الجمهورية الجزائرية الديمقراطية الشعبية
People's Democratic Republic of Algeria
وزارة التعليم العالي والبحث العلمي
Ministry of Higher Education and Scientific Research
جـــامعـة ابن خلدون – تيارت
Ibn Khaldoun University of Tiaret (IKUT)
كلية العلوم التطبيقية
Faculty of Applied Sciences
قسم الهندسة الكهربائية
Department of Electrical Engineering

TP: Control Systems

Specialization : Electronic

Level : Third-Year Undergraduate Students

Semester : 6

Prepared by:

CHAIB Habib MCB Ibn Khaldoun University -Tiaret

Reviewed by:

Full Name	**Academic Rank**	**University**
Belabbas Belkacem	Pr.	Ibn Khaldoun -Tiaret
Berkani Abderrahmane	MCA	Ibn Khaldoun -Tiaret

Academic Year: 2025–2026

Preface

Control engineering is one of the most empowering disciplines of modern technology. It is found at the core of robots, aircraft, electrical drives, chemical reactors, biomedical devices, energy grids and consumer electronics. The design of a controller is thus mastered by a balanced training which supports the rigorous mathematical background gained in the lectures with a strong practical experience.

This laboratory lecture note is written to follow the course "Asservissements Continus et Régulation". Its purpose is twofold: to consolidate the theoretical foundations of linear control via carefully selected exercises and to introduce the student to the modern computer-aided tools that the modern engineer employs daily, namely MATLAB, its Control System Toolbox and Simulink.

The seven practical sessions of this manual cover the whole pedagogical pipeline, from the discovery of the software environment up to the design of a full corrector and its experimental validation on a real laboratory bench. Each session has the following organization: a detailed theoretical review, a clear statement of the objectives, a series of supervised exercises with their fully commented MATLAB solutions, and an open discussion to incite the student to extend the analysis. Particular care has been taken to include intermediate figures (block diagrams, time responses, Bode and Nyquist plots) for the student to be able to compare his/her own results with reliable references.

The only prerequisites are an initial exposure to the Laplace transform, the basic ideas of linear circuit theory, and the elementary notion of a transfer function. Familiarity with MATLAB is welcome but not a prerequisite, and the first practical session is precisely devoted to closing this gap.

I hope this manual will be useful to the reader in developing both the analytical rigor and the practical intuition that characterize a good control engineer. Any feedback is welcome and will be included in future lectures.

CHAIB Habib

Course information

Field	Value
Subject	TP : Asservissements continus et Régulation (Control Systems Lab)
Cycle	Bachelor (LMD)
Recommended year	L3 Electronics / Automatic Control
Semester	Semester 6
Hourly volume	22 h 30 (1 h 30 per week — 15 weeks)
Assessment	Continuous evaluation + final exam

I. Teaching objectives

Upon completion of this practical training, the student shall be able to:

- build the mathematical model of a linear time-invariant (LTI) physical system and translate it into a MATLAB object;
- manipulate the basic block-diagram-reduction operations (series, parallel, feedback) both analytically and numerically;
- analyse the time response of first- and second-order systems and identify the influence of poles and zeros;
- plot and interpret the frequency response of a system on a Bode, Nyquist or Black diagram;
- assess the stability of a closed-loop system through several complementary criteria and quantify its accuracy;
- design a phase-lead compensator using the frequency-response method and check its effect on the closed-loop response;
- connect, adjust and operate a real laboratory bench (position servo, speed servo, temperature, level / flow regulation).

II. Recommended background

The student is expected to be familiar with the following topics: Laplace transform and transfer-function representation, basic linear algebra (matrices, determinants,

eigenvalues), fundamental electronics (op-amps, RC and RLC circuits) and a first exposure to control theory. Although no previous programming experience is mandatory, the practical sessions will require the student to type simple MATLAB instructions and to interpret short scripts.

III. Syllabus at a glance

Session	Title
TP 1	Familiarisation with MATLAB, Control System Toolbox and Simulink
TP 2	System modelling under MATLAB and block-diagram reduction
TP 3	Time analysis of first-, second- and higher-order LTI systems; dominant poles
TP 4	Frequency analysis: Bode, Nyquist and Black diagrams
TP 5	Stability and accuracy of closed-loop systems
TP 6	Design of a phase-lead compensator by the frequency-response method
TP 7	Analog real laboratory benches: position / speed servo, temperature, level / flow

Practical organisation and assessment

How a session is conducted

Every practical session has been designed for an effective duration of one hour and a half. Before coming to the laboratory, the student must:

- read carefully the theoretical reminder placed at the beginning of the corresponding chapter;
- answer the few preliminary questions that prepare for the work to be done with MATLAB;
- prepare a short personal report sheet on which intermediate results, observations and plots will be written.

During the session, students work in groups of two on a single workstation. The supervising teacher follows the progress, provides hints when needed, and grades the active participation. At the end of the session, every group hands in a brief minutes (one or two pages) that summarises the work done, the difficulties encountered and the conclusions reached.

Final report

Within one week after the session, each group submits a complete written report. The report must contain: the cover page, an introduction stating the objectives, the answers to all the questions of the handout, the relevant MATLAB scripts (well commented), the captured figures, and finally a critical analysis of the results. Care must be taken to clearly justify any numerical answer and to keep the report concise (about ten pages per session).

Required equipment

- A computer running MATLAB R2018b or later, with the Control System Toolbox and Simulink installed.
- For TP 7, the dedicated laboratory bench (position / speed servo, temperature, level / flow set-up) and the associated power supply, signal generator and oscilloscope.
- A notebook and a few sheets of millimetre paper for hand-drawn diagrams.

Safety reminder

Although the laboratory benches operate with low voltage, the student must always switch the power supply off before re-wiring the experimental set-up. Liquids must never

be poured close to electronic boards. The general safety rules of the laboratory must be respected at all times.

I. General Introduction

I.1 A brief history of control engineering

The discipline of automatic control predates the digital computer. It stems from mechanical regulators, such as the float-valve devices used by the Greek inventors of the Hellenistic period to regulate the water level in their clepsydrae. The first formal mathematical analysis is, however, generally credited to James Clerk Maxwell, who in 1868 published "On Governors", a paper that analyzed the stability of James Watt's centrifugal flyball governor for steam engines and introduced the use of linear differential equations and characteristic roots to predict whether oscillations would grow or decay.

Then two parallel currents formed the modern field. In the United Kingdom and the United States, during the 1930s and 1940s, Hendrik Bode, Harry Nyquist and Nathaniel B. Nichols developed frequency-domain methods while working at Bell Telephone Laboratories on long-distance amplifier design, methods that proved decisive during the Second World War for fire-control and radar tracking. The modern state-space framework originated in the rediscovery and extension after 1950 of the stability theory of Aleksandr Lyapunov, published already in 1892, and was further reinforced by the papers of Rudolf E. Kálmán in 1960 on observability, controllability and optimal filtering.

Control theory underpins the operation of virtually every engineered system that interacts with its environment in the modern world: aircraft autopilots, automotive cruise control and anti-lock braking systems, industrial robot manipulators, chemical-plant flow and temperature loops, power-grid frequency regulation, biomedical drug-delivery pumps, and the attitude controllers of orbiting satellites. The laboratory exercises in this manual are concerned with the classical frequency-domain and time-domain methods which form the working vocabulary of practising control engineers in industry.

I.2 The function of the laboratory work

In the classroom, when analysing control systems, we must of course discuss idealised models, e.g. linear differential equations with constant coefficients, perfect sensors, instantaneous actuators and noise-free signals. All of these assumptions are violated by real plants. Sensors saturate, actuators are bandwidth and slew-rate bounded, friction is nonlinear, and measurements are stochastically noisy. The lab work is meant to fill that gap, to let the student experience first hand how a controller designed on paper actually behaves when wired to a real physical process.

The exercises in this manual are structured so that each session reinforces a specific concept introduced in the lecture course on Asservissements et Régulation. The first four sessions are simulation based using Matlab, Control System Toolbox and Simulink. The fifth session is simulation and analytical tools for stability. The sixth session is a full controller synthesis exercise — the design of a phase-lead compensator using the frequency response method. The seventh session is devoted to the physical equipment: a DC servomotor, a thermal plant and a level/flow rig representative of process industry.

I.3 Mathematical prerequisites

The student is assumed to be comfortable with the Laplace transform, ordinary differential equations with constant coefficients, complex numbers, and elementary matrix algebra. The following ideas are central and will be used without further introduction.

I.3.1 The Laplace transform

The (one-sided) Laplace transform of a signal f(t) defined for t ≥ 0 is the function of the complex variable s given by:

$$F(s) = \int_0^\infty f(t)e^{-st}dt$$

The transform converts the operations of differentiation and integration into algebraic multiplication and division by s, and it is this property that makes it the natural tool for analysing linear time-invariant differential equations. A function f(t) is recovered from F(s) by the inverse transform, but in engineering practice one almost always uses tables of standard pairs and the partial-fraction expansion rather than the contour-integral definition.

I.3.2 Transfer functions and the s-plane

A linear, time-invariant, single-input single-output system can be characterised by its transfer function G(s) = Y(s)/U(s), the ratio of the Laplace transform of the output to that of the input under zero initial conditions. For physical systems, G(s) is a real-rational function: a ratio of polynomials in s with real coefficients. The roots of the numerator are called the zeros of the system; the roots of the denominator are the poles. The pattern of poles and zeros in the complex s-plane determines almost everything about the system's behaviour: its stability, its speed of response, its degree of oscillation, and its steady-state accuracy. Plotting and reasoning about pole locations is a habit the student should cultivate from the first laboratory session.

I.3.3 Block diagrams

A block diagram is a graphical shorthand for the algebra of transfer functions. A rectangular block labelled G(s) represents multiplication by G(s); a summing junction with a "+" and a "−" represents the algebraic sum of the incoming signals. The two reduction operations the student must master are series combination — two blocks in cascade are equivalent to one block whose transfer function is the product — and the feedback formula:

$$T(s) = G(s) / [\, 1 + G(s)\, H(s) \,]$$

where G(s) is the forward path and H(s) is the feedback path. The denominator 1 + G(s)H(s) is the characteristic polynomial of the closed loop; its roots are the closed-loop poles, and they determine stability.

I.4 The Matlab software ecosystem

Matlab is a numerical-computing environment built around a matrix language and a large library of toolboxes. For control-system work, four components matter most:

- **Base Matlab** — the matrix language, plotting, and scripting/function programming. Indispensable for data manipulation and visualisation.
- **Control System Toolbox** — provides the LTI object (transfer functions, state-space, zero-pole-gain models) along with analysis and design functions: step, impulse, bode, nyquist, nichols, rlocus, margin, pidtune, lqr, place.
- **Simulink** — a block-diagram simulator. Continuous, discrete, and hybrid models are built graphically and integrated with variable- or fixed-step solvers. Essential for non-linear simulation and for closing the loop around plants that cannot be conveniently expressed as a single transfer function.
- **Symbolic Math Toolbox** (used occasionally) — for symbolic Laplace transforms, simplification of expressions, and partial-fraction expansion when one prefers exact rather than numerical arithmetic.

The exercises in this manual rely chiefly on the first three. Where a Symbolic Math step would clarify the algebra, the option is mentioned in the text but a numerical alternative is always provided so that the student can complete the work on any installation of Matlab with the Control System Toolbox enabled.

I.5 Conventions used in this manual

The following conventions are observed throughout.

- Mathematical symbols are typeset in italics (s, ω, K, ζ). Multi-letter function names such as sin or arctan are typeset upright.
- Matlab commands and code listings are shown in a monospaced font on a grey background. Comments begin with %. Lines that would in practice be entered at the Matlab prompt are presented without the prompt.
- Variable names follow standard notation: u for the control (manipulated) input, y for the measured output, r for the reference (set-point), e = r − y for the tracking error, and d for an external disturbance.
- All angular frequencies are in radians per second unless explicitly stated. Phase is in degrees in plots (the Matlab default) and in radians in algebraic expressions.
- Time is in seconds throughout. Where a plant has natural time-constants on a different scale (milliseconds for an electrical loop, minutes for a thermal one), the value is given in the problem statement and converted explicitly.

I.6 Laboratory safety and procedure

Sessions 1 through 6 are purely computational and present no physical hazard beyond ordinary computer-laboratory rules. Session 7 involves physical equipment — rotating shafts, heating elements, pumps, and pressurised vessels — and a number of precautions apply.

- **Read the bench documentation before powering up.** Each physical plant ships with a manufacturer's manual giving the operating ranges, the maximum permissible input voltage or current, the location of emergency stops, and the calibration of sensors. The instructor will indicate which manuals apply to which benches.
- **Start with low gains.** A controller whose gain is set too high can drive an actuator into saturation immediately, with risk of mechanical damage on motor benches and of overheating on thermal benches. Increase gain gradually while watching the response.
- **Never modify wiring with the bench powered on.** Switch off the bench at the panel before changing any connection between the computer's data-acquisition card and the plant.

- **Report anomalies.** Unusual noises, smells of burning insulation, smoke, or unexpected behaviour are reasons to press the emergency stop and call the instructor before proceeding.

I.7 How to use this manual

Each session is structured in the same way. Section 1 of the session states the learning objectives; sections 2 and 3 provide a theoretical reminder and a worked discussion of the relevant Matlab functions; section 4 contains the exercises proper, with Matlab listings and figures that the student is expected to reproduce; section 5 closes with synthesis questions and an open-ended discussion. The student should not simply read the manual: every Matlab listing should be entered, executed, and inspected, and the results compared with the figures shown.

A laboratory report is expected at the end of each session. Reports should include the Matlab code used, the figures produced (clearly labelled with axis titles and units), a discussion of the numerical and graphical results, and a paragraph in answer to each of the synthesis questions. The discussion is the most important part of the report; copying figures without commentary is not sufficient.

TP 1 — Familiarisation with MATLAB, Control System Toolbox and Simulink

1.1 Objectives

This first practical session introduces the working environment in which all the subsequent experiments will be carried out. By the end of the session, the student should be able to:

- start MATLAB and identify the main panels (Command Window, Workspace, Editor, Current Folder);
- use MATLAB as a powerful scientific calculator and produce simple plots;
- manipulate the elementary objects of the Control System Toolbox: transfer functions, zero–pole–gain models, state-space representations;
- build and simulate a simple block diagram with Simulink.

1.2 Theoretical reminder

1.2.1 MATLAB — a numerical computing environment

MATLAB (MATrix LABoratory) is an interactive software dedicated to numerical computation, data visualisation and rapid prototyping. Its strength comes from the matrix as a basic data structure: every variable is treated as a matrix, even scalars (1×1) and vectors (n×1 or 1×n). Operations are then vectorised, which makes the language particularly suitable for control engineering, signal processing and numerical analysis.

The MATLAB language is interpreted: every command typed in the Command Window is executed immediately, and the result is assigned to the variable ans by default if no left-hand side is given. The user can also gather a sequence of instructions in a script (*.m file) or wrap them into a function for re-use.

