[Q_2 + \Gamma^T S(k+1)\Gamma]u(k) = -\Gamma^T S(k+1)\Phi x(k) \\
&u(k) = -\underbrace{[Q_2 + \Gamma^T S(k+1)\Gamma]^{-1}\Gamma^T S(k+1)\Phi}_{K(k)} x(k)
\end{aligned}$$

The optimal solution is provided by the relation:

$$\begin{aligned}
&u(k) = -K(k)x(k) \\
&[r] = -[r \times n] \begin{bmatrix} n \\ \end{bmatrix}
\end{aligned}$$

Without having specified anything about the structure of the feedback, it appears that the

control inputs of the optimal sequence are proportional to the state, as those introduced in the context of the placement of the eigenvalues. However, here, the gain is not time invariant. The structure of a general state feedback is represented in Figure 7.4.

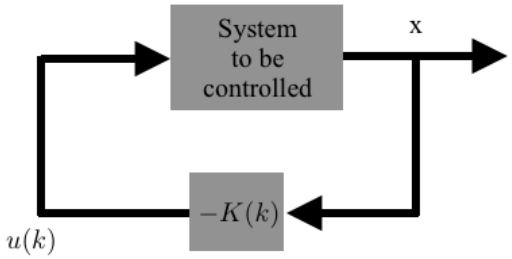


Figure 7.4 – Stucture of a general state feedback.

7.1.7 Algorithm for the design of an optimal control

To ease the computation of the gain $K(k)$, two intermediate variables $R(k)$ and $M(k)$ are introduced. The solution of the Riccati equation being independent of the initial state $x(0)$, it can be computed a priori, and the gain stored in memory for a subsequent implementation of the control if the horizon N is constant. Taking into account that $u(N) = 0$ and that $x(N) = 0$, the gain $K(N)$ is zero, the following steps to set the solution can be proposed:

1. Set as final conditions $S(N) = Q_1$ and $K(N) = 0$;
2. Start with $k = N$;
3. Compute $R(k) = Q_2 + \Gamma^T S(k)\Gamma$ and its inverse;
4. Compute $M(k) = S(k) - S(k)\Gamma R^{-1}(k)\Gamma^T S(k)$;
5. Compute the gain $K(k-1) = R^{-1}(k)\Gamma^T S(k)\Phi$
6. Save the gain $K(k-1)$
7. Set $S(k-1) = \Phi^T M(k)\Phi + Q_1$
8. Set $k = k-1$
9. Restart at 3.

7.1.8 Quadratic linear control (LQR)

As a backwards solution of the minimization problem, $S(k)$ converges to a time invariant value when k decreases.

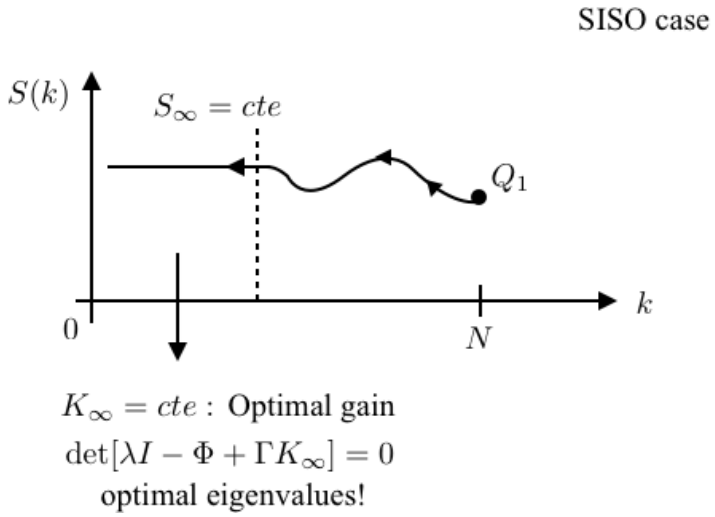


Figure 7.5 – Solution of the Riccati equation.

As most systems operate continuously, $N \to \infty$ and the transient regime can be neglected.

The stationary optimal gain K_∞ is obtained by setting:

$$S_\infty = S(k) = S(k+1)$$
$$S_\infty = \Phi^T [S_\infty - S_\infty \Gamma R^{-1} \Gamma^T S_\infty] \Phi + Q_1$$

and finally by keeping the positive semi-definite solution for S_∞.

The Matlab command that provides these time invariant optimal values is:

$$[K_\infty, S_\infty, E_\infty] = \mathrm{dl}qr(\Phi, \Gamma, Q_1, Q_2)$$

7.2 Summary of the time invariant optimal control solution

Find the sequence of the control $u(k)$ which minimizes the cost function:

$$J = \frac{1}{2}\sum_{k=0}^{N}[\underbrace{\tilde{x}^T(k)Q_1\tilde{x}(k)}_{\substack{\text{Quadratic sum of the deviation with respect to the nominal state}}} + \underbrace{\tilde{u}^T(k)Q_2\tilde{u}(k)}_{\substack{\text{Quadratic sum of the deviation with respect to the nominal state}}}]$$

Q_1 and Q_2 are $n \times n$ and $r \times r$ diagonal weighting matrices, with positive and zero elements.

The optimal solution obtained by the Lagrange method has the form:

$$u(k) = -K(k)x(k)$$

The time-invariant solution for an optimization horizon $N \to \infty$ is obtained by solving the stationary Riccati equation.

$$S_\infty = \Phi^T\left[S_\infty - S_\infty\Gamma\underbrace{(Q_2 + \Gamma^T S_\infty\Gamma)}_{R_\infty}^{-1}\Gamma^T S_\infty\right]\Phi + Q_1$$

$$K_\infty = R_\infty^{-1}\Gamma^T S_\infty\Phi \qquad \text{with the dimensions: } S_\infty : n \times n \text{ and } K_\infty : n \times n$$

$$u(k) = -K_\infty x(k)$$

A control design obtained by this optimal method is less aggressive for the actuators than one obtained by a placement of the eigenvalues. However, the limitation of the exploitation of the actuators is obtained at the expense of losing the explicit control of the absolute and relative damping. Suppressing oscillations by a judicious placement of the eigenvalues is also not possible anymore, which is not acceptable in machine tools applications for example.

7.3 Example: Heat exchanger

Consider the heat exchanger introduced in exercise 3.5.2 and the obtained linearized continuous-time state-space model. The discrete-time state-space model can be obtained numerically by means of the Matlab c2d function. For a sampling period of $h = 0.25$ s, we get:

$$\Phi = \begin{bmatrix} 0.63 & 0.36 \\ 0.09 & 0.85 \end{bmatrix}; \Gamma = \begin{bmatrix} 0.025 \\ 0.115 \end{bmatrix}$$

The only quantity that has to be set being the variable $\tilde{x}_1$, the following weighting matrices are chosen:

$$Q_1 = \begin{bmatrix} 50 & 0 \\ 0 & 0 \end{bmatrix} \text{ and } Q_2 = 1$$

The block diagram for the implementation of the control is represented in Figure 7.6.

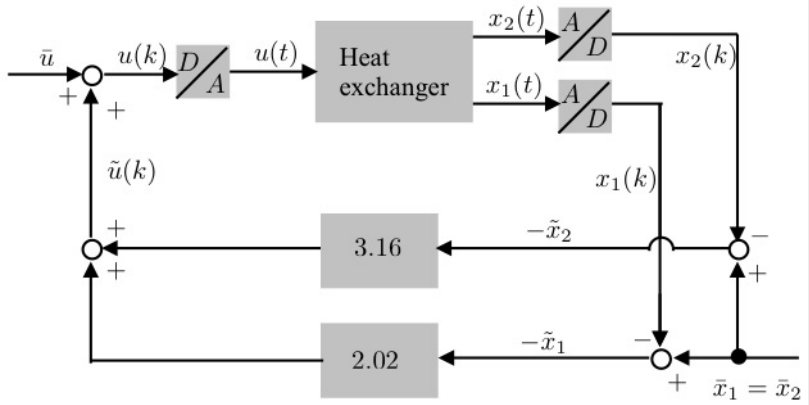


Figure 7.6 – Structure of the optimal control of the exchanger.

The Matlab control file to obtain at the same time a time-varying gain and a time-invariant gain is the following:

```
clear

% Continuous Time Model
% -----------
A=[-2 2; 0.5 -0.75];
B=[0;0.5];
C=[1 0;0 1];
D=[0;0];

% Discrete Time Model
% ----------
h=0.25;
```

```
[F,G]=c2d(A,B,h);

% Time Varying Gain
% ---------
Q1=[50 0;0 0];
Q2=1;
N=20;
S=Q1; K(N,1:2)=[0 0];
for k=N:-1:2,
InvR=inv(Q2+G'*S*G);
M=S-S*G*InvR*G'*S;
K(k-1,1:2)=InvR*G'*S*F;
S=F'*M*F+Q1;
end,
plot(K)

% LQR (Time Invariant Solution)
% --------------
 [Ks,Ss,E]=dlqr(F,G,Q1,Q2);     % Ks = [2.0205 3.1559]
 Ks                             % E = eigenvalues = 0.5333?0.1951i
```

The evolution of the optimal gains is represented in Figure 7.7 .

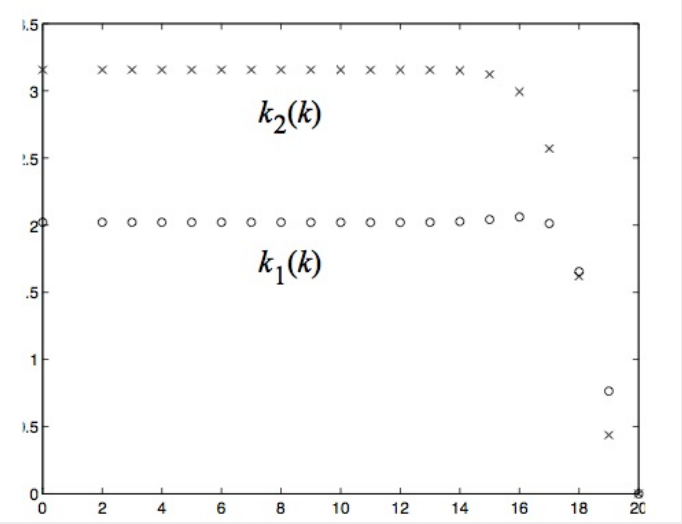


Figure 7.7 – Evolution of the optimal gains with respect to k.

The time-invariant values of the gains for $N \to \infty$ are $k_1 = 2.02$ and $k_2 = 3.16$. These values are used to generate the time responses given in figures 7.8 and 7.9.

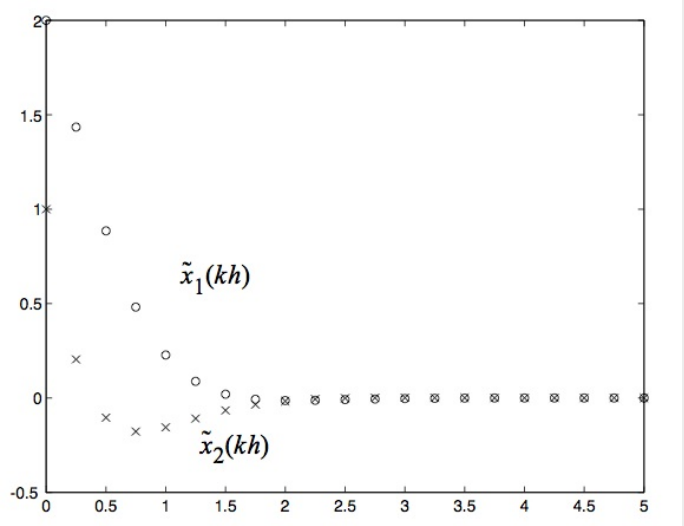


Figure 7.8 – Evolutions of the states with respect to time expressed in seconds.

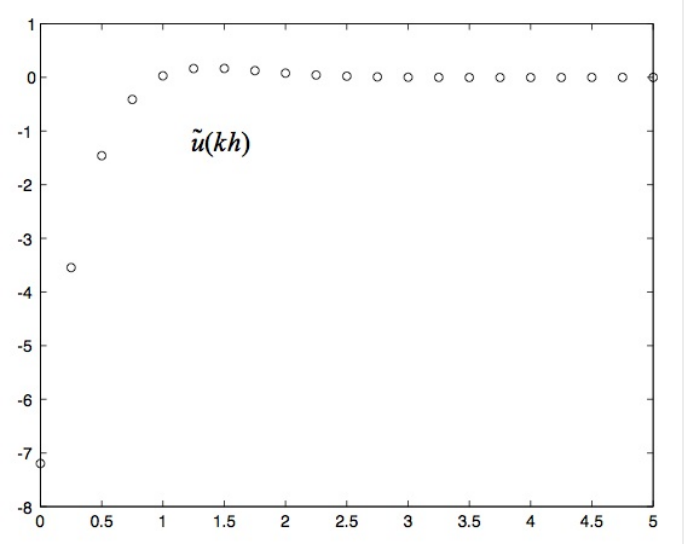


Figure 7.9 – Evolution of the control with respect to time expressed in seconds.

7.4 Optimal estimation

7.4.1 Duality

Controllability and observability are two coupled concepts. We will exploit this feature to derive an optimal estimator from the optimal controller introduced in section 7.1. First, we can show that the system defined by (Φ, Γ) is controllable if and only if the system defined by (Φ^T, Γ^T) is observable. This can be shown by recalling that the system (Φ, Γ) is controllable if and only if

$$\text{rk}\begin{bmatrix} \Gamma & \Phi\Gamma & \dots & \Phi^{n-1}\Gamma \end{bmatrix} = n$$

and the system (Φ^T, Γ^T) is observable if and only if

$$\text{rk}\begin{bmatrix} \Gamma^T \\ \Gamma^T\Phi^T \\ \vdots \\ \Gamma^T(\Phi^T)^{n-1} \end{bmatrix} = n$$

The rank of a matrix being the same as its transpose

$$\text{rk}\begin{bmatrix} \Gamma & \Phi\Gamma & \dots & \Phi^{n-1} & \Gamma \end{bmatrix} = \text{rk}\begin{bmatrix} \Gamma & \Phi\Gamma & \dots & \Phi^{n-1} & \Gamma \end{bmatrix}^T$$

$$= \text{rk}\begin{bmatrix} \Gamma^T \\ \Gamma^T\Phi^T \\ \vdots \\ \Gamma^T(\Phi^{n-1})^T \end{bmatrix} = \text{rk}\begin{bmatrix} \Gamma^T \\ \Gamma^T\Phi^T \\ \vdots \\ \Gamma^T(\Phi^T)^{n-1} \end{bmatrix}$$

Hence (Φ, Γ) is equivalent to (Φ^T, Γ^T) which corresponds to (Φ, C) when exploiting the definition of observability.

