Numerical Mathematics

Lecture Notes

Thomas Bartz-Beielstein

2026-11-09

Table of contents

Preface 1
Lecture Slides . . . 1
Use of AI Tools . . . 1

1. Overview 3

I. Ordinary Differential Equations 5

2. Ordinary Differential Equations 7
2.1. First-Order Differential Equations . . . 11
2.1.1. Separation of Variables . . . 11
2.1.2. Integration of a Differential Equation by Substitution . . . 12
2.1.3. Exact Differential Equations . . . 14
2.2. Linear First-Order Differential Equations . . . 15
2.2.1. Variation of Constants . . . 16
2.2.2. Finding a Particular Solution . . . 17
2.3. Linear First-Order Differential Equations with Constant Coefficients . . . 18
2.3.1. Applications . . . 19
2.4. Linear Second-Order Differential Equations with Constant Coefficients . . . 20
2.5. Integration of the Homogeneous Linear Second-Order Differential Equation . . . 22
2.6. Integration of the Inhomogeneous Linear Second-Order Differential Equation with Constant Coefficients 23
2.7. Spring Pendulum . . . 26
2.7.1. Equation of Motion . . . 26
2.7.2. Free Undamped Oscillation . . . 28
2.7.3. Free Damped Oscillation . . . 32
2.7.4. Strong Damping (Aperiodic Behavior, Overdamped Case) . . . 35
2.7.5. Aperiodic Limiting Case . . . 37
2.7.6. Forced Oscillation . . . 40
2.7.7. Frequency Response . . . 44
2.8. Simple Pendulum . . . 47
2.8.1. Equation of Motion . . . 48
2.8.2. The General Solution of the Equation of Motion . . . 49
2.9. Reduction of Order . . . 49

3. Numerical Integration of a Differential Equation 53
3.1. Explicit Euler Method (Euler's Polygon Method) . . . 54
3.2. Implicit Euler Method . . . 58
3.3. Trapezoidal Method . . . 58
3.4. Simple Pendulum: Euler Methods in Matrix Form . . . 59
3.5. Numerical Error and Instability . . . 60
3.6. Runge-Kutta Methods . . . 63
3.6.1. The Midpoint Method . . . 63
3.6.2. Derivation of the Runge-Kutta Methods . . . 65

4. Differential Equations in Python 69
4.1. Creating a General ODE Solver . . . 70
4.2. A Simple and General Solver: The Forward Euler Method . . . 71
4.3. Extending the Solver to General ODEs . . . 73
4.4. The ODE Solver as a Python Class . . . 74
4.5. ODE Systems . . . 76
4.6. A `ForwardEuler` Class for Systems of ODEs . . . 77
4.7. Using ODE Solvers from SciPy . . . 81

4.8. Exercises . . . 83
4.9. Solutions . . . 85

II. Systems of Linear Differential Equations 93

5. Eigenvalues and Eigenvectors 95
5.1. Definitions . . . 96
5.2. Numerical Eigenvalue Computations . . . 100
5.2.1. The Power Method (Largest Eigenvalue) . . . 100
5.2.2. The Inverse Power Method . . . 102
5.2.3. The Shifted Power Method . . . 102
5.2.4. The QR Decomposition . . . 102
5.2.5. Eigenvalues and Eigenvectors in `numpy.linalg` (Python) . . . 104

6. Systems of Linear Differential Equations 105
6.1. Systems of First-Order Linear Differential Equations with Constant Coefficients . . . 106
6.1.1. Two First-Order Linear Differential Equations with Constant Coefficients . . . 106
6.1.2. Solution of the Homogeneous System for $n = 2$ . . . 107
6.1.3. Homogeneous Systems of First-Order Linear Differential Equations with Constant Coefficients of Arbitrary Order . . . 110

7. Systems of Second-Order Linear Differential Equations in Python 115
7.1. Introduction . . . 115
7.2. Coupled Springs . . . 117

8. Four Special Matrices K_n, C_n, T_n, and B_n 121
8.1. The Matrices K_n and C_n . . . 124
8.2. From K_n to T_n . . . 127
8.3. From T_n to B_n . . . 128
8.4. Matrices in Python . . . 129
8.4.1. Elementary Matrix Functions `eye` and `ones` . . . 129
8.4.2. The Function `toeplitz` . . . 129
8.4.3. The Function `KTBC(n)` . . . 130
8.5. Eigenvalues and Eigenvectors of the Matrices K_n, T_n, B_n, and C_n . . . 131
8.5.1. Part I: $Av = \lambda v$, $A^k v = \lambda^k v$, and the Diagonalization of A . . . 131
8.5.2. Application to Systems of ODEs . . . 133
8.5.3. Symmetric Matrices and Orthonormal Eigenvectors . . . 135
8.5.4. Eigenvectors for Derivatives and Differences . . . 135
8.5.5. Eigenvectors of the Matrices K_n: Discrete Sine Function . . . 136
8.5.6. Eigenvectors of the Matrices C_n: Powers of $\zeta = \mathrm{e}^{2\pi \mathrm{j}/n}$ . . . 137
8.6. Equilibrium and the Stiffness Matrix . . . 140
8.6.1. A Spring Chain . . . 140
8.6.2. Stiffness Matrix and Solution . . . 142
8.6.3. Fixed and Free End . . . 144
8.7. Oscillations by Newton's Law . . . 146
8.7.1. One Mass and One Spring . . . 146

III. Applications 149

9. Turbines and Turbochargers 151
9.1. Definitions . . . 151
9.2. High-Cycle and Low-Cycle Fatigue . . . 151
9.3. Natural Vibration Behavior of Turbomachines . . . 152
9.3.1. Blade and Disk Mode Shapes . . . 152

10. Models for Vibration Analysis 155
10.1. Mathematical Derivation . . . 156
10.2. Interpretation of the Mathematical Considerations . . . 158
10.3. Phase Angle of the Blades . . . 159

IV. Project **161**

11. Project: Vibration Analysis of Turbochargers **163**

References **165**

Appendices **167**

A. Table of Integrals **167**

Preface

These lecture notes cover material from the lecture “Numerical Mathematics”. They are intended for personal use only and must not be passed on.

Gummersbach, 2024-2026

Lecture Slides

Lecture slides that open in the browser are available for some chapters:

- Chapter 2: Ordinary Differential Equations

Use of AI Tools

Transparency notice: parts of these lecture notes were researched, drafted, and translated with the support of artificial intelligence. This information is provided voluntarily for complete transparency, as no legal requirement mandates its disclosure.

1. Overview

Table 1.1.: Overview

No.	Date	Time	Lecture	Room	Material
1	Tue 27.10.26	09:00	Numerical Methods (Kick-off)	0503	
2	Tue 10.11.26	09:00	2. Ordinary Differential Equations; 3. Numerical Integration of a Differential Equation	0503	
3	Tue 17.11.26	09:00	4. Differential Equations in Python; 5. Eigenvalues and Eigenvectors	0503	
4	Tue 24.11.26	09:00	6. Systems of Linear Differential Equations; 7. Systems of Second-Order Linear Differential Equations in Python	0503	
5	Tue 01.12.26	09:00	8. Four Special Matrices K_n, C_n, T_n, and B_n	0503	
6	Tue 08.12.26	09:00	9. Turbines and Turbochargers; 10. Models for Vibration Analysis	0503	
7	Tue 15.12.26	09:00	Project	0503	
8	Tue 12.01.27	09:00	Test Examination, Discussion of Project Results	0503	

Part I.

Ordinary Differential Equations

2. Ordinary Differential Equations

Papula (2015) gives an introduction to the topic in Part IV, Chapter 1, "Gewöhnliche Differentialgleichungen" (ordinary differential equations).

Definition 2.1 (Ordinary Differential Equation)**.** A *differential equation* is a relation between an unknown function $y(x)$, its independent variable x, and an arbitrary number of its derivatives. An *ordinary differential equation* (ODE) is a differential equation in which the independent variable, and hence the derivatives, are one-dimensional. We assume that an ODE can be written in the explicit form

$$\frac{\mathrm{d}^n y}{\mathrm{d}x^n} = f\left(x, y, \frac{\mathrm{d}y}{\mathrm{d}x}, \frac{\mathrm{d}^2 y}{\mathrm{d}x^2}, \dots, \frac{\mathrm{d}^{n-1} y}{\mathrm{d}x^{n-1}}\right), \quad \text{for short} \quad y^{(n)} = f\left(x, y, y', \dots, y^{(n-1)}\right),$$

where f is an arbitrary function of some or all of its arguments. The highest order of derivative n that occurs is called the *order* of the ODE. □

Remark 2.1 (Memory Aid for Integration by Parts). Not entirely correct mathematically, but easy to remember:

$$\int u'v = uv - \int uv' \qquad \square$$

Table 2.1 summarizes the symbols of this chapter. Up to Section 2.7, x denotes the independent variable and a, b denote the coefficients of a general linear ODE. From Section 2.7 onward, t is the independent variable, $x(t)$ the displacement, and b the damper constant. Likewise, A first stands for a scalar amplitude and only in Section 2.9 for a matrix.

Table 2.1.: Symbols in Chapter 02

Symbol	Meaning	Type	Defined in
General Theory			
a	Constant coefficient of a linear ODE ($y' + ay = g(x)$; growth rate in $y' = ay$)	Constant	Equation 2.3
a, b	Constant coefficients of the general linear second-order ODE $y'' + ay' + by = g(x)$	Constant	Definition 2.7
d	Differential (dx, $\frac{\mathrm{d}y}{\mathrm{d}x}$)	Operator	Definition 2.1
e	Euler's number	Constant	Example 2.1
f	Function; right-hand side of the ODE $y' = f(x, y)$ or $y^{(n)} = f(x, y, \dots, y^{(n-1)})$	Function	Definition 2.1
$f(x)$	Coefficient function of the linear ODE $y' + f(x)y = g(x)$	Function	Definition 2.6
$g(x)$	Forcing function of a linear ODE; as a vector $g(x)$ or $g(t)$ for systems	Function	Definition 2.6
j	Imaginary unit, $\mathrm{j}^2 = -1$	Constant	Proposition 2.2
n	Order of an ODE; dimension of a matrix or system	Index	Definition 2.1
x	Independent variable	Scalar	Definition 2.1

Symbol	Meaning	Type	Defined in
x_0, y_0	Initial point; initial value $y(x_0)$	Scalar	Definition 2.3
y	Unknown function $y(x)$ or vector of unknown functions	Function	Definition 2.1
$y_\mathrm{h}, y_\mathrm{p}$	General solution of the homogeneous ODE and particular solution, respectively	Function	Theorem 2.2
y_1, y_2	Basis solutions of a homogeneous linear second-order ODE	Function	Definition 2.8
C; C_1, C_2	Constant of integration; $C_1, \dots, C_n$ constants of integration or coefficients of the general solution	Constant	Example 2.1
C_0	Combined constant of integration of the general solution	Constant	Example 2.2
$K(x)$	Factor function that replaces C in the variation of constants	Function	Section 2.2.1
W	Wronskian $W(y_1; y_2)$	Function	Definition 2.8
$\mathbb{R}$, $\mathbb{C}$	Set of real and of complex numbers, respectively	Set	Example 2.2
α	Real part of a complex eigenvalue, $\lambda_{1,2} = \alpha \pm \mathrm{j}\omega$	Scalar	Proposition 2.2
β	Angular frequency of the forcing function $g(x) = \sin(\beta x)$	Constant	Equation 2.6
λ; $\lambda_{1,2}$	Eigenvalue or root of the characteristic equation (exponent in the ansatz $\mathrm{e}^{\lambda t}$); $\lambda_{1,2}$ the two roots	Scalar	Section 2.3
φ	Phase angle (initial phase angle) of an oscillation; φ_d of the damped oscillation; $\varphi(\omega)$ phase response	Scalar	Equation 2.6
ω	Angular frequency (excitation angular frequency; imaginary part of an eigenvalue)	Scalar	Proposition 2.2
$y', y'', y^{(n)}$	First, second, n-th derivative with respect to the independent variable	Operator	Definition 2.1
$\frac{\partial}{\partial x}$	Partial derivative	Operator	Definition 2.5
$\int \dots \mathrm{d}x$	Integral	Operator	Example 2.2
$\lvert z \rvert$	Absolute value	Operator	Example 2.2
local	$g(y)$ in Definition 2.4; a, b, c and u in Theorem 2.1; u in Proposition 2.1; $g(x; y)$, $h(x; y)$, $u(x; y)$ in Definition 2.5; $K(y)$ in Example 2.7; ansatz coefficients a_0, a_1, a_2, A, B; boundary points $x_1, \dots, x_n$ in Definition 2.3	local	where used

Symbol	Meaning	Type	Defined in
b	Damper constant of the mass-spring oscillator	Constant	Definition 2.9
c	Spring constant	Constant	Theorem 2.7
g	Gravitational acceleration	Constant	Section 2.8
i	Running index (component)	Index	Section 2.9
l	Length of the simple pendulum	Constant	Definition 2.13
m	Mass	Constant	Theorem 2.6
$r(t), w(t)$	Population of rabbits and of wolves, respectively	Function	Example 2.47
$s(t)$	Arc length of the simple pendulum	Function	Section 2.8
t	Time	Scalar	Theorem 2.6
t_1, t_2	Time of the turning point in the overdamped case and in the aperiodic limiting case, respectively	Scalar	Figure 2.10, Figure 2.13
v, v_0	Velocity, $v = \dot{x}$; v_0 initial velocity	Scalar	Definition 2.9
$x(t)$, x_0	Displacement from the rest position; x_0 initial displacement	Function	Theorem 2.7
$x_\mathrm{h}, x_\mathrm{p}$	Homogeneous and particular component of the oscillation solution, respectively	Function	Equation 2.24
x_st	Static displacement, $x_\mathrm{st} = F_0/c$	Scalar	Definition 2.12
$\underline{x}_\mathrm{p}$	Complex particular solution (underline = complex quantity)	Function	Equation 2.18
$y_1, \dots, y_n$	Components of the state vector S in the reduction of order ($y_1 = y$, $y_2 = y'$, …)	Function	Theorem 2.14
z	Complex number	Scalar	Figure 2.14
A, $A(\omega)$	Amplitude of an oscillation; $A(\omega)$ amplitude frequency response, $A_\mathrm{r} = A_\mathrm{max}$ resonance amplitude	Scalar	Section 2.7.2
A	Matrix (coefficient or system matrix); $a_1, \dots, a_n$ entries of its last row	Matrix	Section 2.9
A_d	Amplitude of the damped oscillation $x(t) = A_\mathrm{d}\,\mathrm{e}^{-\delta t}\sin(\omega_\mathrm{d} t + \varphi_\mathrm{d})$	Scalar	Theorem 2.10
F	Force; $F(t)$ external force, F_0 its amplitude, F_1, F_2, F_3 component forces, F_R restoring force, F_G weight	Scalar	Theorem 2.6
$\underline{F}(t)$	Complex excitation force	Function	Equation 2.19
K_0	Excitation amplitude per unit mass, $K_0 = F_0/m$	Constant	Equation 2.17
S	State vector $S(t)$; S_i its i-th component	Vector	Section 2.9
T	Period of oscillation; T_0 of the simple pendulum, T_d of the damped oscillation	Scalar	Section 2.7.2

Symbol	Meaning	Type	Defined in
δ	Damping factor or decay constant, $\delta = b/(2m)$; $\delta_1, \delta_2, \delta_3$ sample values	Constant	Equation 2.10
Δ	Discriminant of the characteristic equation, $\Delta = \delta^2 - \omega_0^2$	Scalar	Equation 2.11
ν_0	Frequency of the simple pendulum, $\nu_0 = \omega_0/(2\pi)$	Scalar	Section 2.8
π	The number pi (italic, KaTeX exception)	Constant	Section 2.7.2
$\varphi(t)$, φ_0	Angle of deflection of the simple pendulum; φ_0 initial deflection	Function	Definition 2.13
$\Psi(\omega)$	Radicand of the resonance curve, $\Psi(\omega) = (\omega_0^2 - \omega^2)^2 + (2\delta\omega)^2$	Function	Section 2.7.7.1
ω_0	Natural angular frequency of the undamped single-mass oscillator, $\omega_0 = \sqrt{c/m}$ (simple pendulum: $\sqrt{g/l}$)	Constant	Equation 2.8
ω_d	Natural angular frequency of the damped oscillation, $\omega_\mathrm{d} = \sqrt{\omega_0^2 - \delta^2}$	Constant	Theorem 2.10
ω_r	Resonance angular frequency, $\omega_\mathrm{r} = \sqrt{\omega_0^2 - 2\delta^2}$	Constant	Theorem 2.13
$\dot{x}, \ddot{x}$	First and second derivative with respect to time t	Operator	Theorem 2.6
Im	Imaginary part	Operator	Section 2.7.6
^{T}	Transpose	Operator	Example 2.49
local	Bodies 1, 2 in Theorem 2.6; turning points A, B in Figure 2.22	local	where used

The solutions of an ODE form a family from which additional conditions select a single solution.

Definition 2.2 (General and Particular Solution)**.** The *general solution* of an ODE of order n comprises all solutions of the equation. In general, it contains n undetermined constants of integration and describes a family of functions. A *particular solution* is a member of this family in which the constants have fixed values. □

The constants cannot be determined from the ODE itself, but only from additional conditions such as initial or boundary conditions. A particular solution then satisfies both the ODE and these requirements.

Example 2.1 (General and Particular Solution)**.** The first-order ODE

$$\frac{\mathrm{d}y}{\mathrm{d}x} = 3y$$

has the general solution $y = C\mathrm{e}^{3x}$ with an arbitrary constant C. It describes a family of exponential functions whose curves depend on the value of C. If $y(0) = 4$ is additionally required, substituting into the general solution yields $4 = C\mathrm{e}^0 = C$. The particular solution is therefore $y = 4\mathrm{e}^{3x}$. □

Definition 2.3 (Initial Value Problem and Boundary Value Problem)**.** In an *initial value problem*, the function value and the values of the first $n-1$ derivatives of the solution $y = y(x)$ of an ODE of order n are prescribed at a point x_0:

$$y(x_0) = y_0, \quad y'(x_0) = y_0', \quad y''(x_0) = y_0'', \quad \dots, \quad y^{(n-1)}(x_0) = y_0^{(n-1)}.$$

In a *boundary value problem*, the function values $y(x_1), y(x_2), \dots, y(x_n)$ of the solution are prescribed at n different points $x_1, x_2, \dots, x_n$. □

2.1. First-Order Differential Equations

2.1.1. Separation of Variables

Definition 2.4 (Separable ODE)**.** A first-order ODE of the form

$$y' = \frac{\mathrm{d}y}{\mathrm{d}x} = f(x)\,g(y)$$

is called *separable* (ODE with separable variables). □

It is solved by separating the variables x and y onto the two sides of the equation, integrating both sides, and solving the result for y. The antiderivatives required for this are collected in Table A.1.