1.2.2 The Control System Toolbox

The Control System Toolbox provides a unified framework for the analysis and design of linear time-invariant (LTI) systems. It defines several object types — transfer function (tf), zero–pole–gain (zpk), state space (ss), frequency response data (frd) — and a large set of functions that work transparently with any of them: step, impulse, lsim, bode, nyquist, nichols, rlocus, margin, etc.

Three classical representations are used throughout this manual. The transfer function $H(s)$ of a single-input single-output system relates the Laplace transform of the output to that of the input through a rational fraction:

$$H(s) = Y(s) / U(s) = (b_m s^m + \dots + b_1 s + b_0) / (a_n s^n + \dots + a_1 s + a_0) \quad (1.1)$$

The zero–pole–gain form factorises the polynomials and reveals the zeros z_i, the poles p_j and the static gain K:

$$H(s) = K (s - z_1)(s - z_2)\dots(s - z_m) / [(s - p_1)(s - p_2)\dots(s - p_n)] \quad (1.2)$$

The state-space form expresses the dynamics through a first-order vector differential equation:

$$\dot{x}(t) = A\, x(t) + B\, u(t), \quad y(t) = C\, x(t) + D\, u(t) \quad (1.3)$$

These three representations are mathematically equivalent for LTI systems; the choice between them is dictated by the algorithm one wants to apply and by the available data.

1.2.3 Simulink

Simulink is a graphical environment for the modelling, simulation and analysis of dynamical systems. The user assembles a block diagram by dragging blocks from a library and connecting them with signal wires. Once the diagram is complete, Simulink integrates the differential equations and displays the result in a Scope block or sends it back to the MATLAB workspace. For a control engineer, Simulink offers a particularly intuitive way to set up a closed-loop simulation involving a reference input, a corrector, a plant and a sensor.

1.3 Preliminary work

1. Recall the definition of the Laplace transform and state the linearity property.
2. Give the transfer function of a unity-gain first-order low-pass filter of time constant $\tau = 0.1$ s.
3. Sketch by hand the step response of this filter.

1.4 Guided exercises

Exercise 1 — Discovering the Command Window

Type the following instructions one after the other in the Command Window and observe the result of each. Pay attention to the role of the semicolon at the end of a line.

```
% MATLAB code
>> a = 3
>> b = 4;
>> c = a + b
>> v = [1 2 3 4 5]
```

```
>> w = 0:0.1:1
>> M = [1 2 3 ; 4 5 6 ; 7 8 10]
>> det(M)
>> inv(M)
>> M.'           % transpose
>> M^2           % matrix square
>> M.^2          % element-wise square
```

Discussion. The semicolon suppresses the automatic echo of a result. The colon operator a:step:b is the natural MATLAB way to generate a regularly spaced vector — it is far more efficient than a for-loop. Note the very important difference between the matrix operations *, ^ and / on the one hand, and the element-wise versions .*, .^ and ./ on the other hand.

Exercise 2 — Plotting elementary curves

Create a script named ex2.m and copy the lines below.

```
% MATLAB code
% ex2.m – basic plotting
t = 0:0.01:2*pi;
y1 = sin(t);
y2 = cos(t);
y3 = sin(t).*exp(-0.5*t);
 figure;
plot(t, y1, 'b-',  'LineWidth', 1.5); hold on;
plot(t, y2, 'r--', 'LineWidth', 1.5);
plot(t, y3, 'g-.', 'LineWidth', 1.5);
grid on;
xlabel('t (s)');
ylabel('Amplitude');
title('Three elementary signals');
legend('sin(t)', 'cos(t)', 'sin(t) e^{-0.5 t}');
```

Question. Modify the script so that the three signals are displayed in three separate panels by means of the function subplot.

Exercise 3 — Defining transfer functions

Use the function tf to define the two systems $H_1(s) = 5/(2s + 1)$ and $H_2(s) = (s + 2)/(s^2 + 3s + 5)$:

```
% MATLAB code
% Building transfer functions
H1 = tf(5, [2 1])
H2 = tf([1 2], [1 3 5])
 % Same systems in zero-pole-gain form
H1z = zpk(H1)
H2z = zpk(H2)
 % Properties
poles_H2  = pole(H2)
zeros_H2  = zero(H2)
gain_H2   = dcgain(H2)
 % Step responses
figure; step(H1, H2, 5);grid on;
legend('H_1', 'H_2');
```

Observe the textual representation of a transfer function in the Command Window. Notice that MATLAB automatically displays the polynomial form by default; the function zpk converts it to the factorised form (1.2).

Exercise 4 — Converting between representations

```
% MATLAB code
% Numerical data of the system
num  = [1 2];
den  = [1 3 5];

sys_tf = tf(num, den);              % transfer function
sys_zp = zpk(sys_tf);               % zero-pole-gain
sys_ss = ss(sys_tf);                % state space

disp('--- Transfer function ---');  sys_tf
disp('--- Zero-pole-gain ---');     sys_zp
disp('--- State space ---');        sys_ss
```

Question. Display the four matrices A, B, C, D of the state-space model. Verify by hand that the transfer function recovered through $C(sI - A)^{-1} B + D$ is indeed equal to $H_2(s)$.

Exercise 5 — A first contact with Simulink

Open Simulink (command simulink at the MATLAB prompt or click on the Simulink icon of the toolbar). Create a new blank model, then drag from the library and connect the following blocks:

- a Step block (library Sources) with Step time = 0 and Final value = 1;
- a Transfer Fcn block (library Continuous) configured as numerator = [1], denominator = [1 2 1];
- a Scope block (library Sinks) to visualise the output.

Set the simulation time to 10 s, run the model and observe the step response. Save the model under the name first_simulink.slx — you will reuse it as a template in the following sessions.

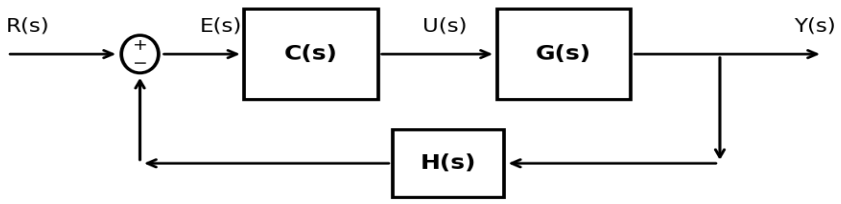

Figure 1.1 — General structure of a closed-loop control system. R(s) is the reference, Y(s) the controlled output, E(s) the error, U(s) the command. C(s) is the controller, G(s) the plant and H(s) the feedback (sensor) transfer function.

1.5 Synthesis questions

4. In what context would you choose a state-space representation rather than a transfer function?
5. Why is it preferable to use the function tf rather than typing the numerical coefficients of every polynomial when one builds a complex model?
6. Cite three advantages of Simulink with respect to a pure script-based simulation.

1.6 Conclusion

At the end of this first session, the student is now able to navigate inside the MATLAB environment, to design a linear system in any of its three canonical forms and to run a

first Simulink simulation. These essential abilities are the foundation on which the succeeding sessions will progressively build the whole control-engineering toolset.

TP 2 — System modelling and block-diagram reduction

2.1 Objectives

- Establish the model of a physical system (electrical, mechanical, electromechanical) and derive its transfer function.
- Translate a block diagram into MATLAB instructions.
- Compute the equivalent transfer function of a series, parallel or feedback interconnection.
- Verify the result with the functions series, parallel and feedback of the Control System Toolbox.

2.2 Theoretical reminder

2.2.1 From the physical equations to the transfer function

Building the model of a physical system is the most important — and the most delicate — step of any control study. The systematic methodology consists of three stages: writing the physical balance equations (Newton's law, Kirchhoff's laws, energy or mass conservation, etc.), linearising them around a chosen operating point when they are non-linear, and finally taking the Laplace transform under zero initial conditions to obtain the transfer function.

2.2.2 Example 1: RC low-pass filter

Consider a simple RC filter with input voltage $v_i(t)$ and output voltage $v_o(t)$ taken across the capacitor. The Kirchhoff voltage law gives $v_i = R\, i + v_o$, and the constitutive equation of the capacitor reads $i = C\, dv_o/dt$. Substituting yields:

$$RC\, dv_o/dt + v_o = v_i \tag{2.1}$$

With zero initial conditions, the Laplace transform leads to the transfer function:

$$H(s) = V_o(s) / V_i(s) = 1 / (1 + RC\, s) = 1 / (1 + \tau\, s) \tag{2.2}$$

where $\tau = RC$ is the time constant of the filter. This is a textbook first-order system.

2.2.3 Example 2: DC motor

Consider a separately-excited DC motor driven by an armature voltage u(t).

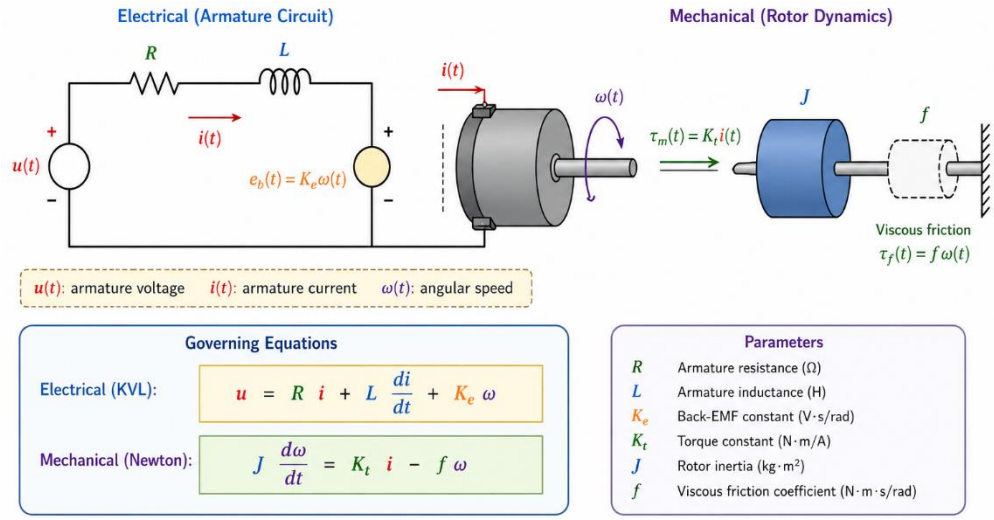

Figure 2.1 — DC motor (separately-excited)

Let i(t) be the armature current, ω(t) the angular speed, R and L the armature resistance and inductance, Ke the back-EMF constant, Kt the torque constant, J the rotor inertia and f the viscous friction coefficient. The two physical equations are:

$$u = R\,i + L\,di/dt + K_e\,\omega, \quad J\,d\omega/dt = K_t\,i - f\,\omega \tag{2.3}$$

Taking the Laplace transform and eliminating the current $I(s)$ yields the speed transfer function:

$$\Omega(s) / U(s) = K_t / [(Ls + R)(Js + f) + K_t\,K_e] \tag{2.4}$$

This is a second-order transfer function. When the electrical time constant L/R is much smaller than the mechanical one J/f, the inductance can be neglected and (2.4) reduces to a first-order model.

2.2.4 Elementary block-diagram operations

Three rules allow one to simplify the most common interconnections of linear blocks.

Series (cascade). Two blocks placed one after the other, with no feedback between them, are equivalent to a single block whose transfer function is the product of the two:

$$G\_eq(s) = G_1(s)\ \cdot G_2(s) \tag{2.5}$$

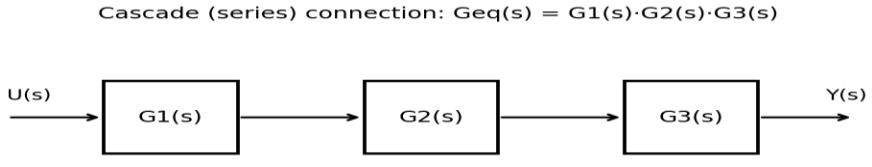

Figure 2.2 — Series connection of three blocks.

Parallel. Two blocks fed by the same input and whose outputs are summed give:

$$G\_eq(s) = G_1(s) + G_2(s) \tag{2.6}$$

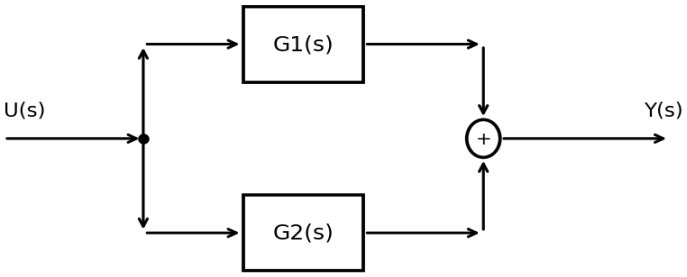

Figure 2.3 — Parallel connection of two blocks.

Negative feedback. With a forward branch $G(s)$ and a feedback branch $H(s)$, the closed-loop transfer function reads:

$$T(s) = G(s) / [1 + G(s)\,H(s)] \tag{2.7}$$

When the feedback gain is unitary ($H(s) = 1$), expression (2.7) simplifies to $T(s) = G(s)/(1 + G(s))$. This is the most common configuration in control engineering.

2.3 Preliminary work

7. Derive the transfer function of an RLC series circuit whose output is the voltage across the capacitor.
8. Write the differential equation governing the angular position of a DC motor (consider that the speed equation has already been established).
9. Recall the formulas (2.5), (2.6) and (2.7) and identify the corresponding MATLAB functions.

2.4 Guided exercises

Exercise 1 — Building a model from physical data

A DC motor has the following characteristics: $R = 1\ \Omega$, $L = 0.5$ H, $K_e = K_t = 0.01$, $J = 0.01$ kg·m², $f = 0.1$ N·m·s. Build a MATLAB script that constructs the speed-to-voltage transfer function (2.4) and plots its step response.

```
% MATLAB code
% ex1.m — DC motor speed transfer function
R  = 1;     L  = 0.5;
Ke = 0.01;  Kt = 0.01;
```

```
J  = 0.01;  f  = 0.1;
 % Direct expansion of (2.4):  num = Kt,   den = (Ls+R)(Js+f) + Kt*Ke
num = Kt;
den = conv([L R], [J f]) + [0 0 Kt*Ke];   % polynomial sum
G   = tf(num, den)
 % DC gain and poles
disp(['DC gain = ', num2str(dcgain(G))])
disp('Poles : '); disp(pole(G))
 % Step response
figure; step(G, 5);
grid on;
title('Step response of the DC motor (speed)');
xlabel('Time (s)'); ylabel('Angular speed (rad/s)');
```

Question. Verify the value of the DC gain by direct substitution of s = 0 in (2.4). Identify the two poles and discuss their relative position.