It is said that the system (Φ^T, Γ^T) is dual to the system (Φ, C) or (Φ, Γ) is dual to (Φ^T, C^T). The following substitutions are valid:

$$\begin{aligned} \Phi &\leftrightarrow \Phi^T \\ \Gamma &\leftrightarrow C^T \end{aligned}$$

7.4.2 Equivalence between the optimal controller and the optimal observer

In the previous paragraph it has been illustrated that the state-variable approaches are derived from the mathematical structure of the system model. So, we can summarize below the structure of the optimal control problem and its solution. Considering the following system with its state feedback:

$$x(k+1) = \Phi x(k) + \Gamma u(k) \tag{7.8}$$
$$y(k) = Cx(k)$$
$$u(k) = -Kx(k)$$
$$x(k+1) = (\Phi - \Gamma K)x(k)$$
$$\Phi_{cl} = \Phi - \Gamma K \tag{7.9}$$

The optimal solution for the feedback gain is obtained from the following equations:

$$K = R^{-1}\Gamma^T S\Phi \tag{7.10}$$
$$R = \Gamma^T S\Gamma + Q_2 \qquad \text{dimension } r \times r \tag{7.11}$$
$$S = \Phi^T[S - S\Gamma R^{-1}\Gamma^T S]\Phi + Q_1 \tag{7.12}$$

If the structure of the state observer problem is defined the same way, we have:

$$\hat{x}(k+1) = \Phi\hat{x}(k) + \Gamma u(k) + L[y(k) - \hat{y}(k)]$$
$$\delta(k) = \hat{x}(k) - x(k)$$
$$\delta(k+1) = (\Phi - LC)\delta(k)$$
$$\Phi_e = \Phi - LC$$

This last equation can be transposed to show the same structure as (7.9):

$$\Phi_e^T = \Phi^T - C^T L^T \tag{7.13}$$

By comparing equations (7.9) and (7.13) we see the duality between the two problems and we can extend the equivalences given in paragraph 7.4.1.

$$\begin{array}{ccc} \Phi & \leftrightarrow & \Phi^T \\ \Gamma & \leftrightarrow & C^T \\ K & \leftrightarrow & L^T \end{array}$$

As a consequence, we can use the solution of the optimal control problem (equations (7.10), (7.11) and (7.12)) to derive the solution of the optimal estimation problem. The equation

(7.10) becomes $L^T = R^{-1}CS\Phi^T$ or

$$L = \Phi S C^T R^{-1} \tag{7.14}$$

The equation (7.11) becomes

$$R = CSC^T + Q_2 \qquad \text{dimension } p \times p \tag{7.15}$$

and (7.12)

$$S = \Phi[S - SC^T R^{-1} CS]\Phi^T + Q_1 \tag{7.16}$$

7.4.3 Corresponding cost function

In 7.4.2, the solution to compute an optimal gain L for the state observer has been derived using the duality principle. However, no information is available regarding the initial cost function which is minimized by this solution.

Looking at the dimension of the weighting matrices Q_1 and Q_2 and by exploiting the following equivalences:

$$\hat{x}(k+1) = \Phi\hat{x}(k) + \Gamma u(k) + L[y(k) - \hat{y}(k)]$$
$$\delta(k) = \hat{x}(k) - x(k)$$

we can define the output error

$$e(k) = y(k) - \hat{y}(k)$$

Then,

$$\delta(k+1) = \hat{x}(k+1) - x(k+1)$$
$$\delta(k+1) = \Phi\hat{x}(k) + \Gamma u(k) + L[y(k) - \hat{y}(k)] - \Phi x(k) - \Gamma u(k)$$
$$\delta(k+1) = \Phi[\hat{x}(k) - x(k)] + L[y(k) - \hat{y}(k)]$$
$$\delta(k+1) = \Phi\delta(k) + Le(k) \tag{7.17}$$

Also,

$$y(k) = Cx(k) = C[\hat{x}(k) - \delta(k)] = C\hat{x}(k) - C\delta(k) = \hat{y}(k) - C\delta(k)$$
$$y(k) - \hat{y}(k) = -C\delta(k)$$
$$e(k) = -C\delta(k) \tag{7.18}$$

Finally, by comparing equations (7.8) and (7.17), the corresponding cost function is

$$J = \frac{1}{2}\sum_{k=0}^{N}[\delta^T(k)Q_1\delta(k) + e^T(k)Q_2e(k)]$$

with the equality constraint

$$\delta(k+1) = \Phi\delta(k) + Le(k)$$

The first term in the cost function minimizes the modeling errors, while the second term minimizes the measurement noise.

7.4.4 Example: Electrical drive

Considering the example 1.6.3 where we measure not only x_1, but also x_2, i.e.:

$$y_1(t) = x_1(t) \tag{7.19}$$
$$y_2(t) = x_2(t) \tag{7.20}$$

Hence, the linear output equation is:

$$y(t) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} \tag{7.21}$$

The corresponding discrete-time model was derived in example 2.8:

$$\Phi = e^{Ah} = \begin{bmatrix} 1 & \frac{1}{a}(e^{ah}-1) \\ 0 & e^{ah} \end{bmatrix} \tag{7.22}$$
$$\Gamma = \frac{b}{a}\begin{bmatrix} \frac{1}{a}(e^{ah}-1) - h \\ (e^{ah}-1) \end{bmatrix} \tag{7.23}$$

Using the duality $\Phi \leftrightarrow \Phi^T, \Gamma \leftrightarrow C^T, K \leftrightarrow L^T$, we can use the dlqr matlab function to get the optimal observer parameters:

$$[L_\infty^T, S_\infty, E_\infty] = dlqr(\Phi^T, C^T, Q_1, Q_2) \tag{7.24}$$

This observer can be modeled in simulink as an extended state-space model:

$$\hat{x}(k+1) = \Phi\hat{x}(k) + \Gamma u(k) + L[y(k) - C\hat{x}(k)] \tag{7.25}$$

$$\hat{x}(k+1) = (\Phi - LC)\hat{x}(k) + \begin{bmatrix}\Gamma & L\end{bmatrix}\begin{bmatrix}u(k)\\ y(k)\end{bmatrix} \tag{7.26}$$

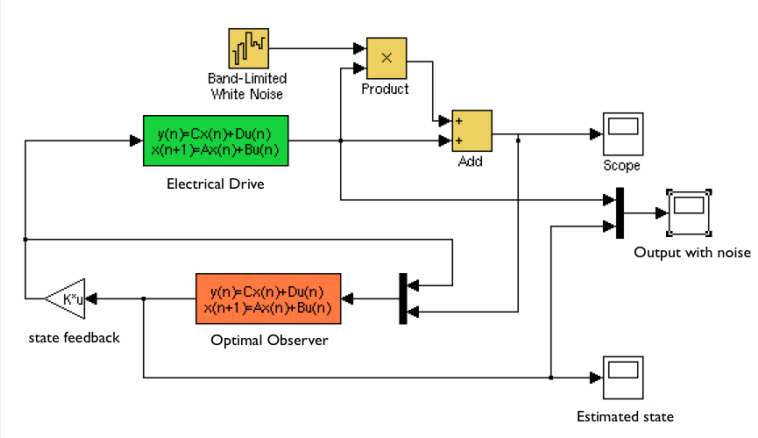


Figure 7.10 – Simulink blok diagram for the validation of the optimal state observer.

The Matlab script to get the corresponding matrices is given below.

```
clear

% Continuous Time Model
% -----------

a=-5;
b=1;
A=[0 1; 0 a];
B=[0;b];
C=[1 0;0 1];
D=[0;0];

% Discrete Time Model
% ----------

h=0.25e-3;
e=exp(a*h);
F=[1 (e-1)/a;0 e];
G=[-b*(h-e/a+1/a(/a;b*(e-1)/a];

% Controller
% ---------

s = -1.8+3.12*j;
```

```
z1=exp(h*s);
z2=exp(h*s');

K = acker(F,G,[z1, z2]);

% Optimal Observer
% --------------

Q1=[1 0;0 1]
Q2=100*[1 0;0 1]

[M,S,R]=dlqr(F', C', Q1, Q2);

L = M';
```

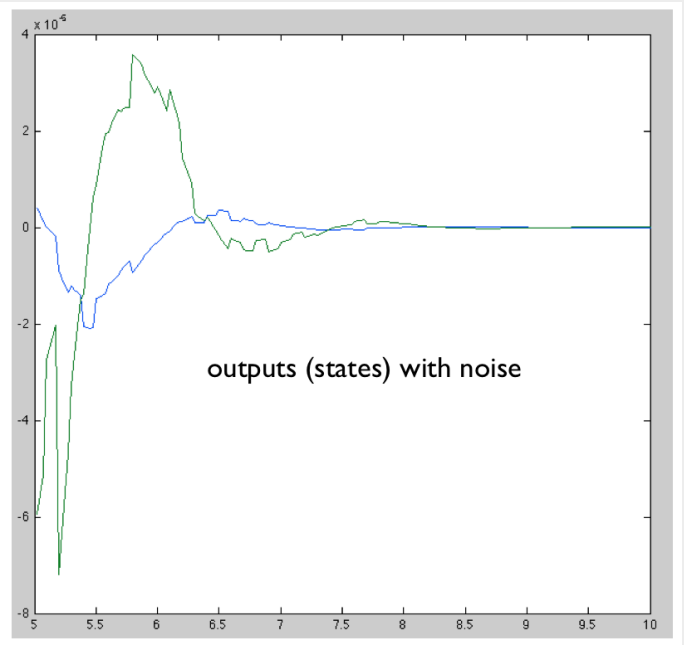


Figure 7.11 – Measured states

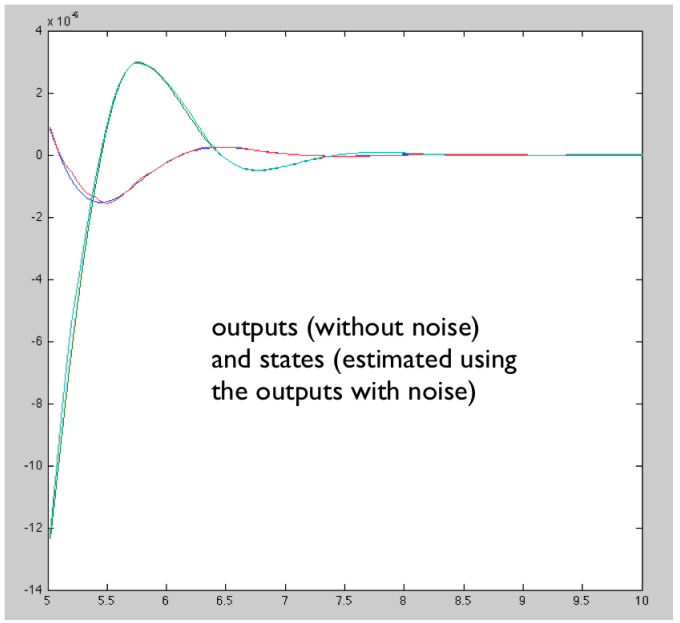


Figure 7.12 – Estimated states

7.5 Some useful definitions

Consider A a real matrix, symmetric ($a_{ij} = a_{ji}$), of dimension $n \times n$, and x and y real vectors, of dimension n.

7.5.1 Real quadratic form

The real quadratic form $x^T Ax = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_i x_j$ is a scalar.

A is positive definite if $x^T Ax > 0$ for $x > 0$ and $x^T Ax = 0$ for $x = 0$. The eigenvalues of a symmetric real matrix are real. If A is positive definite, its eigenvalues are positive (real) and A is regular.

7.5.2 Derivative of a real quadratic form

$$\frac{\partial}{\partial x} x^T Ax = 2Ax = (2x^T A)^T = \begin{bmatrix} \frac{\partial}{\partial x_1} \\ \frac{\partial}{\partial x_2} \end{bmatrix} x^T Ax$$

This form is a column vector.

Reminder:$(AB)^T = B^T A^T$

7.5.3 Bilinear form

The bilinear form $x^T Ay = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_i y_j$ is a scalar.

7.5.4 Derivative of a bilinear form

$$\frac{\partial}{\partial x} x^T Ay = Ay$$

This form is a column vector.

Similarly:

$$\frac{\partial}{\partial x} y^T Ax = A^T y$$

As A is symmetric, $A^T = A$.

7.6 Exercises

7.6.1 Time delay

Problem

Consider the discrete time delay described by the following difference equation:

$$y(k+2) = u(k).$$

1. Determine an optimal time invariant controller to control this system using identity matrices for Q_1 and Q_2.

2. Comment the result and compare with a dead-beat controller.

Solution

Consider the equation $y(k+2) = u(k)$.