Example 2.2 (Separation of Variables)**.** The ODE $y' = y$ can be solved for $y \neq 0$ by separation of variables:

$$\frac{\mathrm{d}y}{\mathrm{d}x} = y \Rightarrow \frac{\mathrm{d}y}{y} = \mathrm{d}x \Rightarrow \int \frac{\mathrm{d}y}{y} = \int \mathrm{d}x \Rightarrow \ln|y| = x + C.$$

The constants of integration arising on both sides have been combined into a single constant. The solution is now in implicit form. Solving for y yields

$$|y| = \mathrm{e}^{x+C} = \mathrm{e}^{C}\mathrm{e}^{x} \Rightarrow y = \pm\mathrm{e}^{C}\mathrm{e}^{x}.$$

When C runs through all real numbers, e^{C} runs through all positive and $\pm\mathrm{e}^{C}$ through all nonzero real numbers. We set

$$y = \pm\mathrm{e}^{C}\mathrm{e}^{x} = C_0\mathrm{e}^{x}, \quad C_0 = \pm\mathrm{e}^{C} \neq 0.$$

Since $y = 0$ also solves the ODE $y' = y$, the general solution is

$$y = C_0\mathrm{e}^{x}, \quad C_0 \in \mathbb{R}.$$ □

Remark 2.2 (Logarithmic Terms). If logarithmic terms such as $\ln|y|$ in the example above arise when integrating an ODE, it is convenient to write the constant of integration C in the form $\ln|C|$. This is possible because, as C runs through all real numbers, so does $\ln|C|$, see Figure 2.1. □

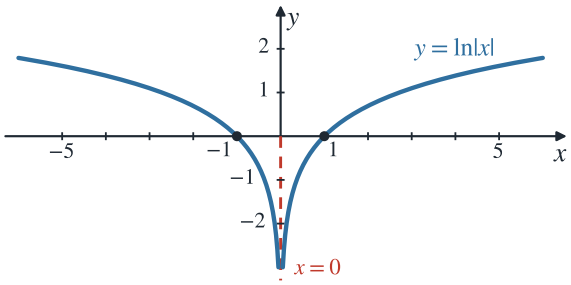


Figure 2.1.: Graph of the function $y = \ln|x|$ for $x \neq 0$. The function is even, has the zeros $x = \pm 1$, and tends to $-\infty$ as $x \to 0$; the y-axis $x = 0$ (dashed) is a vertical asymptote. The range is all of $\mathbb{R}$.

With this note, we solve the ODE $y' = y$ once more. After separating the variables and integrating, we write the constant of integration as $\ln|C|$ with $C \neq 0$:

$$\frac{\mathrm{d}y}{\mathrm{d}x} = y \Rightarrow \frac{\mathrm{d}y}{y} = \mathrm{d}x \Rightarrow \int \frac{\mathrm{d}y}{y} = \int \mathrm{d}x \Rightarrow \ln|y| = x + \ln|C|.$$

The logarithmic terms are combined,

$$\ln|y| - \ln|C| = \ln\left|\frac{y}{C}\right| = x,$$

and taking the exponential yields

$$\left|\frac{y}{C}\right| = \mathrm{e}^{x} \Rightarrow \frac{y}{C} = \pm\mathrm{e}^{x} \Rightarrow y = \pm C\mathrm{e}^{x}.$$

With $C_0 = \pm C$ and including the solution $y = 0$, the general solution of the ODE $y' = y$ is therefore again

$$y = C_0\mathrm{e}^{x}, \quad C_0 \in \mathbb{R}.$$

Example 2.3 (Separation of Variables with an Initial Value)**.** We solve the initial value problem

$$x + xy' = 0, \qquad y(0) = 2.$$

For $x \neq 0$, we divide by x and obtain $1 + y' = 0$, that is,

$$\frac{\mathrm{d}y}{\mathrm{d}x} = -1.$$

The right-hand side depends neither on x nor on y, so the variables are already separated. Integrating both sides of $\mathrm{d}y = -\mathrm{d}x$ yields

$$\int \mathrm{d}y = -\int \mathrm{d}x \Rightarrow y = -x + C, \quad C \in \mathbb{R}.$$

The initial condition $y(0) = C = 2$ gives the solution of the initial value problem

$$y = 2 - x.$$

Check: with $y' = -1$, we have $x + x \cdot (-1) = 0$ for all x, hence also at $x = 0$, where we were not allowed to divide by x. □

Example 2.4 (Separation of Variables with a Singular Solution)**.** We determine the general solution of the ODE

$$y' + y^2 = 0.$$

For $y \neq 0$, we separate the variables and integrate:

$$\frac{\mathrm{d}y}{\mathrm{d}x} = -y^2 \Rightarrow -\frac{\mathrm{d}y}{y^2} = \mathrm{d}x \Rightarrow -\int \frac{\mathrm{d}y}{y^2} = \int \mathrm{d}x \Rightarrow \frac{1}{y} = x + C.$$

Solving for y yields the general solution

$$y = \frac{1}{x + C}, \quad C \in \mathbb{R}.$$

Check: $y' = -\frac{1}{(x+C)^2} = -y^2$. Each of these solutions is defined only on an interval that does not contain the pole $x = -C$.

In separating the variables, we divided by y^2 and thereby excluded the case $y = 0$. The zero function $y = 0$ also solves the ODE, but it is not contained in the family of solutions for any value of C. Such a solution is called *singular*. It must be stated separately. □

2.1.2. Integration of a Differential Equation by Substitution

Theorem 2.1 (Integration of an ODE by Substitution)**.** *A first-order ODE of the type*

$$y' = f(ax + by + c)$$

is transformed by the substitution

$$u = ax + by + c \tag{2.1}$$

into the separable ODE

$$u' = a + bf(u)$$

for the auxiliary function $u = u(x)$. □

Proof. Differentiating Equation 2.1 with respect to x yields $u' = a + by'$. With $y' = f(ax + by + c) = f(u)$ it follows that

$$u' = \frac{\mathrm{d}u}{\mathrm{d}x} = a + bf(u).$$

The right-hand side depends only on u, so this ODE can be solved by separation of the variables x and u. □

The solution $u = u(x)$ is then substituted into Equation 2.1, and the equation is solved for y.

Proposition 2.1 (Substitution $u = y/x$)**.** *A first-order ODE of the type*

$$y' = f\left(\frac{y}{x}\right)$$

is transformed by the substitution $u = y/x$, i.e., $y = xu$, into the separable ODE

$$u + xu' = f(u)$$

for the auxiliary function $u = u(x)$. □

Proof. Differentiating $y = xu$ with respect to x yields $y' = u'x + u$. Since $y' = f(y/x) = f(u)$, the claim follows. □

Both substitutions follow the same procedure: carry out the substitution, integrate the ODE for the auxiliary function $u = u(x)$ by separation of variables, undo the substitution, and solve the equation for y.

Example 2.5 (Substitution $u = ax + by + c$)**.** We solve the ODE

$$y' = 2x - y.$$

The right-hand side has the form $f(ax + by + c)$ with $a = 2$, $b = -1$, $c = 0$, and $f(u) = u$. By Theorem 2.1, the substitution

$$u = 2x - y, \qquad u' = 2 - y',$$

leads to the separable ODE

$$u' = 2 - u.$$

For $u \neq 2$, we separate the variables and write the constant of integration as $-\ln|C|$:

$$\frac{\mathrm{d}u}{2-u} = \mathrm{d}x \Rightarrow -\ln|2-u| = x - \ln|C| \Rightarrow \ln\left|\frac{2-u}{C}\right| = -x.$$

Taking exponentials yields $2 - u = C\mathrm{e}^{-x}$, where the sign is absorbed into the constant C. The value $C = 0$ gives the initially excluded solution $u = 2$. Hence

$$u = 2 - C\mathrm{e}^{-x}, \quad C \in \mathbb{R}.$$

The back-substitution $y = 2x - u$ yields the general solution

$$y = 2x - 2 + C\mathrm{e}^{-x}, \quad C \in \mathbb{R}.$$

Check: $y' = 2 - C\mathrm{e}^{-x}$ and $2x - y = 2x - (2x - 2 + C\mathrm{e}^{-x}) = 2 - C\mathrm{e}^{-x}$. □

Example 2.6 (Substitution $u = y/x$)**.** We solve the ODE

$$y' = \frac{x + 2y}{x} \qquad (x \neq 0).$$

Splitting the right-hand side yields

$$y' = 1 + 2\,\frac{y}{x},$$

that is, the type $y' = f(y/x)$ with $f(u) = 1 + 2u$. By Proposition 2.1, the substitution $u = y/x$, i.e., $y = xu$, leads to

$$u + xu' = 1 + 2u \Rightarrow xu' = 1 + u.$$

For $u \neq -1$, we separate the variables and write the constant of integration as $\ln|C|$:

$$\frac{\mathrm{d}u}{1+u} = \frac{\mathrm{d}x}{x} \Rightarrow \ln|1+u| = \ln|x| + \ln|C| = \ln|Cx|.$$

Taking exponentials yields $1 + u = Cx$, where the sign is again absorbed into C. The value $C = 0$ gives the initially excluded solution $u = -1$. Hence $u = Cx - 1$, and the back-substitution $y = xu$ yields the general solution

$$y = Cx^2 - x, \quad C \in \mathbb{R}.$$

Check: $y' = 2Cx - 1$ and

$$\frac{x + 2(Cx^2 - x)}{x} = \frac{2Cx^2 - x}{x} = 2Cx - 1.$$ □

2.1.3. Exact Differential Equations

Definition 2.5 (Exact ODE)**.** A first-order ODE of the form

$$g(x;y)\,\mathrm{d}x + h(x;y)\,\mathrm{d}y = 0, \tag{2.2}$$

in another notation

$$y' = -\frac{g(x;y)}{h(x;y)},$$

is called *exact* if the integrability condition

$$\frac{\partial g}{\partial y} = \frac{\partial h}{\partial x}$$

is satisfied. □

The continuous factor functions $g(x;y)$ and $h(x;y)$ of an exact ODE Equation 2.2 are the partial derivatives of an unknown function $u(x;y)$:

$$\frac{\partial u}{\partial x} = g(x;y) \quad \text{and} \quad \frac{\partial u}{\partial y} = h(x;y).$$

From these relations, $u(x;y)$ is determined by indefinite integration. The general solution of the exact ODE Equation 2.2 is then given in the implicit form

$$u(x;y) = C$$

where C is a constant of integration.

Example 2.7 (Exact Differential Equation)**.** We solve the ODE

$$y' = \frac{\mathrm{d}y}{\mathrm{d}x} = \frac{2x+y}{y-x}.$$

Multiplying by $(y-x)\,\mathrm{d}x$ yields $(y-x)\,\mathrm{d}y = (2x+y)\,\mathrm{d}x$. Bringing all terms to one side, we obtain the form Equation 2.2,

$$(2x+y)\,\mathrm{d}x + (x-y)\,\mathrm{d}y = 0,$$

with the factor functions

$$g(x;y) = 2x+y \quad \text{and} \quad h(x;y) = x-y.$$

The integrability condition is satisfied,

$$\frac{\partial g}{\partial y} = 1 = \frac{\partial h}{\partial x},$$

so the ODE is exact. We determine $u(x;y)$ in two steps. Integrating $\partial u/\partial x = g(x;y)$ with respect to x yields

$$u(x;y) = \int (2x+y)\,\mathrm{d}x = x^2 + xy + K(y),$$

where the constant of integration $K(y)$ may still depend on y. Differentiating with respect to y and comparing with $h(x;y)$ gives

$$\frac{\partial u}{\partial y} = x + K'(y) = x - y \Rightarrow K'(y) = -y \Rightarrow K(y) = -\frac{y^2}{2}.$$

We may omit an additive constant in $K(y)$ because it is absorbed into the constant C of the solution. Thus

$$u(x;y) = x^2 + xy - \frac{y^2}{2},$$

and the general solution in implicit form reads

$$x^2 + xy - \frac{y^2}{2} = C, \quad C \in \mathbb{R}.$$

Solving this quadratic equation for y yields the explicit form

$$y = x \pm \sqrt{3x^2 - 2C}.$$

Check: implicit differentiation of $u(x;y) = C$ with respect to x yields $2x+y+xy'-yy' = 0$, hence $y'(y-x) = 2x+y$, which is the given ODE. □

2.2. Linear First-Order Differential Equations

Definition 2.6 (Linear First-Order ODE)**.** A first-order ODE is called *linear* if it can be written in the form

$$y' + f(x)\, y = g(x)$$

The function $f(x)$ is called the *coefficient function*, the function $g(x)$ the *forcing function*. If the forcing function is absent, the ODE is called *homogeneous*, otherwise *inhomogeneous*. □

The homogeneous linear first-order ODE

$$y' + f(x)\, y = 0$$

is separable. Separation of variables yields the general solution

$$y = C\mathrm{e}^{-\int f(x)\,\mathrm{d}x}, \quad C \in \mathbb{R}.$$

The following two examples follow Papula (2015, sec. IV.2.5.2).

Example 2.8 (Homogeneous Linear First-Order ODE)**.** We seek the general solution of

$$x^2 y' + y = 0 \quad \text{or} \quad y' + \frac{1}{x^2}\, y = 0, \qquad x \neq 0.$$

For $y \neq 0$, separation of variables yields

$$\frac{\mathrm{d}y}{\mathrm{d}x} = -\frac{y}{x^2} \Rightarrow \frac{\mathrm{d}y}{y} = -\frac{\mathrm{d}x}{x^2}.$$

Integrating both sides gives

$$\int \frac{\mathrm{d}y}{y} = -\int x^{-2}\,\mathrm{d}x \Rightarrow \ln|y| = \frac{1}{x} + \ln|K|, \quad K \neq 0,$$

i.e., $\ln|y/K| = 1/x$ and hence $y/K = \pm\mathrm{e}^{1/x}$. With $C = \pm K$, the general solution is

$$y = C\mathrm{e}^{1/x}, \quad C \in \mathbb{R}.$$

The trivial solution $y = 0$ is included as the special case $C = 0$. □

Example 2.9 (Homogeneous Linear First-Order ODE with an Initial Value)**.** We seek the solution of the initial value problem

$$y' - 2xy = 0, \qquad y(0) = 5.$$

Separation of variables yields

$$\frac{\mathrm{d}y}{\mathrm{d}x} = 2xy \Rightarrow \frac{\mathrm{d}y}{y} = 2x\,\mathrm{d}x \Rightarrow \int \frac{\mathrm{d}y}{y} = \int 2x\,\mathrm{d}x \Rightarrow \ln|y| = x^2 + \ln|C|,$$

i.e., $\ln|y/C| = x^2$ and hence the general solution

$$y = C\mathrm{e}^{x^2}, \quad C \in \mathbb{R}.$$

The initial value gives $y(0) = C\mathrm{e}^0 = C = 5$. The initial value problem therefore has the solution

$$y = 5\mathrm{e}^{x^2}.$$ □

2.2.1. Variation of Constants

An inhomogeneous linear first-order ODE

$$y' + f(x)\,y = g(x)$$

is solved by *variation of constants* in two steps. First, the associated homogeneous ODE $y' + f(x)\,y = 0$ is integrated by separation of variables:

$$y_\mathrm{h} = C\mathrm{e}^{-\int f(x)\,\mathrm{d}x}, \quad C \in \mathbb{R}.$$

Then the constant of integration C is replaced by a still unknown function $K(x)$. Substituting the ansatz

$$y = K(x)\,\mathrm{e}^{-\int f(x)\,\mathrm{d}x}$$

into the inhomogeneous ODE yields a simple first-order ODE for the factor function $K(x)$, which is solved by indefinite integration (antiderivatives in Table A.1).

The following two examples follow Papula (2015, sec. IV.2.5.3.1).

Example 2.10 (Variation of Constants)**.** We seek the general solution of

$$y' + \frac{y}{x} = \cos(x), \qquad x \neq 0.$$

Separation of variables yields the general solution of the homogeneous ODE,

$$y_\mathrm{h} = \frac{C}{x}, \quad C \in \mathbb{R}.$$

Variation of constants leads to the ansatz

$$y = K(x)\,\frac{1}{x}$$

with the derivative

$$y' = K'(x)\,\frac{1}{x} - K(x)\,\frac{1}{x^2} = \frac{K'(x)}{x} - \frac{K(x)}{x^2}.$$

Substituting into the inhomogeneous ODE yields

$$K'(x) = x\cos(x) \Rightarrow K(x) = \int x\cos(x)\,\mathrm{d}x.$$

Integration by parts yields

$$K(x) = \cos(x) + x\sin(x) + C_0, \quad C_0 \in \mathbb{R}.$$

The inhomogeneous ODE therefore has the general solution

$$y(x) = \frac{\cos(x) + x\sin(x) + C_0}{x}, \quad C_0 \in \mathbb{R}. \qquad \square$$

Example 2.11 (Variation of Constants with an Initial Value)**.** We seek the general solution and the particular solution for the initial value $y(1) = 2$ of the ODE

$$y' - 3y = x\mathrm{e}^{4x}.$$

Separation of variables shows that the associated homogeneous ODE $y' - 3y = 0$ has the general solution

$$y_\mathrm{h} = K\mathrm{e}^{3x}, \quad K \in \mathbb{R}.$$

Variation of constants leads to the ansatz

$$y = K(x)\,\mathrm{e}^{3x} \quad \text{with} \quad y' = K'(x)\,\mathrm{e}^{3x} + 3K(x)\,\mathrm{e}^{3x}.$$

Substituting into the inhomogeneous ODE yields

$$K'(x)\,\mathrm{e}^{3x} + 3K(x)\,\mathrm{e}^{3x} - 3K(x)\,\mathrm{e}^{3x} = K'(x)\,\mathrm{e}^{3x} = x\mathrm{e}^{4x} \Rightarrow K'(x) = x\mathrm{e}^{x}.$$

Integration by parts yields

$$K(x) = \int x\mathrm{e}^{x}\,\mathrm{d}x = (x-1)\,\mathrm{e}^{x} + C, \quad C \in \mathbb{R}.$$

The general solution of the inhomogeneous ODE is therefore

$$y = [(x-1)\,\mathrm{e}^{x} + C]\,\mathrm{e}^{3x} = (x-1)\,\mathrm{e}^{4x} + C\mathrm{e}^{3x}, \quad C \in \mathbb{R}.$$

The initial value gives

$$y(1) = 0 \cdot \mathrm{e}^{4} + C\mathrm{e}^{3} = 2 \Rightarrow C = 2\mathrm{e}^{-3}.$$

The particular solution is therefore

$$y = (x-1)\,\mathrm{e}^{4x} + 2\mathrm{e}^{3(x-1)}. \qquad \square$$

2.2.2. Finding a Particular Solution

Theorem 2.2 (Solution Set of an Inhomogeneous Linear ODE)**.** *The general solution $y = y(x)$ of an inhomogeneous linear first-order ODE*

$$y' + f(x)\,y = g(x)$$

is the sum of the general solution $y_\mathrm{h} = y_\mathrm{h}(x)$ of the associated homogeneous ODE

$$y' + f(x)\,y = 0$$

and an arbitrary particular solution $y_\mathrm{p} = y_\mathrm{p}(x)$ of the inhomogeneous ODE:

$$y(x) = y_\mathrm{h}(x) + y_\mathrm{p}(x).$$

This also holds for linear ODEs of higher order. □

Proof. If y is an arbitrary solution of the inhomogeneous ODE, then, by linearity, the difference $y - y_\mathrm{p}$ solves the homogeneous ODE and is therefore of the form y_h. Conversely, $y_\mathrm{h} + y_\mathrm{p}$ solves the inhomogeneous ODE. □

It therefore suffices to find a particular solution y_p and to add it to the general solution y_h of the homogeneous ODE, which is obtained by separation of variables. The ansatz for y_p depends on the type of the coefficient function $f(x)$ and on the form of the forcing function $g(x)$.