Exercise 2 — Series, parallel and feedback

Three subsystems are defined by:

$$G_1(s) = 1 / (s + 1), \quad G_2(s) = 2 / (s^2 + 3\,s + 2), \quad H(s) = 1 / (s + 5)$$

```
% MATLAB code
% ex2.m — block-diagram reduction
G1 = tf(1, [1 1]); G2 = tf(2, [1 3 2]); H  = tf(1, [1 5]);
 % Series : G12(s) = G1(s) * G2(s)
G12_series = series(G1, G2);
disp('Series : '); G12_series
 % Parallel : Gp(s) = G1(s) + G2(s)
Gp = parallel(G1, G2);
disp('Parallel : '); Gp
 % Negative feedback : T(s) = G12 / (1 + G12 * H)
T = feedback(G12_series, H);
disp('Feedback : '); T
 % Same result obtained by direct algebra :
T_check = G12_series / (1 + G12_series*H);
T_check = minreal(T_check);    % cancel hidden common factors
disp('Same closed loop, computed by algebra : '); T_check
```

Question. Check that the operator * gives the same result as series. What is the role of minreal in the previous code?

Exercise 3 — Reduction of a complex diagram

Consider the closed-loop system of Figure 1.1 with C(s) = (s + 2)/(s + 5), G(s) = 4/(s² + 2s + 4) and H(s) = 1 (unity feedback). Compute by hand the closed-loop transfer function, then verify the result with MATLAB.

```
% MATLAB code
% ex3.m — closed-loop transfer function
C = tf([1 2], [1 5]); G = tf(4, [1 2 4]);
OL = C*G;        % open-loop
CL = feedback(OL, 1);            % unity feedback
CL = minreal(CL);
disp('Closed-loop transfer function : '); CL
 % Verify with the canonical formula CL = OL / (1 + OL)
CL_check = minreal(OL / (1 + OL));  disp('Verification : '); CL_check
```

Exercise 4 — Working with Simulink

Reproduce in Simulink the closed-loop diagram of Exercise 3. Drag a Step block, two Transfer Fcn blocks (for C(s) and G(s)), a Sum block configured as |+−, and a Scope block. Connect the negative input of the Sum block to the output of G(s). Run the simulation over 8 seconds and compare the response with the one obtained in MATLAB.

2.5 Synthesis questions

10. Cite three modelling techniques that yield a linear transfer function.
11. How does the function feedback handle the algebraic loop introduced when 1 + GH = 0?
12. What is the practical use of the function minreal? What kind of poles and zeros does it cancel?

2.6 Conclusion

The student has now built two complete LTI models based on physical considerations and verified the algebraic reduction of complex block diagrams with the dedicated MATLAB functions. From here on any closed-loop system given in the manual will be manipulated either by transfer-function algebra or by a Simulink model (without preference).

TP 3 — Time-domain analysis of LTI systems

3.1 Objectives

- Identify the canonical time response of a first-order system and extract its time constant and DC gain.
- Recognise the four operating regimes of a second-order system according to the damping ratio.
- Compute the classical time-domain performance criteria: rise time, settling time, peak overshoot, peak time.
- Investigate the influence of higher-order poles and zeros, and introduce the notion of dominant poles.

3.2 Theoretical reminder

3.2.1 First-order systems

A first-order system is characterised by the canonical transfer function:

$$H(s) = K / (1 + \tau\, s) \tag{3.1}$$

where K is the static gain (or DC gain) and τ the time constant. Under a unit-step input, the time response reads:

$$y(t) = K\,(1 - e^{\wedge}(-t/\tau)) \tag{3.2}$$

It is a strictly monotonous curve that reaches 63.2 % of its final value after one time constant, 86.5 % after two, and 95 % after three. The 5 % settling time is consequently estimated by $t_s \approx 3\,\tau$ and the 2 % settling time by $t_s \approx 4\,\tau$.

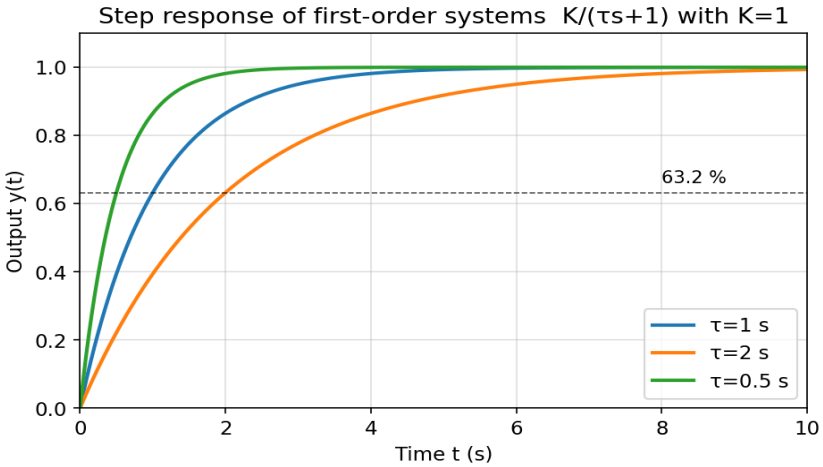

Figure 3.1 — Step response of first-order systems for three values of the time constant.

3.2.2 Second-order systems

The standard form of a second-order system is:

$$H(s) = K\,\omega_n^2 / (s^2 + 2\,\zeta\,\omega_n\,s + \omega_n^2) \tag{3.3}$$

with ω_n the natural (un-damped) angular frequency and ζ the damping ratio. Four cases must be distinguished depending on the position of the poles in the complex plane:

- $\zeta > 1$ over-damped — two distinct real poles, slow non-oscillatory response;
- $\zeta = 1$ critically damped — a double real pole, no overshoot, fastest non-oscillatory response;
- $0 < \zeta < 1$ under-damped — two complex-conjugate poles, oscillatory response with overshoot;
- $\zeta = 0$ un-damped — purely imaginary poles, sustained oscillations.

For an under-damped second-order system, the unit-step response is:

$$y(t) = K\,[\,1 - (1/\sqrt{(1-\zeta^2)})\,e^{\wedge}(-\zeta\,\omega_n\,t)\,\sin(\omega d\,t + \varphi)\,] \tag{3.4}$$

with the damped angular frequency $\omega d = \omega_n\,\sqrt{(1-\zeta^2)}$ and $\varphi = \arctan(\sqrt{(1-\zeta^2)}/\zeta)$. The corresponding performance criteria are:

$$M_p\,(\%) = 100\,\cdot \exp(-\pi\,\zeta / \sqrt{(1-\zeta^2)}) \tag{3.5}$$

$$t_p = \pi / \omega d \qquad (\textit{peak time}) \tag{3.6}$$

$$t_s \approx 4 / (\zeta\,\omega_n) \qquad (\textit{2 \% settling time}) \tag{3.7}$$

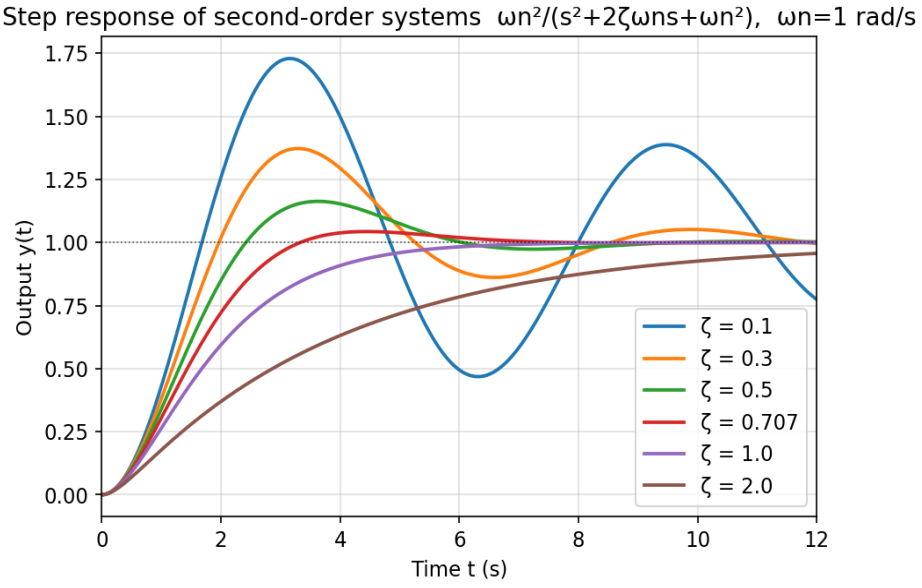

Figure 3.2 — Step response of second-order systems for several values of the damping ratio (ω_n = 1 rad/s).

3.2.3 Higher-order systems and dominant poles

An LTI system of order n possesses n poles, each contributing a term in the time response. When some poles lie far to the left in the complex plane compared with the others, their contribution decays rapidly and the long-term behaviour is governed almost exclusively by the slower poles, called the dominant poles. A common rule of thumb states that a pole can be neglected if its real part is at least five times more negative than that of the dominant poles, provided that its residue is small.

This idea is at the heart of model simplification: a high-order transfer function can often be replaced — for design purposes — by a second-order approximation that retains the two dominant poles. The reduced model is much easier to handle while remaining sufficiently accurate.

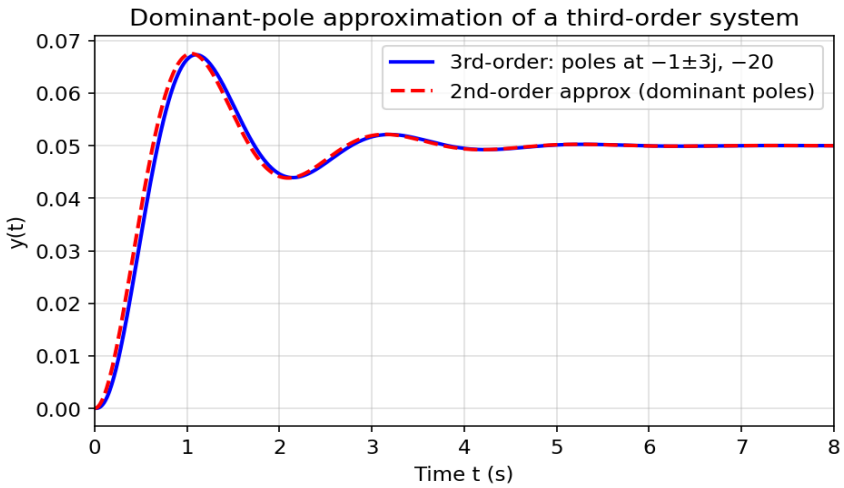

Figure 3.3 — Comparison between a third-order system and its second-order dominant-pole approximation.

3.3 Preliminary work

13. From the response of Figure 3.1, graphically estimate the three time constants.
14. Compute by hand the percentage overshoot of a second-order system with $\zeta = 0.5$.
15. Justify the dominant-pole concept by writing the partial-fraction expansion of a third-order step response.

3.4 Guided exercises

Exercise 1 — Identification of a first-order system

Consider the system $H_1(s) = 5 / (1 + 0.4\ s)$. Plot its step response, then identify by graphical construction the time constant and the static gain. Verify the agreement with the rule $t_{53\%} = \tau$.

```
% MATLAB code
% ex1_tp3.m – first-order identification
K  = 5;
tau = 0.4;
H1 = tf(K, [tau 1]);
 [y, t] = step(H1);
figure; plot(t, y, 'b', 'LineWidth', 1.5); grid on; hold on;
 % Graphical reading
yf = y(end);
yline(yf,'k:');             yline(0.632*yf,'k:');
plot(tau, 0.632*yf, 'ro', 'MarkerFaceColor','r');
xline(tau,'k:');
text(tau+0.05, 0.55*yf, '\tau','FontSize',12);
xlabel('Time t (s)'); ylabel('y(t)');
title('Step response of a first-order system');
 % Automatic performance reading
info = stepinfo(H1);
disp(info)
```

Exercise 2 — Influence of the damping ratio

Plot on the same graph the step responses of the systems given in Table 3.1.

Damping ζ	ω_n (rad/s)	Transfer function H(s)
0.1	1	$1 / (s^2 + 0.2\ s + 1)$
0.3	1	$1 / (s^2 + 0.6\ s + 1)$
0.5	1	$1 / (s^2 + 1.0\ s + 1)$

0.707	1	$1 / (s^2 + 1.414\,s + 1)$
1.0	1	$1 / (s^2 + 2\,s + 1)$
2.0	1	$1 / (s^2 + 4\,s + 1)$

```
% MATLAB code
% ex2_tp3.m – varying the damping
wn  = 1;
zetas = [0.1 0.3 0.5 0.707 1.0 2.0];
 figure; hold on;
for i = 1:length(zetas)
    z = zetas(i);
    H = tf(wn^2, [1 2*z*wn wn^2]);
    [y,t] = step(H, 0:0.01:12);
    plot(t, y, 'LineWidth', 1.5,...
        'DisplayName', sprintf('\zeta = %.3f', z));
end
grid on; yline(1,'k:');
xlabel('Time t (s)'); ylabel('y(t)');
legend('Location','best');
title('Step responses of second-order systems');
```

Question. For each value of ζ, use stepinfo to extract the percentage overshoot, the peak time and the 2 % settling time. Compare the numerical values with formulas (3.5) – (3.7).

Exercise 3 — Higher-order system and dominant poles

Consider the third-order system:

$$H(s) = 10 / [(s + 1 - 3j)(s + 1 + 3j)(s + 20)] = 10 / [(s^2 + 2\,s + 10)(s + 20)]$$

Compare its step response with the response of the second-order approximation obtained by discarding the fast pole at $s = -20$.

```
% MATLAB code
% ex3_tp3.m – dominant poles
H_full = tf(10, conv([1 2 10],[1 20]));
% Drop the fast pole at -20: divide gain by 20 to preserve the DC gain
H_dom  = tf(10/20, [1 2 10]);
 figure;
```

```
step(H_full, 'b', H_dom, 'r--', 8);
legend('Third-order system','Dominant-pole approximation');
grid on;
title('Validation of the dominant-pole approximation');
 % Compare key indicators
disp('Full third-order model :'); stepinfo(H_full)
disp('Dominant approximation :'); stepinfo(H_dom)
```

Exercise 4 — Influence of a zero

Consider the family of second-order systems $H_a(s) = (a\,s + 1)/(s^2 + s + 1)$ for $a \in \{0, 1, 2, 5\}$. Plot the four step responses on the same figure and comment on the influence of the parameter a.

```
% MATLAB code
% ex4_tp3.m — adding a zero to a second-order system
a_values = [0 1 2 5];
figure; hold on;
for k = 1:length(a_values)
    a = a_values(k);
    num = [a 1];
    den = [1 1 1];
    H   = tf(num, den);
    [y,t] = step(H, 0:0.01:15);
    plot(t, y, 'LineWidth', 1.5,...
        'DisplayName', sprintf('a = %d', a));
end
grid on;
xlabel('t (s)'); ylabel('y(t)');
legend('Location','best');
title('Influence of a zero on the second-order response');
```

Discussion. A zero in the left half-plane that lies close to the dominant poles introduces a derivative effect on the response: the rise becomes faster but, for small zero positions, an undershoot or a marked overshoot appears. The influence of the zero becomes negligible when its modulus is large compared with that of the dominant poles.

3.5 Synthesis questions

16. How does the position of the dominant poles in the complex plane influence the speed of response?
17. What relationship exists between the overshoot M_p and the angle θ that the dominant poles make with the imaginary axis?
18. In what circumstances is the dominant-pole approximation no longer valid?