The choice of the state variables $x_1(k) = y(k)$ and $x_2(k) = y(k+1)$ leads to:

$$\left| \begin{array}{lcl} x_1(k+1) & = & x_2(k) \\ x_2(k+1) & = & u(k) \end{array} \right.$$

The state model is: $x(k+1) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k)$, thus:

$$\Phi = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \Gamma = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, Q_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, Q_2 = q_2 = 1 \text{ and}$$
$$S_\infty = \begin{bmatrix} s_1 & s \\ s & s_2 \end{bmatrix}, R_\infty = Q_2 + \Gamma^T S_\infty \Gamma = 1 + \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} s_1 & s \\ s & s_2 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = 1 + s_2$$

S_∞ is solution of $S_\infty = \Phi^T \left\{ S_\infty - S_\infty \Gamma R_\infty^{-1} \Gamma^T S_\infty \right\} \Phi + Q_1$

$$\begin{bmatrix} s_1 & s \\ s & s_2 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \left\{ \begin{bmatrix} s_1 & s \\ s & s_2 \end{bmatrix} - \frac{\begin{bmatrix} s_1 & s \\ s & s_2 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} s_1 & s \\ s & s_2 \end{bmatrix}}{1+s_2} \right\} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$
$$= \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} s_1 - \frac{s^2}{1+s_2} & s - \frac{s_2 s}{1+s_2} \\ s - \frac{s_2 s}{1+s_2} & s_2 - \frac{s_2^2}{1+s_2} \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 + s_1 - \frac{s^2}{1+s_2} \end{bmatrix}$$

from which: $S_\infty = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$ and $K_\infty = \begin{bmatrix} 0 & 0 \end{bmatrix}$

For a time delay of two sampling periods, we just have to wait during two sampling periods so that the input appears at the output. We are in presence of a dead-beat response, even without feedback ($K = 0$).

7.6.2 Optimal electrical drive

Problem

The dynamics of a drive whose speed is measured is given by:

$$\begin{aligned} \dot{\omega}(t) &= a\omega(t) + bu(t) \\ y(t) &= \omega(t) \end{aligned}$$

The equivalent discrete state-space model is: (with $x = \omega$)

$$\begin{aligned} x(k+1) &= \phi x(k) + \gamma u(k) \\ y(k) &= x(k) \end{aligned}$$

where $\phi = e^{ah}$ and $\gamma = \frac{b}{a}(\phi - 1)$.

1. Describe this system by a state-space model that is valid around an operating point: $\bar{x} = c$ (c: speed reference).

2. Sketch the block diagram including the imposing of the operating point.

3. Determine the feedback gain of optimal time invariant state with $h = 25$ ms; $a = -5$ and $b = 1$. Chose $Q_1 = 100$ and $Q_2 = 1$.

Solution

- **Operating point**

$$\begin{aligned} &\bar{x}(k+1) = \phi\bar{x}(k) + \gamma\bar{u}(k) \\ &c = \phi c + \gamma\bar{u} \\ &\Rightarrow \bar{u} = \frac{1}{\gamma}(1-\phi)c \end{aligned}$$

- **Block diagram**

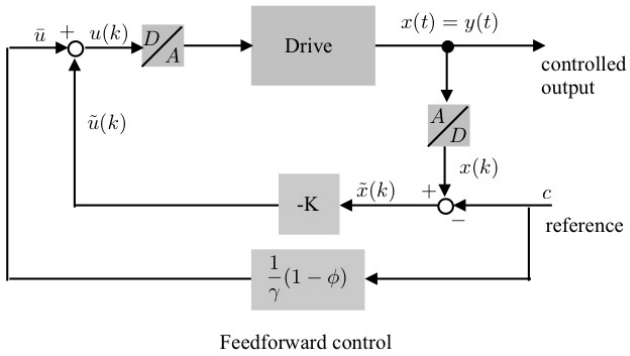


Figure 7.13 – Electrical drive with a feedforward and an optimal feedback action

- **Optimal control**

System to be controlled:

$$\tilde{x}(k+1) = \phi\tilde{x}(k) + \gamma\tilde{u}(k)$$
$$\tilde{y}(k) = \tilde{x}(k)$$

- **Controllability**

The system is of the first order. The controllability matrix: $G = \gamma$ is of rank one, so the system is controllable if $\gamma \neq 0$.

- **Time-invariant control gain**

$$S_\infty = s \text{ “matrix" of dimension } 1 \times 1$$
$$R_\infty = r = Q_2 + s\gamma^2 = 1.166$$
$$s = \phi^2\left[s - s^2\gamma^2\frac{1}{Q_2 + s\gamma^2}\right] + Q_1 = 301$$

The last equation is written:

$$s^2\gamma^2 + s(Q_2 - Q_2\phi^2 - Q_1\gamma^2) - Q_1Q_2 = 0$$

It is the positive solution of this second degree equation that allows to compute the gain $K = \frac{1}{r}\gamma s\phi = 5.35$.

- **M-file**

```
clear
% Continuous-time model of the electrical drive
% ----------------------
a=-5;
b=1;
% Discrete-time model of the electrical drive
% ---------------------
h=25e-3:
F=exp(a*h);
G=b*(F-1)/a;
% Optimal controller
% ---------
Q1=100;
Q2=1;
% Matlab command
% -------
[K,S,E]=dlqr(F,G,Q1,Q2)
% LQR solution % --------------
Sol=roots([G^2(Q2-Q2*F^2-Q1*G^2)-Q1*Q2]);
Sinf=max(Sol)
Rinf=Q2+G'*Sinf*G;
Kinf=inv(Rinf)*G'*Sinf*F
```

- **Simulation**

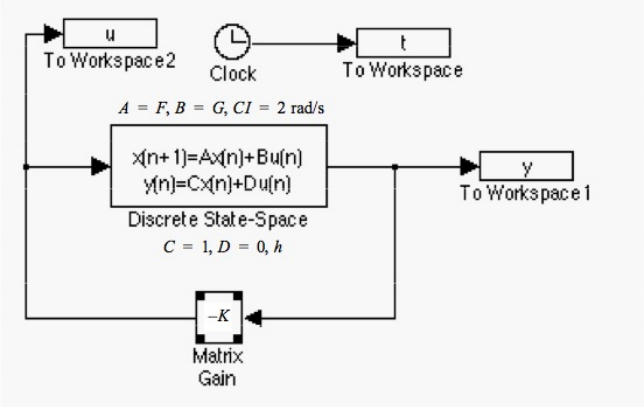


Figure 7.14 – Simulink diagram for the simulation of the optimal electrical drive.

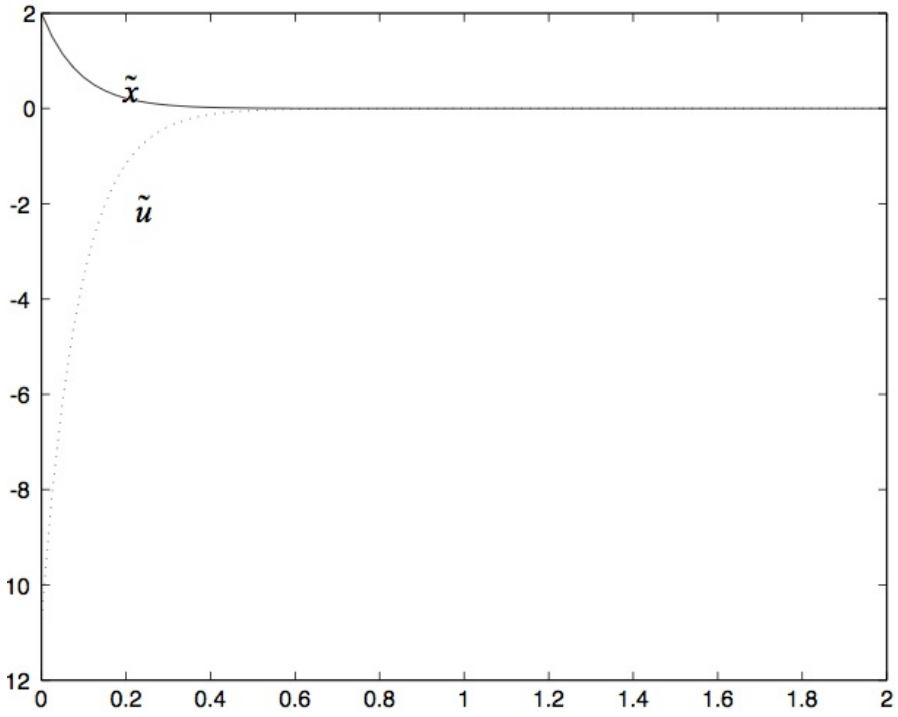


Figure 7.15 – Optimal state and input responses of the electrical drive.

7.6.3 LQR Design

Problem

Consider the following first-order system:

$$x(k+1) = \varphi x(k) + \begin{bmatrix} 2 & 1 \end{bmatrix} \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix}$$

1. Design a LQR controller for this system with $Q_1 = q$ and $Q_2 = \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix}$
2. How could you characterize the closed-loop dynamics?

Solution

1. Stationary solution

$$s = \varphi\left\{ s - s\begin{bmatrix}2 & 1\end{bmatrix}\left(\begin{bmatrix}1 & 0\\ 0 & 4\end{bmatrix} + \begin{bmatrix}2\\ 1\end{bmatrix}\begin{bmatrix}2 & 1\end{bmatrix} s\right)^{-1}\begin{bmatrix}2\\ 1\end{bmatrix} s\right\}\varphi + q$$

$$s = \varphi\left\{ s - s\begin{bmatrix}2 & 1\end{bmatrix}\left(\begin{bmatrix}1 & 0\\ 0 & 4\end{bmatrix} + \begin{bmatrix}4 & 2\\ 2 & 1\end{bmatrix} s\right)^{-1}\begin{bmatrix}2\\ 1\end{bmatrix} s\right\}\varphi + q$$

$$s = \varphi\left\{ s - s\begin{bmatrix}2 & 1\end{bmatrix}\begin{bmatrix}1+4s & 2s\\ 2s & 4+s\end{bmatrix}^{-1}\begin{bmatrix}2\\ 1\end{bmatrix} s\right\}\varphi + q$$

$$s = \varphi\left\{ s - \frac{s^2}{4+17s}\begin{bmatrix}2 & 1\end{bmatrix}\begin{bmatrix}4+s & -2s\\ -2s & 1+4s\end{bmatrix}\begin{bmatrix}2\\ 1\end{bmatrix}\right\}\varphi + q$$

$$s = \varphi\left\{ s - \frac{s^2}{4+17s}\begin{bmatrix}8 & 1\end{bmatrix}\begin{bmatrix}2\\ 1\end{bmatrix}\right\}\varphi + q = \varphi\left\{ s - \frac{17s^2}{4+17s}\right\}\varphi + q$$

$$s = \frac{4s}{4+17s}\varphi^2 + q$$

$$4s + 17s^2 = 4s\varphi^2 + 4q + 17qs$$

$$17s^2 + (4 - 4\varphi^2 - 17q)s - 4q = 0$$

$$K = R^{-1}\Gamma^T s\varphi = \frac{1}{4+17s}\begin{bmatrix}4+s & -2s\\ -2s & 1+4s\end{bmatrix}\begin{bmatrix}2\\ 1\end{bmatrix} s\varphi$$

$$K = \frac{s\varphi}{4+17s}\begin{bmatrix}8\\ 1\end{bmatrix}$$

2. Closed-loop dynamics

$$\Phi_{bf} = \varphi - \Gamma K = \varphi - \begin{bmatrix} 2 & 1 \end{bmatrix} \frac{s\varphi}{4+17s} \begin{bmatrix} 8 \\ 1 \end{bmatrix}$$

$$\Phi_{bf} = \varphi - \Gamma K = \varphi - \frac{17s\varphi}{4+17s} = \frac{4\varphi}{4+17s}$$

7.6.4 Optimal observer

Problem

Consider the following MIMO system with $y(k) = x(k)$ and:

$$x(k+1) = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} x(k) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(k)$$

1. Design an optimal state observer with $Q_1 = qI$ and $Q_2 = I$ (q is a positive constant). Provide all the equations and indicate the dimensions and the form of all matrices (no need to solve explicitly the equations).

2. Write the equation of the observer with its inputs and outputs. Also indicate the dimensions and the form of all matrices.

Solution

- **Observability**

$$C = I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \text{ and } \Phi = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}$$

$$rank(Q) = rank \begin{bmatrix} C \\ C\Phi \end{bmatrix} = rank \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} = 2$$

$$Q_1 = \begin{bmatrix} q & 0 \\ 0 & q \end{bmatrix} \text{ and } Q_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

- **Optimal observer**

$$S = \begin{bmatrix} s_1 & s \\ s & s_2 \end{bmatrix}$$

$$R = CSC^T + Q_2 = S + I$$

$$S = \Phi \left[S - SC^T R^{-1} CS \right] \Phi^T + Q_1 = \Phi S \left[I - (S+I)^{-1} S \right] \Phi^T + qI$$

$$(S+I)^{-1} = \begin{bmatrix} s_1+1 & s \\ s & s_2+1 \end{bmatrix}^{-1} = \frac{1}{(s_1+1)(s_2+1) - s^2} \begin{bmatrix} s_2+1 & -s \\ -s & s_1+1 \end{bmatrix}$$

$$\begin{aligned} (S+I)^{-1} &= \tfrac{1}{(s_1+1)(s_2+1)-s^2} \begin{bmatrix} s_2+1 & -s \\ -s & s_1+1 \end{bmatrix} \begin{bmatrix} s_1 & s \\ s & s_2 \end{bmatrix} \\ &= \tfrac{1}{(s_1+1)(s_2+1)-s^2} \begin{bmatrix} s_1(s_2+1)-s^2 & s(s_2+1)-ss_2 \\ -ss_1+s(s_1+1) & s_2(s_1+1)-s^2 \end{bmatrix} \end{aligned}$$

$$\begin{aligned} I-(S+I)^{-1} &= \tfrac{1}{(s_1+1)(s_2+1)-s^2} \begin{bmatrix} s_1(s_2+1)-s^2 & s(s_2+1)-ss_2 \\ -ss_1+s(s_1+1) & s_2(s_1+1)-s^2 \end{bmatrix} \\ &= \tfrac{1}{(s_1+1)(s_2+1)-s^2} \begin{bmatrix} (s_1+1)(s_2+1)-s^2-s_1(s_2+1)+s^2 & -s(s_2+1)+ss_2 \\ ss_1-s(s_1+1) & (s_1+1)(s_2+1)-s^2-s_2(s_1+1)+s^2 \end{bmatrix} \\ &= \tfrac{1}{(s_1+1)(s_2+1)-s^2} \begin{bmatrix} s_2+1 & -s \\ -s & s_1+1 \end{bmatrix} \end{aligned}$$

$$\begin{aligned} S &= \Phi S \left[I-(S+I)^{-1} S \right] \Phi^T + qI \\ &= \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} s_1 & s \\ s & s_2 \end{bmatrix} \left[I-(S+I)^{-1} S \right] \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} q & 0 \\ 0 & q \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0 \\ s_1 & s \end{bmatrix} \tfrac{1}{(s_1+1)(s_2+1)-s^2} \begin{bmatrix} s_2+1 & -s \\ -s & s1+1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} q & 0 \\ 0 & q \end{bmatrix} \end{aligned}$$

$$\begin{bmatrix} s_1 & s \\ s & s_2 \end{bmatrix} = \begin{bmatrix} q & 0 \\ 0 & \frac{s_1(s_2+1)-s^2}{(s_1+1)(s_2+1)-s^2} + q \end{bmatrix}$$

$$s_1 = q$$

$$s = 0 \text{ and } S = \begin{bmatrix} q & 0 \\ 0 & s_2 \end{bmatrix}$$

$$s_2 = \frac{s_1(s_2+1)-s^2}{(s_1+1)(s_2+1)-s^2} + q = \frac{q(s_2+1)}{(q+1)(s_2+1)} + q = \frac{q}{(q+1)} + q = \frac{2q+q^2}{q+1} = q\frac{q+2}{q+1}$$

$$R = S + I = \begin{bmatrix} q+1 & 0 \\ 0 & s_2+1 \end{bmatrix}$$

$$\begin{aligned} L &= \Phi S C^T R^{-1} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} q & 0 \\ 0 & s_2 \end{bmatrix} \frac{1}{(q+1)(s_2+1)} \begin{bmatrix} s_2+1 & 0 \\ 0 & q+1 \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0 \\ q & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{q+1} & 0 \\ 0 & \frac{1}{s_2+1} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ \frac{q}{q+1} & 0 \end{bmatrix} \end{aligned}$$