Theorem 2.3 (Procedure for Finding a Particular Solution)**.** *In many cases, an inhomogeneous linear first-order ODE $y' + f(x)\,y = g(x)$ can be solved as follows:*

1. *Integrate the associated homogeneous ODE $y' + f(x)\,y = 0$ by separation of variables, with the general solution*

$$y_\mathrm{h} = C\mathrm{e}^{-\int f(x)\,\mathrm{d}x}, \quad C \in \mathbb{R}.$$

2. *Determine a particular solution y_p of the inhomogeneous ODE with a suitable ansatz that still contains one or more unknown parameters.*
3. *The general solution of the inhomogeneous ODE is then the sum*

$$y = y_\mathrm{h} + y_\mathrm{p}.$$

□

Example 2.12 (Finding a Particular Solution)**.** We seek the general solution of

$$y' - \tan(x)\,y = 2\sin(x).$$

Integrating the homogeneous ODE $y' - \tan(x)\,y = 0$ by separation of variables yields

$$y_\mathrm{h} = \frac{C}{\cos(x)}, \quad C \in \mathbb{R}.$$

For the particular solution we choose the ansatz

$$y_\mathrm{p} = A\cos(x)$$

with the parameter A. Substituting y_p and $y_\mathrm{p}' = -A\sin(x)$ into the inhomogeneous ODE yields the determining equation

$$y_\mathrm{p}' - \frac{\sin(x)}{\cos(x)}\,y_\mathrm{p} = -A\sin(x) - A\sin(x) = 2\sin(x) \Rightarrow A = -1.$$

With the particular solution $y_\mathrm{p} = -\cos(x)$, the general solution of the inhomogeneous ODE is

$$y = \frac{C}{\cos(x)} - \cos(x), \quad C \in \mathbb{R}.$$

□

2.3. Linear First-Order Differential Equations with Constant Coefficients

Linear first-order ODEs with constant coefficients have the form

$$y' + ay = g(x), \qquad a \in \mathbb{R}. \tag{2.3}$$

They are the special case of Definition 2.6 with the constant coefficient function $f(x) = a$.

The associated homogeneous ODE

$$y' + ay = 0$$

is solved by separation of variables or by the exponential ansatz

$$y_\mathrm{h} = C\mathrm{e}^{\lambda x}, \quad C \in \mathbb{R}.$$

Substituting into the homogeneous ODE yields the *characteristic equation* for the parameter λ:

$$y_\mathrm{h}' + ay_\mathrm{h} = \lambda C\mathrm{e}^{\lambda x} + aC\mathrm{e}^{\lambda x} = 0 \Rightarrow \lambda + a = 0 \Rightarrow \lambda = -a.$$

The homogeneous ODE $y' + ay = 0$ therefore has the general solution

$$y_\mathrm{h} = C\mathrm{e}^{-ax}, \quad C \in \mathbb{R}.$$

The examples in this section follow Papula (2015, sec. IV.2.6).

Example 2.13 (Homogeneous Linear ODE with Constant Coefficients)**.** The general solution $y_\mathrm{h} = C\mathrm{e}^{-ax}$ with $C \in \mathbb{R}$ can be stated directly:

1. $y' + 4y = 0$ has, since $a = 4$, the general solution $y_\mathrm{h} = C\mathrm{e}^{-4x}$.
2. $y' - 0.5\,y = 0$ has, since $a = -0.5$, the general solution $y_\mathrm{h} = C\mathrm{e}^{0.5\,x}$.
3. $-3y' + 18y = 0$ becomes $y' - 6y = 0$ after division by -3 and therefore has the general solution $y_\mathrm{h} = C\mathrm{e}^{6x}$. □

The inhomogeneous ODE Equation 2.3 is integrated either by variation of constants (Section 2.2.1) or by finding a particular solution according to Theorem 2.3. Since the ansatz for y_p here depends essentially on the function type of the forcing function $g(x)$, the second method is usually more convenient.

Example 2.14 (Linear ODE with Constant Coefficients)**.** We seek the general solution of

$$y' + 2y = 2x^2 - 4.$$

The associated homogeneous ODE $y' + 2y = 0$ has the general solution

$$y_\mathrm{h} = C\mathrm{e}^{-2x}, \quad C \in \mathbb{R}.$$

For the particular solution of the inhomogeneous ODE we choose the ansatz

$$y_\mathrm{p} = a_2x^2 + a_1x + a_0.$$

With $y_\mathrm{p}' = 2a_2x + a_1$, substitution into the inhomogeneous ODE gives

$$y_\mathrm{p}' + 2y_\mathrm{p} = 2a_2x + a_1 + 2(a_2x^2 + a_1x + a_0) = 2x^2 - 4.$$

Sorting by descending powers of x yields

$$2a_2x^2 + (2a_2 + 2a_1)x + (a_1 + 2a_0) = 2x^2 + 0x - 4.$$

Comparing coefficients gives the equations

$$\begin{aligned} 2a_2 = 2 &\Rightarrow a_2 = 1, \\ 2a_2 + 2a_1 = 0 &\Rightarrow 2 + 2a_1 = 0 \Rightarrow a_1 = -1, \\ a_1 + 2a_0 = -4 &\Rightarrow -1 + 2a_0 = -4 \Rightarrow a_0 = -3/2. \end{aligned}$$

The particular solution of the inhomogeneous ODE is therefore

$$y_\mathrm{p} = x^2 - x - \frac{3}{2}.$$

The general solution is

$$y = C\mathrm{e}^{-2x} + x^2 - x - \frac{3}{2}, \quad C \in \mathbb{R}.$$ □

Example 2.15 (Linear ODE with Constant Coefficients and an Initial Value)**.** We seek the solution of the initial value problem

$$y' + 5y = -26\sin(x), \qquad y(0) = 0.$$

The homogeneous ODE $y' + 5y = 0$ has the general solution

$$y_\mathrm{h} = C\mathrm{e}^{-5x}, \quad C \in \mathbb{R}.$$

Since the forcing function is a sine function with the angular frequency 1, we choose the ansatz for the particular solution

$$y_\mathrm{p} = A\sin(x) + B\cos(x) \quad \text{with} \quad y_\mathrm{p}' = A\cos(x) - B\sin(x).$$

Substituting into the inhomogeneous ODE and sorting by sine and cosine terms yields

$$(5A - B)\sin(x) + (A + 5B)\cos(x) = -26\sin(x) + 0\cos(x).$$

Comparing coefficients gives the linear system of equations

$$\begin{aligned} 5A - B &= -26, \\ A + 5B &= 0. \end{aligned}$$

The second equation gives $A = -5B$. Substituting into the first equation yields $-26B = -26$, so $B = 1$ and $A = -5$. The general solution of the inhomogeneous ODE is therefore

$$y = C\mathrm{e}^{-5x} - 5\sin(x) + \cos(x), \quad C \in \mathbb{R}.$$

The initial value gives $y(0) = C + 1 = 0$, so $C = -1$. The initial value problem therefore has the solution

$$y = -\mathrm{e}^{-5x} - 5\sin(x) + \cos(x). \qquad \square$$

2.3.1. Applications

Linear first-order ODEs with constant coefficients describe many processes in physics and engineering. The following two application examples follow Papula (2015, sec. IV.2.7).

Example 2.16 (Radioactive Decay)**.** Let $n = n(t)$ be the number of atomic nuclei of a radioactive substance still present at time t. The number $\mathrm{d}n$ of nuclei decaying in a short time interval $\mathrm{d}t$ is proportional both to $\mathrm{d}t$ and to n. With the radioactive decay constant $\lambda > 0$, we therefore have

$$\mathrm{d}n = -\lambda n\,\mathrm{d}t \quad \text{or} \quad \frac{\mathrm{d}n}{\mathrm{d}t} + \lambda n = 0.$$

The minus sign expresses that the number of nuclei decreases. The decay is thus described by a homogeneous linear first-order ODE with constant coefficients. Its general solution is

$$n(t) = C\mathrm{e}^{-\lambda t}, \quad C \in \mathbb{R}.$$

If $n(0) = n_0$ is the initial number of nuclei, then $C = n_0$, which yields the law of radioactive decay

$$n(t) = n_0\mathrm{e}^{-\lambda t}, \quad t \geq 0. \qquad \square$$

Example 2.17 (AC Circuit)**.** An ohmic resistor R and an inductor L are connected in series to a voltage source with the sinusoidal alternating voltage $u(t) = \hat{u}\sin(\omega t)$. By Kirchhoff's voltage law, the sum of the partial voltages $u_L(t) = L\,\mathrm{d}i/\mathrm{d}t$ (law of induction) and $u_R(t) = Ri$ (Ohm's law) equals the applied voltage. The current $i = i(t)$ therefore satisfies the inhomogeneous linear first-order ODE

$$L\frac{\mathrm{d}i}{\mathrm{d}t} + Ri = \hat{u}\sin(\omega t).$$

The homogeneous ODE $L\,\mathrm{d}i/\mathrm{d}t + Ri = 0$ has the general solution

$$i_\mathrm{h}(t) = C\mathrm{e}^{-\frac{R}{L}t}, \quad C \in \mathbb{R}.$$

For the particular solution we choose the ansatz

$$i_\mathrm{p}(t) = \hat{\imath}\sin(\omega t + \varphi)$$

with the peak value $\hat{\imath}$ and the zero phase angle φ. Substituting into the ODE yields

$$\omega L\hat{\imath}\cos(\omega t+\varphi)+R\hat{\imath}\sin(\omega t+\varphi)=\hat{u}\sin(\omega t).$$

After applying the addition theorems and sorting by $\sin(\omega t)$ and $\cos(\omega t)$, comparing coefficients leads to the nonlinear system of equations

$$\begin{aligned} R\hat{\imath}\cos\varphi-\omega L\hat{\imath}\sin\varphi&=\hat{u},\\ \omega L\hat{\imath}\cos\varphi+R\hat{\imath}\sin\varphi&=0. \end{aligned}$$

Squaring and adding both equations eliminates φ, since $\sin^2\varphi+\cos^2\varphi=1$:

$$\left(R^2+(\omega L)^2\right)\hat{\imath}^2=\hat{u}^2\Rightarrow\hat{\imath}=\frac{\hat{u}}{\sqrt{R^2+(\omega L)^2}}.$$

Dividing the second equation by $R\hat{\imath}\cos\varphi$ gives

$$\tan\varphi=-\frac{\omega L}{R}\Rightarrow\varphi=-\arctan\left(\frac{\omega L}{R}\right).$$

The general solution is therefore

$$i(t)=C\mathrm{e}^{-\frac{R}{L}t}+\frac{\hat{u}}{\sqrt{R^2+(\omega L)^2}}\sin\left(\omega t-\arctan\left(\frac{\omega L}{R}\right)\right),\quad t\geq 0.$$

The exponentially decaying direct current $i_{\mathrm{h}}(t)$ has practically vanished after a short transient phase ("transient" component). What remains is the *steady-state solution* $i_{\mathrm{p}}(t)$, an alternating current with the peak value $\hat{\imath}=\hat{u}/Z$, where $Z=\sqrt{R^2+(\omega L)^2}$ is the impedance of the circuit. The current lags behind the voltage by the phase angle $\arctan(\omega L/R)$. □

2.4. Linear Second-Order Differential Equations with Constant Coefficients

Definition 2.7 (Linear Second-Order ODE with Constant Coefficients)**.** A *linear second-order ODE with constant coefficients* is an ODE of the form

$$y''+ay'+by=g(x),\qquad a,b\in\mathbb{R}.$$

It is called *homogeneous* if the forcing function $g(x)$ is absent, otherwise *inhomogeneous*. □

The examples for Definition 2.7 follow Papula (2015, sec. IV.3.1), those on the solutions of the homogeneous ODE Papula (2015, sec. IV.3.2).

Example 2.18 (Linear and Nonlinear Second-Order ODEs 1)**.** The ODEs

$$y''+y=0,\qquad y''+2y'-3y=2x-4\qquad\text{and}\qquad 2y''-4y'+20y=\cos(x)$$

are linear with constant coefficients. The first is homogeneous, the other two are inhomogeneous. After division by 2, the third takes the form of Definition 2.7. □

Example 2.19 (Linear and Nonlinear Second-Order ODEs 2)**.** The ODEs

$$y''+xy'+y=0\qquad\text{and}\qquad x^3y''+x^2y'-xy=\mathrm{e}^x$$

are linear but have nonconstant coefficients. □

Example 2.20 (Linear and Nonlinear Second-Order ODEs 3)**.** The ODEs

$$y''+y'+y^2=0\qquad\text{and}\qquad y'y''+y=x$$

are nonlinear: in the first, y appears to the second power, and the second contains the product $y'y''$. □

Example 2.21 (Solutions of the Oscillation Equation)**.** The *oscillation equation*

$$y'' + \omega^2 y = 0, \qquad \omega > 0,$$

has, among others, the solutions $y_1 = \sin(\omega x)$ and $y_2 = \cos(\omega x)$. Differentiating twice and substituting confirms this:

$$y_1'' + \omega^2 y_1 = -\omega^2 \sin(\omega x) + \omega^2 \sin(\omega x) = 0,$$
$$y_2'' + \omega^2 y_2 = -\omega^2 \cos(\omega x) + \omega^2 \cos(\omega x) = 0.$$

Since the ODE is linear and homogeneous, $C_1 \sin(\omega x)$, $C_2 \cos(\omega x)$, and

$$y = C_1 \sin(\omega x) + C_2 \cos(\omega x), \quad C_1, C_2 \in \mathbb{R},$$

are solutions as well. However, not every linear combination of two solutions contains two independent parameters. The functions $y_1 = \sin(\omega x)$ and $y_2 = 2\sin(\omega x)$ also solve the oscillation equation, but their linear combination

$$C_1 \sin(\omega x) + 2C_2 \sin(\omega x) = (C_1 + 2C_2)\sin(\omega x) = C_3 \sin(\omega x)$$

contains only the single parameter C_3. This observation leads to the concept of basis solutions in Definition 2.8. □

Example 2.22 (Complex Solution of the Oscillation Equation)**.** The oscillation equation $y'' + \omega^2 y = 0$ is also solved by the complex-valued exponential function $y = \mathrm{e}^{\mathrm{j}\omega x}$. With $y' = \mathrm{j}\omega\, \mathrm{e}^{\mathrm{j}\omega x}$ and $y'' = \mathrm{j}^2\omega^2 \mathrm{e}^{\mathrm{j}\omega x} = -\omega^2 \mathrm{e}^{\mathrm{j}\omega x}$, it follows that

$$y'' + \omega^2 y = -\omega^2 \mathrm{e}^{\mathrm{j}\omega x} + \omega^2 \mathrm{e}^{\mathrm{j}\omega x} = 0.$$

Since the coefficients are real, the real and imaginary parts of y are real solutions as well. Euler's formula gives

$$y_1 = \mathrm{Re}\,(\mathrm{e}^{\mathrm{j}\omega x}) = \cos(\omega x) \quad \text{and} \quad y_2 = \mathrm{Im}\,(\mathrm{e}^{\mathrm{j}\omega x}) = \sin(\omega x),$$

i.e., the solutions already known from Example 2.21. □

Definition 2.8 (Wronskian and Basis Solutions)**.** Two solutions $y_1 = y_1(x)$ and $y_2 = y_2(x)$ of a homogeneous linear second-order ODE with constant coefficients of the type

$$y'' + ay' + by = 0$$

are called *basis solutions* if their *Wronskian*

$$W(y_1; y_2) = \begin{vmatrix} y_1 & y_2 \\ y_1' & y_2' \end{vmatrix} = y_1 y_2' - y_1' y_2$$

is nonzero, $W(y_1; y_2) \neq 0$. □

Theorem 2.4 (General Solution of a Homogeneous Linear Second-Order ODE with Constant Coefficients)**.** *According to Papula (2015), the general solution of a homogeneous linear second-order ODE with constant coefficients of the type*

$$y'' + ay' + by = 0$$

is the linear combination of two linearly independent solutions (basis solutions) y_1 and y_2:

$$y(x) = C_1 y_1(x) + C_2 y_2(x), \quad C_1, C_2 \in \mathbb{R}.$$ □

Example 2.23 (Wronskian of the Oscillation Equation)**.** The solutions $y_1 = \sin(\omega x)$ and $y_2 = \cos(\omega x)$ of the oscillation equation $y'' + \omega^2 y = 0$ have the Wronskian

$$W(y_1; y_2) = \begin{vmatrix} \sin(\omega x) & \cos(\omega x) \\ \omega\cos(\omega x) & -\omega\sin(\omega x) \end{vmatrix} = -\omega\left[\sin^2(\omega x) + \cos^2(\omega x)\right] = -\omega \neq 0.$$

They are therefore basis solutions, and by Theorem 2.4 the general solution of the oscillation equation is

$$y(x) = C_1 \sin(\omega x) + C_2 \cos(\omega x), \quad C_1, C_2 \in \mathbb{R}.$$ □

Example 2.24 (Wronskian of Two Exponential Functions)**.** The homogeneous ODE $y'' - 4y' - 5y = 0$ has, among others, the solutions $y_1 = \mathrm{e}^{5x}$ and $y_2 = \mathrm{e}^{-x}$, as substitution confirms. The Wronskian decides whether they are basis solutions:

$$W(y_1; y_2) = \begin{vmatrix} \mathrm{e}^{5x} & \mathrm{e}^{-x} \\ 5\mathrm{e}^{5x} & -\mathrm{e}^{-x} \end{vmatrix} = -\mathrm{e}^{4x} - 5\mathrm{e}^{4x} = -6\mathrm{e}^{4x} \neq 0.$$