3.6 Conclusion

The location of poles essentially dictates the time response of an LTI system. A first order system is identified by a single time constant while a second order system is identified by the pair (ω_n, ζ). For higher order systems the concept of dominant-poles allows the engineer to reduce the model without losing the fidelity of the long term behaviour. These observations are central to the design decisions that will be taken in forthcoming sessions.

TP 4 — Frequency analysis of LTI systems

4.1 Objectives

- Plot and interpret the Bode diagram of an LTI system (magnitude in dB and phase in degrees).
- Plot and interpret the Nyquist locus of a system in the complex plane.
- Use the Black (Nichols) chart to evaluate the gain and phase margins of a closed-loop system.
- Identify the characteristic frequencies (cut-off frequency, resonance frequency, bandwidth).

4.2 Theoretical reminder

4.2.1 The frequency response

The frequency response of an LTI system is defined as the complex function $H(j\omega)$ obtained by substituting $s = j\omega$ in its transfer function. Physically it describes how a sinusoidal input $u(t) = U_0 \sin(\omega t)$ is transformed by the system into the steady-state output $y(t) = U_0 \, |H(j\omega)| \, \sin(\omega t + \arg H(j\omega))$. The frequency response therefore contains the magnitude $|H(j\omega)|$ — usually expressed in decibels — and the phase $\arg H(j\omega)$.

4.2.2 Bode diagram

The Bode diagram represents the magnitude (in dB) and the phase (in degrees) of the frequency response as functions of the logarithm of the angular frequency. Its main practical advantage is that the magnitude and the phase contributions of cascaded blocks simply add, which makes the graphical construction of a complex transfer function very fast.

The magnitude in decibels reads:

$$|H(j\omega)|_{dB} = 20 \log_{10} |H(j\omega)| \tag{4.1}$$

Each elementary factor of $H(s)$ (constant gain, integrator, first-order pole or zero, second-order block) contributes a known asymptotic slope on the magnitude diagram (−20 dB/dec, −40 dB/dec, etc.) and a known phase contribution. The complete diagram is the sum of these asymptotes corrected by the precise curve at every corner frequency.

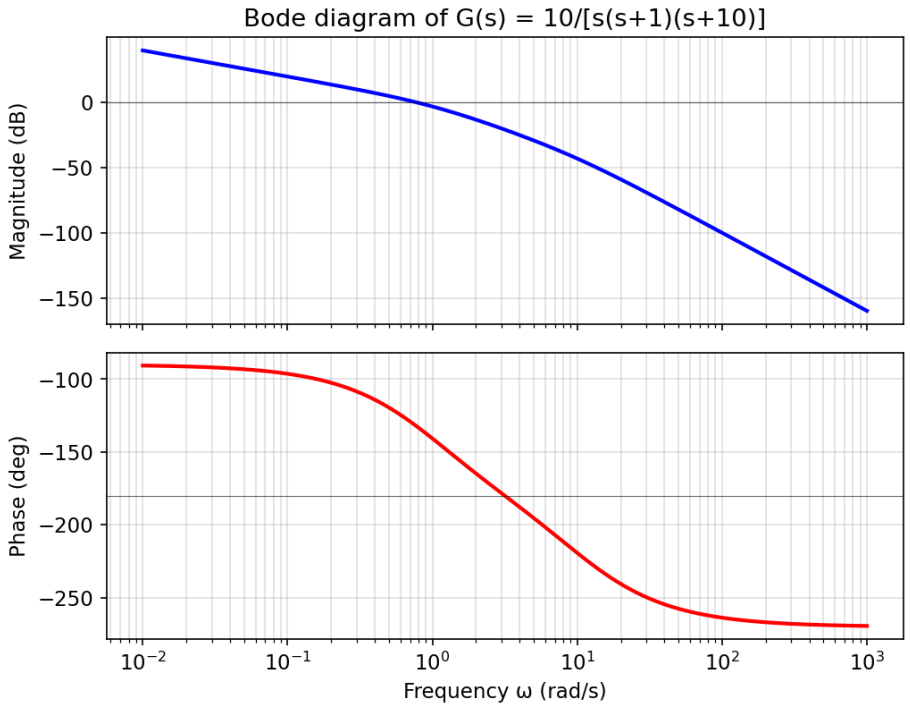

Figure 4.1 — Bode diagram of G(s) = 10/[s(s+1)(s+10)].

4.2.3 Nyquist locus

The Nyquist locus is the parametric curve described in the complex plane by the point of affix H(jω) when ω varies from 0 to +∞. It is a direct geometric representation of the frequency response. The Nyquist criterion, which links the encirclements of the critical point (−1, 0) to the stability of the closed-loop system, will be detailed in TP 5.

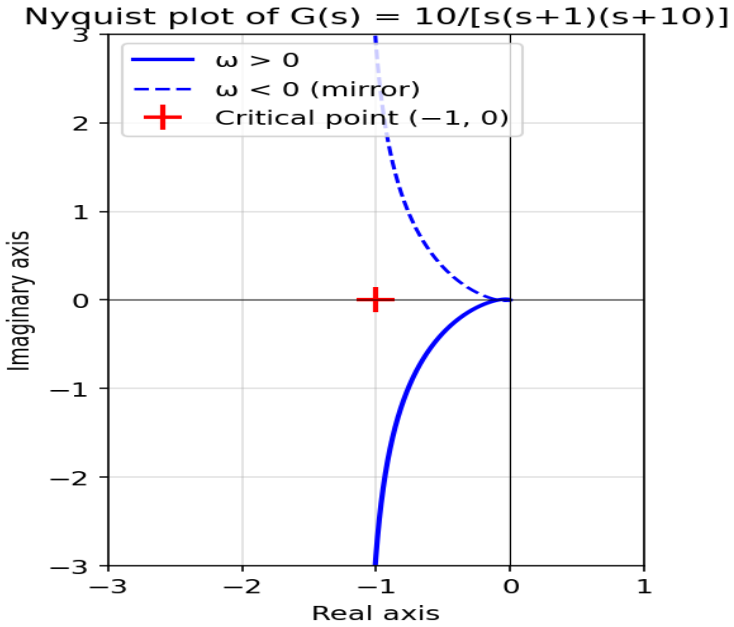

Figure 4.2 — Nyquist locus of the same system. The critical point (−1, 0) is drawn in red.

4.2.4 Black (Nichols) chart

The Black chart — known as the Nichols chart in the English-speaking world — combines the two pieces of information of the Bode diagram on a single plane: the horizontal axis is the phase (in degrees) and the vertical axis is the magnitude (in dB). The frequency ω is the parameter along the curve. Superimposed on this plot, M-circles and N-circles allow the engineer to directly read the magnitude and phase of the closed-loop system from the curve of the open-loop system. This combined view is particularly convenient for the analysis of the stability margins.

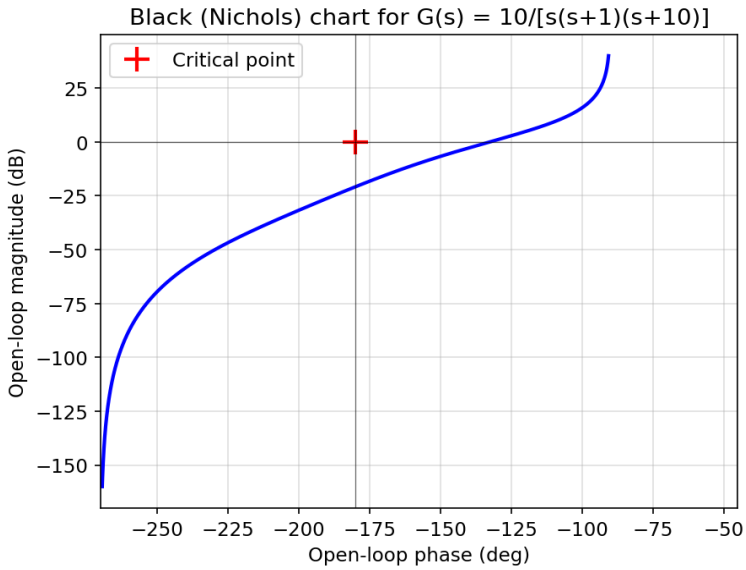

Figure 4.3 — Black (Nichols) chart of G(s) = 10/[s(s+1)(s+10)].

4.2.5 Characteristic frequencies

- Cut-off frequency (−3 dB): the angular frequency at which the magnitude has fallen 3 dB below its DC value. It bounds the so-called passband of the system.
- Resonance frequency: for under-damped second-order systems, the angular frequency at which the magnitude reaches its maximum. It is given by $\omega r = \omega_n \sqrt{(1 - 2\zeta^2)}$ and exists only for $\zeta < 1/\sqrt{2}$.
- Gain crossover frequency ωgc : the angular frequency at which the open-loop magnitude crosses 0 dB.
- Phase crossover frequency ωpc : the angular frequency at which the open-loop phase crosses −180°.

4.3 Preliminary work

19. Sketch the asymptotic Bode diagram of $H(s) = 1/(1 + 0.1\ s)$.
20. Compute the magnitude and the phase at $\omega = 10$ rad/s.

21. Recall the asymptotic phase contribution of a pure integrator.

4.4 Guided exercises

Exercise 1 — Bode diagram of an elementary system

Plot the Bode diagram of H(s) = 1 / (1 + 0.1 s). Verify the asymptotic break at ω = 10 rad/s and the phase of −45° exactly at this corner frequency.

```
% MATLAB code
% ex1_tp4.m — Bode of a first-order system
H = tf(1, [0.1 1]);
 figure;
bode(H, {0.1, 1000});
grid on;
title('Bode diagram of  1 / (1 + 0.1 s)');
 % Numerical reading at ω = 10 rad/s
[mag, phase] = bode(H, 10);
fprintf('|H(j10)| dB = %.2f\n', 20*log10(mag));
fprintf('arg H(j10)  = %.2f deg\n', phase);
```

Exercise 2 — Bode of a third-order system

Reproduce Figure 4.1 and compare with the result obtained by manual asymptotic construction.

```
% MATLAB code
% ex2_tp4.m — Bode of  10/[s(s+1)(s+10)]
G = tf(10, conv([1 0], conv([1 1],[1 10])));
figure;
bode(G, {0.01, 1000});
grid on;
title('Bode diagram of  G(s) = 10 / [s(s+1)(s+10)]');
 % Locate the gain crossover frequency and phase crossover
[Gm, Pm, Wcg, Wcp] = margin(G);
fprintf('Gain crossover  wgc = %.3f rad/s\n', Wcp);
fprintf('Phase crossover wpc = %.3f rad/s\n', Wcg);
fprintf('Gain margin  GM  = %.2f dB\n', 20*log10(Gm));
fprintf('Phase margin PM  = %.2f deg\n', Pm);
```

Exercise 3 — Nyquist locus

```
% MATLAB code
% ex3_tp4.m – Nyquist of the same system
G = tf(10, conv([1 0], conv([1 1],[1 10])));
 figure;
nyquist(G);
axis equal; grid on;
title('Nyquist plot of  G(s) = 10 / [s(s+1)(s+10)]');
```

Question. Locate the critical point (−1, 0) and the point where the locus crosses the negative real axis. Determine its abscissa and deduce the gain margin.

Exercise 4 — Black (Nichols) chart

```
% MATLAB code
% ex4_tp4.m – Nichols / Black plot
G = tf(10, conv([1 0], conv([1 1],[1 10])));
 figure;
nichols(G);
ngrid;
title('Black (Nichols) chart of  G(s) = 10 / [s(s+1)(s+10)]');
```

Question. The function ngrid superimposes the so-called M-circles and N-circles. Read graphically the value of the resonance peak $|T(j\omega_r)|$ of the closed-loop system.

Exercise 5 — Second-order system, resonance and bandwidth

Consider $H(s) = 100 / (s^2 + 4\,s + 100)$. Identify graphically the resonance peak and the bandwidth at −3 dB.

```
% MATLAB code
% ex5_tp4.m – second-order frequency response
wn = 10; z = 0.2;
H  = tf(wn^2, [1 2*z*wn wn^2]);
 figure;
bode(H);
grid on;
title('Bode diagram of a lightly damped second-order system');
 % Resonance and bandwidth
```

```
wr_th = wn*sqrt(1-2*z^2);
Mr_th = 1/(2*z*sqrt(1-z^2));
fprintf('Theoretical resonance frequency : %.3f rad/s\n', wr_th);
fprintf('Theoretical resonance peak       : %.2f dB\n', 20*log10(Mr_th));
 [mag, phase, w] = bode(H);
mag = squeeze(mag);
[Mr, ir] = max(mag);
fprintf('Numerical resonance frequency  : %.3f rad/s\n', w(ir));
fprintf('Numerical resonance peak         : %.2f dB\n', 20*log10(Mr));
```

4.5 Synthesis questions

22. Why is the magnitude expressed in decibels in the Bode diagram?
23. What is the relationship between the Nyquist locus and the polar locus of H(jω)?
24. What information does the Nichols chart provide that the Bode diagram does not?
25. How does the bandwidth of a closed-loop system relate to the speed of its time response?

4.6 Conclusion

The three classical frequency domain representations - Bode, Nyquist and Black - emphasize a particular aspect of the system. The decomposition in elementary blocks is best expressed in the Bode diagram, the Nyquist locus gives a clear geometric picture of the stability of the closed-loop system, the Black chart is the natural framework when dealing directly with the closed-loop magnitude and phase. The next session will combine these complementary points of view to analyze the stability and the accuracy of a feedback system.

TP 5 — Stability and accuracy of closed-loop systems

5.1 Objectives

- Apply the Routh–Hurwitz criterion to assess the stability of a continuous-time LTI system.
- Determine the gain and phase margins of a system from its Bode, Nyquist or Black diagram.
- Plot and use the root locus to predict the behaviour of a closed-loop system when a gain varies.
- Compute the steady-state error for the canonical step, ramp and parabolic inputs.

5.2 Theoretical reminder

5.2.1 Stability of a closed-loop LTI system

A continuous-time LTI system is BIBO-stable if and only if all the poles of its transfer function have strictly negative real parts. For a closed-loop system $T(s) = G(s)/[1 + G(s)\,H(s)]$, stability therefore amounts to studying the roots of the characteristic equation:

$$1 + G(s)\,H(s) = 0 \tag{5.1}$$

5.2.2 The Routh–Hurwitz criterion

The Routh–Hurwitz criterion provides a purely algebraic test on the coefficients of the characteristic polynomial without explicitly computing the roots. Given the polynomial

$$a_n s^n + a_{n-1} s^{n-1} + \ldots + a_1 s + a_0 \tag{5.2}$$

one builds the Routh table by arranging the coefficients on two rows, then completing each subsequent row through the rule:

$$b_i = -\,(1/a_{n-1}) \cdot \det\left(\begin{bmatrix} a_n & a_{n-2i} \\ a_{n-1} & a_{n-2i-1} \end{bmatrix} \right)$$

The system is stable if and only if all the elements of the first column of the table are strictly positive. Any change of sign in this column indicates the presence of a pole with positive real part, and the number of sign changes equals the number of such unstable poles.