- **Equation of the observer**

$$\hat{x}(k+1) = \Phi\hat{x}(k) + \Gamma u(k) + L\left[y(k) - C\hat{x}(k)\right]$$

$$\hat{x}(k+1) = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \hat{x}(k) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(k) + \begin{bmatrix} 0 & 0 \\ \frac{q}{q+1} & 0 \end{bmatrix} \left[y(k) - \hat{x}(k)\right]$$

$$\left| \begin{array}{l} \hat{x}_1(k+1) = u(k) \\ \hat{x}_2(k+1) = \hat{x}(k) + \frac{q}{q+q}\left[y_1(k) - \hat{x}_1(k)\right] = \frac{1}{q+1}\hat{x}_1(k) + \frac{q}{q+1}y_1(k) \end{array} \right.$$

The second measurements is not exploited by the observer (an Ackemann design with only one measure should hence also be fine).

8 Dynamic programming

8.1 Bellman’s principle of optimality

The purpose of this chapter is to present a brief introduction to *dynamic programming*. Dynamic programming was developed by R.E. Bellman in the later 1950s (Bellman 1957, Bellman and Dreyfus 1962, Bellman and Kalaba 1965). It can be used to solve control problems for nonlinear, time-varying systems, and it is straightforward to program. The optimal control is expressed as a *state-variable feedback* in graphical or tabular form. Dynamic programming is based on Bellman’s *principle of optimality*:

An optimal policy has the property that no matter what the previous decision (i.e., controls) have been, the remaining decisions must constitute an optimal policy with regard to the state resulting from those previous decisions.

(8.1)

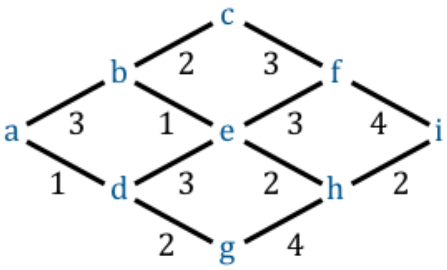


Figure 8.1 – Aircraft routing network.

We shall see that the principle of optimality serves to *limit the number of potentially optimal control strategies that must be investigated.* It also implies that optimal control strategies must be determined by working *backward* from the final stage; the optimal control problem is inherently a *backward-in-time problem.*

A simple routing example will serve to demonstrate all the points made so far.

8.1.1 Example: An aircraft Routing Problem

An aircraft can fly from left to right along the paths shown in Figure 8.1. Intersections $a, b, c, \ldots$ represent cities, and the numbers represent the fuel required to complete each path. Let us use the principle of optimality to solve the minimum-fuel problem.

We can quickly construct a state-variable feedback that shows both the optimal cost and the optimal control from *any* node to i. First, however, we must define what we mean by *state* in this example.

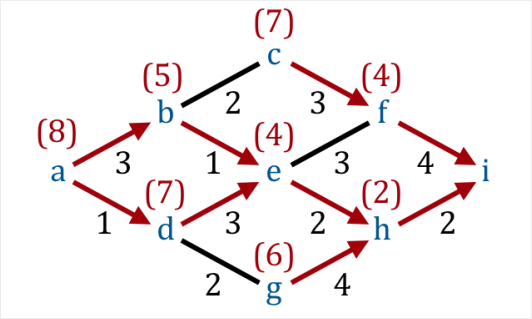


Figure 8.2 – Minimum-fuel state feedback control law.

We can label stages $k = 0$ through $k = N = 4$ of the decision-making process as shown in Fig. 8.2. (The arrowheads and numbers in parentheses should initially be disregarded: we shall show how to fill them in later). At each stage $k = 0, 1, \ldots, N-1$ a decision is required, and N is the final stage.

The current *state* is the node where we are making the current decision. Thus, the initial state is $x_0 = a$. At stage 1, the stage can be $x(1) = b$ or $x(1) = d$. Similarly, $x(2) = c, e$ or g; $x(3) = f$ or h; and the final state is constrained to be $x(N) = x(4) = i$.

To control $u(k)$ at stage k can be considered to be $u(k) = \pm 1$, where $u(k) = 1$ results in a move-up, and $u(k) = -1$ results in a move down to stage $k+1$.

Now it is clear that all we have on our hands is a minimum-fuel problem with fixed final state, constrained control and state values.

To find a minimum-fuel feedback control law using the principle of optimality, start at $k = N = 4$. No decision is required here, so decrement k to 3.

If $x(3) = f$, the optimal (only) control is $u(3) = -1$, and the cost is then 4. This is indicated by placing (4) above node f, and placing an arrowhead on path $f \to i$. If $x(3) = h$, the optimal control is $u(3) = 1$, with a cost of 2, which is now indicated on the figure.

Now decrement k to 2. If $x(2) = c$, then $u(2) = -1$ with a cost to go of $4+3 = 7$. This information is added to the figure. If $x(2) = e$, then we must make a decision. If we apply $u(2) = 1$ to get to f, and then go via the optimal path to i, the cost is $4+3 = 7$. On the other hand, if we apply $u(2) = -1$ at e and go to h, the cost is $2+2 = 4$. Hence, at e the optimal decision is $u(2) = -1$ with a cost to go of 4. Add this information to the figure.

If $x(2) = g$, there is only one choice: $u(2) = 1$ with a cost to go of 6.

By successively decrementing k and continuing to compare the control possibilities allowed by the principle of optimality, we can fill in the remainder of the control decisions (arrowheads) and optimal costs to go shown in Fig. 8.2. It should be clearly realized that the only control sequences we are allowed to consider are those *the last portions of which are optimal sequences.*

Note that when $k = 0$, a control of either $u(0) = 1$ or $u(0) = -1$ yields the same cost to go of 8; the optimal control for $k = 0$ is not unique.

To examine what we have just constructed, suppose we now are told to find the minimum-fuel path from node d to the destination i. All we need to do is begin at d and follow the arrows! The optimal control $u^*(k)$ and the cost to go at each stage k are determined if we know the value of $x(k)$. This, however, is exactly the meaning of state-variable feedback. Therefore, the grid labeled with arrows as in Fig. 8.2 is just a graphical *state feedback control law* for the minimum-fuel problem!

Our feedback law tells us how to get from *any* state to the fixed signal state $x(4) = x(N) = i$. If

we change the $final$ state, however (e.g., to $x(3) = x(N) = f$), then the entire grid must be redone.

Several points should be noted about this example. First, note that, according to Fig. 8.2, there are two paths from a to i with the same cost of 8 : $a \to b \to e \to h \to i$ and $a \to d \to e \to h \to i$. Evidently, the optimal solution found by dynamic programming may not be unique.

Second, suppose we had attempted, in ignorance of the optimality principle, to determine an optimal route from a to i by working *forward*. Then a near-sighted decision maker at a would compare the costs of traveling to b and d, and decide to go to d. The next myopic decision would take it to g. From there on there is no choice: he must go via h to i. The net cost of this strategy is $1+2+4+2=9$, which is nonoptimal.

We can say that “any portion of an optimal path is optimal". For example, an optimal route from a to e is $a \to b \to e$, while the optimal route from e to i is $e \to h \to i$.

Finally, let us point out that Bellman's principle of optimality has reduced the number of required calculations by *restricting the number of decisions that must be made.* We could determine the optimal route from a to i by comparing *all* possible paths from a to i. This would require many more calculations than we used.

We shall now discuss the application of Bellman's principle of optimality to the control of dynamic systems.

8.2 Discrete-time systems

We discussed optimal control for linear dynamical systems in chapter 7 in the cases in which the states and the control variables can vary freely. Dynamic programming, on the other hand, can easily be applied to nonlinear systems, and the more constraints there are on the control and state variables, the easier the solution.

Let the system be

$$x(k+1) = f[x(k), u(k), k] \tag{8.2}$$

Suppose we associate with this system the performance index

$$J[x(i), i] = \phi[N, x(n)] + \sum_{k=1}^{N-1} L[x(k), u(k), k], \tag{8.3}$$

where $[i, N]$ is the time interval of interest. We have shown the dependance of J on the initial time and state.
We want to use the principle of optimality (8.1) to select the control sequence $u(k)$ to minimize

(8.3). First, we need to determine what (8.1) looks like in this situation. Let us express it in a more mathematical form.

Suppose we have computed the optimal cost $J^*[x(k+1), k+1]$ from time $k+1$ to the terminal time N for all possible states $x(k+1)$, and that we have also found the optimal control sequences from time $k+1$ to N for all $x(k+1)$. The optimal cost results when the optimal control sequence $u^*(k+1), u^*(k+2), \cdots, u^*(N-1)$ is applied to the system with a state of $x(k+1)$. Note that the optimal control sequence depends on $x(k+1)$.

If we apply any arbitrary control $u(k)$ at time k and then use the known optimal control sequence from $k+1$ on, the resulting cost will be

$$L[x(k), u(k), k] + J^*[x(k+1), k+1], \tag{8.4}$$

where $x(k)$ is the state at time k, and $x(k+1)$ is given by (8.2). According to Bellman, the optimal cost from time k is equal to

$$J^*[x(k), k] = \min_{u(k)}\{L[x(k), u(k), k] + J^*[x(k+1), k+1]\}, \tag{8.5}$$

and the optimal control $u^*(k)$ at time k is the $u(k)$ that achieves this minimum.

Equation (8.5) is the principle of optimality for discrete systems. Its importance lies in the fact that it allows us to optimize over *only one control vector at a time* by working backward from N. It is called the *functional equation of dynamic programming* and is the basis for computer implementation of Bellman's method.

In general we can specify additional constraints, such as the requirement that $u(k)$ belong to some admissible control set.

Without realizing it, we worked Example 8.1.1 by applying (8.5). Let us consider next a systems-oriented example.

8.2.1 Example: Optimal Control of a Discrete System Using Dynamic Programming

Let the system

$$x(k+1) = x(k) + u(k) \tag{8.6}$$

have an associated performance index of

$$J(0) = x^2(N) + \frac{1}{2}\sum_{k=0}^{N-1} u^2(k), \tag{8.7}$$

where the final time N is 2. The control is constrained to take on values of

$$u(k) = -1, -0.5, 0, 0.5, 1, \tag{8.8}$$

and the state is constrained to take only the values

$$x(k) = 0, 0.5, 1.0, 1.5 \tag{8.9}$$

The control value constraint is not unreasonable, since a minimum-time optimal control takes on only values of $0, \pm 1$. The state value constraint in this problem is also reasonable, since if the state initially takes on one of the admissible values (8.9), then under the influence of the allowed controls (8.8) subsequent states will take on integer or half-integer values. An equivalent constraint to (8.9) is therefore $x(0) = 0, 0.5, 1.0, 1.5$, and

$$0 \leqslant x(k) \leqslant 1.5. \tag{8.10}$$

This is a positivity condition and a magnitude constraint on the state, which is often reasonable in physical situations.

In an actual application N would almost certainly be greater than 2, but to handle large N we would use a computer program designed on the basis of our work in this example.

Now, the optimal control problem is to find an admissible control sequence $u^*(0), u^*(1)$ that minimizes $J(0)$ while resulting in an admissible state trajectory $x^*(0), x^*(1), x^*(2)$. We should like for $u^*(k)$ to be specified as a state feedback control law.

To solve this problem, we can use the principle of optimality in the form (8.5).

First, we set up a grid of $x(k)$ versus k as shown in Fig. 8.3 (Disregard the arrowheads, numbers above the lines and numbers in parentheses for now.) the lines on the figure are drawn according to the state equation (8.6). For each admissible $x(k)$, the lines extend toward $x(k+1) = x(k) + u(k)$, with the control value $u(k)$ written at the end of each line. Only $u(k)$ that are both admissible by (8.8) and result in admissible values of $x(k+1)$ given in (8.9) are considered. Thus, for example, if $x(0) = 1.0$, then the admissible $u(0)$ are $-1, 0.5, 0, 0.5$, which result respectively in

$$x(1) = x(0) + u(0) = 1 - 1 = 0,$$
$$x(1) = 1 - 0.5 = 0.5,$$
$$x(1) = 1 + 0 = 1,$$
$$x(1) = 11 + 0.5 = 1.5.$$

The lines emanating from the node $x(0) = 1.0$ are directed toward these values of $x(1)$.