The two solutions are thus basis solutions, and the general solution is

$$y(x) = C_1 \mathrm{e}^{5x} + C_2 \mathrm{e}^{-x}, \quad C_1, C_2 \in \mathbb{R}.$$ □

2.5. Integration of the Homogeneous Linear Second-Order Differential Equation

A fundamental basis of the homogeneous ODE

$$y'' + ay' + by = 0$$

can be determined with the ansatz $y = \mathrm{e}^{\lambda x}$. Substitution into the homogeneous ODE yields the characteristic equation

$$\lambda^2 + a\lambda + b = 0.$$

Proposition 2.2 (Basis Solutions According to the Roots of the Characteristic Equation)**.** *The basis solutions y_1 and y_2 of the homogeneous ODE $y'' + ay' + by = 0$ depend on the nature of the solutions λ_1 and λ_2 of the characteristic equation $\lambda^2 + a\lambda + b = 0$:*

1. $\lambda_1 \neq \lambda_2$ *real:*
$$y_1 = \mathrm{e}^{\lambda_1 x} \quad \textit{and} \quad y_2 = \mathrm{e}^{\lambda_2 x}.$$
2. $\lambda_1 = \lambda_2 = \lambda$ *real:*
$$y_1 = \mathrm{e}^{\lambda x} \quad \textit{and} \quad y_2 = x\,\mathrm{e}^{\lambda x}. \tag{2.4}$$
3. $\lambda_{1,2} = \alpha \pm \mathrm{j}\omega$ *complex conjugate with* $\omega \neq 0$*:*
$$y_1 = \mathrm{e}^{\alpha x}\sin(\omega x) \quad \textit{and} \quad y_2 = \mathrm{e}^{\alpha x}\cos(\omega x). \tag{2.5}$$

□

Proof. In case 1, $\mathrm{e}^{\lambda_i x}$ solves the ODE because λ_i satisfies the characteristic equation. In case 3, the same holds for the complex solution $\mathrm{e}^{\lambda_1 x} = \mathrm{e}^{\alpha x}(\cos(\omega x) + \mathrm{j}\sin(\omega x))$. Since the coefficients are real, its real and imaginary parts are then solutions as well. In case 2, $\lambda = -a/2$. Substituting $x\,\mathrm{e}^{\lambda x}$ yields $(\lambda^2 + a\lambda + b)\,x\,\mathrm{e}^{\lambda x} + (2\lambda + a)\,\mathrm{e}^{\lambda x} = 0$. The Wronskians $(\lambda_2 - \lambda_1)\,\mathrm{e}^{(\lambda_1+\lambda_2)x}$, $\mathrm{e}^{2\lambda x}$, and $-\omega\,\mathrm{e}^{2\alpha x}$ of the three cases are nonzero. □

In all three cases, the general solution of the homogeneous ODE is obtained, by Theorem 2.4, as the linear combination $y(x) = C_1 y_1(x) + C_2 y_2(x)$ with $C_1, C_2 \in \mathbb{R}$.

The following examples follow Papula (2015, sec. IV.3.3). The solutions of the characteristic equation $\lambda^2 + a\lambda + b = 0$ are $\lambda_{1,2} = -a/2 \pm \sqrt{a^2/4 - b}$.

Example 2.25 (Two Distinct Real Solutions)**.** The ODE $y'' + 2y' - 8y = 0$ has the characteristic equation

$$\lambda^2 + 2\lambda - 8 = 0 \Rightarrow \lambda_1 = 2, \quad \lambda_2 = -4.$$

By case 1 of Proposition 2.2, $y_1 = \mathrm{e}^{2x}$ and $y_2 = \mathrm{e}^{-4x}$ are basis solutions. The general solution is

$$y = C_1\mathrm{e}^{2x} + C_2\mathrm{e}^{-4x}, \quad C_1, C_2 \in \mathbb{R}.$$ □

Example 2.26 (Double Real Solution)**.** The ODE $y'' - 8y' + 16y = 0$ has the characteristic equation

$$\lambda^2 - 8\lambda + 16 = (\lambda - 4)^2 = 0 \Rightarrow \lambda_1 = \lambda_2 = 4.$$

By case 2 of Proposition 2.2, $y_1 = \mathrm{e}^{4x}$ and $y_2 = x\,\mathrm{e}^{4x}$ are basis solutions. The general solution is

$$y = C_1\mathrm{e}^{4x} + C_2\,x\,\mathrm{e}^{4x} = (C_1 + C_2 x)\,\mathrm{e}^{4x}, \quad C_1, C_2 \in \mathbb{R}.$$ □

Example 2.27 (Complex Conjugate Solutions)**.** The ODE $y'' + 4y' + 13y = 0$ has the characteristic equation $\lambda^2 + 4\lambda + 13 = 0$ with the solutions

$$\lambda_{1,2} = -2 \pm \sqrt{4 - 13} = -2 \pm \sqrt{-9} = -2 \pm 3\mathrm{j},$$

i.e., $\alpha = -2$ and $\omega = 3$. By case 3 of Proposition 2.2, $y_1 = \mathrm{e}^{-2x}\sin(3x)$ and $y_2 = \mathrm{e}^{-2x}\cos(3x)$ are basis solutions. The general solution is

$$y = \mathrm{e}^{-2x}\left[C_1\sin(3x) + C_2\cos(3x)\right], \quad C_1, C_2 \in \mathbb{R}.$$ □

Example 2.28 (Homogeneous Linear Second-Order ODE). $y''+3y'-4y=0$: The characteristic equation $\lambda^2+3\lambda-4=0$ has the solutions $\lambda_1=1$ and $\lambda_2=-4$ (case 1). The general solution is

$$y = C_1 \mathrm{e}^{x} + C_2 \mathrm{e}^{-4x}, \quad C_1, C_2 \in \mathbb{R}. \qquad \square$$

Example 2.29 (Further Homogeneous Linear Second-Order ODE). $y'' + 4y' + 20y = 0$: The characteristic equation $\lambda^2 + 4\lambda + 20 = 0$ has the solutions $\lambda_{1,2} = -2 \pm \sqrt{4-20} = -2 \pm 4\mathrm{j}$ (case 3 with $\alpha = -2$ and $\omega = 4$). The general solution is

$$y = \mathrm{e}^{-2x} \left[C_1 \sin(4x) + C_2 \cos(4x) \right], \quad C_1, C_2 \in \mathbb{R}. \qquad \square$$

Example 2.30 (Further Homogeneous Linear Second-Order ODE). $y'' + 20y' + 100y = 0$: The characteristic equation $\lambda^2 + 20\lambda + 100 = (\lambda+10)^2 = 0$ has the double solution $\lambda_{1,2} = -10$ (case 2). The general solution is

$$y = (C_1 + C_2 x)\, \mathrm{e}^{-10x}, \quad C_1, C_2 \in \mathbb{R}. \qquad \square$$

2.6. Integration of the Inhomogeneous Linear Second-Order Differential Equation with Constant Coefficients

As for linear first-order ODEs (Theorem 2.2), the following holds:

Theorem 2.5 (Solution Set of an Inhomogeneous Linear Second-Order ODE with Constant Coefficients). *The general solution of an inhomogeneous linear second-order ODE with constant coefficients of the type*

$$y'' + ay' + by = g(x)$$

is the sum of the general solution y_h of the associated homogeneous ODE

$$y'' + ay' + by = 0$$

and an arbitrary particular solution y_p of the inhomogeneous ODE:

$$y(x) = y_\mathrm{h}(x) + y_\mathrm{p}(x). \qquad \square$$

Proof. The proof of Theorem 2.2 carries over verbatim, because here, too, the difference of two solutions of the inhomogeneous ODE solves the homogeneous ODE. $\square$

The solution procedure thus corresponds to finding a particular solution for linear first-order ODEs (Theorem 2.3). First, one determines the general solution y_h of the homogeneous ODE $y'' + ay' + by = 0$ (Section 2.5). Then one determines a particular solution y_p of the inhomogeneous ODE with an ansatz that depends essentially on the type of the forcing function $g(x)$. The sum $y = y_\mathrm{h} + y_\mathrm{p}$ is the general solution of the inhomogeneous ODE.

Table 2.2 compiles the ansatz functions for the forcing functions most common in applications (Papula 2015, Table 2 in Sec. IV.3.4). Which ansatz to choose depends on whether a number read off from the forcing function (c, $\mathrm{j}\beta$, or $c + \mathrm{j}\beta$) is a solution of the characteristic equation $\lambda^2 + a\lambda + b = 0$. $P_n(x)$ denotes a given polynomial function of degree n. $Q_n(x)$ and $R_n(x)$ are polynomials of degree n with unknown coefficients.

Table 2.2.: Ansatz functions for a particular solution y_p of the inhomogeneous ODE $y'' + ay' + by = g(x)$ depending on the type of the forcing function $g(x)$

Forcing function $g(x)$	Case	Ansatz $y_\mathrm{p}(x)$	Parameters
1. Polynomial function $P_n(x)$	$b \neq 0$	$Q_n(x)$	Coefficients of Q_n
	$a \neq 0, b = 0$	$x\, Q_n(x)$	Coefficients of Q_n
	$a = b = 0$	$x^2\, Q_n(x)$	Coefficients of Q_n

Forcing function $g(x)$	Case	Ansatz $y_\mathrm{p}(x)$	Parameters
2. Exponential function e^{cx}	c is not a solution of the characteristic equation	$A\,\mathrm{e}^{cx}$	A
	c is an r-fold solution ($r = 1, 2$)	$A\,x^r\,\mathrm{e}^{cx}$	A
3. $P_n(x)\,\mathrm{e}^{cx}$	c is not a solution of the characteristic equation	$Q_n(x)\,\mathrm{e}^{cx}$	Coefficients of Q_n
	c is an r-fold solution ($r = 1, 2$)	$x^r\,Q_n(x)\,\mathrm{e}^{cx}$	Coefficients of Q_n
4. $\sin(\beta x)$, $\cos(\beta x)$, or a linear combination of both	$\mathrm{j}\beta$ is not a solution of the characteristic equation	$A\sin(\beta x) + B\cos(\beta x)$	A, B
	$\mathrm{j}\beta$ is a solution	$x\,[A\sin(\beta x) + B\cos(\beta x)]$	A, B
5. $P_n(x)\,\mathrm{e}^{cx}\sin(\beta x)$ or $P_n(x)\,\mathrm{e}^{cx}\cos(\beta x)$	$c + \mathrm{j}\beta$ is not a solution of the characteristic equation	$\mathrm{e}^{cx}\,[Q_n(x)\sin(\beta x) + R_n(x)\cos(\beta x)]$	Coefficients of Q_n, R_n
	$c + \mathrm{j}\beta$ is a solution	$x\,\mathrm{e}^{cx}\,[Q_n(x)\sin(\beta x) + R_n(x)\cos(\beta x)]$	Coefficients of Q_n, R_n

The following remarks apply to Table 2.2:

1. The polynomials $Q_n(x)$ and $R_n(x)$ in the ansatz have the same degree n as the polynomial $P_n(x)$ of the forcing function.
2. An additional constant factor in the forcing function does not change the ansatz. For $g(x) = 3\sin(2x)$, the same ansatz applies as for $g(x) = \sin(2x)$.
3. The parameters of the ansatz are determined such that y_p satisfies the inhomogeneous ODE. With a correctly chosen ansatz, substitution and comparison of coefficients always lead to a uniquely solvable system of equations for the parameters.
4. If the forcing function consists of several additive forcing terms, the ansatz is the sum of the ansatz functions for the individual terms (Example 2.33).
5. If the forcing function is a product $g(x) = g_1(x) \cdot g_2(x)$, the product $y_\mathrm{p} = y_{\mathrm{p}1} \cdot y_{\mathrm{p}2}$ of the ansatz functions for the two factors $g_1(x)$ and $g_2(x)$ succeeds in many, but not all, cases.
6. For periodic forcing functions $g(x) = \sin(\beta x)$ or $g(x) = \cos(\beta x)$, the complex ansatz

$$\underline{y}_\mathrm{p}(x) = A\,\mathrm{e}^{\mathrm{j}(\beta x + \varphi)} \tag{2.6}$$

with the amplitude A and the phase angle φ as parameters is also frequently used. The underline marks a complex quantity.

Example 2.31 (Linear Second-Order ODE with Constant Coefficients)**.** We seek the general solution of

$$y'' + 10y' - 24y = 12x^2 + 14x + 1.$$

The homogeneous ODE $y'' + 10y' - 24y = 0$ has the characteristic equation

$$\lambda^2 + 10\lambda - 24 = 0 \Rightarrow \lambda_1 = -12, \quad \lambda_2 = 2.$$

By Proposition 2.2, the basis solutions are

$$y_1 = \mathrm{e}^{-12x} \quad \text{and} \quad y_2 = \mathrm{e}^{2x},$$

from which the general solution of the homogeneous ODE follows:

$$y_\mathrm{h} = C_1\mathrm{e}^{-12x} + C_2\mathrm{e}^{2x}, \quad C_1, C_2 \in \mathbb{R}.$$

We obtain a particular solution of the inhomogeneous ODE with the ansatz

$$y_\mathrm{p} = a_2x^2 + a_1x + a_0$$

and its derivatives

$$y_\mathrm{p}' = 2a_2x + a_1 \quad \text{and} \quad y_\mathrm{p}'' = 2a_2.$$

Substituting into the inhomogeneous ODE yields

$$2a_2 + 10(2a_2x + a_1) - 24(a_2x^2 + a_1x + a_0) = 12x^2 + 14x + 1.$$

Comparing coefficients gives the equations

$$\begin{aligned} -24a_0 + 10a_1 + 2a_2 &= 1, \\ -24a_1 + 20a_2 &= 14, \\ -24a_2 &= 12 \end{aligned}$$

with the solution $a_2 = -1/2$, $a_1 = -1$, and $a_0 = -1/2$. Hence

$$y_\mathrm{p} = -\frac{1}{2}x^2 - x - \frac{1}{2}$$

is a particular solution of the inhomogeneous ODE. Its general solution is

$$y = C_1\mathrm{e}^{-12x} + C_2\mathrm{e}^{2x} - \frac{1}{2}x^2 - x - \frac{1}{2}, \quad C_1, C_2 \in \mathbb{R}. \qquad \square$$

The following two examples follow Papula (2015, sec. IV.3.4).

Example 2.32 (Ansatz Functions for Different Forcing Functions)**.** We solve the inhomogeneous ODE

$$y'' + y' - 2y = g(x)$$

for six different forcing functions $g(x)$. The homogeneous ODE $y'' + y' - 2y = 0$ has the characteristic equation $\lambda^2 + \lambda - 2 = 0$ with the solutions $\lambda_1 = 1$ and $\lambda_2 = -2$ and hence the general solution

$$y_\mathrm{h} = C_1\mathrm{e}^{x} + C_2\mathrm{e}^{-2x}, \quad C_1, C_2 \in \mathbb{R}.$$

The ansatz for y_p depends on the type of $g(x)$. If the exponent c of a forcing function e^{cx} (or $c + \mathrm{j}\beta$ for $\mathrm{e}^{cx}\sin(\beta x)$) is itself a solution of the characteristic equation, the ansatz is multiplied by x:

Forcing function $g(x)$	Ansatz y_p	Reason
$10x + 1$	$a_1x + a_0$	polynomial, $b = -2 \neq 0$
$x^2 - 4x + 3$	$a_2x^2 + a_1x + a_0$	polynomial, $b = -2 \neq 0$
$3\mathrm{e}^{4x}$	$A\mathrm{e}^{4x}$	$c = 4$ is not a solution
$6\mathrm{e}^{x}$	$Ax\mathrm{e}^{x}$	$c = 1$ is a simple solution
$x\mathrm{e}^{x}$	$x\mathrm{e}^{x}(a_1x + a_0)$	$c = 1$ is a simple solution
$3\sin(2x)$	$A\sin(2x) + B\cos(2x)$	2j is not a solution

1. $g(x) = 10x + 1$: With $y_\mathrm{p} = a_1x + a_0$, $y_\mathrm{p}' = a_1$, and $y_\mathrm{p}'' = 0$, it follows that

   $$a_1 - 2(a_1x + a_0) = -2a_1x + (a_1 - 2a_0) = 10x + 1.$$

   Comparing coefficients gives $-2a_1 = 10$, so $a_1 = -5$, and $a_1 - 2a_0 = 1$, so $a_0 = -3$. The general solution is

   $$y = C_1\mathrm{e}^{x} + C_2\mathrm{e}^{-2x} - 5x - 3.$$

2. $g(x) = x^2 - 4x + 3$: In the same way, one obtains $y_\mathrm{p} = -\frac{1}{2}x^2 + \frac{3}{2}x - \frac{5}{4}$ and

   $$y = C_1\mathrm{e}^{x} + C_2\mathrm{e}^{-2x} - \frac{1}{2}x^2 + \frac{3}{2}x - \frac{5}{4}.$$

3. $g(x) = 3\mathrm{e}^{4x}$: Substituting $y_\mathrm{p} = A\mathrm{e}^{4x}$ yields $(16 + 4 - 2)A\mathrm{e}^{4x} = 3\mathrm{e}^{4x}$, so $A = 1/6$ and

   $$y = C_1\mathrm{e}^{x} + C_2\mathrm{e}^{-2x} + \frac{1}{6}\mathrm{e}^{4x}.$$

4. $g(x) = 6\mathrm{e}^{x}$: With $y_\mathrm{p} = Ax\mathrm{e}^{x}$, $y_\mathrm{p}' = (A + Ax)\,\mathrm{e}^{x}$, and $y_\mathrm{p}'' = (2A + Ax)\,\mathrm{e}^{x}$, division by e^{x} yields

   $$2A + Ax + A + Ax - 2Ax = 3A = 6 \Rightarrow A = 2.$$

   The general solution is

   $$y = C_1\mathrm{e}^{x} + C_2\mathrm{e}^{-2x} + 2x\mathrm{e}^{x} = (C_1 + 2x)\,\mathrm{e}^{x} + C_2\mathrm{e}^{-2x}.$$

5. $g(x) = x\mathrm{e}^x$: With $y_\mathrm{p} = \mathrm{e}^x(a_1x^2 + a_0x)$ and the derivatives

$$y_\mathrm{p}' = \mathrm{e}^x\left(a_1x^2 + (a_0 + 2a_1)x + a_0\right),$$
$$y_\mathrm{p}'' = \mathrm{e}^x\left(a_1x^2 + (a_0 + 4a_1)x + 2a_0 + 2a_1\right)$$

substituting, dividing by e^x, and sorting yields

$$6a_1x + (3a_0 + 2a_1) = x + 0.$$

Comparing coefficients gives $a_1 = 1/6$ and $a_0 = -1/9$. The general solution is

$$y = C_1\mathrm{e}^x + C_2\mathrm{e}^{-2x} + \left(\frac{1}{6}x^2 - \frac{1}{9}x\right)\mathrm{e}^x.$$