5.2.3 Gain margin and phase margin

The gain margin (GM) and the phase margin (PM) are two scalar indicators of robustness, extracted from the open-loop frequency response. They measure how far the system stands from the stability boundary and are read on the Bode, Nyquist or Black diagrams.

Gain margin: extra gain (in dB) that could be inserted in the loop before instability is reached, measured at the phase crossover frequency ωpc where the phase equals −180°:

$$GM\ (dB) = -\ 20\ log_{10}\ |G(j\ \omega pc)| \tag{5.3}$$

Phase margin: extra phase lag (in degrees) that could be introduced before instability is reached, measured at the gain crossover frequency ωgc where the magnitude equals 0 dB:

$$PM = 180° + arg\ G(j\ \omega gc) \tag{5.4}$$

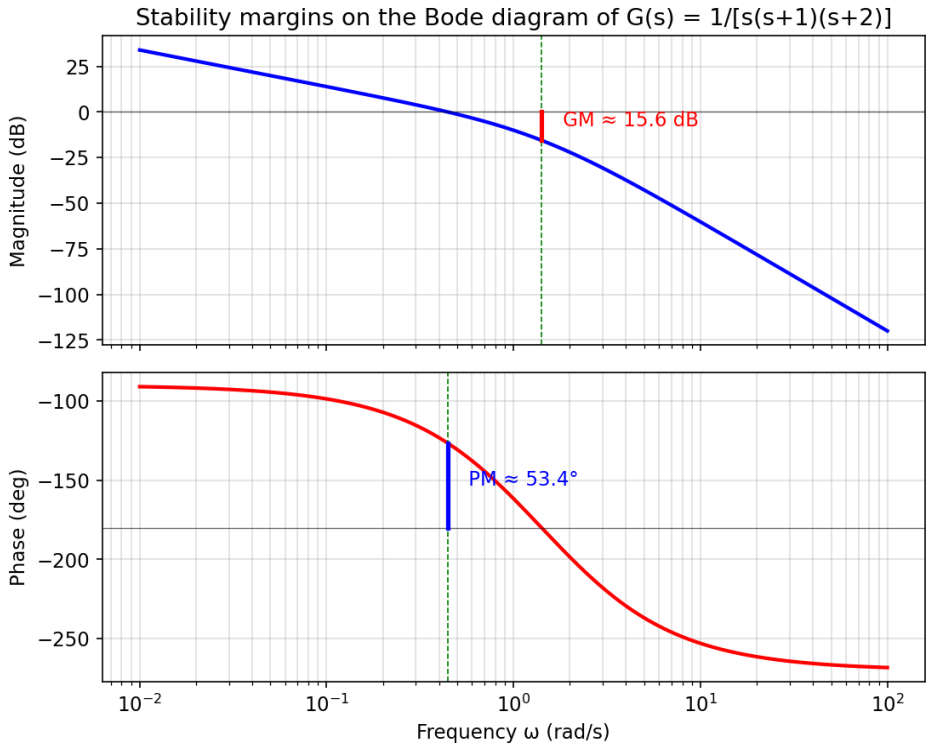

Figure 5.1 — Gain margin and phase margin read directly on the Bode diagram of G(s) = 1/[s(s+1)(s+2)].

A widespread rule of thumb is to specify a phase margin of $PM \geq 45°$ and a gain margin of $GM \geq 6\ dB$. Smaller values lead to a closed-loop response that exhibits significant overshoot or oscillations; larger values yield a sluggish response.

5.2.4 The root locus

The root locus is the geometric place described by the poles of the closed-loop system when one parameter (typically a static gain K) varies from 0 to +∞. It directly visualises the migration of the dominant poles and is therefore an extremely useful design tool. The root locus is built using a small set of geometric rules — branch starting and ending points, asymptotic directions, real-axis behaviour, breakaway and break-in points — that the reader is supposed to know from the course.

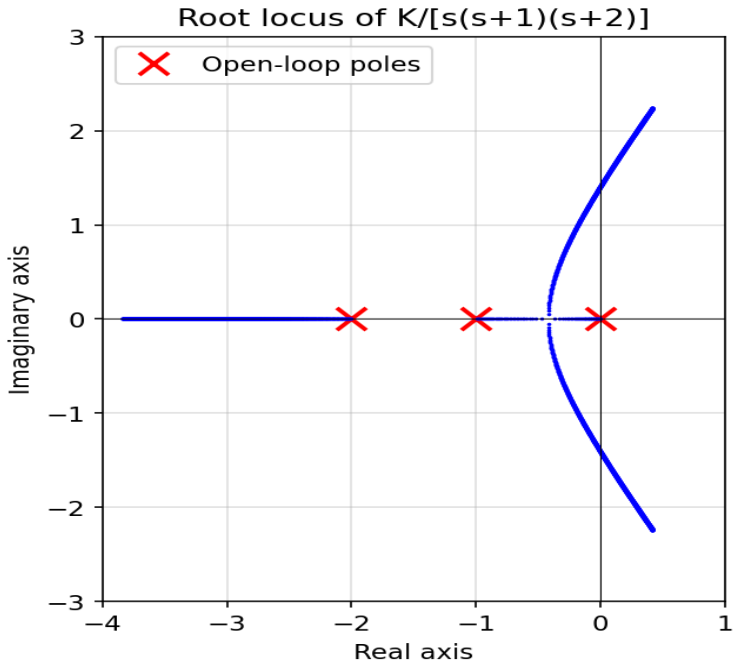

Figure 5.2 — Root locus of K/[s(s+1)(s+2)].

5.2.5 Steady-state error and system type

For a unity-feedback closed-loop system, the final-value theorem applied to the error gives:

$$e_\infty = \lim_{s\to 0}\ s\cdot R(s)/[1+G(s)] \tag{5.5}$$

The result depends on the number of integrators in the open-loop transfer function, called the type of the system, and on the nature of the reference input. The classical results are summarised in Table 5.1, where one defines the static, velocity and acceleration error constants by $K_p = \lim_{s\to 0} G(s)$, $K_v = \lim_{s\to 0} s\,G(s)$ and $K_a = \lim_{s\to 0} s^2\,G(s)$.

System type	Step input R(s) = 1/s	Ramp input R(s) = 1/s²	Parabolic input R(s) = 1/s³
Type 0	$1 / (1 + K_p)$	∞	∞
Type 1	0	$1 / K_v$	∞
Type 2	0	0	$1 / K_a$
Type ≥ 3	0	0	0

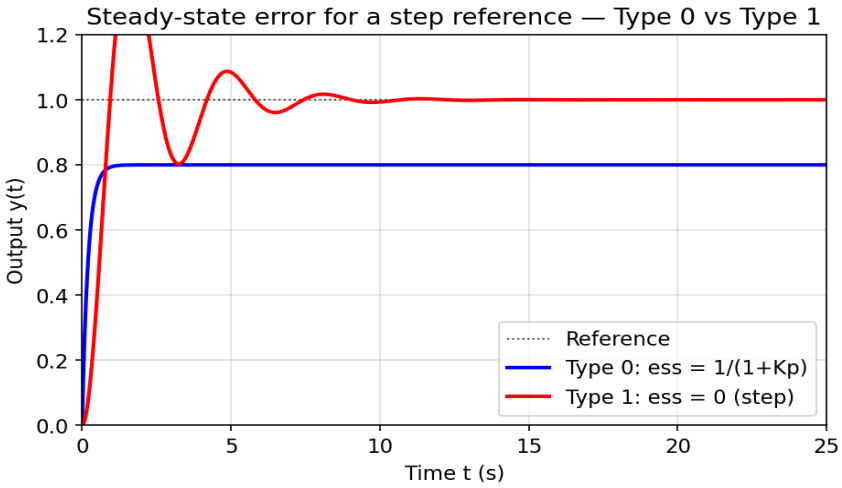

Figure 5.3 — Step response of a Type 0 and of a Type 1 closed-loop system. The Type 1 system tracks the step input with zero steady-state error.

5.3 Preliminary work

26. Apply the Routh–Hurwitz criterion to the polynomial $s^4 + 2\,s^3 + 3\,s^2 + 4\,s + 5$.
27. Sketch the root locus of $K / [s\,(s + 2)]$.
28. Compute the steady-state error of a Type 1 system whose velocity error constant equals $K_v = 10$, when fed by a unit ramp.

5.4 Guided exercises

Exercise 1 — Routh table

Study the stability of the system whose characteristic polynomial is $P(s) = s^4 + 3\,s^3 + 5\,s^2 + 4\,s + K$. Determine the range of values of K for which the closed-loop system is stable. Compare with the result given by the MATLAB function roots.

```
% MATLAB code
% ex1_tp5.m — Routh table / stability range
K_values = [0.5 1 3 6 12 20];
for K = K_values
    p = [1 3 5 4 K];    r = roots(p);
    fprintf('\nK = %.1f\n', K);    disp(r);
    if all(real(r) < 0)
        fprintf('System stable\n');
    else
        fprintf('System UNSTABLE\n');
    end
```

```
end
```

Exercise 2 — Stability margins

Consider the open-loop system G(s) = 1 / [s(s+1)(s+2)]. Compute the gain and phase margins and verify their interpretation on the Bode diagram.

```
% MATLAB code
% ex2_tp5.m — stability margins
G = tf(1, conv([1 0], conv([1 1],[1 2])));
 [Gm, Pm, Wcg, Wcp] = margin(G);
fprintf('Gain margin    GM = %.2f dB\n', 20*log10(Gm));
fprintf('Phase margin   PM = %.2f deg\n', Pm);
fprintf('Gain crossover     w_gc = %.3f rad/s\n', Wcp);
fprintf('Phase crossover    w_pc = %.3f rad/s\n', Wcg);
 figure; margin(G); grid on;
```

Question. Estimate the maximum gain K that can be inserted in the loop before instability occurs. Verify this prediction using rlocus.

Exercise 3 — Root locus

```
% MATLAB code
% ex3_tp5.m — root locus
G = tf(1, conv([1 0], conv([1 1],[1 2])));
 figure;
rlocus(G); grid on; axis equal;
title('Root locus of  K / [ s (s+1) (s+2) ]');
 % Marginally stable gain
K_marg = 6;            % theoretical value
hold on;
T = feedback(K_marg*G, 1);
poles_marg = pole(T);
plot(real(poles_marg), imag(poles_marg), 'mo',...
    'MarkerFaceColor','m', 'MarkerSize',8);
legend('Locus','Poles at K = 6');
```

Exercise 4 — Steady-state error

Consider the unity-feedback system whose open-loop transfer function is given below. Compute the steady-state error in response to a step, a ramp and a parabolic reference.

```
% MATLAB code
% ex4_tp5.m — steady-state error
G   = tf(10, conv([1 1],[1 0]));   % type-1 open loop
T   = feedback(G, 1);              % closed loop, unity feedback
 % --- Step reference
t  = 0:0.01:8;
r  = ones(size(t));
y  = lsim(T, r, t);
err_step = r(end) - y(end);
fprintf('Error to a step  : %.4f\n', err_step);
 % --- Ramp reference
r  = t;
y  = lsim(T, r, t);
err_ramp = r(end) - y(end);
fprintf('Error to a ramp  : %.4f\n', err_ramp);
 % --- Parabolic reference
r  = 0.5 * t.^2;
y  = lsim(T, r, t);
err_par = r(end) - y(end);
fprintf('Error to a parabola : %.4f\n', err_par);
 % Verification with the limit formulas
Kp = dcgain(G);                    % = inf for a type-1 system
Kv = dcgain(tf('s')*G);
fprintf('\nKp = %.3f   Kv = %.3f\n', Kp, Kv);
```

5.5 Synthesis questions

29. Why does the Routh–Hurwitz criterion become difficult to apply when one of the rows of the table contains a zero?
30. Show that increasing the type of the system improves the steady-state accuracy but is generally detrimental to the closed-loop stability.
31. Cite three pieces of information that the root locus immediately reveals.

32. What is the link between the phase margin and the damping ratio of the dominant closed-loop poles?

5.6 Conclusion

The two most important figures of merit of a control loop are stability and steady state accuracy. The Routh-Hurwitz criterion tells you in a binary way if the system is stable or not, whereas the gain and phase margins are measures of the robustness of the system to plant uncertainty. The type of the system also determines the steady-state error to the canonical inputs. The synthesis problem is to design a corrector which guarantees, for the same time, sufficient margins, fast response and small steady-state error. This is the topic of the next session.

TP 6 — Design of a phase-lead compensator

6.1 Objectives

- Recognise the situations in which a phase-lead compensator is the appropriate corrector.
- Derive the analytical expression of the phase-lead transfer function and characterise its frequency response.
- Apply the frequency-response method to determine the parameters of the compensator (α, T, gain K).
- Validate the obtained corrector by comparing the closed-loop time and frequency responses with the uncompensated reference.

6.2 Theoretical reminder

6.2.1 General form of the corrector

A phase-lead corrector is a first-order corrector that injects a positive phase contribution in a chosen frequency range. Its transfer function reads:

$$C(s) = Kc \cdot (1 + a\,T\,s) / (1 + T\,s), \quad a > 1, \quad T > 0 \tag{6.1}$$

The zero $-1/(aT)$ lies closer to the origin than the pole $-1/T$. Hence in the intermediate frequency range, the phase contribution of the zero (positive) prevails over that of the pole, and the global phase contribution is positive — between 0° and 90°.

6.2.2 Frequency response

The maximum positive phase contribution is reached at the geometric mean of the corner frequencies:

$$\omega_m = 1 / (T \sqrt{a}) \tag{6.2}$$

and its value is:

$$\varphi_m = \arcsin((a - 1) / (a + 1)) \tag{6.3}$$

At this same frequency, the magnitude of C is:

$$| C(j\,\omega_m) | = Kc \sqrt{a} \tag{6.4}$$

Equation (6.3) is the cornerstone of the design procedure: choosing α directly fixes the maximum phase boost. Typical values are $\alpha \in [3, 20]$; larger values yield bigger phase boosts but lead to amplification of high-frequency noise (because of the high gain at HF)

and to a larger spread between the corner frequencies. For $\alpha = 10$, $\varphi_m \approx 55°$; for $\alpha = 20$, $\varphi_m \approx 65°$.