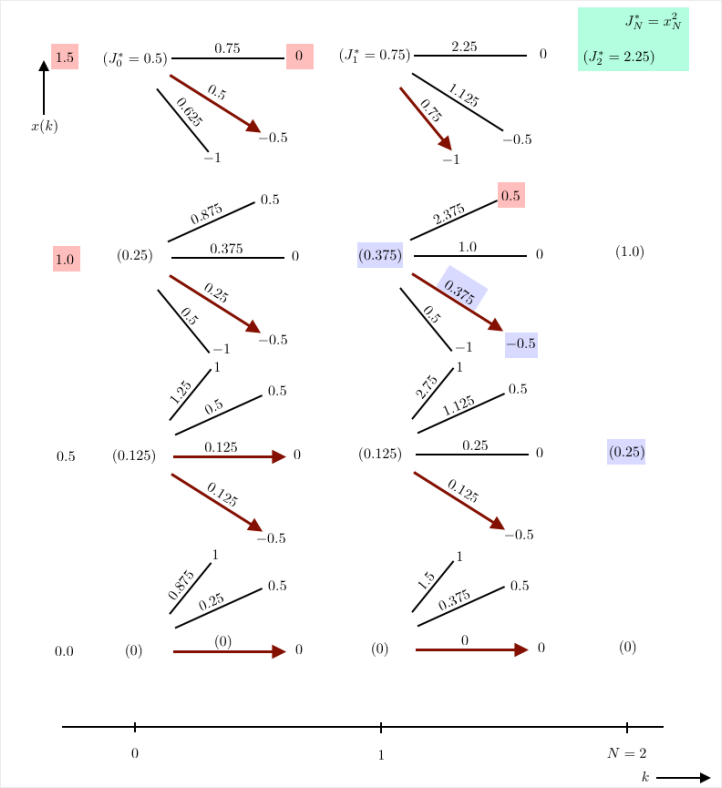


Figure 8.3 – Decision grid for solving the optimal control problem by dynamic programming.

Figure 8.3 incorporates all of the information in the state equation (8.6) and the constraints (8.8), (8.9). To compute the optimal control using (8.5), we need to work with the cost function (8.7). Write

$$J(k) = \frac{1}{2}u^2(k) + J^*(k+1) \tag{8.11}$$

as the *admissible costs* at time k; an admissible cost $J(k)$ is one whose last portion $J^*(k+1)$ is optimal. In terms of these $J(k)$, (8.5) becomes

$$J^*(k) = \min_{u(k)}[J(k)]. \tag{8.12}$$

(The dependence of $J(k)$ and $J^*(k)$ on $x(k)$ is not explicitly shown.) Write down (8.11) for each k beneath the grid in Fig. 8.3.
To compute $u^*(k)$ and $J^*(k)$ for each $x(k)$ using (8.12), we need to work backward from $k = N$. First, let $k = N = 2$. The final costs

$$J^*(N) = x^2(N) \tag{8.13}$$

must be evaluated for each admissible final state. $J^*(N)$ represents the penalty for being in the final state with value $x(N)$. These costs are written down in parentheses in Fig. 8.3 at the location of each of the final states $x(N) = 0, x(N) = 0.5, x(N) = 1.0$, and $x(N) = 1.5$. They are $J^*(2) = 1.5^2 = 2.25$ for $x(2) = 1.5$, $J^*(2) = 1.0$ for $x(2) = 1.0$, and so on.

Now decrement to $k = 1$. For each possible state $x(1)$ and for each admissible control $u(1)$ we must now compute (8.11). Begin with $x(1) = 1.5$. If we apply $u(1) = 0$, then $x(2) = x(1) + u(1) = 1.5$, so that $J^*(2) = 2.25$. According to (8.11),

$$J(1) = u^2(1)/2 + J^*(2) = 0^2/2 + 2.25 = 2.25 \tag{8.14}$$

This cost is written above the line representing $u(1) = 0$ at $x(1) = 1.5$.

Now suppose we apply $u(1) = -0.5$ when $x(1) = 1.5$. Then $x(2) = x(1) + u(1) = 1.0$, and so $J^*(2) = 1.0$. Thus

$$J(1) = \frac{u^2(1)}{2} + J^*(2) = \frac{(-0.5)^2}{2} + 1.0 = 1.125. \tag{8.15}$$

Write this above the line for $u(1) = -0.5$ applied to $x(1) = 1.5$.

If we apply $u(1) = -1$ when $x(1) = 1.5$, then $x(2) = x(1) + u(1) = 0.5$, so that $J^*(2) = 0.25$. Thus, in this case,

$$J(1) = \frac{u^2(1)}{2} + J^*(2) = \frac{(-1)^2}{2} + 0.25 = 0.75. \tag{8.16}$$

Having computed $J(1)$ for each control possibility with $x(1) = 1.5$, we now need to decide on the value of $u^*(1)$ if $x(1) = 1.5$. According to (8.12), $u^*(1)$ is simply the value of $u(1)$ that yields the smallest $J(1)$. Therefore

$$u^*(1) = -1, J^*(1) = 0.75. \tag{8.17}$$

This can be shown on the figure by placing an arrowhead on the $u(1) = -1$ control path leading from $x(1) = 1.5$ and placing the values of $J^*(1)$ in parentheses at the location corresponding to $x(1) = 1.5$.

Now we focus our attention on $x(1) = 1.0$. Using (8.11) to determine $J(1)$ for the admissible values of control $u(k) = 0.5, 0, -0.5, -1$, we get the numbers shown above each line emanating from $x(1) = 1.0$ in the figure. The smallest value of $J(1)$ is 0.375, which occurs for $u(1) = -0.5$.

Hence, if $x(1) = 1.0$, then

$$u^*(1) = -0.5, \quad J^*(1) = 0.375 \tag{8.18}$$

This is indicated by placing an arrowhead on the $u(1) = -0.5$ path and $J^*(1) = 0.375$ in parentheses at the location $x(1) = 1.0$.

In a similar manner we obtain $u^*(1)$ and $J^*(1)$ for $x(1) = 0.5$ and $x(1) = 0.0$. See the figure for the results.

Now we decrement k to 0. Let $x(0) = 1.5$. For this value of the initial state, we use (8.11) with each of the possible control values $u(0) = 0, -0.5$, and -1 to compute the associated values of $J(0) = u^2(0)/2 + J^*(1)$, where $J^*(1)$ is the number in parentheses at the state $x(1)$ resulting from $x(1) = 1.5 + u(0)$. The values of $J(0)$ found are shown in the figure. The smallest value of $J(0)$ is 0.5, which occurs for $u(0) = -0.5$; so if $x(0) = 1.5$, then

$$u^*(0) = -0.5, \quad J^*(0) = 0.5. \tag{8.19}$$

In a similar fashion we compute $u^*(0)$ and $J^*(0)$ for $x(0) = 1.0, 0.5$ and 0.0. The resulting values are indicated in the figure. Note that if $x(0) = 0.5$, then a control of either $u(0) = 0$ or $u(0) = -0.5$ results in the optimal cost of $J^*(0) = 0.125$; the optimal control for this initial state is not unique.

To see what we have just constructed, we can redraw Fig. 8.3 showing only the optimal controls $u^*(k)$ and costs to go $J^*(k)$. See Fig. 8.4. Now, suppose $x(0) = 1.0$. Then to find the optimal state trajectory we need only follow the arrows. It is

$$x^*(0) = 1.0 \quad x^*(1) = 0.5 \quad x^*(0) = 0. \tag{8.20}$$

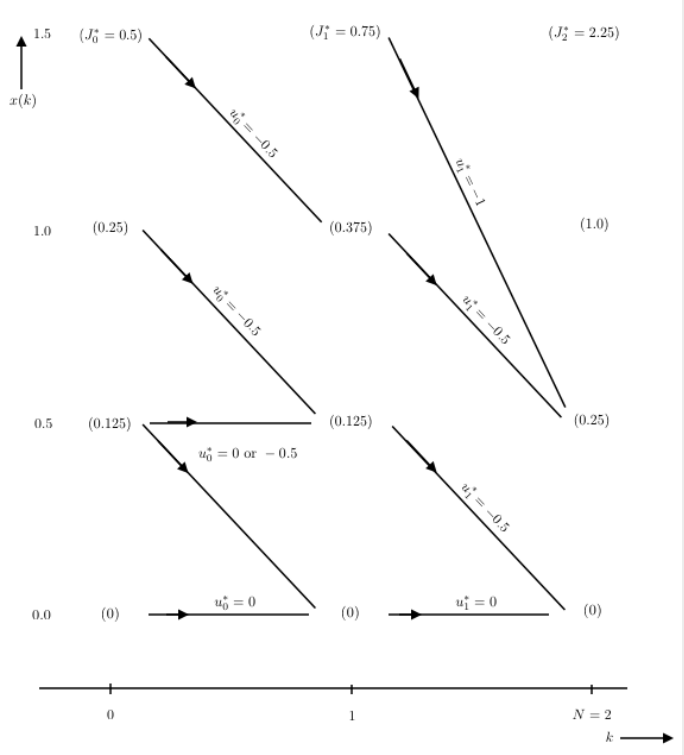


Figure 8.4 – State feedback control law found by dynamic programming.

Before we even apply the control, we know the optimal cost will be

$$J^*(0) = 0.25. \tag{8.21}$$

The optimal control sequence to be applied to the plant is

$$u^*(0) = -0.5, \quad u^*(1) = -0.5. \tag{8.22}$$

Whatever the initial state $x(0)$ is, the optimal control $u^*(0)$ and cost to go $J^*(0)$ are determined by Fig. 8.4 once $x(0)$ is given. The current control $u^*(k)$ is given once the current state $x(k)$ is known; hence the figure is just the optimal control law given in *state feedback* form.
In the dynamic programming approach more constraints mean that fewer control possibilities must be examined; constraints *simplify* the dynamic programming problem.

Figure 8.4 has the form of a *vector field* that tends to force the state down toward the origin as k increases. By changing the control weighting in $J(0)$ in (8.7), the shape of the vector field changes. For instance, if we modify the cost index to

$$J(0) = x^2(2) + 2\sum_{k=0}^{1} u^2(k) \tag{8.23}$$

and repeat the above procedure, the state feedback control law of Fig. 8.5 results. A heavier control weighting has resulted in a tendency to use less control effort. The family of optimal state trajectories shown in figures like these is called a *field of extremals*.

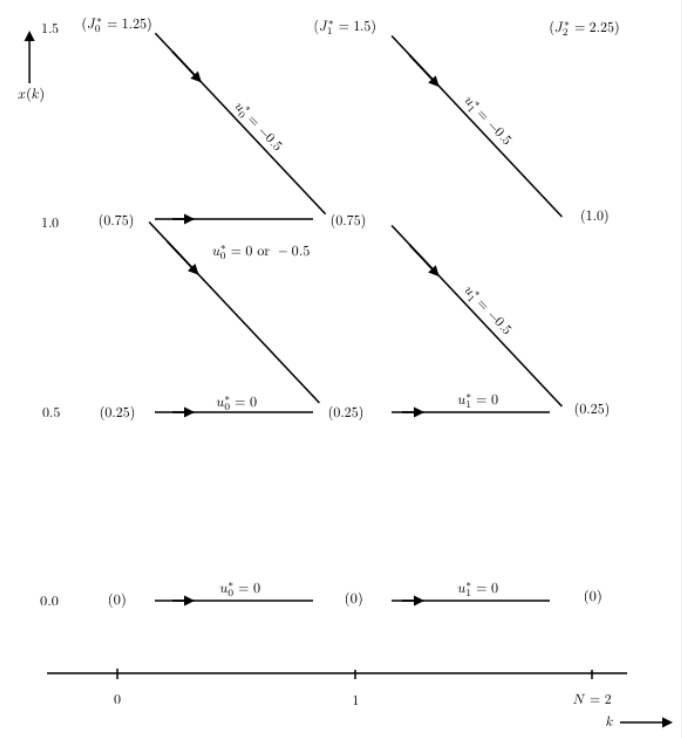


Figure 8.5 – State feedback control for modified performance index.

This problem could be solved without using the optimality principle; but then *every possible* control sequence would need to be examined, and the required number of calculations would drastically increase. We see here how easy it is to extend the dynamic programming approach to nonlinear systems.

8.2.2 Example: Discrete Linear Quadratic Regulator via Dynamic Programming

Let the system

$$x(k+1) = \Phi x(k) + \Gamma u(k) \tag{8.24}$$

have associated performance index

$$J(i) = \frac{1}{2} x^T(N) S(N) x(N) + \frac{1}{2} \sum_{k=i}^{N-1} [x^T(k) Q x(k) + u^T(k) R u(k)] \tag{8.25}$$

with $S(N) \geqslant 0, Q \geqslant 0$, and $R > 0$. If the system and weighting matrices are time-varying, the development to follow still holds. It is desired to find the optimal control $u^*(k)$ on the fixed time interval $[i, N]$ that minimizes $J(i)$. The initial state $x(i)$ is given and the final state $x(N)$ is free.