6. $g(x) = 3\sin(2x)$: With $y_\mathrm{p} = A\sin(2x) + B\cos(2x)$, $y_\mathrm{p}' = 2A\cos(2x) - 2B\sin(2x)$, and $y_\mathrm{p}'' = -4A\sin(2x) - 4B\cos(2x)$, sorting yields

$$(-6A - 2B)\sin(2x) + (2A - 6B)\cos(2x) = 3\sin(2x) + 0\cos(2x).$$

The linear system of equations $-6A - 2B = 3$, $2A - 6B = 0$ has the solution $A = -9/20$ and $B = -3/20$. The general solution is

$$y = C_1\mathrm{e}^x + C_2\mathrm{e}^{-2x} - \frac{9}{20}\sin(2x) - \frac{3}{20}\cos(2x).$$

In all cases, $C_1, C_2 \in \mathbb{R}$. □

Example 2.33 (Forcing Function Consisting of Several Terms)**.** We seek the general solution of

$$y'' - 6y' + 9y = 2\mathrm{e}^x + 9x - 15.$$

The homogeneous ODE $y'' - 6y' + 9y = 0$ has the characteristic equation $\lambda^2 - 6\lambda + 9 = (\lambda - 3)^2 = 0$ with the double solution $\lambda_{1,2} = 3$ and hence the general solution

$$y_\mathrm{h} = (C_1 + C_2x)\,\mathrm{e}^{3x}, \quad C_1, C_2 \in \mathbb{R}.$$

The forcing function consists of the terms $g_1(x) = 2\mathrm{e}^x$ and $g_2(x) = 9x - 15$. The corresponding ansatz functions are $y_\mathrm{p1} = A\mathrm{e}^x$ and $y_\mathrm{p2} = a_1x + a_0$. Since the ODE is linear, the sum

$$y_\mathrm{p} = y_\mathrm{p1} + y_\mathrm{p2} = A\mathrm{e}^x + a_1x + a_0$$

is a suitable ansatz for $g = g_1 + g_2$. With $y_\mathrm{p}' = A\mathrm{e}^x + a_1$ and $y_\mathrm{p}'' = A\mathrm{e}^x$, it follows that

$$A\mathrm{e}^x - 6(A\mathrm{e}^x + a_1) + 9(A\mathrm{e}^x + a_1x + a_0) = 4A\mathrm{e}^x + 9a_1x + (9a_0 - 6a_1) = 2\mathrm{e}^x + 9x - 15.$$

Comparing coefficients gives $4A = 2$, $9a_1 = 9$, and $9a_0 - 6a_1 = -15$, so $A = 1/2$, $a_1 = 1$, and $a_0 = -1$. The general solution is

$$y = (C_1 + C_2x)\,\mathrm{e}^{3x} + \frac{1}{2}\mathrm{e}^x + x - 1, \quad C_1, C_2 \in \mathbb{R}.$$ □

2.7. Spring Pendulum

2.7.1. Equation of Motion

Theorem 2.6 (Newton's Three Laws of Motion)**.**

1. *Newton's first law (law of inertia) states that a body remains at rest or moves in a straight line at constant velocity as long as no external force acts on it. It is the special case of the second law in which the sum of the forces acting on the body is zero.*
2. *Newton's second law states that the acceleration of a body is directly proportional to the force exerted on it and inversely proportional to its mass:*

$$F = m\ddot{x},$$

where F is the sum of all forces acting on the body, m its mass, and $\ddot{x}$ its acceleration, i.e., the second derivative of its position $x(t)$ with respect to time t.

3. *Newton's third law states that every force gives rise to an equal and opposite counterforce: if body 1 exerts a force on body 2, then body 2 exerts a force of equal magnitude but opposite direction on body 1.* □

Here and in the following, dots denote derivatives with respect to time, $\dot{x} = \mathrm{d}x/\mathrm{d}t$ and $\ddot{x} = \mathrm{d}^2x/\mathrm{d}t^2$.

Theorem 2.7 (Hooke's Law)**.** *The force exerted on an elastic body is directly proportional to its deformation:*

$$F = -cx,$$

where F is the force exerted on the body, c the spring constant, and x the displacement of the body from its rest position. □

The minus sign indicates that the force acts opposite to the displacement, i.e., always toward the rest position. From Newton's second law and Hooke's law we derive the equation of motion of an elastic spring pendulum.

Definition 2.9 (Mass-Spring Oscillator)**.** A *mass-spring oscillator* (spring pendulum) consists of a mass m that hangs from a spring with the spring constant $c > 0$ and is slowed down by a damper with the *damper constant* $b \geq 0$, see Figure 2.2. Its displacement $x(t)$ is measured from the rest position $x = 0$; $v = \dot{x}$ is its velocity. In addition, an external force $F(t)$ can act on the mass. □

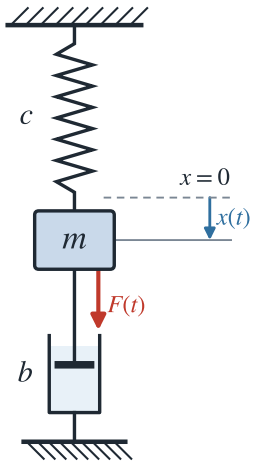


Figure 2.2.: Damped mass-spring oscillator (spring pendulum) with spring constant c, mass m, damper constant b, and external force $F(t)$. The displacement $x(t)$ is measured from the rest position $x = 0$ and is positive downward. Adapted from Papula (2015).

Three forces act on the mass:

1. By Hooke's law, the restoring force of the spring is
$$F_1 = -c\,x(t).$$
2. According to experience, the damping force is proportional to the velocity and always opposes the motion:
$$F_2 = -bv = -b\dot{x}(t).$$
3. In addition, there is an externally applied force, e.g., via an eccentric:
$$F_3 = F(t).$$

Further forces such as gravity or the inertial force are neglected. By Newton's second law,

$$m\ddot{x} = F_1 + F_2 + F_3,$$

from which (taking friction into account) the second-order ODE

$$m\ddot{x} = -cx - b\dot{x} + F(t) \quad \text{or} \quad m\ddot{x} + b\dot{x} + cx = F(t)$$

follows. The motion of a mass-spring oscillator is thus described by a linear second-order ODE with constant coefficients. Depending on whether an external force and damping are present, we distinguish in the following the free undamped, the free damped, and the forced oscillation. If the damping is too strong, the system no longer oscillates but returns aperiodically to the rest position (overdamped case, Section 2.7.4). Definition 2.10 summarizes the oscillation types (Papula 2015, sec. IV.4.1.1).

Definition 2.10 (Differential Equation of a Mechanical Oscillation (Oscillation Equation))**.** The motion of a mass-spring oscillator (Definition 2.9) is described by the linear second-order ODE with constant coefficients

$$m\ddot{x} + b\dot{x} + cx = F(t) \tag{2.7}$$

It is called the *oscillation equation*. Here, m is the mass, b the damper constant, c the spring constant, and $F(t)$ the time-dependent external force acting on the system.

Three special types of oscillation are distinguished:

1. *Free undamped oscillation*: Neither an external force nor damping acts on the system, i.e., $F(t) \equiv 0$ and $b = 0$:
$$m\ddot{x} + cx = 0.$$
2. *Free damped oscillation*: No external force acts on the system, but it is damped, i.e., $F(t) \equiv 0$ and $b \neq 0$:
$$m\ddot{x} + b\dot{x} + cx = 0.$$
3. *Forced oscillation*: A time-dependent periodic force $F(t) = F_0 \sin(\omega t)$ with the *excitation angular frequency* ω acts on the system:
$$m\ddot{x} + b\dot{x} + cx = F_0 \sin(\omega t).$$ □

2.7.2. Free Undamped Oscillation

In what follows, we need the addition theorems for sine and cosine as well as the representation of a linear combination of sine and cosine as a phase-shifted sine oscillation.

Theorem 2.8 (Addition Theorem for the Sine)**.**

$$\sin(x + y) = \sin(x)\cos(y) + \cos(x)\sin(y)$$ □

Lemma 2.1 (Phase-Shifted Sine Oscillation)**.** *For $C_1, C_2 \in \mathbb{R}$, not both zero,*

$$C_1 \sin(x) + C_2 \cos(x) = A \sin(x + \varphi)$$

holds with the amplitude

$$A = \sqrt{C_1^2 + C_2^2}$$

and the phase angle φ, which is uniquely determined up to multiples of 2π by

$$\cos(\varphi) = \frac{C_1}{A} \quad \textit{and} \quad \sin(\varphi) = \frac{C_2}{A}.$$

Depending on the quadrant,

$$\varphi = \begin{cases} \arctan(C_2/C_1) & \textit{for } C_1 > 0, \\ \arctan(C_2/C_1) + \pi & \textit{for } C_1 < 0, \\ \pi/2 & \textit{for } C_1 = 0,\ C_2 > 0, \\ -\pi/2 & \textit{for } C_1 = 0,\ C_2 < 0. \end{cases}$$ □

Proof. Since $(C_1/A)^2 + (C_2/A)^2 = 1$, the point $(C_1/A, C_2/A)$ lies on the unit circle. Hence there is an angle φ with $\cos(\varphi) = C_1/A$ and $\sin(\varphi) = C_2/A$, and it is unique up to multiples of 2π. The addition theorem for the sine yields

$$\begin{aligned} A\sin(x + \varphi) &= A\sin(x)\cos(\varphi) + A\cos(x)\sin(\varphi) \\ &= A\sin(x)\frac{C_1}{A} + A\cos(x)\frac{C_2}{A} \\ &= C_1 \sin(x) + C_2 \cos(x). \end{aligned}$$

For $C_1 \neq 0$, $\tan(\varphi) = C_2/C_1$. The arctangent returns only angles between $-\pi/2$ and $\pi/2$, that is, in the first or fourth quadrant, where $\cos(\varphi) > 0$. For $C_1 > 0$, this is the angle sought. For $C_1 < 0$, $\cos(\varphi) < 0$ and the angle sought is shifted by π. This shift leaves $\tan(\varphi)$ unchanged but reverses the signs of sine and cosine. For $C_1 = 0$, $\cos(\varphi) = 0$ and $\sin(\varphi) = C_2/|C_2| = \pm 1$, so $\varphi = \pm\pi/2$. □

Remark 2.3 (Computing the Phase Angle in Python). In Python, `numpy.arctan2(C2, C1)` computes the phase angle without a case distinction. □

2.7.2.1. General Solution

Theorem 2.9 (General Solution of the Free Undamped Oscillation)**.** *In the simplest case, without damping and without external force, the ODE (equation of motion) of the spring pendulum reads*

$$m\ddot{x} + cx = 0 \quad \textit{or} \quad \ddot{x} + \omega_0^2 x = 0 \tag{2.8}$$

with $\omega_0^2 = c/m$.

The characteristic equation associated with Equation 2.8,

$$\lambda^2 + \omega_0^2 = 0$$

has the complex conjugate solutions

$$\lambda_{1,2} = \pm \mathrm{j}\omega_0.$$

By Equation 2.5, the fundamental basis of the homogeneous ODE Equation 2.8 is then

$$y_1 = \sin(\omega_0 t) \quad \textit{and} \quad y_2 = \cos(\omega_0 t),$$

from which the general solution of the equation of motion follows:

$$x(t) = C_1 \sin(\omega_0 t) + C_2 \cos(\omega_0 t), \tag{2.9}$$

with the angular frequency $\omega_0 = \sqrt{c/m}$ *and* $C_1, C_2 \in \mathbb{R}$. □

Remark 2.4 (Alternative Representation of the Phase-Shifted Sine Oscillation). By Lemma 2.1, the general solution Equation 2.9 can also be represented as a phase-shifted sine oscillation

$$x(t) = A \sin(\omega_0 t + \varphi), \quad t \geq 0,$$

with the amplitude A of the oscillation. □

2.7.2.2. Interpretation of the General Solution

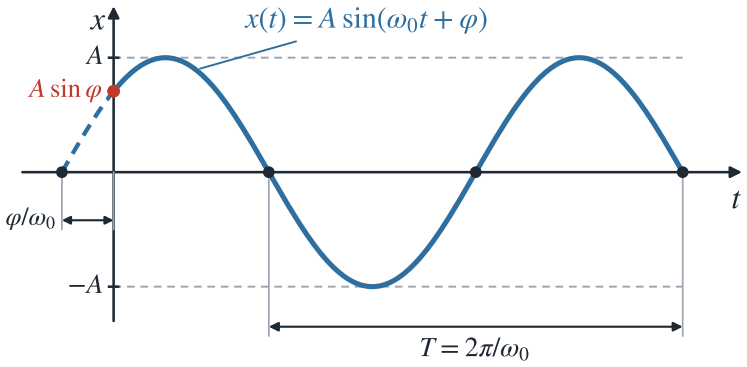


Figure 2.3.: Displacement-time law of a harmonic oscillation $x(t) = A\sin(\omega_0 t + \varphi)$ (general solution) with amplitude A, phase shift φ/ω_0, and period $T = 2\pi/\omega_0$. The initial value is $x(0) = A\sin\varphi$; the dashed part lies outside the domain $t \geq 0$. Adapted from Papula (2015).

Lemma 2.2 (Period of the Oscillation)**.** *The system oscillates harmonically with the angular frequency* $\omega_0 = \sqrt{c/m}$*, which is called the* natural angular frequency *(natural frequency for short). It has the period of oscillation*

$$T = \frac{2\pi}{\omega_0} = 2\pi\sqrt{m/c},$$

see Figure 2.3. □

Proof. $x(t) = A\sin(\omega_0 t + \varphi)$ is periodic with the period $T = 2\pi/\omega_0$ because

$$x(t+T) = A\sin(\omega_0(t+T) + \varphi) = A\sin(\omega_0 t + 2\pi + \varphi) = A\sin(\omega_0 t + \varphi) = x(t).$$

□

The amplitude and the phase angle φ of the oscillation depend on the initial conditions $x(0)$ and $v(0)$.

2.7.2.3. Influence of the Spring Constant and the Mass on the Natural Angular Frequency

The natural angular frequency $\omega_0 = \sqrt{c/m}$ depends only on the ratio of the spring constant c to the mass m, not on the amplitude. The spring constant is not a force; it states how strongly the restoring force $F_1 = -c\,x$ grows with the displacement. Writing the equation of motion Equation 2.8 in the form

$$\ddot{x} = -\frac{c}{m}\,x = -\omega_0^2\,x$$

reveals the reason: for the same displacement x, the acceleration of the mass toward the rest position is the larger, the larger c/m is. The mass then returns to the rest position more quickly, and the oscillation runs faster.

1. *Constant mass, modified spring constant.* A stiffer spring (larger c) exerts a larger restoring force on the same mass at the same displacement. The natural angular frequency grows with the square root of the spring constant, and the period decreases accordingly:
$$\omega_0 \sim \sqrt{c}, \qquad T \sim \frac{1}{\sqrt{c}}.$$
Quadrupling c doubles ω_0 and halves T (Figure 2.4). Accordingly, a softer spring makes the system oscillate more slowly.
2. *Constant spring constant, modified mass.* A larger mass is more inert: the same restoring force gives it a smaller acceleration. The natural angular frequency decreases with the square root of the mass, and the period grows:
$$\omega_0 \sim \frac{1}{\sqrt{m}}, \qquad T \sim \sqrt{m}.$$
Quadrupling m halves ω_0 and doubles T (Figure 2.5). Accordingly, a lighter mass oscillates faster.

In general, the natural angular frequency changes by the factor $\sqrt{k}$ when c is multiplied by the factor k, and by the factor $1/\sqrt{k}$ when m is multiplied by k. If c and m change by the same factor, ω_0 remains unchanged. For example, $m = 1\,\mathrm{kg}$ and $c = 4\,\mathrm{N/m}$ give $\omega_0 = 2\,\mathrm{s}^{-1}$ and $T = \pi\,\mathrm{s} \approx 3.14\,\mathrm{s}$. With $c = 16\,\mathrm{N/m}$, one obtains $\omega_0 = 4\,\mathrm{s}^{-1}$ and $T \approx 1.57\,\mathrm{s}$; with $m = 4\,\mathrm{kg}$ and $c = 4\,\mathrm{N/m}$, by contrast, $\omega_0 = 1\,\mathrm{s}^{-1}$ and $T \approx 6.28\,\mathrm{s}$.

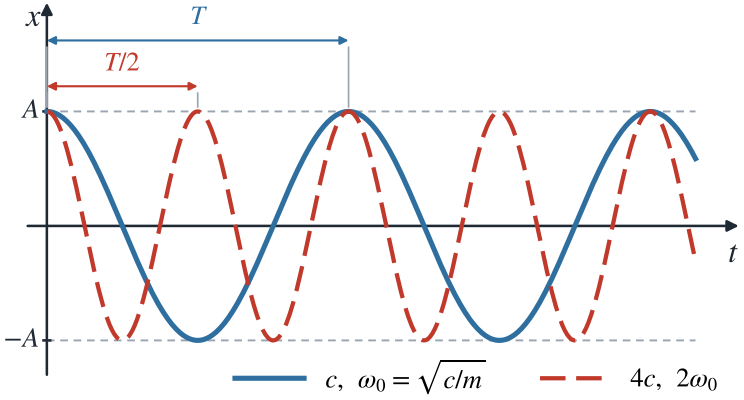


Figure 2.4.: Influence of the spring constant on the natural angular frequency at constant mass m: in each case, the mass is displaced by A and released from rest, $x(t) = A\cos(\omega_0 t)$. Quadrupling the spring constant from c (solid) to $4c$ (dashed) doubles the natural angular frequency and halves the period T. Amplitude and phase angle remain the same.

Both dependencies can be explored in the following animation by changing the spring constant c at fixed m or the mass m at fixed c.

> **Animation: Harmonic Oscillation and Mass-Spring Oscillator**
>
> The animation at https://advm1.gm.fh-koeln.de/~bartz/bart26y/ shows Figure 2.3 and the mass-spring oscillator from Figure 2.2 (without external force) side by side. The point on the curve $x(t)$ and the mass move in sync. The mass m, the spring constant c, and the initial conditions $x(0)$ and $v(0)$ can be adjusted; the amplitude A, the phase angle φ, and the period T adapt accordingly. For an easier visualization, the animation shows the displacement as positive upward, so that the mass and the point on the curve are at the same height; the checkbox “x positive downward” sets the direction of Figure 2.2. The equation of motion does not change.