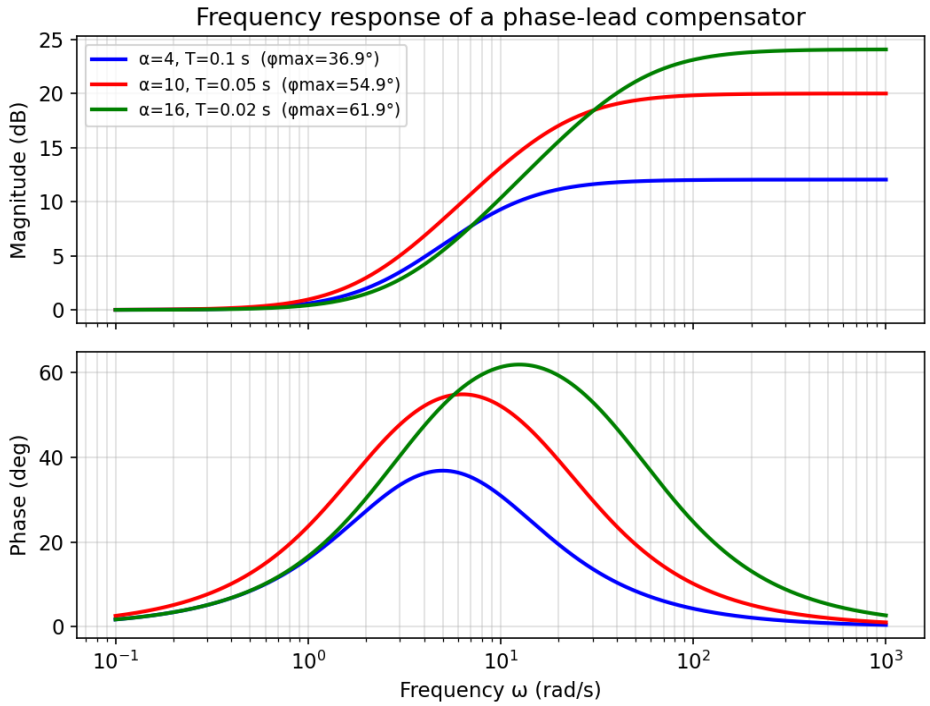

Figure 6.1 — Bode diagram of a phase-lead compensator for three values of (a, T). Note the local positive phase bump.

6.2.3 Frequency-response design procedure

The design proceeds in seven steps:

1. Set the static gain K_c so as to satisfy the steady-state error specification.
2. Plot the Bode diagram of $K_c \cdot G(s)$ and read the current phase margin PM_act.
3. Compute the additional phase needed: $\Delta\varphi$ = PM_des − PM_act + Δ_security (typically Δ_security = 5° – 12°).
4. Deduce α from (6.3) with $\varphi_m = \Delta\varphi$.
5. Determine the new gain crossover frequency ω_m: it is the frequency at which $|K_c \cdot G(j\omega)| = -10 \log_{10} \alpha$ dB.
6. Compute T from (6.2): $T = 1 / (\omega_m \sqrt{\alpha})$.
7. Verify the closed-loop response (time and frequency) and adjust if necessary.

6.3 Preliminary work

33. Establish (6.3) starting from the phase expression of (6.1).

34. Show that the additional gain introduced at the new crossover frequency is $10 \log_{10} \alpha$ dB.
35. Why is it usually preferred to apply a phase-lead corrector rather than to simply reduce the loop gain?

6.4 Guided exercise — design of a complete lead corrector

Specifications. Consider the plant $G(s) = 10 / [\, s\,(s + 1)\,(s + 10)\,]$. Design a phase-lead corrector that guarantees a steady-state error to a unit ramp of $e_\infty \le 0.1$ and a phase margin $PM \ge 50°$.

Step 1 — Static gain Kc

The plant is of Type 1, hence the steady-state error to a unit ramp is $e_\infty = 1/K_v$ with $K_v = \lim_{s\to 0} s \cdot Kc \cdot G(s)$. With $G(s)$ above one obtains $K_v = Kc$, so the requirement $1/K_v \le 0.1$ gives $Kc \ge 10$. One takes $Kc = 10$.

Step 2 — Phase margin of the gain-only compensated plant

```
% MATLAB code
% lead_design.m — phase-lead compensator
G  = tf(10, conv([1 0], conv([1 1],[1 10])));
Kc = 10;
G_K = Kc * G;
 [Gm, Pm_act, Wcg, Wcp] = margin(G_K);
fprintf('Phase margin with Kc = 10 alone : %.2f deg\n', Pm_act);
figure; margin(G_K); grid on;
title('Bode of Kc * G(s) — phase margin to be improved');
```

One typically observes a phase margin of the order of 20° — clearly insufficient.

Step 3 — Required phase boost

Aiming at PM_des = 50° and adding a 10° safety margin to compensate for the frequency shift introduced by the corrector itself, one needs $\Delta\varphi = 50° - PM_{act} + 10°$. With a typical value $PM_{act} = 20°$, $\Delta\varphi \approx 40°$.

Step 4 — Computation of α

From (6.3): $a = (1 + \sin\Delta\varphi) / (1 - \sin\Delta\varphi)$. For $\Delta\varphi = 40°$, $a \approx 4.6$.

Step 5 — Determination of ωm

The phase-lead corrector also raises the magnitude by $10 \log_{10} a \approx 6.6\ dB$ at $\omega = \omega m$. The new crossover frequency is therefore the one at which the gain of $Kc \cdot G(s)$ is approximately −6.6 dB. Reading the Bode diagram (or using MATLAB) one finds $\omega m \approx 2.4\ rad/s$.

Step 6 — Computation of T

From (6.2): $T = 1 / (\omega m \sqrt{a}) \approx 0.19\ s$. The zero is then at $1/(a\ T) \approx 1.13\ rad/s$ and the pole at $1/T \approx 5.2\ rad/s$.

Step 7 — Verification

```
% MATLAB code
% Continuation of lead_design.m
alpha = 4.6;
phi_m = asind((alpha-1)/(alpha+1));
fprintf('Maximum phase boost  phi_m = %.2f deg\n', phi_m);
 % Choose ωm from the Bode diagram of Kc*G (−10 log α point)
wm = 2.4;
T  = 1/(wm*sqrt(alpha));
fprintf('Parameters of the corrector : alpha = %.2f, T = %.3f s\n', alpha, T);
 C = Kc * tf([alpha*T 1], [T 1]);
disp('Corrector C(s) :'); zpk(C)
 % Open-loop and closed-loop transfer functions
OL_uncomp = Kc * G;
OL_comp   = C  * G;
CL_uncomp = feedback(OL_uncomp, 1);
CL_comp   = feedback(OL_comp  , 1);
 [~, Pm_uncomp] = margin(OL_uncomp);
[~, Pm_comp  ] = margin(OL_comp  );
fprintf('PM before compensation : %.2f deg\n', Pm_uncomp);
fprintf('PM after  compensation : %.2f deg\n', Pm_comp  );
 % Bode diagrams
figure;
bode(OL_uncomp,'b', OL_comp,'r'); grid on;
legend('Kc * G(s) (without lead)','C(s) * G(s) (with lead)');
title('Open-loop frequency response');
 % Step responses
```

```
figure;
step(CL_uncomp,'b', CL_comp,'r', 6); grid on;
legend('Without lead corrector','With lead corrector');
title('Closed-loop step response');
```

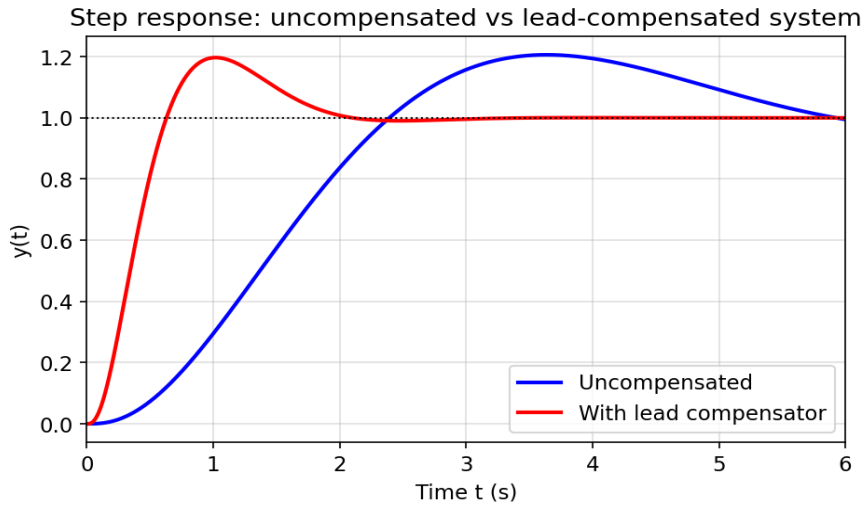

Figure 6.2 — Step response of the closed-loop system before and after the introduction of the lead compensator.

Reading. The corrector reduces the overshoot, slightly accelerates the response and considerably improves the phase margin. A residual fine-tuning of α or of ωm can be done if necessary.

6.5 Synthesis questions

36. Why is the security margin Δ_security necessary in the design procedure?
37. What is the practical limitation of using a single lead corrector with α too large?
38. Cite an alternative corrector that improves the steady-state accuracy without changing the high-frequency behaviour.
39. In what situation would you choose a phase-lag (or lead-lag) corrector rather than a pure phase-lead?

6.6 Conclusion

The phase-lead compensator is one of the most used correctors of classical control. It is designed in the frequency domain to increase the phase margin and speed up the closed-loop response at the expense of a moderate amount of high-frequency gain. The design process is systematic and can be completely automated in MATLAB. The last practical

session is to apply the techniques developed during the manual on a real laboratory bench.

TP 7 — Real laboratory benches: position, speed, temperature, level and flow

7.1 Objectives

- Connect and operate a real laboratory bench involving an actuator, a sensor and an analog controller.
- Identify the dynamical model of a real plant from its open-loop response.
- Tune the parameters of a PID corrector with one of the classical empirical methods (Ziegler–Nichols, trial-and-error).
- Investigate the experimental performance of a closed-loop system and compare it with the predictions of MATLAB / Simulink.

7.2 Theoretical reminder

7.2.1 The PID corrector

The proportional–integral–derivative (PID) corrector remains by far the most widely used controller in industrial applications. Its parallel form is:

$$C(s) = Kp + Ki/s + Kd \cdot s = Kp \cdot (1 + 1/(Ti\, s) + Td\, s) \qquad (7.1)$$

with Kp the proportional gain, Ti the integral time constant and Td the derivative time constant. The integral action eliminates the steady-state error on a step input, while the derivative action introduces a predictive component that improves the damping. In practice, the pure derivative is replaced by a filtered derivative $Td\, s / (1 + Td\, s / N)$ with N between 5 and 20, in order to avoid amplifying the high-frequency noise.

7.2.2 Empirical tuning rules (Ziegler–Nichols)

Two classical empirical methods avoid the explicit identification of the plant model. The first method (open-loop, Ziegler–Nichols I) requires only the step response of the plant. From the inflection tangent, one measures the delay time L and the time constant T, then reads the suggested PID parameters from Table 7.1.

Corrector type	Kp	Ti	Td
P	T / L	∞	0
PI	0.9 T / L	3.33 L	0
PID	1.2 T / L	2 L	0.5 L

The second method (closed-loop, Ziegler–Nichols II) requires no time response: starting from a P-only corrector, the proportional gain is gradually increased until the closed-loop system exhibits sustained oscillations of period *Tu* at the so-called ultimate gain *Ku*. The PID parameters then read:

Corrector type	Kp	Ti	Td
P	0.5 Ku	∞	0
PI	0.45 Ku	Tu / 1.2	0
PID	0.6 Ku	Tu / 2	Tu / 8

These rules give a first usable tuning that the engineer can then refine according to the specifications. The student is encouraged to compare the obtained settings with those predicted by the frequency-response method of TP 6.

7.3 Bench A — Position and speed servo

7.3.1 Description

The position servo bench is built around a small DC motor coupled, through a gearbox, to an output shaft fitted with an angular position sensor (potentiometer or incremental encoder). A tachogenerator provides the speed feedback signal. The motor is driven by a power amplifier whose input is the command voltage delivered by an analog PID corrector. The complete architecture is shown in Figure 7.1.

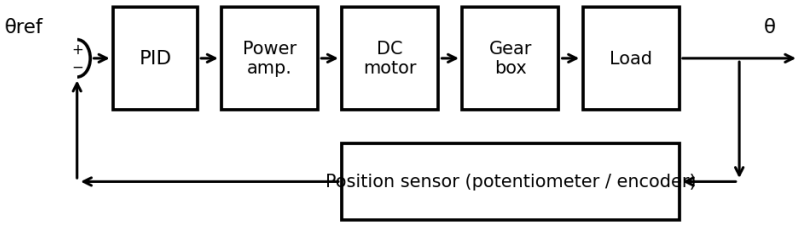

Figure 7.1 — Block diagram of the position servo bench.

7.3.2 Experimental work

1. Connect the bench to the supply. Switch on the equipment, then record the open-loop step response of the speed loop (input: a voltage step of 1 V, output: the tacho-generator voltage). Save the curve.

2. From the recorded curve, identify the parameters of a first-order model $G(s) = K/(1 + \tau s)$.
3. Reproduce the experiment in Simulink using your identified model. Compare the simulated curve with the experimental one.
4. Close the speed loop with a P-only corrector and progressively increase the gain. Record the closed-loop step response for three values of Kp.
5. Replace the P-corrector by a PI corrector tuned with the Ziegler–Nichols rules. Compare the rise time, overshoot and steady-state error of the new closed-loop response with the previous ones.
6. For the position loop, repeat the procedure with a PID corrector. The position transfer function contains an extra integrator (since position is the integral of speed); therefore the system is naturally of Type 1.

7.3.3 MATLAB simulation

```
% MATLAB code
% bench_position.m — simulation of the position servo
% --- Identified plant (speed loop)
K_speed   = 12.5;          % V / V  -- to be replaced by your value
tau_speed = 0.15;          % s
G_speed   = tf(K_speed, [tau_speed 1]);
 % --- Position plant (speed + integrator)
G_pos = G_speed * tf(1, [1 0]);
 % --- PID corrector, parallel form
Kp = 4;  Ki = 8;  Kd = 0.05;
C  = pid(Kp, Ki, Kd);
 % --- Closed loop
T_pos = feedback(C*G_pos, 1);
 figure;
step(T_pos, 2); grid on;
title('Closed-loop step response of the position servo');
ylabel('Angular position (rad)'); xlabel('Time (s)');
```

7.4 Bench B — Temperature regulation

7.4.1 Description

The temperature bench consists of a heating element (resistor or Peltier module) immersed in a small thermal mass, a Pt100 or thermocouple sensor with its measuring transmitter, and an analog or digital PID controller. The dynamics of a thermal process are typically well represented by a first-order model with a transport delay:

$$G(s) = K \cdot e^{-L s} / (1 + \tau s) \qquad (7.2)$$

where the time constant τ is generally large (tens of seconds to minutes) and the delay L of the order of a few seconds.

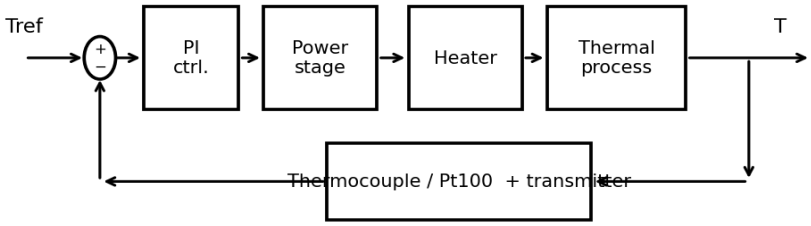

Figure 7.2 — Block diagram of the temperature regulation bench.