To begin, let $k = N$ and write

$$J^*(N) = \frac{1}{2}x^T(N)S(N)x(N), \tag{8.26}$$

which is the penalty for being in state $x(N)$ at time N. Now decrement k to $N-1$ and write

$$J(N-1) = \frac{1}{2}x^T(N-1)Qx(N-1) + \frac{1}{2}u^T(N-1)Ru(N-1) + \frac{1}{2}x^T(N)S(N)x(N). \tag{8.27}$$

According to (8.5), we need to find $u^*(N-1)$ by minimizing (8.27). To do this, use state equation (8.24) to write

$$J(N-1) = \frac{1}{2}x^T(N-1)Qx(N-1) + \frac{1}{2}u^T(N-1)Ru(N-1) + \frac{1}{2}[\Phi x(N-1) + \Gamma u(N-1)]^T S(N)[\Phi x(N-1) + \Gamma u(N-1)]. \tag{8.28}$$

Since there are no constraints, the minimum of $J(N-1)$ is found by setting

$$0 = \frac{\partial J(N-1)}{\partial u(N-1)} = Ru(N-1) + \Gamma^T S(N)[\Phi x(N-1) + \Gamma u(N-1)]. \tag{8.29}$$

Solving for the optimal control yields

$$u^*(N-1) = -[\Gamma^T S(N)\Gamma + R]^{-1}\Gamma^T S(N)\Phi x(N-1). \tag{8.30}$$

Defining the Kalman gain as

$$K(N-1) \equiv [\Gamma^T S(N)\Gamma + R]^{-1}\Gamma^T S(N)\Phi, \tag{8.31}$$

we can write

$$u^*(N-1) = -K(N-1)x(N-1). \tag{8.32}$$

The optimal cost to go from $k = N-1$ is found by substituting (8.31) into (8.28). If we do this

and simplify, the result is

$$J^*(N-1) = \frac{1}{2}x^T(N-1)\{[\Phi - \Gamma K(N-1)]^T S(N)[\Phi - \Gamma K(N-1)] + K^T(N-1)RK(N-1) + Q\}x(N-1). \quad (8.33)$$

If we define

$$S(N-1) \equiv [\Phi - \Gamma K(N-1)]^T S(N)[\Phi - \Gamma K(N-1)] + K(N-1)^T RK(N-1) + Q, \quad (8.34)$$

this can be written

$$J^*(N-1) = \frac{1}{2}x^T(N-1)S(N-1)x(N-1). \quad (8.35)$$

Now decrement to $k = N-2$. Then

$$J(N-2) = \frac{1}{2}x^T(N-2)Qx(N-2) + \frac{1}{2}u^T(N-2)Ru(N-2) + J^*(N-1) \quad (8.36)$$

are the admissible costs for $N-2$, since these are the costs that are optimal from $k = N-1$ on. To determine $u^*(N-2)$, according to (6.5), we must minimize (8.36).
Now, let us be a little tricky to save some work. Note that (8.36) is of the same form as (8.27). The optimal control and cost to go for $k = N-2$ are therefore given by (8.31), (8.32) and (8.34), (8.35) with N there replaced by $N-1$.
If we continued to decrement k and apply the optimality principle, the result for each $k = N-1, \cdots, 1, 0$ is

$$K(k) = [\Gamma^T S(k+1)\Gamma + R]^{-1}\Gamma^T S(k+1)\Phi, \quad (8.37)$$

$$u^*(k) = -K(k)x(k), \quad (8.38)$$

$$S(k) = [\Phi - \Gamma K(k)]^T S(k+1)[\Phi - \Gamma K(k)] + K^T(k)RK(k) + Q, \quad (8.39)$$

$$J^*(k) = \frac{1}{2}x^T(k)S(k)x(k), \quad (8.40)$$

where the final condition $S(N)$ for (8.39) is given in (8.25).
Equation (8.19) is the Joseph stabilized Riccati equation, and so these results are identical to those found in chapter 7 (optimal control) using the Lagrange multiplier approach.

8.3 Exercises

8.3.1 Vessel Loading

Problem

A vessel is to be loaded with N items. Each item $k = 1, ..., N$ has a weight of $w(k)$ and a known value of $v(k)$. The maximum allowed weight of the cargo is W. We want to determine the most valuable cargo load without overloading the vessel.

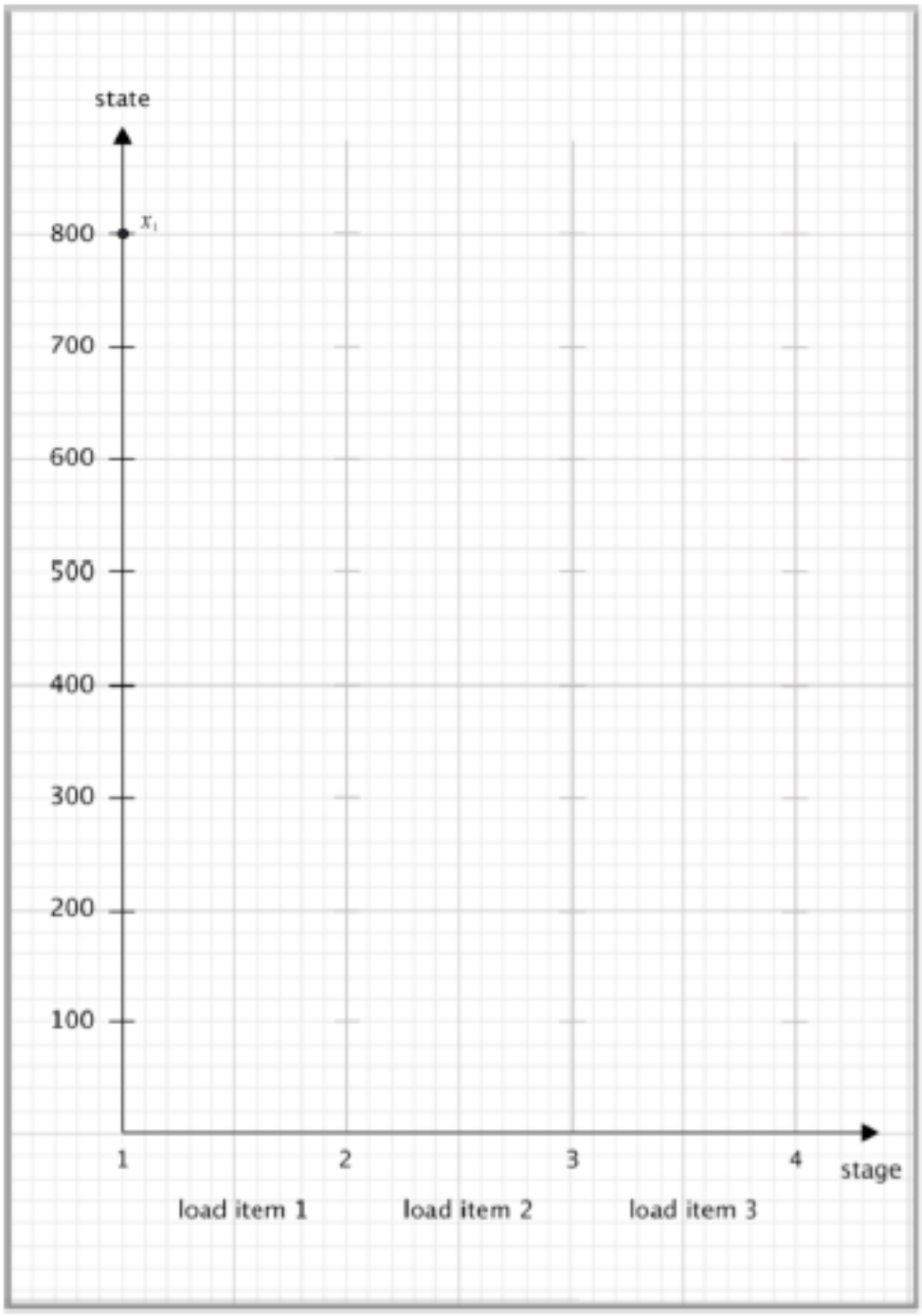
With $u(k)$ the number of units of item k loaded and $x(k)$ the weight capacity available for items $k, k+1, ..., N$, the state equation is $x(k+1) = x(k) - u(k)w(k)$.

The problem is to maximize $J_1 = \sum_{k=1}^{N} u(k)v(k)$ subject to the constraint $\sum_{k=1}^{N} u(k)w(k) \leqslant W$

For $N = 3$ and $W = 800$ kg

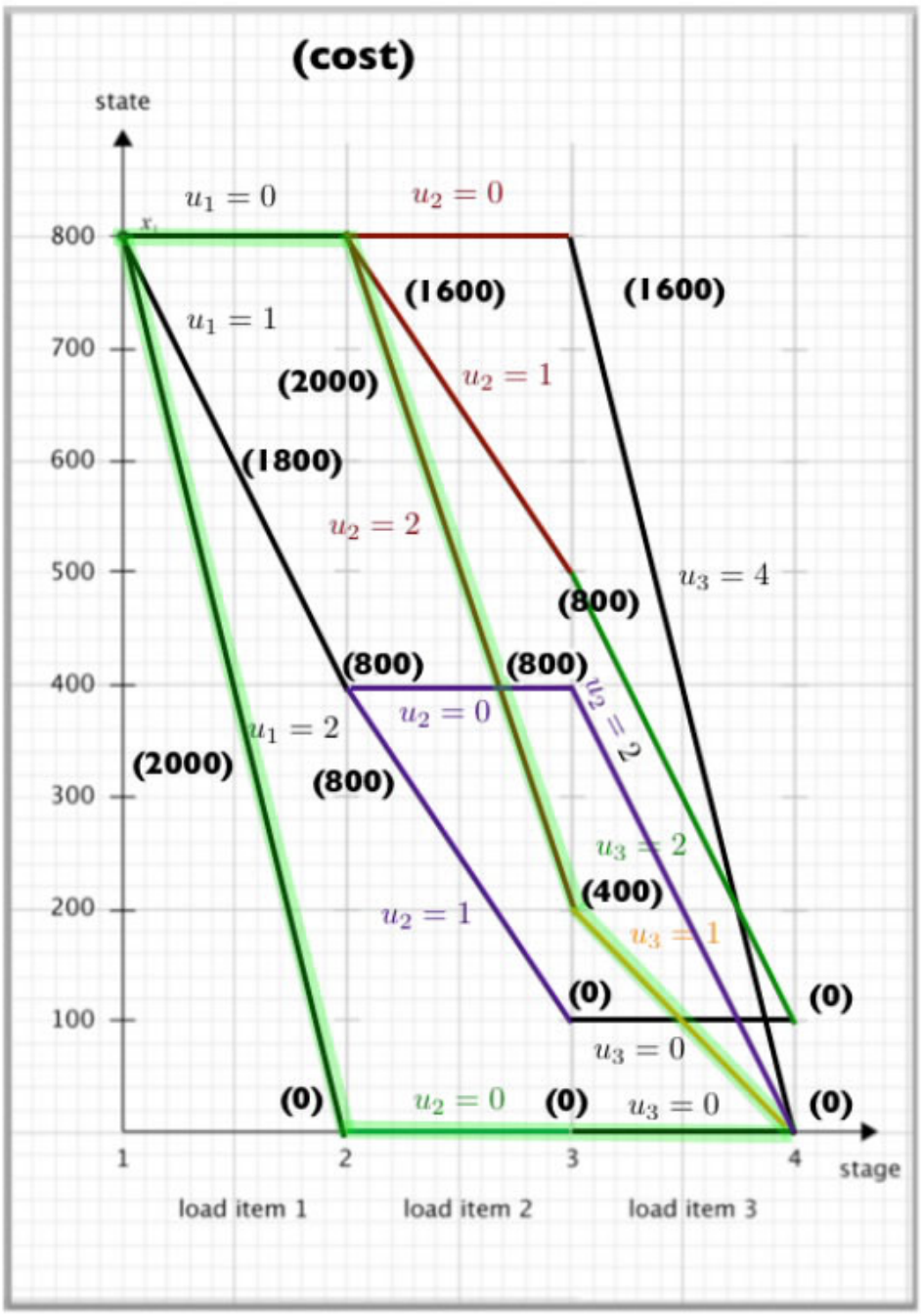
Item	Weight (kg)	Value (CHF)
1	400	1000
2	300	800
3	200	400

1. From the initial state $x(1) = W$, compute the possible values of $x(2)$. From the possible values of $x(2)$, compute the possible values of $x(3)$. Place them in the grid provided below.
2. Find the optimal loading policy $(u^*(1), u^*(2), u^*(3))$ knowing that $J^*(4) = 0$.


state
x_1
800
700
600
500
400
300
200
100
1
2
3
4
stage
load item 1
load item 2
load item 3

Solution

There is two optimal solutions: (2, 0, 0) and (0, 2, 1).



The maximum value of the load is 2000, either with 2 item of type 1, or two items of type 2 and one of type 3.