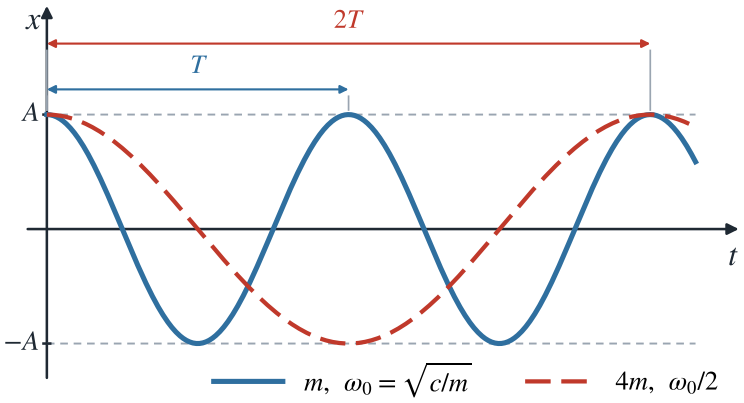


Figure 2.5.: Influence of the mass on the natural angular frequency at constant spring constant c: in each case, the mass is displaced by A and released from rest, $x(t) = A\cos(\omega_0 t)$. Quadrupling the mass from m (solid) to $4m$ (dashed) halves the natural angular frequency and doubles the period T. Amplitude and phase angle remain the same.

Example 2.34 (Undamped Oscillation Starting from Rest)**.** Let the system start from rest with the displacement x_0 and the velocity $v_0 = 0$. With the derivative of Equation 2.9,

$$\dot{x}(t) = C_1\omega_0\cos(\omega_0 t) - C_2\omega_0\sin(\omega_0 t),$$

the conditions $x(0) = x_0$ and $\dot{x}(0) = 0$ yield the values

$$C_1 = 0 \quad \text{and} \quad C_2 = x_0.$$

Substituting into the general solution Equation 2.9 yields the particular solution

$$x(t) = x_0\cos(\omega_0 t),$$

where x_0 is the displacement of the mass at time $t = 0$. The mass-spring oscillator thus oscillates with the natural angular frequency $\omega_0 = \sqrt{c/m}$ and the amplitude $A = x_0$, see Figure 2.6. □

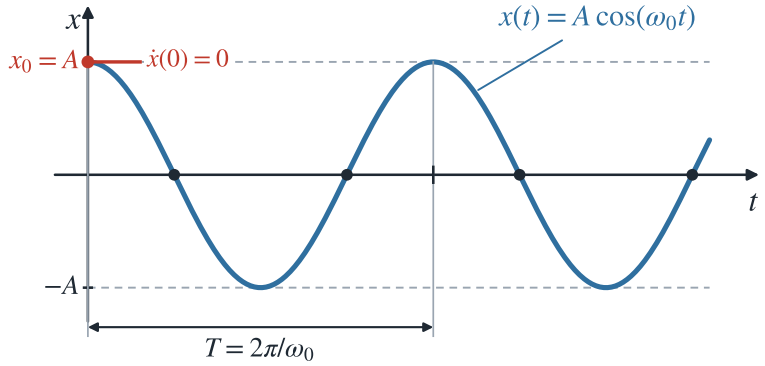


Figure 2.6.: Particular solution $x(t) = A\cos(\omega_0 t)$ of the undamped mass-spring oscillator (spring pendulum) for the initial values $x(0) = x_0 = A$ and $\dot{x}(0) = v(0) = 0$: the oscillation starts at the maximum with a horizontal tangent and has the natural angular frequency ω_0 and the period $T = 2\pi/\omega_0$. Adapted from Papula (2015).

The following numerical example follows Papula (2015, sec. IV.4.1.2).

Example 2.35 (Free Undamped Oscillation)**.** We investigate the motion of an oscillatory mechanical system with the mass $m = 10$ kg and the spring constant $c = 250$ N/m under the initial conditions

$$x(0) = 0.6\,\text{m} \quad \text{and} \quad v(0) = 0\,\text{m/s}.$$

The natural angular frequency of the system is

$$\omega_0 = \sqrt{\frac{c}{m}} = \sqrt{\frac{250\,\text{N/m}}{10\,\text{kg}}} = 5\,\text{s}^{-1}.$$

By Equation 2.9, the equation of motion $\ddot{x} + 25x = 0$ has the general solution

$$x(t) = C_1 \sin(5t) + C_2 \cos(5t).$$

We omit the units in the intermediate calculations. From $x(0) = C_2 = 0.6$ and $\dot{x}(0) = 5C_1 = 0$ it follows that $C_1 = 0$ and $C_2 = 0.6$. The system thus oscillates harmonically according to

$$x(t) = 0.6\,\text{m} \cdot \cos(5\,\text{s}^{-1} \cdot t), \quad t \geq 0.$$

The period is $T = 2\pi/\omega_0 = 0.4\pi\,\text{s} \approx 1.26\,\text{s}$, the amplitude $A = 0.6$ m. This is the special case $x_0 = 0.6$ m of Example 2.34.

What does the value $\omega_0 = 5\,\text{s}^{-1}$ mean? The system neither performs five oscillations per second, nor does one oscillation take five seconds. The natural angular frequency states how fast the phase angle $\omega_0 t$ in the argument of the cosine grows, here by 5 rad per second; the unit s^{-1} is to be read as rad/s. One full oscillation, i.e., the path from $x = A$ via $x = -A$ back to $x = A$, corresponds to the phase angle $2\pi \approx 6.28$ rad. For this, the system needs the period $T = 2\pi/(5\,\text{s}^{-1}) \approx 1.26$ s, i.e., slightly more than one second. The number of full oscillations per second is the frequency

$$\nu_0 = \frac{\omega_0}{2\pi} = \frac{1}{T} = \frac{5}{2\pi}\,\text{s}^{-1} \approx 0.80\,\text{Hz}.$$

In one second, the system thus performs about 0.8 full oscillations; five full oscillations take $5T \approx 6.3$ s. Angular frequency and frequency differ by the factor 2π. □

2.7.3. Free Damped Oscillation

In a free damped oscillation, a damping force acts on the mass in addition to the restoring force. The ODE of the spring pendulum then reads

$$m\ddot{x} + b\dot{x} + cx = 0$$

or

$$\ddot{x} + 2\delta\dot{x} + \omega_0^2 x = 0 \tag{2.10}$$

with the damping factor $\delta = b/(2m)$ and the natural angular frequency $\omega_0 = \sqrt{c/m}$.

The associated characteristic equation

$$\lambda^2 + 2\delta\lambda + \omega_0^2 = 0 \tag{2.11}$$

has the solutions

$$\lambda_{1,2} = -\delta \pm \sqrt{\delta^2 - \omega_0^2},$$

whose nature is decided by the discriminant $\Delta = \delta^2 - \omega_0^2$:

1. $\Delta < 0$, i.e., $\delta < \omega_0$: weakly damped oscillation,
2. $\Delta = 0$, i.e., $\delta = \omega_0$: aperiodic limiting case (critically damped case),
3. $\Delta > 0$, i.e., $\delta > \omega_0$: aperiodic motion (overdamped case).

Theorem 2.10 (General Solution of the Equation of Motion of a Free Damped Spring Pendulum)**.** *The solutions of Equation 2.10 belong to one of three types of oscillation:*

1. Weakly damped oscillation *($\delta < \omega_0$): The characteristic equation has the complex conjugate solutions*

$$\lambda_{1,2} = -\delta \pm \text{j}\sqrt{\omega_0^2 - \delta^2} = -\delta \pm \text{j}\omega_\text{d},$$

where $\omega_\text{d} = \sqrt{\omega_0^2 - \delta^2}$ is the natural angular frequency *of the damped system. The general solution is*

$$x(t) = \text{e}^{-\delta t}\left[C_1 \sin(\omega_\text{d} t) + C_2 \cos(\omega_\text{d} t)\right]$$

with $C_1, C_2 \in \mathbb{R}$, or equivalently

$$x(t) = A_\text{d}\text{e}^{-\delta t} \sin(\omega_\text{d} t + \varphi_\text{d})$$

with the amplitude $A_\text{d} > 0$ and the phase angle $0 \leq \varphi_\text{d} \leq 2\pi$.

2. Aperiodic limiting case *($\delta = \omega_0$): The characteristic equation has the double real solution $\lambda_{1,2} = -\delta$. The general solution is*
$$x(t) = [C_1 t + C_2]\, \mathrm{e}^{-\delta t} \tag{2.12}$$
with $C_1, C_2 \in \mathbb{R}$.

3. Aperiodic motion under strong damping *(overdamped case, $\delta > \omega_0$): The characteristic equation has two distinct real negative solutions λ_1 and λ_2. The general solution is*
$$x(t) = C_1 \mathrm{e}^{\lambda_1 t} + C_2 \mathrm{e}^{\lambda_2 t}$$
with $C_1, C_2 \in \mathbb{R}$. □

Proof. The three cases follow from Proposition 2.2 with $a = 2\delta$, $b = \omega_0^2$, and $\lambda_{1,2} = -\delta \pm \sqrt{\Delta}$. The representation with A_d and φ_d follows from the sine addition theorem with $C_1 = A_\mathrm{d} \cos\varphi_\mathrm{d}$ and $C_2 = A_\mathrm{d} \sin\varphi_\mathrm{d}$. □

2.7.3.1. Weak Damping (Underdamped Case)

Theorem 2.11 (General Solution of the Equation of Motion of a Free Damped Spring Pendulum under Weak Damping)**.** *Under weak damping, $\Delta = \delta^2 - \omega_0^2 < 0$, i.e., $\delta < \omega_0$. With the abbreviation*
$$\omega_\mathrm{d}^2 = \omega_0^2 - \delta^2 > 0$$
the characteristic equation Equation 2.11 has the complex conjugate solutions
$$\lambda_{1,2} = -\delta \pm \sqrt{\delta^2 - \omega_0^2} = -\delta \pm \sqrt{-(\omega_0^2 - \delta^2)} = -\delta \pm \sqrt{-\omega_\mathrm{d}^2} = -\delta \pm \mathrm{j}\omega_\mathrm{d}.$$
By Equation 2.5 with $\alpha = -\delta$ and $\omega = \omega_\mathrm{d}$, the general solution stated in Theorem 2.10 follows:
$$x(t) = A_\mathrm{d} \mathrm{e}^{-\delta t} \sin(\omega_\mathrm{d} t + \varphi_\mathrm{d}). \tag{2.13}$$
□

This is a damped oscillation whose natural angular frequency is reduced compared with the undamped oscillation,
$$\omega_\mathrm{d} = \sqrt{\omega_0^2 - \delta^2} < \omega_0,$$
see Figure 2.7.

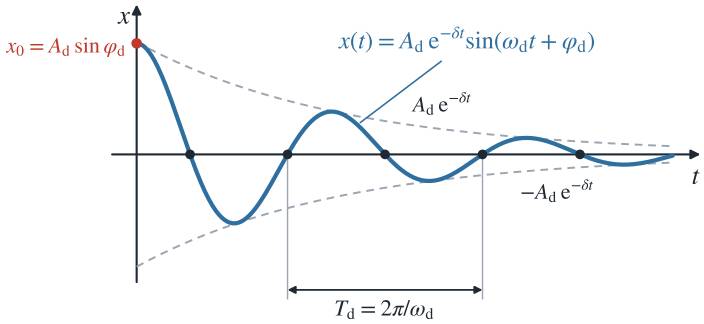


Figure 2.7.: Time history $x(t) = A_\mathrm{d} \mathrm{e}^{-\delta t} \sin(\omega_\mathrm{d} t + \varphi_\mathrm{d})$ of a mass-spring oscillator under weak damping ($\delta < \omega_0$), starting from rest with the displacement $x_0 = A_\mathrm{d} \sin\varphi_\mathrm{d}$. The amplitude decays exponentially within the dashed envelope $\pm A_\mathrm{d} \mathrm{e}^{-\delta t}$. The zero crossings are spaced $T_\mathrm{d}/2$ apart, with the period $T_\mathrm{d} = 2\pi/\omega_\mathrm{d}$. Adapted from Papula (2015).

Example 2.36 (Weakly Damped Oscillation Starting from Rest)**.** Consider the free damped oscillation Equation 2.10,
$$m\ddot{x} + b\dot{x} + cx = 0 \qquad \text{or} \qquad \ddot{x} + 2\delta\dot{x} + \omega_0^2 x = 0,$$
with the damping factor $\delta = b/(2m) > 0$ and the natural angular frequency $\omega_0 = \sqrt{c/m}$ of the undamped system. Let the system be weakly damped ($\delta < \omega_0$) and start from rest at time $t = 0$ with the displacement $x_0 = A > 0$ and the velocity $v_0 = 0$. We seek the amplitude A_d and the phase angle φ_d of the particular solution.

Approach. The starting point is the general solution for weak damping (case 1 in Theorem 2.10, see Section 2.7.3.1)

$$x(t) = A_\mathrm{d} \mathrm{e}^{-\delta t} \sin(\omega_\mathrm{d} t + \varphi_\mathrm{d}), \qquad \omega_\mathrm{d} = \sqrt{\omega_0^2 - \delta^2} > 0,$$

with $A_\mathrm{d} > 0$. The two parameters are determined from the initial conditions.

Step 1: Velocity. By the product rule,

$$\dot{x}(t) = -\delta A_\mathrm{d} \mathrm{e}^{-\delta t} \sin(\omega_\mathrm{d} t + \varphi_\mathrm{d}) + \omega_\mathrm{d} A_\mathrm{d} \mathrm{e}^{-\delta t} \cos(\omega_\mathrm{d} t + \varphi_\mathrm{d}) = A_\mathrm{d} \mathrm{e}^{-\delta t} \left[\omega_\mathrm{d} \cos(\omega_\mathrm{d} t + \varphi_\mathrm{d}) - \delta \sin(\omega_\mathrm{d} t + \varphi_\mathrm{d})\right].$$

Step 2: Initial conditions. For $t = 0$, $\mathrm{e}^0 = 1$, so

$$x(0) = A_\mathrm{d} \sin \varphi_\mathrm{d} = A,$$

$$\dot{x}(0) = A_\mathrm{d} \left(\omega_\mathrm{d} \cos \varphi_\mathrm{d} - \delta \sin \varphi_\mathrm{d}\right) = 0.$$

Step 3: Phase angle. Since $A_\mathrm{d} \neq 0$, the second equation yields

$$\omega_\mathrm{d} \cos \varphi_\mathrm{d} = \delta \sin \varphi_\mathrm{d}.$$

The first equation yields $\sin \varphi_\mathrm{d} = A/A_\mathrm{d} > 0$, so $\delta > 0$ and $\omega_\mathrm{d} > 0$ imply $\cos \varphi_\mathrm{d} > 0$ as well. The angle therefore lies in the first quadrant, $0 < \varphi_\mathrm{d} < \pi/2$, and division by $\delta \cos \varphi_\mathrm{d}$ gives

$$\tan \varphi_\mathrm{d} = \frac{\omega_\mathrm{d}}{\delta} \qquad \Longrightarrow \qquad \varphi_\mathrm{d} = \arctan\left(\frac{\omega_\mathrm{d}}{\delta}\right).$$

The principal value of the arctangent is the correct choice here because for positive arguments it also lies between 0 and $\pi/2$.

Step 4: Sine of the phase angle. For $0 < \varphi_\mathrm{d} < \pi/2$, $\sin \varphi_\mathrm{d} = \tan \varphi_\mathrm{d} / \sqrt{1 + \tan^2 \varphi_\mathrm{d}}$. With $\delta^2 + \omega_\mathrm{d}^2 = \delta^2 + \omega_0^2 - \delta^2 = \omega_0^2$, it follows that

$$\sin \varphi_\mathrm{d} = \frac{\omega_\mathrm{d}/\delta}{\sqrt{1 + \omega_\mathrm{d}^2/\delta^2}} = \frac{\omega_\mathrm{d}}{\sqrt{\delta^2 + \omega_\mathrm{d}^2}} = \frac{\omega_\mathrm{d}}{\omega_0} \qquad \text{and likewise} \qquad \cos \varphi_\mathrm{d} = \frac{\delta}{\omega_0}.$$

Step 5: Amplitude. Substituting into the first initial condition $A_\mathrm{d} \sin \varphi_\mathrm{d} = A$ yields

$$A_\mathrm{d} = \frac{A}{\sin \varphi_\mathrm{d}} = \frac{\omega_0}{\omega_\mathrm{d}} A.$$

Check. With the results of Steps 4 and 5,

$$x(0) = \frac{\omega_0}{\omega_\mathrm{d}} A \cdot \frac{\omega_\mathrm{d}}{\omega_0} = A \qquad \text{and} \qquad \dot{x}(0) = \frac{\omega_0}{\omega_\mathrm{d}} A \left(\omega_\mathrm{d} \frac{\delta}{\omega_0} - \delta \frac{\omega_\mathrm{d}}{\omega_0}\right) = 0.$$

Result. The particular solution is

$$x(t) = A_\mathrm{d} \mathrm{e}^{-\delta t} \sin(\omega_\mathrm{d} t + \varphi_\mathrm{d})$$

with the parameters

$$A_\mathrm{d} = \frac{\omega_0}{\omega_\mathrm{d}} A \qquad \text{and} \qquad \varphi_\mathrm{d} = \arctan\left(\frac{\omega_\mathrm{d}}{\delta}\right). \qquad \square$$

The numerical examples on damped oscillation follow Papula (2015, sec. IV.4.1.3).