7.4.2 Experimental work

7. Record the open-loop step response of the heating circuit (input: a step of the heating-power command, output: the temperature read by the sensor). Estimate the static gain K, the time constant τ and the delay L.
8. Apply the open-loop Ziegler–Nichols method (Table 7.1, last row) to obtain a first PID tuning.
9. Run the closed loop with the obtained PID parameters. Record the response to a 5 °C step change of the set-point.
10. Refine the PID parameters in order to reduce the overshoot below 10 %. Compare the obtained values to the Ziegler–Nichols ones.

7.5 Bench C — Level / flow regulation

7.5.1 Description

The level / flow bench comprises a transparent tank fitted with a level sensor, a pump driven by a variable-frequency drive (manipulating the inflow) and a manual valve at the outlet. The inflow can also be measured by a flow transmitter (electromagnetic or

rotameter type), enabling either a level or a flow control loop. The block diagram is given in Figure 7.3.

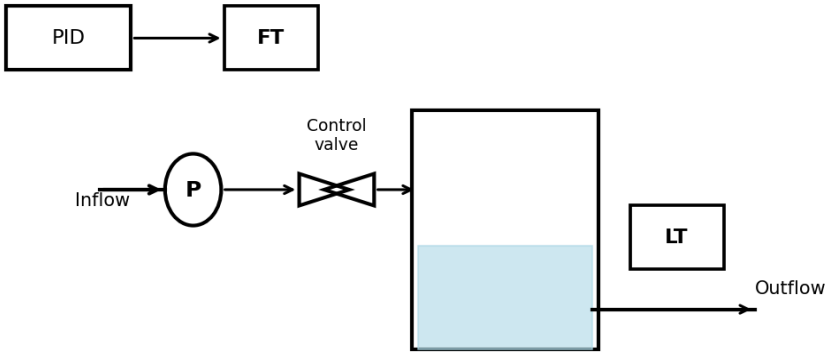

Figure 7.3 — Block diagram of the level / flow regulation bench.

7.5.2 Modelling

The mass-balance equation of the tank gives $A \cdot dh/dt = q_{in} - q_{out}$ with A the cross-section of the tank, h the water level and q_{in}, q_{out} the inflow and outflow. Assuming a turbulent flow through the outlet valve, $q_{out} = \beta \sqrt{h}$, which is non-linear. Linearisation around an operating point yields a first-order behaviour:

$$\Delta h(s) / \Delta q_{in}(s) = R / (1 + R\, A\, s) \tag{7.3}$$

where R denotes the linearised hydraulic resistance of the outlet.

7.5.3 Experimental work

11. Bring the level to its mid-range value, then introduce a small step of the inflow command. Record the level response and extract the model parameters.
12. Design a PI corrector that guarantees a steady-state error of zero on a step set-point change and a settling time of less than 30 s. Compute Kp and Ti with the frequency-response method of TP 6.
13. Implement the corrector and validate the closed-loop response on the bench. Investigate the rejection of a disturbance simulated by a sudden opening of the outlet valve.
14. Optional: configure the bench in cascade control, where an outer level loop sends its output as the set-point of an inner flow loop. Compare the disturbance-rejection performance with that of the single-loop architecture.

7.5.4 Cascade control — MATLAB simulation

```
% MATLAB code
% bench_level_cascade.m — cascade control of the level
```

```
% Inner flow loop : fast first-order model
G_flow  = tf(1, [0.5 1]);
C_flow  = pid(2, 5, 0);
T_flow  = feedback(C_flow*G_flow, 1);
  Outer level loop : first-order model with the inner loop as actuator
G_level = T_flow * tf(1, [10 0.5]);
C_level = pid(0.8, 0.05, 0);
 % Closed loop, outer
T_lvl   = feedback(C_level*G_level, 1);
 figure;
step(T_lvl, 80); grid on;
title('Cascade-controlled level — set-point step response');
ylabel('Tank level (m)');
xlabel('Time (s)');
```

7.6 Synthesis questions

40. Why is the integral action systematically introduced when one wants to ensure zero steady-state error on a step set-point change?
41. What is the practical role of the derivative filter and of the saturation block in an industrial PID implementation?
42. Cite three advantages of cascade control with respect to a single-loop architecture.
43. How are the empirical Ziegler–Nichols rules affected by a process with a long dead time?

7.7 Conclusion

This last practical session has exposed the student to the unavoidable mismatch between an idealised academic model and the real behaviour of a physical plant: non-modelled dynamics, transport delays, sensor noise, actuator saturation and friction. The PID corrector - robust, well understood and easily tunable - turns out to be a very versatile tool. The seven chapters of this handbook have thus addressed the entire chain of competences expected from a junior control engineer: modelling, analysis, design and experimental validation.

Appendix C — Tutorial Problems and Worked Solutions

The following tutorial problems are intended for self-study between laboratory sessions. They are arranged in the order of the corresponding TPs and graded from elementary to moderately demanding. A complete worked solution is given for each problem so that the student can check both the final numerical answer and the line of reasoning.

Problem 1 — Step response of a first-order system

Consider the first-order plant $G(s) = 4 / (2s + 1)$. (a) Identify the static gain K and the time constant τ. (b) Compute the value of the step response at $t = \tau$, $t = 3\tau$ and $t = 5\tau$ for a unit-step input. (c) State the 5% settling time.

Solution

(a) Putting $G(s)$ in the canonical form $K / (\tau s + 1)$ gives $K = 4$ and $\tau = 2$ s.

(b) The unit-step response of a first-order system is $y(t) = K\,(1 - e^{-t/\tau})$. At $t = \tau$, $y = 4(1 - e^{-1}) \approx 2.528$. At $t = 3\tau$, $y \approx 3.801$ (95%). At $t = 5\tau$, $y \approx 3.973$ (99.3%).

(c) The 5% settling time is $t_s = 3\tau = 6$ s.

Problem 2 — Second-order characteristics from the transfer function

Given $G(s) = 25 / (s^2 + 4s + 25)$: identify ω_n and ζ, then predict the overshoot, the peak time and the 5% settling time of the unit-step response.

Solution

Comparing with the canonical form $\omega_n^2 / (s^2 + 2\zeta\omega_n s + \omega_n^2)$ gives $\omega_n^2 = 25$, hence $\omega_n = 5$ rad/s, and $2\zeta\omega_n = 4$, hence $\zeta = 0.4$.

Damped natural frequency $\omega\_d = \omega_n\sqrt{(1 - \zeta^2)} = 5 \times 0.9165 = 4.583$ rad/s. Overshoot $M_p = \exp(-\zeta\pi/\sqrt{(1-\zeta^2)}) = \exp(-1.371) \approx 25.4\%$. Peak time $t_p = \pi/\omega\_d \approx 0.685$ s. 5% settling time $t_s \approx 3/(\zeta\omega_n) = 3/2 = 1.5$ s.

Problem 3 — Block-diagram reduction

Three blocks are connected as shown: G_1 and G_2 in series in the forward path, with a unity-gain feedback wrapped around their cascade, and a third block G_3 placed after the feedback junction. Find the overall transfer function from input $R(s)$ to output $Y(s)$.

Solution

The inner loop ($G_1\,G_2$ with unity negative feedback) reduces to $G_1\,G_2 / (1 + G_1\,G_2)$. The cascade with G_3 then gives:

$$T(s) = Y(s) / R(s) = G_1(s)\,G_2(s)\,G_3(s) / [\,1 + G_1(s)\,G_2(s)\,]$$

Note that G_3 does not appear in the characteristic polynomial $1 + G_1\,G_2$; closed-loop stability is fixed by the inner loop only, while the outer cascade with G_3 modifies only the forward gain and the zero pattern.

Problem 4 — Routh-Hurwitz criterion

For the closed-loop characteristic polynomial $P(s) = s^4 + 2s^3 + 3s^2 + 4s + K$, find the range of K for which the closed-loop system is stable.

Solution

The Routh array is constructed as follows. Coefficients of the polynomial:

- s^4 : 1, 3, K
- s^3 : 2, 4, 0

Computing the s^2 row:

$$b_1 = (2\times3 - 1\times4)/2 = 1, \quad b_2 = (2\times K - 1\times0)/2 = K$$

s^2 : 1, K. Computing the s row:

$$c_1 = (1\times4 - 2\times K)/1 = 4 - 2K$$

s^1 : $4 - 2K$, 0. Finally s^0 : K. For stability all first-column entries must be positive:

- $2 > 0$ ✓ (always);
- $1 > 0$ ✓ (always);
- $4 - 2K > 0 \Rightarrow K < 2$;
- $K > 0$.

Conclusion: the closed loop is stable for $0 < K < 2$.

Problem 5 — Phase margin and crossover frequency

For the open-loop transfer function $L(s) = 10 / [\,s\,(s + 1)\,(s + 5)\,]$ in unity-feedback configuration, estimate the gain crossover frequency and the phase margin without resorting to Matlab.

Solution

At the gain crossover frequency ω_c the magnitude of L(jω) is unity. The poles are at 0, −1 and −5, so the magnitude is approximately

$$|L(j\omega)| \approx 10 / [\, \omega \cdot \sqrt{(1+\omega^2)} \cdot 5 \cdot \sqrt{(1+(\omega/5)^2)}\,]$$

For low ω (ω << 1), |L| ≈ 10/(5ω) = 2/ω, giving |L| = 1 at ω ≈ 2 rad/s. Refining by including the (s+1) factor: at ω = 2, √5 ≈ 2.236, so |L| ≈ 10/(2 × 2.236 × 5 × 1.077) ≈ 0.415. We need a lower ω; trying ω = 1.2: |L| ≈ 10/(1.2 × 1.562 × 5 × 1.028) ≈ 1.04. So ω_c ≈ 1.2 rad/s.

Phase: arg L(jω_c) = −90° − arctan(ω_c) − arctan(ω_c/5) ≈ −90° − 50.2° − 13.5° = −153.7°. Phase margin PM = 180° + arg L(jω_c) ≈ 26.3°. The margin is positive — the closed loop is stable — but small, and a substantial overshoot is to be expected. A phase-lead compensator (TP6) would improve transient behaviour considerably.

Problem 6 — Steady-state error of a Type-1 system

A unity-feedback system has open-loop transfer function L(s) = 20 / [s (s + 4)]. Find the steady-state error for (a) a unit-step reference, (b) a unit-ramp reference, (c) a parabolic reference r(t) = t²/2.

Solution

The system is of type 1 (one pole at the origin in L(s)). The error coefficients are:

- Position constant $K_p = \lim_{s\to 0} L(s) = \infty \Rightarrow$ step error e_ss = 1/(1+K_p) = 0.
- Velocity constant $K_v = \lim_{s\to 0} s\, L(s) = 20/4 = 5 \Rightarrow$ ramp error e_ss = 1/K_v = 0.2.
- Acceleration constant $K_a = \lim_{s\to 0} s^2 L(s) = 0 \Rightarrow$ parabolic error e_ss = ∞.

Tracking a parabolic input with zero asymptotic error would require a Type-2 system (an additional integrator in the loop).

Problem 7 — Phase-lead compensator design

A plant has transfer function G(s) = 5 / [s (s + 1)]. Design a phase-lead compensator C(s) = K (1 + ατ s) / (1 + τ s) that gives a phase margin of approximately 50° while keeping a velocity-error constant K_v ≥ 5.

Solution

Step 1 — Set the gain. $K_v = \lim_{s\to 0} s\, C(s)\, G(s) = K \times 5 = 5K$. For K_v = 5 take K = 1. The uncompensated open loop is then $L_0(s) = 5/[s(s+1)]$.

Step 2 — Phase margin of L_0. At gain crossover, $|L_0(j\omega)| = 5/(\omega\sqrt{(1+\omega^2)}) = 1$, which gives $\omega_c \approx 2.06$ rad/s. Phase: $-90° - \arctan(2.06) = -90° - 64.1° = -154.1°$, so $PM_0 \approx 25.9°$.

Step 3 — Required additional phase. Target 50° plus 5° safety margin = 55°. Phase to add $\varphi_m = 55° - 25.9° = 29.1°$.

Step 4 — Compute α. From $\sin\varphi_m = (\alpha - 1)/(\alpha + 1)$:

$$a = (1 + \sin 29.1°) / (1 - \sin 29.1°) = 1.486/0.514 = 2.89$$

Step 5 — At the new crossover, the compensator contributes $10 \log_{10}(\alpha) \approx 4.6$ dB of gain. We need the original $|L_0|$ to be −4.6 dB ≈ 0.589 at the new crossover. Solving $5/(\omega\sqrt{(1+\omega^2)}) = 0.589$ gives $\omega_m \approx 2.65$ rad/s.

Step 6 — Place the compensator so that its maximum phase coincides with ω_m:

$$\tau = 1 / (\omega_m \sqrt{a}) = 1 / (2.65 \times 1.700) = 0.222 \Rightarrow a\tau = 0.642$$

The final compensator is:

$$C(s) = (1 + 0.642\, s) / (1 + 0.222\, s)$$

Step 7 — Verification. With this C(s) in cascade with G(s), the new open loop has phase margin close to the target of 50° and the velocity-error constant is unchanged.

Problem 8 — Ziegler-Nichols PID tuning

A process responds to a unit-step in input with an S-shaped curve of the type usual in thermal systems. The delay before the response begins is estimated by drawing the tangent at the inflection point at L = 1.5 s, and the process time-constant is T = 6 s; the asymptotic gain is K = 2. Tune a PID controller by the Ziegler-Nichols open-loop (reaction-curve) rule.

Solution

The Ziegler-Nichols open-loop rule for a PID controller of standard form $K_p (1 + 1/(T_i s) + T_d s)$ is:

- $K_p = 1.2\ T\ /\ (K\ L) = 1.2 \times 6\ /\ (2 \times 1.5) = 2.4$
- $T_i = 2\ L = 3.0$ s
- $T_d = 0.5\ L = 0.75$ s

These values give a closed-loop response with about a quarter-amplitude decay; some authors recommend reducing K_p by a factor of 0.6 and increasing T_i by 25% for less oscillatory tracking. The values above are an excellent starting point that one then fine-tunes in the laboratory.

Problem 9 — Cascade control

In a heat-exchanger temperature loop, the primary measurement is the outlet temperature T (slow, time-constant 60 s) and the secondary measurement is the steam flow F (fast, time-constant 3 s). Explain qualitatively why a cascade structure — outer T-controller setting the set-point of an inner F-controller — outperforms a single PID controller acting directly on the steam valve.

Solution

Single-loop control has to deal with two different types of disturbance, variations of steam-supply pressure (that affect flow at constant valve opening) and variations of feed-water temperature (that affect outlet temperature directly). With only one loop, the outer controller can only react after the disturbance has propagated through the slow exchanger dynamics, which results in large transient excursions.

In the cascade configuration the inner flow loop is fast: a change in the steam supply produces an immediate mismatch between F and its set-point, which is corrected by the inner controller within a few seconds -- before the outlet temperature has had time to deviate appreciably. The outer temperature controller sees a much more predictable plant whose dynamics are essentially those of the exchanger alone, and it can be tuned for tighter outlet-temperature regulation than would be safe on the original loop. As a rule of thumb the inner loop should be at least 3 times faster than the outer. Here 60/3 = 20 which is comfortable.