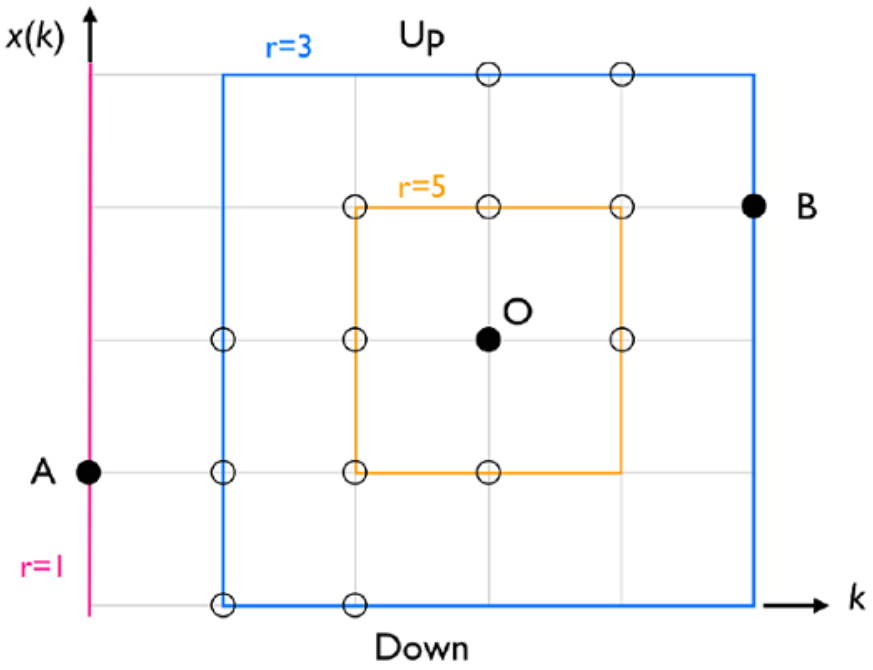
8.3.2 Quadrotor

Problem

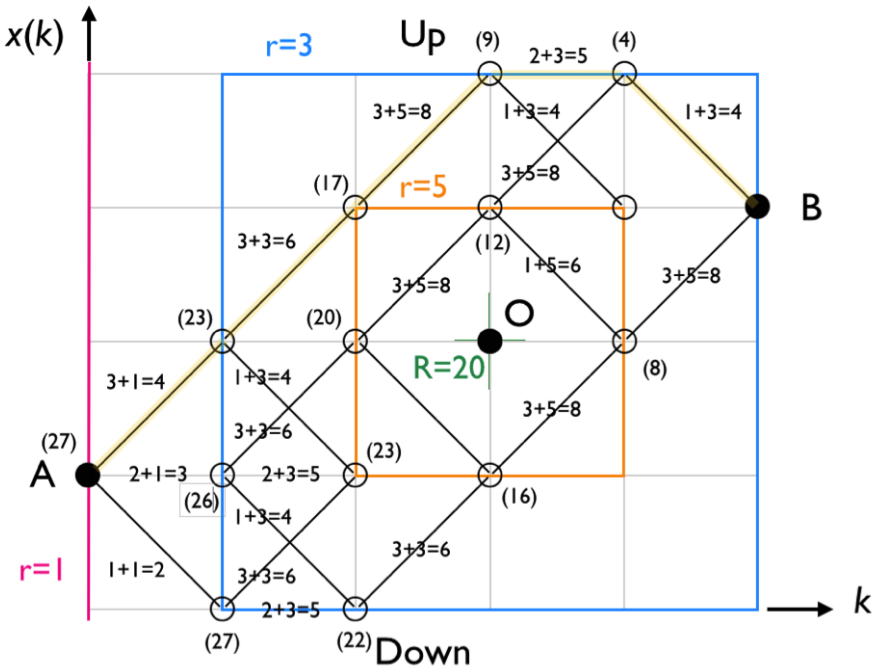
A quadrotor moves in a vertical plan between two points A and B while avoiding an obstacle located at point O. It can only go down in diagonal, straight horizontally or up in diagonal at each stage k with an input $u(k)$ of 1, 2, or 3 respectively. The displacement cost on each segment is $u(k) + r(k)$, where $r(k)$ is the repulsion potential at the origin of the segment (colored potential lines with the corresponding value). There is no final cost at point B. The possible states $x(k)$ are the black or white dots.

1. Determine graphically and draw the optimal path for the quadrator below. *Show the cost of each segment in the plan (you can ask for additional sheets with the map if you want.*

2. What is the total cost for the manoeuver and the corresponding sequence of inputs ?



Solution


x(k)
r=3
Up
(9)
2+3=5
(4)
3+5=8
1+3=4
1+3=4
(17)
r=5
3+5=8
B
3+3=6
(12)
1+5=6
3+5=8
3+5=8
(23)
(20)
O
R=20
(8)
3+1=4
1+3=4
3+5=8
3+3=6
(27)
A
2+1=3
(23)
2+3=5
(26)
(16)
1+3=4
3+3=6
r=1
1+1=2
3+3=6
2+3=5
(27)
(22)
Down
k

9 Decentralized Navigation Functions

by *Laleh Makarem*, React Group, School of Engineering, EPFL

9.1 Potential Functions

For many applications a robot or a team of robots can accomplish the task more efficiently and precisely than humans. In the next chapters, we will study examples of applications in which a group of robots create a formation or collaborate to reach a consensus. In this chapter we study the problem of collision avoidance for a single robot. Then we extend the solutions for a team of robots in a way that each robot is responsible for its safety.

The way swarms of insects or flocks of birds can travel in large, dense groups without colliding has inspired many techniques in robotics. Even in the presence of external obstacles these member of groups are capable of smoothly avoiding collisions. There is strong reasons to believe that the rules, or protocols, each member of the group follows are quite basic, and yet the collective or global motion is quite remarkable. The potential function method that we are studying here is based on models of smooth reactions toward destination and against collision hazards.

A potential function is a differentiable real-valued function $\phi : \mathbb{R}^m \rightarrow \mathbb{R}$. The value of a potential function can be seen as energy and thus the gradient of the potential function is force. The gradient is a vector $\bigtriangledown\phi(q) = D\phi(q)^T = [\frac{\partial\phi}{\partial q_1}, \ldots, \frac{\partial\phi}{\partial q_m}]^T$ which points in the direction that locally increases the ϕ function. We can use the gradient to define a vector field, which assigns a vector to every point in the working space.

The potential function approach directs a robot as if it were a particle moving in a gradient vector field like an electrical field. A gradient can be intuitively viewed as a force vector acting on a positively charged particle (robot) which is attracted to a negatively charged goal (destination). Obstacles also have positive charge which forms a repulsive force directing the robot away form obstacles. The combination of repulsive and attractive forces hopefully directs robot from the start location to the goal location while avoiding obstacles (Fig.9.1).

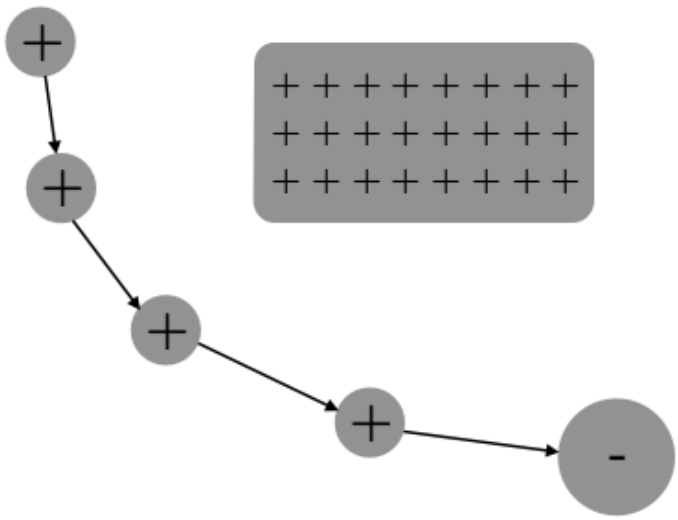


Figure 9.1 – The negative charge (destination) attracts the particle (robot) and the positive charges (obstacle) repels it, resulting in a path shown on dashed line.

Note that in this chapter, we mainly deal with first-order systems and the gradients are velocity vectors instead of force vectors. The main rational behind this, is that robots are normally capable of controlling their speed. Potential functions can be viewed as a framework where the robots move from a “high-value" state to a “low-value" state. The robot follows a path “downhill" by following the negated gradient of the potential function. Following such a path is called gradient descent:

$$\dot{c(t)} = - \bigtriangledown \phi(c(t)) \tag{9.1}$$

The robot completes its motion when it reaches a point where the gradient vanishes. i.e. it has reached a $q^\star$ where $\bigtriangledown\phi(q^\star) = 0$. Such a point $q^\star$ is called a critical point of ϕ. The point $q^\star$ is either a maximum, a minimum or a saddle point (Fig.9.2). For determining the type of critical point, we should look at the second derivative of ϕ .

For real-values functions, this second derivative is the Hessian matrix

$$\begin{pmatrix} \frac{\partial^2 \phi}{\partial {q_1}^2} & \cdots & \frac{\partial^2 \phi}{\partial q_1 \partial q_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 \phi}{\partial q_1 \partial q_n} & \cdots & \frac{\partial^2 \phi}{\partial {q_n}^2} \end{pmatrix}$$

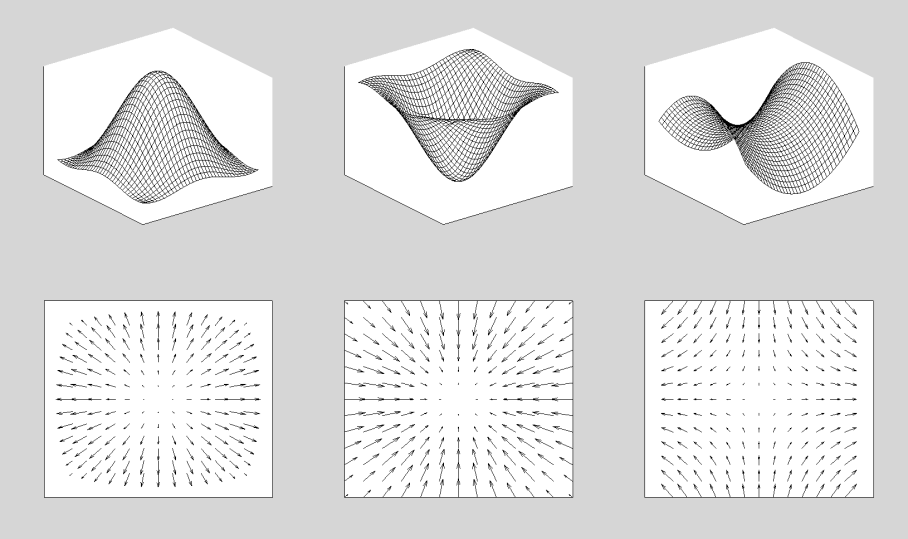


Figure 9.2 – Different types of critical points. Top graphs show the functions, down graphs show the gradients.

When the Hessian is nonsingular at $q^\star$, the critical point at $q^\star$ is called non-degenerate, meaning that the critical point is isolated. If the Hessian is positive-definite, the critical point is a local minimum, if it is negative-definite, then the critical point is a local maximum otherwise the critical point is a saddle point.

In practice, we do not have to compute the Hessian because the robot generally completes its motion at a local minimum, not at a local maximum or a saddle point. Since gradient descent decreases the function ϕ, the robot can not arrive at a local maximum. However, even if the robot starts at a local maximum point and if the ϕ function is non-degenerate, any little perturbation (for example some noise from sensors) gets the robot out of that point. Therefore, the only critical point where the robot can end up is a local minimum. In many applications this is where the destination of the robot is.

9.1.1 Additive Attractive/Repulsive Potential

The intuition behind the attractive/repulsive potential is straight forward. The goal attracts the robot while the obstacles repel it. The sum of these effects draws the robot to the goal while deflecting it from obstacles. The potential function can be constructed as the sum of attractive and repulsive potentials

$$\phi(q) = \phi_{att}(q) + \phi_{rep}(q) \tag{9.2}$$

The attractive potential

Depending on the application, there are several criteria that the attractive potential field ϕ_{att}

should satisfy. First, ϕ_{att} should be monotonically increasing with distance from destination of the robot. The simplest choice is the conic potential, measuring a scaled distance to the goal, i.e , $\phi(q) = \zeta d(q, q_{goal})$ where the ζ is the scaling parameter and q_{goal} is the destination of the robot. The attractive gradient is $\bigtriangledown\phi(q) = \frac{\zeta}{d(q,q_{goal})}(q - q_{goal})$. The gradient points away from the goal with magnitude ζ at all points of the configuration space except the goal, where it is undefined. Starting from any point other than the goal, by following the negated gradient, a path is traced toward the goal.

Using this method, gradient descent may have “chattering" problems very near to the origin. Because there is a discontinuity in the attractive gradient at that point. For this reason, we can use a quadratic potential function that is continuously differentiable.

$$\phi_{att} = \frac{1}{2}\zeta d^2(q, q_{goal}) \tag{9.3}$$

with the gradient

$$\bigtriangledown\phi_{att} = \bigtriangledown(\frac{1}{2}\zeta d^2(q, q_{goal})) = \frac{1}{2}\zeta \bigtriangledown d^2(q, q_{goal}) = \zeta(q - q_{goal}) \tag{9.4}$$

which is a vector based at q, point away from q_{goal}, and has a magnitude proportional to the distance from q to q_{goal}. The farther away q is from q_{goal}, the bigger the magnitude of the vector. In other words, when the robot is far away from the goal, the robot quickly approaches it. When the robot is close to the goal, the robot slowly approaches it. This feature is useful for mobile robots because it reduces “overshoot" of the goal (resulting from step quantization). However, it grows without bound as q moves away from q_{goal}. If q_{start} is far from q_{goal}, this may produce a desired velocity that is too large. For this reason, we may choose to combine the quadratic and conic potential so that the conic potential attracts the robot when it is far from q_{goal} and the quadratic potential attracts the robot when it is near q_{goal}. Of course for sake of continuity it is necessary that the gradient be defined at the boundary between the conic and quadratic portions. Such a field can be defined by:

$$\phi_{att}(q) = \left\{ \begin{array}{ll} \frac{1}{2}\zeta d^2(q, q_{goal}) & d(q, q_{goal}) \leq d^{\star}_{goal} \\ \\ d^{\star}_{goal}\zeta d(q, q_{goal}) - \frac{1}{2}\zeta(d^{\star}_{goal})^2 & d(q, q_{goal}) \geq d^{\star}_{goal} \end{array} \right\} \tag{9.5}$$

and in this case we have

$$\nabla\phi_{att}(q) = \left\{ \begin{array}{ll} \frac{1}{2}\zeta(q - q_{goal}) & d(q, q_{goal}) \leq d^{\star}_{goal} \\ \\ \frac{d^{\star}_{goal}\zeta(q-q_{goal})}{d(q,q_{goal})} & d(q, q_{goal}) > d^{\star}_{goal} \end{array} \right\} \tag{9.6}$$

where $d^{\star}_{goal}$ is the threshold distance from the goal where the potential switches between the conic and quadratic functions. The gradient is well-defined in the boundary of the two functions. At the boundary the $d(q, q_{goal}) = d^{\star}_{goal}$, the gradient of the quadratic function is equal to the gradient of the conic potential $\nabla\phi_{att}(q) = \zeta(q - q_{goal})$.