Example 2.37 (Weak Damping)**.** Consider an oscillatory mechanical system with the mass $m = 10\,\mathrm{kg}$, the damper constant $b = 80\,\mathrm{kg/s}$, and the spring constant $c = 250\,\mathrm{N/m}$. The initial conditions are

$$x(0) = 0.6\,\mathrm{m} \quad \text{and} \quad v(0) = 0\,\mathrm{m/s}.$$

The damping factor and the natural angular frequency of the undamped system are

$$\delta = \frac{b}{2m} = \frac{80\,\mathrm{kg/s}}{20\,\mathrm{kg}} = 4\,\mathrm{s}^{-1} \quad \text{and} \quad \omega_0 = \sqrt{\frac{c}{m}} = 5\,\mathrm{s}^{-1}.$$

The equation of motion (without units) is therefore $\ddot{x} + 8\dot{x} + 25x = 0$. Since $\delta < \omega_0$, the system is weakly damped and oscillates with the natural angular frequency

$$\omega_\mathrm{d} = \sqrt{\omega_0^2 - \delta^2} = \sqrt{25 - 16}\,\mathrm{s}^{-1} = 3\,\mathrm{s}^{-1}.$$

The general solution of the ODE is therefore

$$x(t) = \mathrm{e}^{-4t}\left[C_1 \sin(3t) + C_2 \cos(3t)\right]$$

or, in the form of a phase-shifted sine oscillation,

$$x(t) = A_\mathrm{d}\mathrm{e}^{-4t} \sin(3t + \varphi_\mathrm{d}). \tag{2.14}$$

We consider two ways of solving:

1. First, the solution using the formulas from Example 2.36: By Example 2.36 with $A = 0.6$ m, the amplitude and the phase angle are

   $$A_\mathrm{d} = \frac{\omega_0}{\omega_\mathrm{d}} A = \frac{5}{3} \cdot 0.6\,\mathrm{m} = 1\,\mathrm{m} \quad \text{and} \quad \varphi_\mathrm{d} = \arctan\left(\frac{\omega_\mathrm{d}}{\delta}\right) = \arctan\left(\frac{3}{4}\right) \approx 0.6435.$$

   The system therefore performs the damped oscillation

   $$x(t) = 1\,\mathrm{m} \cdot \mathrm{e}^{-4\,\mathrm{s}^{-1}t} \sin(3\,\mathrm{s}^{-1} \cdot t + 0.6435), \quad t \geq 0.$$

2. Alternatively, we substitute the initial conditions directly into Equation 2.14. With the derivative

   $$\dot{x}(t) = A_\mathrm{d}\mathrm{e}^{-4t}\left[3\cos(3t + \varphi_\mathrm{d}) - 4\sin(3t + \varphi_\mathrm{d})\right]$$

   we obtain (without units) the system of equations

   $$\begin{aligned} \text{I:} \quad & x(0) = A_\mathrm{d} \sin\varphi_\mathrm{d} = 0.6, \\ \text{II:} \quad & \dot{x}(0) = A_\mathrm{d}\left(3\cos\varphi_\mathrm{d} - 4\sin\varphi_\mathrm{d}\right) = 0. \end{aligned}$$

   Since $A_\mathrm{d} \neq 0$, II yields

   $$4\sin\varphi_\mathrm{d} = 3\cos\varphi_\mathrm{d} \Rightarrow \tan\varphi_\mathrm{d} = \frac{3}{4} \Rightarrow \varphi_\mathrm{d} = \arctan\left(\frac{3}{4}\right) \approx 0.6435.$$

   With $\sin\varphi_\mathrm{d} = 3/5$ (cf. $\sin\varphi_\mathrm{d} = \omega_\mathrm{d}/\omega_0$ in Example 2.36), I yields

   $$A_\mathrm{d} = \frac{0.6\,\mathrm{m}}{\sin\varphi_\mathrm{d}} = \frac{0.6\,\mathrm{m}}{3/5} = 1\,\mathrm{m}.$$

□

2.7.4. Strong Damping (Aperiodic Behavior, Overdamped Case)

Theorem 2.12 (General Solution of the Equation of Motion of a Free Damped Spring Pendulum under Strong Damping)**.** *Under strong damping, $\Delta = \delta^2 - \omega_0^2 > 0$, i.e., $\delta > \omega_0$. The characteristic equation Equation 2.11 then has the two distinct real solutions*

$$\lambda_1 = -\delta + \sqrt{\delta^2 - \omega_0^2} < 0 \quad \textit{and} \quad \lambda_2 = -\delta - \sqrt{\delta^2 - \omega_0^2} < 0,$$

i.e., $\lambda_2 < \lambda_1 < 0$. The general solution of the ODE Equation 2.10 is a linear combination of two monotonically decreasing exponential functions,

$$x(t) = C_1\mathrm{e}^{\lambda_1 t} + C_2\mathrm{e}^{\lambda_2 t}, \quad C_1, C_2 \in \mathbb{R}. \tag{2.15}$$

The mechanical system no longer oscillates but approaches the equilibrium state asymptotically, see Figure 2.8. □

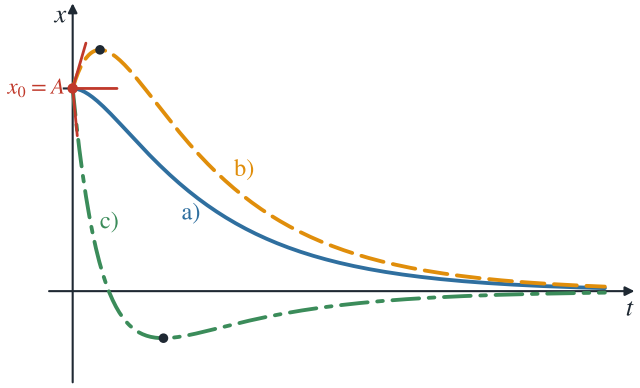


Figure 2.8.: Time history of the motion of a mass-spring oscillator under strong damping ($\delta > \omega_0$, overdamped case) for three different initial velocities with the same initial position $x_0 = A > 0$ (red, with the initial tangents of slope v_0): a) from rest ($v_0 = 0$), b) outward, i.e., away from the equilibrium position ($v_0 > 0$), c) inward, i.e., toward the equilibrium position with sufficient velocity ($v_0 < \lambda_2 A$), so that the mass passes through the equilibrium position once. The dots mark the turning points of the motion ($\dot{x} = 0$). Adapted from Papula (2015).

Example 2.38 (Overdamped Case: Motion Starting from Rest)**.** If the system starts from rest, the initial conditions are $x(0) = A$ and $v(0) = 0$. For the coefficients of the general solution Equation 2.15, $x(0) = A$ and $\dot{x}(0) = 0$ yield the linear system of equations

$$\begin{aligned} C_1 + C_2 &= A, \\ \lambda_1 C_1 + \lambda_2 C_2 &= 0 \end{aligned}$$

with the solutions

$$C_1 = \frac{\lambda_2}{\lambda_2 - \lambda_1} A \quad \text{and} \quad C_2 = \frac{\lambda_1}{\lambda_1 - \lambda_2} A.$$

The particular solution is therefore

$$x(t) = \frac{\lambda_2}{\lambda_2 - \lambda_1} A \mathrm{e}^{\lambda_1 t} + \frac{\lambda_1}{\lambda_1 - \lambda_2} A \mathrm{e}^{\lambda_2 t},$$

see Figure 2.9. □

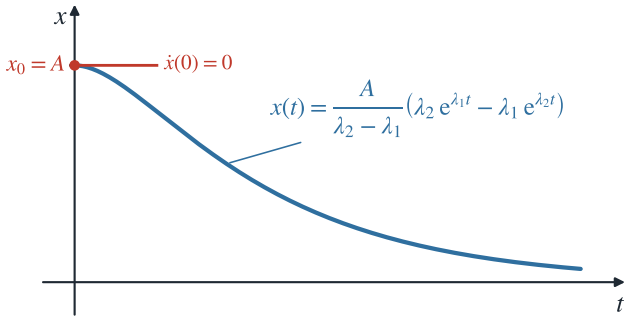


Figure 2.9.: Time history of the motion in the overdamped case (strong damping, $\delta > \omega_0$) for the motion from rest with the initial values $x(0) = x_0 = A$ and $v(0) = \dot{x}(0) = 0$: the displacement starts with a horizontal tangent and approaches the equilibrium position $x = 0$ monotonically, without oscillating. Adapted from Papula (2015).

Example 2.39 (Overdamped Case: Motion Starting from the Equilibrium Position)**.** If the system starts from the equilibrium position with the velocity $v_0 > 0$, the initial conditions are $x(0) = 0$ and $v(0) = v_0$. From

$$x(0) = C_1 + C_2 = 0 \quad \text{and} \quad \dot{x}(0) = \lambda_1 C_1 + \lambda_2 C_2 = v_0$$

we obtain the coefficients

$$C_1 = \frac{v_0}{\lambda_1 - \lambda_2} \quad \text{and} \quad C_2 = -\frac{v_0}{\lambda_1 - \lambda_2}$$

and hence the particular solution

$$x(t) = \frac{v_0}{\lambda_1 - \lambda_2} \left(e^{\lambda_1 t} - e^{\lambda_2 t} \right),$$

see Figure 2.10. □

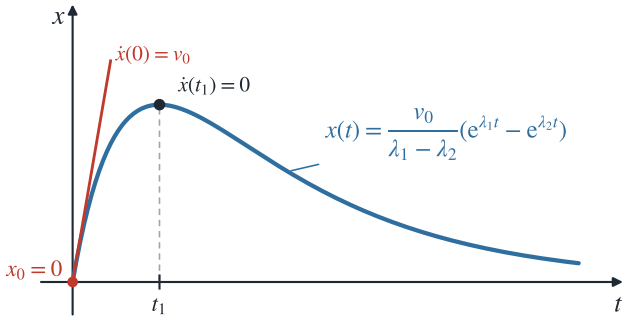


Figure 2.10.: Time history of the motion in the overdamped case (strong damping, $\delta > \omega_0$) for the motion out of the equilibrium position with the initial values $x(0) = 0$ and $v(0) = \dot{x}(0) = v_0 > 0$: the mass first moves away from the equilibrium position, reaches its largest displacement at time $t_1 = \ln(\lambda_2/\lambda_1)/(\lambda_1 - \lambda_2)$ (turning point, $\dot{x}(t_1) = 0$), and then creeps back into the equilibrium position without oscillating. Adapted from Papula (2015).

Example 2.40 (Strong Damping)**.** Consider the equation of motion

$$\ddot{x} + 6\dot{x} + 8.75\, x = 0$$

with the initial values $x(0) = 0$ and $v(0) = 8$. The characteristic equation has two distinct negative solutions:

$$\lambda^2 + 6\lambda + 8.75 = 0 \Rightarrow \lambda_1 = -2.5, \quad \lambda_2 = -3.5.$$

The general solution is therefore

$$x(t) = C_1 e^{-2.5\, t} + C_2 e^{-3.5\, t}.$$

From $x(0) = C_1 + C_2 = 0$ it follows that $C_2 = -C_1$. With $\dot{x}(0) = -2.5\, C_1 - 3.5\, C_2 = 8$ we obtain $-2.5\, C_1 + 3.5\, C_1 = C_1 = 8$, so $C_1 = 8$ and $C_2 = -8$. The particular solution is

$$x(t) = 8 \left(e^{-2.5\, t} - e^{-3.5\, t} \right), \quad t \geq 0.$$

It is the special case $v_0 = 8$ of Example 2.39 and describes an aperiodic motion (overdamped case) with the turning point at

$$t_1 = \frac{\ln(\lambda_2/\lambda_1)}{\lambda_1 - \lambda_2} = \ln(1.4) \approx 0.34,$$

see Figure 2.10. □

2.7.5. Aperiodic Limiting Case

2.7.5.1. General Solution of the Equation of Motion in the Aperiodic Limiting Case

For $\Delta = \delta^2 - \omega_0^2 = 0$ we have the aperiodic limiting case. It separates the oscillations from the aperiodic motions. What is separated are the values of the damping factor $\delta = b/(2m)$: at a fixed natural angular frequency ω_0, they fall into two ranges with fundamentally different motion:

- For $\delta < \omega_0$ (weak damping, Section 2.7.3.1), the system *oscillates*. The mass swings about the rest position with the angular frequency $\omega_d = \sqrt{\omega_0^2 - \delta^2}$, passes through it infinitely often, and comes to rest with exponentially decreasing amplitude. The special case $\delta = 0$ is the free undamped oscillation (Section 2.7.2).
- For $\delta > \omega_0$ (strong damping, overdamped case, Section 2.7.4), the system does *not oscillate*. The mass creeps back to the rest position and passes through it at most once.

The aperiodic limiting case $\delta = \omega_0$ lies exactly on the boundary between these ranges. It belongs to the aperiodic motions itself: as in the overdamped case, the mass passes through the rest position at most once (Figure 2.11). If the damping is increased starting from $\delta = 0$, ω_d becomes smaller and smaller and the period $T_\mathrm{d} = 2\pi/\omega_\mathrm{d}$ larger and larger. For $\delta \to \omega_0$, $\omega_\mathrm{d} \to 0$ and $T_\mathrm{d} \to \infty$: a full oscillation would never be completed. The limiting case is thus the smallest damping at which the system no longer oscillates. For the damper constant, $\delta = \omega_0$ means that

$$b = 2\sqrt{c\,m}.$$

If the mass is released from rest, it returns to the rest position faster in the limiting case than with any stronger damping, without overshooting the rest position as with weaker damping.

Table 2.3.: Types of motion of the free mass-spring oscillator depending on the damping factor δ

Case	Condition	Solutions of the characteristic equation	Motion	Section
undamped	$\delta = 0$	$\lambda_{1,2} = \pm\mathrm{j}\omega_0$	harmonic oscillation with constant amplitude	Section 2.7.2
weak damping	$0 < \delta < \omega_0$	$\lambda_{1,2} = -\delta \pm \mathrm{j}\omega_\mathrm{d}$	damped oscillation, infinitely many zero crossings	Section 2.7.3.1
aperiodic limiting case	$\delta = \omega_0$	$\lambda_{1,2} = -\delta$ (double)	aperiodic, at most one zero crossing	Section 2.7.5
strong damping (overdamped case)	$\delta > \omega_0$	$\lambda_{1,2} = -\delta \pm \sqrt{\delta^2 - \omega_0^2}$ (real, distinct)	aperiodic, at most one zero crossing	Section 2.7.4

In the limiting case, the characteristic equation (Equation 2.11) has the double negative real solution

$$\lambda_{1,2} = -\delta.$$

By Equation 2.4, the general solution of the ODE Equation 2.10 is the function given in Equation 2.12,

$$x(t) = [C_1 t + C_2]\,\mathrm{e}^{-\delta t}, \quad C_1, C_2 \in \mathbb{R}. \tag{2.16}$$

The oscillatory mechanical system behaves similarly to the overdamped case. It moves aperiodically out of its initial position and approaches the equilibrium state asymptotically, see Figure 2.11.

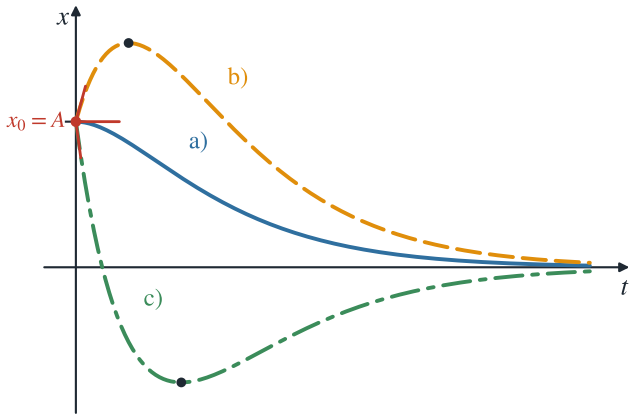


Figure 2.11.: Time history of the oscillation of a mass-spring oscillator in the aperiodic limiting case, $x(t) = [A + (v_0 + \delta A)\,t]\,\mathrm{e}^{-\delta t}$, for three different initial conditions: a) from rest ($v_0 = 0$), b) outward, i.e., away from the equilibrium position ($v_0 > 0$), c) inward, i.e., toward the equilibrium position with sufficient velocity ($v_0 < -\delta A$). The initial position is the same in all three cases: $x_0 = A > 0$ (red, with the initial tangents of slope v_0). The dots mark the turning points of the motion. Adapted from Papula (2015).

2.7.5.2. Special Cases of the Aperiodic Limiting Case

Example 2.41 (Aperiodic Limiting Case: Motion Starting from Rest)**.** If the system starts from rest, the initial conditions are $x(0) = A$ and $v(0) = 0$. For the constants of the general solution Equation 2.12, it follows that

$$x(0) = C_2 = A \quad \text{and} \quad \dot{x}(0) = C_1 - \delta C_2 = 0,$$

i.e., $C_1 = \delta A$ and $C_2 = A$. The particular solution is

$$x(t) = [\delta A t + A]\, \mathrm{e}^{-\delta t}, \quad t \geq 0.$$

The system moves aperiodically out of rest and approaches the equilibrium state asymptotically, see Figure 2.12. □

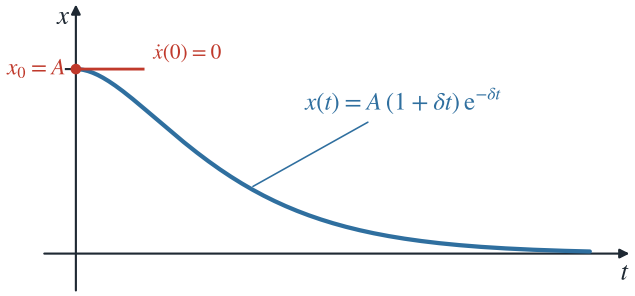


Figure 2.12.: Time history of the motion in the aperiodic limiting case, $x(t) = A\,(1 + \delta t)\,\mathrm{e}^{-\delta t}$, for the initial values $x(0) = A$ and $v(0) = 0$ (red: starting point with horizontal tangent). Adapted from Papula (2015).

Example 2.42 (Aperiodic Limiting Case: Motion Starting from the Equilibrium Position)**.** If the system, located in the equilibrium position, is set in motion at time $t = 0$ with the velocity $v_0 > 0$, the initial conditions are $x(0) = 0$ and $v(0) = v_0$.

With Equation 2.16

$$x(t) = [C_1 t + C_2]\, \mathrm{e}^{-\delta t}, \quad C_1, C_2 \in \mathbb{R}.$$

it follows from

$$x(0) = C_2 = 0 \quad \text{and} \quad \dot{x}(0) = C_1 - \delta C_2 = v_0$$

that $C_1 = v_0$ and $C_2 = 0$.

The particular solution is

$$x(t) = v_0\, t\, \mathrm{e}^{-\delta t}.$$

Owing to the initial velocity, the mass first moves aperiodically out of the equilibrium position, reaches its turning point at time $t_2 = 1/\delta$, and then approaches the equilibrium state asymptotically, see Figure 2.13. □

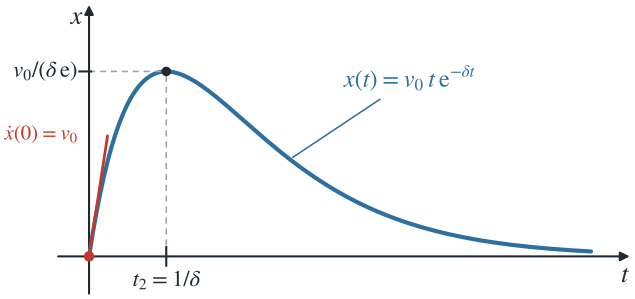


Figure 2.13.: Time history of the motion in the aperiodic limiting case, $x(t) = v_0\, t\, \mathrm{e}^{-\delta t}$, for the initial values $x(0) = 0$ and $v(0) = v_0 > 0$ (red: starting point and initial tangent). The turning point lies at $t_2 = 1/\delta$ with the maximum displacement $x(t_2) = v_0/(\delta\,\mathrm{e})$. Adapted from Papula (2015).

Example 2.43 (Aperiodic Limiting Case)**.** Consider an oscillatory mechanical system with the spring constant $c = 200$ N/m and the damper constant $b = 60$ kg/s. How large must the oscillating mass m be for the aperiodic limiting case to occur?