Problem 10 — Discretisation for digital implementation

A continuous lead compensator C(s) = (1 + 0.6 s) / (1 + 0.2 s) is to be implemented on a microcontroller sampling at T_s = 10 ms. Derive the discrete equivalent using the Tustin (bilinear) transform.

Solution

The bilinear substitution $s = (2/T_s)(z-1)/(z+1)$ gives, with $T_s = 0.01$:

$$C(z) = (1 + 0.6 \times 200\,(z-1)/(z+1)) \,/\, (1 + 0.2 \times 200\,(z-1)/(z+1))$$

Multiplying numerator and denominator by (z+1) and collecting:

$$C(z) = [\,121\,z - 119\,] \,/\, [\,41\,z - 39\,]$$

or, normalised to leading coefficient 1 in the denominator, $C(z) = (2.951\,z - 2.902)/(z - 0.951)$. The corresponding difference equation is $u_k = 0.951\,u_{k-1} + 2.951\,e_k - 2.902\,e_{k-1}$, which is straightforward to implement.

Problem 11 — Effect of a non-minimum-phase zero

Compare the unit-step responses of two systems with the same poles but different zeros: $G_1(s) = (s + 2) / [(s + 1)(s + 4)]$ and $G_2(s) = (-s + 2) / [(s + 1)(s + 4)]$. Comment on the qualitative difference.

Solution

Both systems have the same poles (−1 and −4) and the same steady-state gain $G(0) = 2/4 = 0.5$. The difference lies in the zero: G_1 has a left-half-plane zero at $s = -2$ (minimum-phase), while G_2 has a right-half-plane zero at $s = +2$ (non-minimum-phase).

The unit-step response of G_1 is monotonically increasing from 0 towards 0.5. The response of G_2, however, initially moves in the wrong direction: it goes negative before turning around and rising to its final value of 0.5. This "inverse response" is a hallmark of non-minimum-phase systems and represents a fundamental limitation on the speed of closed-loop control: no matter how the controller is designed, the closed loop must allow time for the inverse-response transient to play out before the output can track a reference change.

Physical examples include boiler drum level (cold feed-water briefly contracts the steam bubbles and lowers the level before the long-term filling effect takes over), bicycle counter-steering, and the longitudinal control of certain aircraft. The student should recognise non-minimum-phase plants from their pole-zero map and adjust expectations of closed-loop bandwidth accordingly.

Problem 12 — Disturbance rejection vs reference tracking

For a unity-feedback loop with controller C(s) and plant G(s), write the closed-loop transfer functions from reference r and from output-disturbance d to the output y. Comment on the engineering implication when C(s) contains an integrator.

Solution

With $L(s) = C(s)\,G(s)$, elementary algebra gives:

$$Y(s) = [\, L / (1 + L) \,]\, R(s) + [\, 1 / (1 + L) \,]\, D(s) = T(s)\, R(s) + S(s)\, D(s)$$

where T(s) is the complementary sensitivity and S(s) is the sensitivity function. Note the identity $T + S = 1$, which holds for every frequency.

If C(s) contains an integrator (a pole at $s = 0$), then $L(s) \to \infty$ as $s \to 0$, so $T(0) = 1$ (perfect steady-state tracking of step references) and $S(0) = 0$ (perfect steady-state rejection of step disturbances). This is precisely the reason why the integral action in a PI or PID controller

is essential whenever zero steady-state offset is required, whether the disturbance is a load change on the plant or a slow drift in the set-point.

The identity $T + S = 1$ also illuminates a fundamental trade-off: making T close to 1 over a frequency range (good tracking) forces S close to 0 over the same range (good disturbance rejection), but the same identity prevents one from being made arbitrarily small without affecting the other elsewhere on the frequency axis. This is the essence of the Bode sensitivity integral and a recurring theme in robust-control design.

Appendix A — MATLAB / Control System Toolbox cheat sheet

This appendix gathers the MATLAB instructions most frequently used throughout the manual. The functions are grouped by purpose. The complete documentation can always be accessed from MATLAB through the command help function_name or doc function_name.

A.1 Building LTI models

Instruction	Description
sys = tf(num, den)	Build a transfer function from its numerator and denominator coefficients.
sys = zpk(z, p, k)	Build a zero-pole-gain model from the row vectors of zeros, poles and gain.
sys = ss(A, B, C, D)	Build a state-space model from its four matrices.
s = tf('s')	Define the Laplace variable to write transfer functions in symbolic style.
sys = minreal(sys)	Cancel out hidden common factors between numerator and denominator.
sys = c2d(sys, Ts)	Discretise a continuous LTI model with sampling period Ts.

A.2 Interconnections

Instruction	Description
series(G1, G2)	Cascade (series) connection.
parallel(G1, G2)	Parallel connection.
feedback(G, H)	Negative feedback connection (default sign = −1).
feedback(G, H, +1)	Positive feedback connection.

append(...)	Diagonal interconnection of several SISO models into a MIMO one.

A.3 Time-domain analysis

Instruction	Description
step(sys)	Plot the unit-step response.
impulse(sys)	Plot the unit-impulse response.
lsim(sys, u, t)	Simulate the response to an arbitrary input.
initial(sys, x0)	Response of a state-space system to initial conditions.
stepinfo(sys)	Return rise time, settling time, overshoot, peak time, peak value.
pole(sys), zero(sys)	Numerical computation of poles and zeros.
dcgain(sys)	Static (DC) gain of an LTI model.

A.4 Frequency-domain analysis

Instruction	Description
bode(sys)	Bode diagram of an LTI model.
nyquist(sys)	Nyquist locus.
nichols(sys); ngrid	Black (Nichols) chart with closed-loop circles.
margin(sys)	Return and plot gain and phase margins.
bandwidth(sys)	Return the −3 dB cut-off frequency.
rlocus(sys)	Root locus of $K\,G(s)$ as K varies from 0 to ∞.
rlocfind(sys)	Interactively read a gain on the root locus.

A.5 Corrector design

Instruction	Description
pid(Kp, Ki, Kd)	Build a parallel-form PID corrector.
pidtune(G, 'PID')	Automatic tuning of a PID corrector.
sisotool(G)	Graphical design environment for SISO correctors.
systune, looptune	Advanced multi-variable tuning (Robust CST).

A.6 Useful Simulink blocks

Block	Library	Role
Step	Sources	Step input signal.
Sine Wave	Sources	Sinusoidal input.
Transfer Fcn	Continuous	Continuous LTI transfer function.
Integrator	Continuous	Continuous-time integrator.
PID Controller	Continuous	Parallel-form PID, with optional anti-windup.
Sum	Math Operations	Linear combination of input signals (configurable signs).
Gain	Math Operations	Multiplication by a constant.
Saturation	Discontinuities	Hard limits on a signal.
Scope	Sinks	Time-domain visualisation.
To Workspace	Sinks	Export of a signal to the MATLAB workspace.

A.7 Common pitfalls and good practice

The following comments assemble errors that occur repeatedly in the laboratory reports and which the student should take care to avoid. None of these is a conceptual difficulty, they are all matters of attention to detail, but every one of them has caused a working controller to behave inexplicably, and eat an afternoon of debugging.
- Always check the units. In seconds, in rad/s in calculations, and in Hz on the instrumentation displays. A factor of 2π is easy to lose and very hard to regain.

- Create s = tf('s') and describe transfer functions symbolically — G = 10/(s*(s+1)*(s+10)) — rather than as coefficient vectors. The list is more explicit in intent and quicker for the instructor to review.

- Inspect poles with pole(G) or pzmap(G) prior to bodeplot or step. Step will not reveal a pole in the right half-plane . The response just diverges off the screen .

- Operators to combine two LTI objects (* for series, + for parallel, feedback() for closed loop). Manual polynomial multiplication is error prone.

- Use minreal() after a series or feedback connection when an obvious pole/zero cancellation is expected. It removes hidden modes that would otherwise mask the structure of the resulting transfer function.

- Plot the step response with a time vector you deliberately choose: step(G, 0:0.01:20) instead of letting Matlab pick. The sampling of the plot must be fine enough to resolve the fastest dynamics; a coarse default vector will hide overshoot.

- Label each axis. A figure without units on the x-axis, without a quantity name on the y-axis and without a legend is not a result – it is a decoration. The grading rubric for the lab reports penalizes unlabelled figures very heavily.

- Save script file You should produce a single .m file for each laboratory session. When run from a clean workspace, the .m file should produce all figures and numbers in the report. This practice underpins reproducible engineering work.

Appendix B — Useful tables and formulas

B.1 Laplace transforms

Time-domain f(t)	Laplace transform F(s)
$\delta(t)$	1
$1(t)$	$1 / s$
$t \cdot 1(t)$	$1 / s^2$
$t^n \cdot 1(t)$	$n! / s^{n+1}$
$e^{(-at)} \cdot 1(t)$	$1 / (s + a)$
$\sin(\omega t) \cdot 1(t)$	$\omega / (s^2 + \omega^2)$
$\cos(\omega t) \cdot 1(t)$	$s / (s^2 + \omega^2)$
$e^{(-at)} \sin(\omega t) \cdot 1(t)$	$\omega / [(s + a)^2 + \omega^2]$
$e^{(-at)} \cos(\omega t) \cdot 1(t)$	$(s + a) / [(s + a)^2 + \omega^2]$

B.2 Useful properties

Property	Formula
Linearity	$L\{a f + b g\} = a F(s) + b G(s)$
First derivative	$L\{f'(t)\} = s F(s) - f(0^-)$
Integral	$L\{\int_0^t f(\tau)\, d\tau\} = F(s) / s$
Initial-value theorem	$f(0^+) = \lim_{s\to\infty} s F(s)$
Final-value theorem	$f(\infty) = \lim_{s\to 0} s F(s)$ (if poles of $s F(s)$ lie in left half-plane)
Convolution	$L\{(f * g)(t)\} = F(s) G(s)$

Time shift	L{f(t − a) 1(t − a)} = e^(−a s) F(s)

B.3 Performance indicators of a second-order system

Indicator	Formula	Comments
Damped frequency ωd	$\omega_n \sqrt{(1 - \zeta^2)}$	Oscillation frequency of an under-damped system.
Resonance frequency ωr	$\omega_n \sqrt{(1 - 2\,\zeta^2)}$	Exists only for $\zeta < 1/\sqrt{2}$.
Resonance peak Mr	$1 / (2\,\zeta \sqrt{(1 - \zeta^2)})$	Magnitude of the closed-loop resonance peak.
Percentage overshoot Mp	$100 \cdot \exp(-\pi\,\zeta / \sqrt{(1 - \zeta^2)})$	Closed-form expression for the first peak.
Peak time tp	$\pi / \omega d$	Time of the first overshoot.
Settling time t_s (2 %)	$4 / (\zeta\,\omega_n)$	Time to remain within ± 2 % of the final value.
Rise time t_r (0 – 100 %)	$(\pi - \arccos \zeta) / \omega d$	Time from 0 to the first crossing of the final value.

B.4 Notation and glossary of symbols

The following list summarises the principal symbols and abbreviations used throughout this manual. The student is encouraged to keep this table at hand during the laboratory sessions and the writing of reports.

Symbol	Meaning
s	Laplace complex variable, $s = \sigma + j\omega$.
t	Time, in seconds unless otherwise stated.
ω	Angular frequency, in radians per second.

f	Cyclic frequency, in Hz; $\omega = 2\pi f$.
G(s)	Plant transfer function (forward path).
H(s)	Feedback-path transfer function; H = 1 for unity feedback.
C(s)	Controller (compensator) transfer function.
L(s)	Loop transfer function, L = C G H.
T(s)	Closed-loop transfer function from reference to output.
S(s)	Sensitivity function, $S = 1/(1 + L)$; S + T = 1.
u, y, r, e, d	Control input, output, reference, error (r − y), disturbance.
ω_n	Undamped natural frequency of a second-order system.
ζ	Damping ratio of a second-order system (dimensionless).
τ	Time constant of a first-order pole; pole at $s = -1/\tau$.
K	Static (DC) gain of a transfer function: K = G(0).
K_p, K_v, K_a	Position, velocity and acceleration error constants.
ω_c	Gain crossover frequency, where $\|L(j\omega)\| = 1$.
ω_p	Phase crossover frequency, where $\arg L(j\omega) = -180°$.
PM, GM	Phase margin and gain margin of the open loop.
Mp	Percentage overshoot of the step response.
t_s, t_p, t_r	Settling, peak and rise times.
T_s	Sampling period for digital implementation.
z	Variable of the z-transform, $z = e^{sT_s}$.

LTI	Linear time-invariant.
SISO / MIMO	Single-input single-output / multiple-input multiple-output.
PID	Proportional-integral-derivative controller.
CST	Control System Toolbox (Matlab).
DAQ	Data acquisition card or system.

B.5 Closing remarks

The present manual contains seven laboratory sessions on the basic material of a classical course in continuous time linear control. The student who has completed all the exercises, reproduced the figures and answered the questions on synthesis will have attained the practical fluency required of a graduate in any engineering curriculum in which automatic control is a recognized discipline. The manual suggests several natural extensions: digital control and discrete-time design, state-space methods and pole-placement, optimal control (LQR/LQG), system identification, robust control, and the design of model predictive controllers. Entry points for each of these directions are provided in the next section by means of the references.

The Control System Toolbox contains functions for each of these advanced topics, and the Matlab documentation should be considered a permanent companion to the textbooks below. It is highly recommended for students working on a project or a final year thesis to read a primary reference and perform hands-on experiments in Matlab/Simulink classical control rewards the practitioner who develops intuition by simulating representative examples repeatedly.

References

[1] R. C. Dorf and R. H. Bishop, Modern Control Systems, 13th ed., Pearson, 2017.

[2] K. Ogata, Modern Control Engineering, 5th ed., Prentice Hall, 2010.

[3] G. F. Franklin, J. D. Powell and A. Emami-Naeini, Feedback Control of Dynamic Systems, 8th ed., Pearson, 2019.

[4] B. C. Kuo and F. Golnaraghi, Automatic Control Systems, 10th ed., Wiley, 2017.

[5] N. S. Nise, Control Systems Engineering, 7th ed., Wiley, 2015.

[6] K. J. Åström and R. M. Murray, Feedback Systems: An Introduction for Scientists and Engineers, 2nd ed., Princeton University Press, 2021.

[7] J. C. Doyle, B. A. Francis and A. R. Tannenbaum, Feedback Control Theory, Macmillan, 1992.

[8] MathWorks, Control System Toolbox User's Guide, R2023b, 2023.

[9] MathWorks, Simulink User's Guide, R2023b, 2023.

[10] P. de Larminat, Automatique appliquée, 2e éd., Hermès–Lavoisier, 2009.

[11] Y. Granjon, Automatique : systèmes linéaires, non linéaires, temps continu, temps discret, Dunod, 2015.

[12] Ph. de Larminat and Y. Thomas, Automatique : commande des systèmes linéaires, Hermès, 2004.