The repulsive potential

A repulsive potential is supposed to keep the robot away from any obstacle in the work space of the robot. The strength of the repulsive force depends on the proximity of the robot to the obstacle. The closer the robot gets to the obstacle, the stronger the repulsive force should be. It is clear that like attractive potential, many function could be good candidate for repulsive potential. Here, we give an example of such a function (Fig.9.3):

$$\phi_{rep}(q) = \left\{ \begin{array}{ll} \frac{1}{2}\eta(\frac{1}{D(q)} - \frac{1}{Q^{\star}})^2 & D(q) \leq Q^{\star} \\ \\ 0 & D(q) > Q^{\star} \end{array} \right\} \tag{9.7}$$

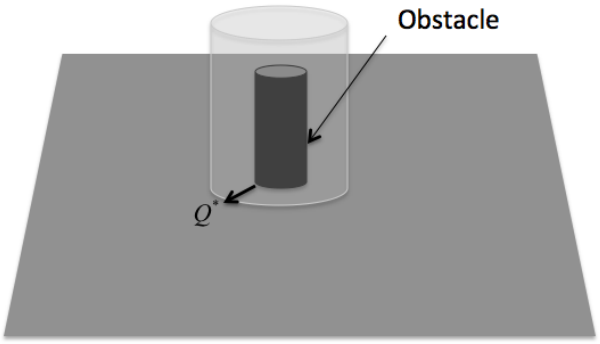


Figure 9.3 – Repulsive force from the obstacle. To keep consistency with different sensor models, this force could only be activated near the obstacles.

where the $Q^\star \in \mathbb{R}$ is the distance that allows the robot to ignore the obstacles further than this distance away. η can be viewed as a gain on repulsive gradient. These scalars are usually determined by trial and error depending on the application.

Gradient descent

The gradient descent approach is a well-known first order solution to find local minimum in optimization problems, The idea is very simple and straight forward. Starting at the initial configuration, taking a small step in the direction opposite to the gradient. This gives a new configuration, and the process is repeated until the gradient is zero.

Gradient descent algorithm
Input: Function $\phi(q)$
output: A sequence of points toward the minimum point $\{q(0), q(1), \ldots, q(i)\}$
$q(0) = q_{initialpoint}$
$i = 0$
repeat until $\bigtriangledown \phi q(i) = 0$
$q(i+1) = q(i) + \alpha(i) \bigtriangledown \phi q(i)$
$i = i + 1$
end repeat

In the above algorithm, the notation $q(i)$ is used to show the value of q at the ith iteration and the final path consists of the sequence of iterations $q(0), q(1), \cdot, q(i)$. The scalar $\alpha(i)$ determines the step size in the ith iteration. It is important that $\alpha(i)$ is small enough so the robot is not allowed to suddenly jump into an obstacle. At the same time, α should be large enough so that the algorithm could converge in a reasonable amount of time. Finally, it is unlikely in a gradient descent method that we will ever exactly satisfy the condition $\bigtriangledown \phi(q(i)) = 0$. For this reason, this condition is often replaced with more robust condition $|| \bigtriangledown \phi(q(i))|| < \epsilon$, in which ϵ is chosen to be sufficiently small, based on the application requirements.

9.1.2 Local Minima Problem

The problem with all gradient descent algorithms is the possible existence of local minima in the potential function. For an appropriate value of $\alpha(i)$ it is possible to guarantee that the algorithm will converge to a minimum point, which in not necessarily the global minimum point. This means that there is no guarantee that the gradient descent method will find the destination of the robot.

In Figure 9.4 the robot is initially attracted to the goal as it approaches the open box-shaped obstacle. The goal continues to attract the robot, but the obstacle starts to force the robot out. the robot reaches a point where the effect of the obstacle's repulsion is equal to the attraction

of the goal. In other words, the robot has reached a $q^\star$ where $\bigtriangledown\phi(q^\star) = 0$ and $q^\star$ is not clearly the destination of the robot. Note that this problem is not limited to concave obstacles like the open box here. So the potential function technique is not yet complete for coordination of robots.

There are potential functions other than the attractive/repulsive potential. Many of these potential functions are efficient to compute and can be computed in real time. Unfortunately, they all have one problem which is the existence of local minima not corresponding to the destination of the robot. Three different approaches can be exploited to tackle the problem of local minima. The first two approaches are adding rotational and random potentials near the obstacles (Fig.9.5). Keen readers can check out Randomized Path Planner methods (RPP). In the RPP methods, the robot follows the negative gradient of the specified potential function and when stuck at a local minimum, it initiates a series of random walks. Often the random walks let the RPP to escape the local minimum and in that case, the negative gradient to the goal is followed again. The third approach defines a family of potential functions with only one local minimum. This family of potential functions are called navigation potential functions or in short navigation functions. We will discuss navigation functions in the upcoming subsections.

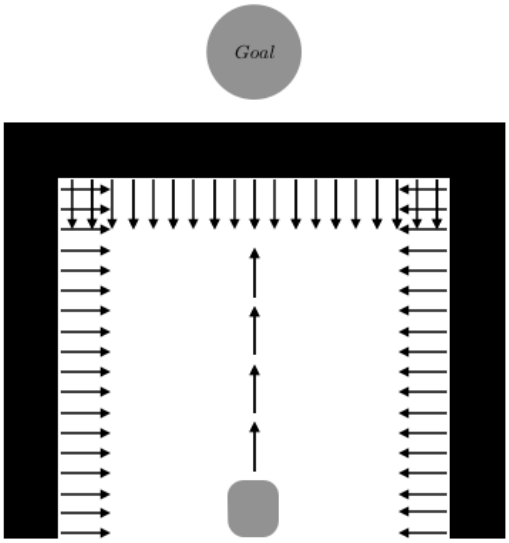


Figure 9.4 – The robot moves toward the local minimum point inside the obstacle using the potential function method.

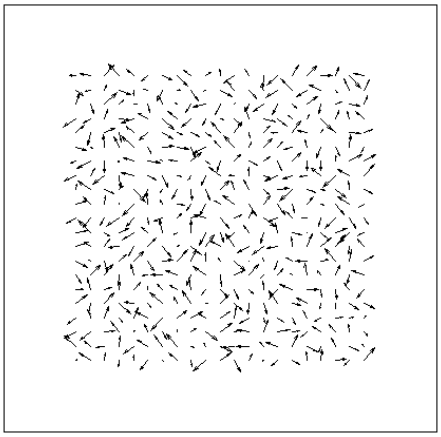


Figure 9.5 – Random vector field could be added to the gradient of the potential function to get the robot out of local minimum point.

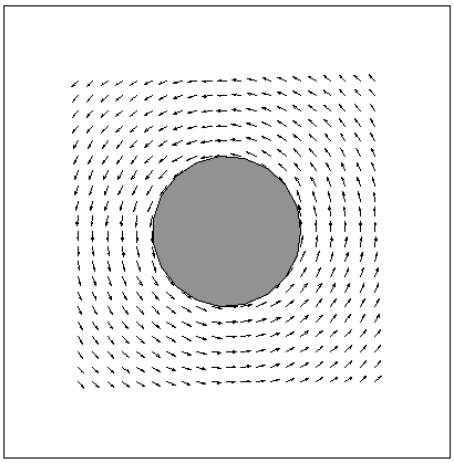


Figure 9.6 – Rotational vector field could be added to the gradient of the potential function to let the robot rotate around the obstacle.

It is worth mentioning that the potential function method has no limitation on the dimension of the robot working space. So, unless stated otherwise, algorithms introduced in this chapter could be used for mobile robots or robot manipulators.

9.1.3 Navigation Function

Thus far we studied potential field methods and their draw backs. In this subsection we introduce a family of potential functions that have only one minimum point. This family of potential functions is called navigation functions.

A potential function is called a navigation function if it:

- is smooth

- is uniformly maximal on boundaries of the work space
- has a minimum at q_{goal} in the connected component of the free space that contains q_{goal} and
- is Morse.

A Morse function is one whose critical points are all non-degenerate. This means that critical points are isolated and if a Morse function is used for gradient descent, any random perturbation will destabilize saddles and maxima.

9.1.4 Example: Comparison of potential and navigation functions

A mobile robot is supposed to move from its initial position to a destination in two slightly different environments and avoid the collision with obstacles. (Fig.9.7 and Fig.9.8). The goal is to compare a navigation function and a potential function in performance of the robot in both environments.

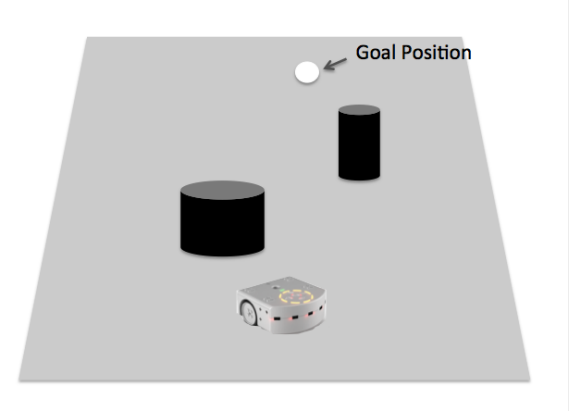


Figure 9.7 – An environment with two obstacles, the robot tries to reach the goal position.

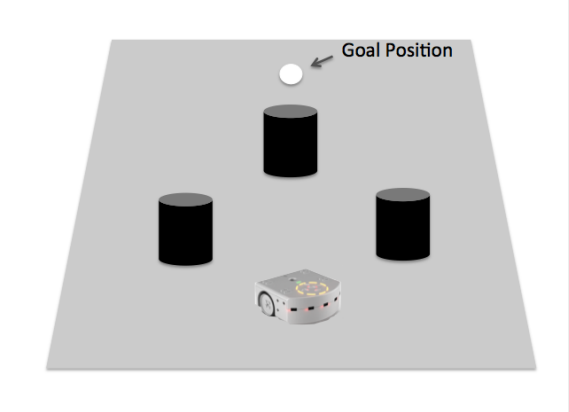


Figure 9.8 – An environment with three obstacles, as in the first environment, the robot tries to reach the goal position.

As the potential function, we chose the one introduced in the previous section:

$$\phi_1(q) = \phi_att(q) + \phi_rep(q)$$

$$\phi_{att}(q) = \left\{ \begin{array}{ll} \frac{1}{2}\zeta d^2(q, q_{goal}) & d(q, q_{goal}) \leq d^\star_{goal} \\ d^\star_{goal}\zeta d(q, q_{goal}) - \frac{1}{2}\zeta (d^\star_{goal})^2 & d(q, q_{goal}) \geq d^\star_{goal} \end{array} \right\}$$

$$\phi_{rep}(q) = \left\{ \begin{array}{ll} \frac{1}{2}\eta(\frac{1}{D(q)} - \frac{1}{Q^\star})^2 & D(q) \leq Q^\star \\ 0 & D(q) > Q^\star \end{array} \right\}$$

A navigation function could be also introduced for a single mobile robot as such:

$$\phi_2(q) = \frac{d^2(q, q_{goal})}{(d(q, q_{goal})^{2k} + \beta(q))^{1/k}}$$

$$\beta(q) = \prod \beta_i(q)\beta_i(q) = d^2(q, q_i) - r_i^2$$

q_i and r_i are the position and the radius of the obstacle number i, respectively. Readers are encouraged to verify if ϕ_2 is a navigation function as an additional exercise.
A mobile robot would find its way in each environment using a navigation function (Fig.9.9 and Fig.9.10). However, using the potential function, it fails in getting to the goal point in one of the environments (Fig.9.11 and Fig.9.12).

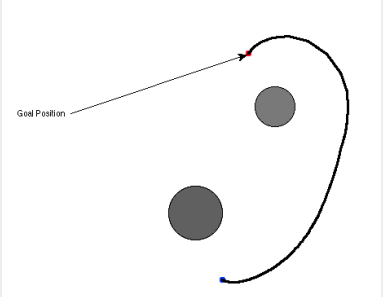


Figure 9.9

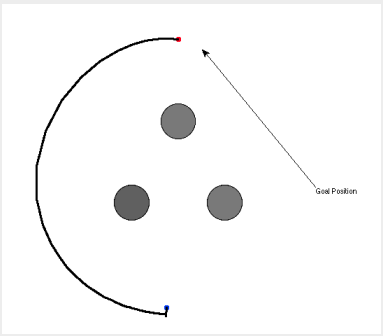


Figure 9.10

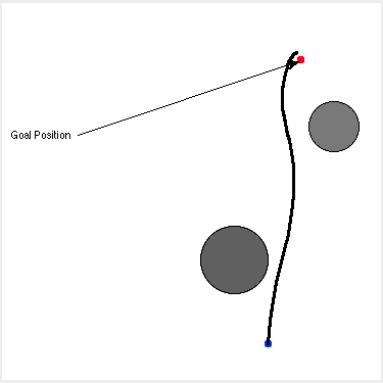


Figure 9.11

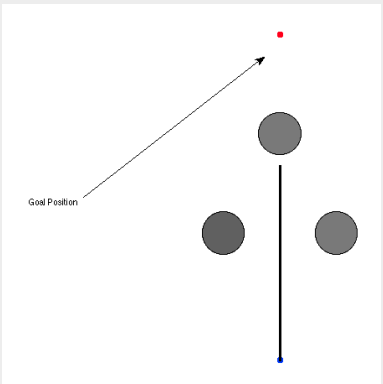


Figure 9.12

9.1.5 Decentralized Navigation Function

Till here, we studied the problem of coordination of a single mobile robot in a static environment. In practice, robots are supposed to deal not only with dynamically moving obstacle but also other robots. Autonomous underwater vehicles (AUVs) are helping scientists to obtain a precise map of oceans as well as verification of presence of different microscopic life under water. Unmanned aerial vehicles are especially useful for penetrating in areas that may be

too dangerous for manned aircrafts. The National Oceanic and Atmospheric Administration began using the unmanned aerial vehicles in 2006 as a hurricane hunter. Autonomous ground vehicles also knowns as driverless cars are hot topic of research nowadays, since Google got licenses for driverless vehicles in some states in US. All these examples motivated the researchers to expand the domain of navigation functions to deal with dynamic obstacles and other robots. The idea is simple, design a navigation function which is attractive toward the goal position and repulsive from all obstacles and other robots. The great advantages of this method are, first its relatively low complexity with respect to the number of robots. Second it is applicable to many tasks which involve different types of robots. In the case study students will be introduced to a formation problem, in which four quadrotors are supposed to fly in a common area avoiding collisions.