The aperiodic limiting case occurs for $\delta = \omega_0$, i.e., for

$$\frac{b}{2m} = \sqrt{\frac{c}{m}} \quad \text{or} \quad \frac{b^2}{4m^2} = \frac{c}{m}.$$

Solving for m gives

$$m = \frac{b^2}{4c} = \frac{(60\,\text{kg/s})^2}{4 \cdot 200\,\text{N/m}} = 4.5\,\text{kg}.$$

For $m > 4.5$ kg, $\delta < \omega_0$, so the system performs a damped oscillation. For $m < 4.5$ kg, it moves aperiodically (overdamped case). □

2.7.6. Forced Oscillation

In a forced oscillation, an external periodic force

$$F(t) = F_0 \sin(\omega t)$$

with the amplitude F_0 and the excitation angular frequency ω acts on the mass in addition to the restoring force and the damping force. This force can also be represented in complex form, see Equation 2.19. The equation of motion of the spring pendulum then reads

$$m\ddot{x} + b\dot{x} + cx = F_0 \sin(\omega t)$$

or, with the abbreviations $\delta = b/(2m)$, $\omega_0^2 = c/m$, and $K_0 = F_0/m$,

$$\ddot{x} + 2\delta\dot{x} + \omega_0^2 x = K_0 \sin(\omega t). \tag{2.17}$$

2.7.6.1. Weakly Damped Forced Oscillation

Under weak damping ($\delta < \omega_0$), the homogeneous ODE associated with Equation 2.17,

$$\ddot{x} + 2\delta\dot{x} + \omega_0^2 x = 0$$

is solved, according to Section 2.7.3.1, by the function

$$x(t) = A_\text{d} \mathrm{e}^{-\delta t} \sin(\omega_\text{d} t + \varphi_\text{d})$$

It describes a damped oscillation with the natural angular frequency

$$\omega_\text{d} = \sqrt{\omega_0^2 - \delta^2}.$$

For the particular solution of the inhomogeneous ODE Equation 2.17 we use, following Equation 2.6, the complex ansatz function

$$\underline{x}_\text{p}(t) = A\mathrm{e}^{\mathrm{j}(\omega t - \varphi)} = A\cos(\omega t - \varphi) + \mathrm{j}A\sin(\omega t - \varphi). \tag{2.18}$$

In the complex calculation it is more convenient to write the argument in the form $\omega t - \varphi$. The external force $F(t)$ is also converted into complex form:

$$\underline{F}(t) = F_0 \mathrm{e}^{\mathrm{j}\omega t} = F_0 \cos(\omega t) + \mathrm{j}F_0 \sin(\omega t). \tag{2.19}$$

The complex and the real representations are therefore related by

$$x_\text{p} = \mathrm{Im}(\underline{x}_\text{p}) = \mathrm{Im}\left[A\mathrm{e}^{\mathrm{j}(\omega t - \varphi)}\right] = A\sin(\omega t - \varphi)$$

and

$$F(t) = \mathrm{Im}(\underline{F}(t)) = \mathrm{Im}\left[F_0 \mathrm{e}^{\mathrm{j}\omega t}\right] = F_0 \sin(\omega t).$$

In complex form, the equation of motion Equation 2.17 reads

$$\ddot{\underline{x}} + 2\delta\dot{\underline{x}} + \omega_0^2 \underline{x} = K_0 \mathrm{e}^{\mathrm{j}\omega t}.$$

We compute the time derivatives of the ansatz function $\underline{x}_\mathrm{p}(t)$ from Equation 2.18,

$$\dot{\underline{x}}_p(t) = \mathrm{j}\omega A \mathrm{e}^{\mathrm{j}(\omega t - \varphi)} = \mathrm{j}\omega A \mathrm{e}^{\mathrm{j}\omega t}\mathrm{e}^{-\mathrm{j}\varphi}$$

and

$$\ddot{\underline{x}}_p(t) = -\omega^2 A \mathrm{e}^{\mathrm{j}(\omega t - \varphi)} = -\omega^2 A \mathrm{e}^{\mathrm{j}\omega t}\mathrm{e}^{-\mathrm{j}\varphi}.$$

Substituting the derivatives into the equation of motion yields

$$-\omega^2 A \mathrm{e}^{\mathrm{j}\omega t}\mathrm{e}^{-\mathrm{j}\varphi} + 2\delta \mathrm{j}\omega A \mathrm{e}^{\mathrm{j}\omega t}\mathrm{e}^{-\mathrm{j}\varphi} + \omega_0^2 A \mathrm{e}^{\mathrm{j}\omega t}\mathrm{e}^{-\mathrm{j}\varphi} = K_0 \mathrm{e}^{\mathrm{j}\omega t}.$$

Division by $A\mathrm{e}^{\mathrm{j}\omega t} \neq 0$ and subsequent multiplication by $\mathrm{e}^{\mathrm{j}\varphi}$ yields

$$-\omega^2 + 2\delta \mathrm{j}\omega + \omega_0^2 = \frac{K_0}{A}\mathrm{e}^{\mathrm{j}\varphi}.$$

Sorting the terms into real and imaginary parts yields

$$(\omega_0^2 - \omega^2) + \mathrm{j}\, 2\delta\omega = \frac{K_0}{A}\mathrm{e}^{\mathrm{j}\varphi}. \tag{2.20}$$

The left-hand side of Equation 2.20 is a complex number in algebraic form with the real part $\omega_0^2 - \omega^2$ and the imaginary part $2\delta\omega$. The right-hand side is the same number in polar form with the modulus K_0/A and the argument φ.

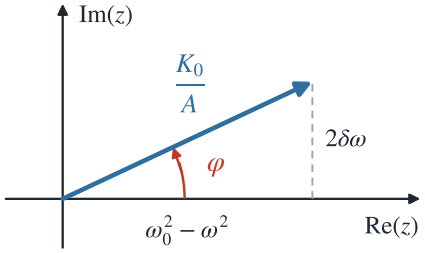


Figure 2.14.: Representation of the complex number $z = (\omega_0^2 - \omega^2) + \mathrm{j}\, 2\delta\omega$ in the complex plane: the real part $\omega_0^2 - \omega^2$, the imaginary part $2\delta\omega$, the modulus K_0/A, and the argument (phase angle) φ form a right triangle. Adapted from Papula (2015).

With the Pythagorean theorem (see Figure 2.14) we obtain

$$\left[\omega_0^2 - \omega^2\right]^2 + \left[2\delta\omega\right]^2 = \left[\frac{K_0}{A}\right]^2 \tag{2.21}$$

and

$$\tan(\varphi) = \frac{2\delta\omega}{\omega_0^2 - \omega^2}. \tag{2.22}$$

From Equation 2.21 and Equation 2.22, the amplitude A and the phase angle φ can be determined. For the amplitude we have

$$A = \frac{K_0}{\sqrt{(\omega_0^2 - \omega^2)^2 + (2\delta\omega)^2}} = \frac{F_0}{m\sqrt{(\omega_0^2 - \omega^2)^2 + (2\delta\omega)^2}}. \tag{2.23}$$

For the phase angle φ we distinguish three cases:

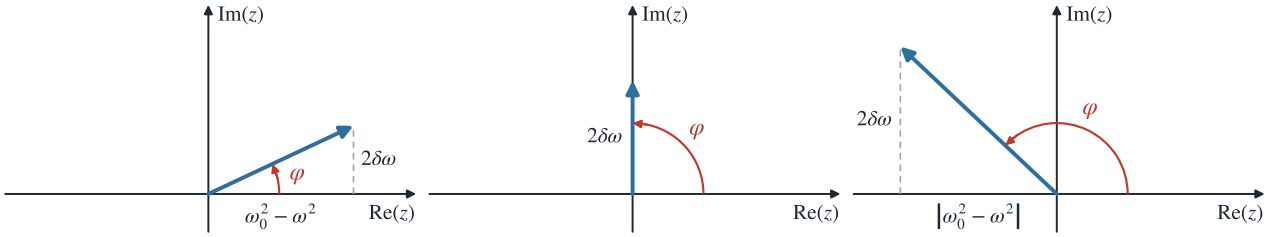
1. $\omega < \omega_0$: $\varphi = \arctan\left(\frac{2\delta\omega}{\omega_0^2 - \omega^2}\right)$, see Figure 2.15a.
2. $\omega = \omega_0$: $\varphi = \frac{\pi}{2}$, see Figure 2.15b.
3. $\omega > \omega_0$: $\varphi = \arctan\left(\frac{2\delta\omega}{\omega_0^2 - \omega^2}\right) + \pi$, see Figure 2.15c.

The particular solution of the ODE Equation 2.17 then reads, in complex form,

$$\underline{x}_\mathrm{p} = A\mathrm{e}^{\mathrm{j}(\omega t - \varphi)}$$

with the amplitude A from Equation 2.23 and the phase angle φ. The sought real solution is its imaginary part:

$$x_\mathrm{p} = \mathrm{Im}(\underline{x}_\mathrm{p}) = A\sin(\omega t - \varphi).$$



(a) $\omega < \omega_0$: $0 < \varphi < \pi/2$ (b) $\omega = \omega_0$: $\varphi = \pi/2$ (c) $\omega > \omega_0$: $\pi/2 < \varphi < \pi$

Figure 2.15.: Phase angle φ of the forced oscillation as the argument of the complex number $(\omega_0^2 - \omega^2) + \mathrm{j}\, 2\delta\omega$ in the three cases (same scale, δ fixed). Adapted from Papula (2015).

Under weak damping ($\delta < \omega_0$), the equation of motion Equation 2.17 therefore has the general solution

$$x(t) = x_\mathrm{h} + x_\mathrm{p} = A_\mathrm{d} \mathrm{e}^{-\delta t} \sin(\omega_\mathrm{d} t + \varphi_\mathrm{d}) + A \sin(\omega t - \varphi).$$

The motion of the weakly damped system is thus composed of two oscillations that superpose without disturbing each other.

The homogeneous equation of motion $\ddot{x} + 2\delta\dot{x} + \omega_0^2 x = 0$ provides the contribution

$$x_\mathrm{h}(t) = A_\mathrm{d} \mathrm{e}^{-\delta t} \sin(\omega_\mathrm{d} t + \varphi_\mathrm{d}), \tag{2.24}$$

a damped oscillation with the natural angular frequency $\omega_\mathrm{d} = \sqrt{\omega_0^2 - \delta^2}$. Since it is damped, it dies out after a certain time, the transient phase. It is therefore also called the "transient component", see Figure 2.16.

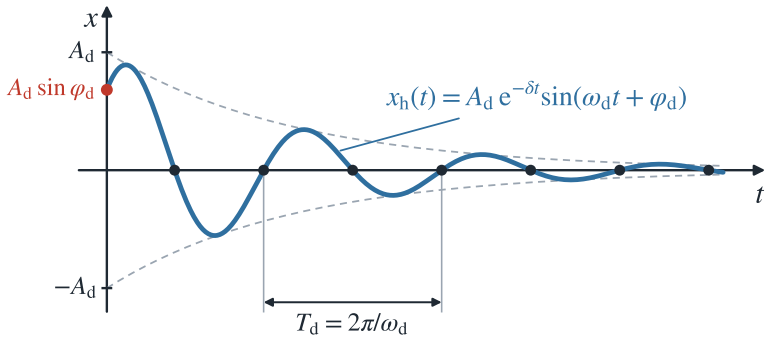


Figure 2.16.: "Transient" contribution $x_\mathrm{h}(t) = A_\mathrm{d} \mathrm{e}^{-\delta t} \sin(\omega_\mathrm{d} t + \varphi_\mathrm{d})$ of the homogeneous equation of motion to the total solution: a damped oscillation with the envelope $\pm A_\mathrm{d} \mathrm{e}^{-\delta t}$ (dashed), the initial value $A_\mathrm{d} \sin \varphi_\mathrm{d}$, and the period $T_\mathrm{d} = 2\pi/\omega_\mathrm{d}$. Adapted from Papula (2015).

The contribution of the inhomogeneous equation of motion $\ddot{x} + 2\delta\dot{x} + \omega_0^2 x = K_0 \sin(\omega t)$ to the total oscillation is the particular solution

$$x_\mathrm{p}(t) = A \sin(\omega t - \varphi), \tag{2.25}$$

an undamped oscillation with the angular frequency ω of the exciter. The external force $F(t)$ thus excites the system to a harmonic oscillation, see Figure 2.17. Such a contribution is called a "forced oscillation".

After the transient phase, the transient component of the solution has vanished, so that only the forced oscillation remains, see Figure 2.18.

The mechanical system then oscillates without damping at the angular frequency ω of the exciter. The amplitude and the phase shift φ between the excitation and the steady-state solution $x_\mathrm{p}(t)$ depend on the excitation angular frequency, $A = A(\omega)$ and $\varphi = \varphi(\omega)$.

Example 2.44 (Forced Oscillation)**.** This numerical example is not taken from Papula (2015). It uses characteristic data similar to Example 2.37. A mass-spring oscillator with $m = 10$ kg, $b = 40$ kg/s, and $c = 250$ N/m is excited by the

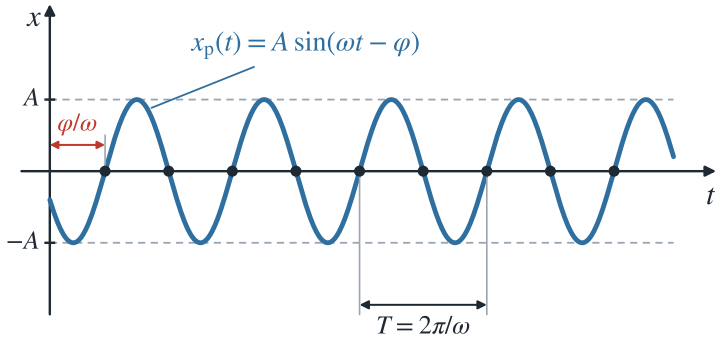


Figure 2.17.: "Forced" contribution $x_p(t) = A\sin(\omega t - \varphi)$ of the inhomogeneous equation of motion to the total solution: an undamped oscillation with the excitation angular frequency ω, the amplitude A, the time shift φ/ω, and the period $T = 2\pi/\omega$. Same scale as in Figure 2.16. Adapted from Papula (2015).

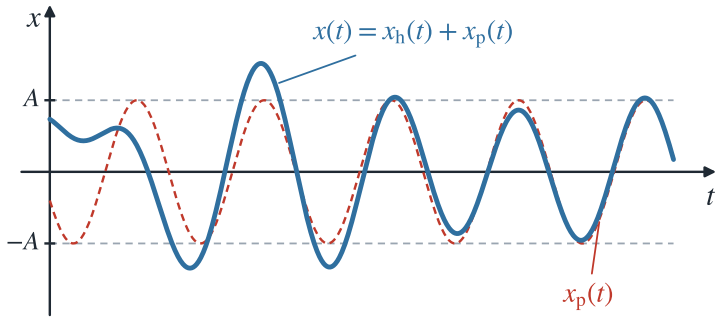


Figure 2.18.: Total solution $x(t) = x_h(t) + x_p(t)$ (solid) composed of the components in Figure 2.16 and Figure 2.17: after the transient phase, it passes into the forced oscillation $x_p(t)$ (dashed).

force $F(t) = 100\,\mathrm{N} \cdot \sin(3\,\mathrm{s}^{-1} \cdot t)$ and starts from the rest position, $x(0) = 0$ and $v(0) = 0$. We omit the units in the intermediate calculations. With

$$\delta = \frac{b}{2m} = 2, \qquad \omega_0 = \sqrt{\frac{c}{m}} = 5, \qquad K_0 = \frac{F_0}{m} = 10, \qquad \omega = 3$$

the equation of motion Equation 2.17 reads

$$\ddot{x} + 4\dot{x} + 25x = 10\sin(3t).$$

Since $\delta < \omega_0$, the system is weakly damped with $\omega_\mathrm{d} = \sqrt{25-4} = \sqrt{21}$. By Equation 2.23 and Equation 2.22, the amplitude and the phase angle of the forced oscillation are

$$A = \frac{10}{\sqrt{(25-9)^2 + (2\cdot 2\cdot 3)^2}} = \frac{10}{\sqrt{256+144}} = \frac{10}{20} = 0.5$$

and, since $\omega < \omega_0$,

$$\varphi = \arctan\left(\frac{12}{16}\right) = \arctan\left(\frac{3}{4}\right) \approx 0.6435,$$

i.e., $\sin\varphi = 3/5$ and $\cos\varphi = 4/5$. In the form of Theorem 2.10, the general solution is

$$x(t) = \mathrm{e}^{-2t}\left[C_1\sin(\sqrt{21}\,t) + C_2\cos(\sqrt{21}\,t)\right] + 0.5\sin(3t-\varphi).$$

From $x(0) = C_2 - 0.5\sin\varphi = C_2 - 0.3 = 0$ it follows that $C_2 = 0.3$. From

$$\dot{x}(0) = \sqrt{21}\,C_1 - 2C_2 + 1.5\cos\varphi = \sqrt{21}\,C_1 - 0.6 + 1.2 = 0$$

it follows that $C_1 = -0.6/\sqrt{21} \approx -0.1309$. The particular solution is therefore

$$x(t) = \mathrm{e}^{-2t}\left[-0.1309\sin(\sqrt{21}\,t) + 0.3\cos(\sqrt{21}\,t)\right] + 0.5\sin(3t-0.6435)$$

(in m, $t \geq 0$). Since $\mathrm{e}^{-2t} < 0.01$ for $t > 2.3$ s, the transient component has practically died out by then. Afterwards, the system oscillates in steady state with the amplitude 0.5 m and the excitation angular frequency $3\,\mathrm{s}^{-1}$, lagging behind the excitation by $\varphi/\omega \approx 0.21$ s. □

2.7.7. Frequency Response

Definition 2.11 (Frequency Response)**.** The dependence of the amplitude A and the phase shift φ of the forced oscillation on the excitation angular frequency ω is called the *frequency response*. The function $A(\omega)$ is called the *amplitude frequency response*, the function $\varphi(\omega)$ the *phase frequency response*. □

2.7.7.1. Frequency Response of the Amplitude

By Equation 2.23, the amplitude of the forced oscillation is

$$A(\omega) = \frac{F_0}{m\sqrt{(\omega_0^2-\omega^2)^2 + (2\delta\omega)^2}}. \tag{2.26}$$

This function is also called the *resonance curve*. Its value at $\omega = 0$ describes the static case.

Definition 2.12 (Static Case)**.** The *static case* is the limiting case $\omega \to 0$ of the forced oscillation. The excitation force then changes so slowly that at every instant it acts like a force that is constant in time. A constant force F_0 stretches the spring by the *static displacement*

$$x_\mathrm{st} = \frac{F_0}{c}. \tag{2.27}$$

□

Remark 2.5 (Remark on the Static Case). Strictly speaking, the excitation force $F_0 \sin(\omega t)$ vanishes for $\omega = 0$. What is meant is therefore the limit $\omega \to 0$: the period $2\pi/\omega$ of the excitation becomes arbitrarily long, and the force passes through its values between $-F_0$ and F_0 arbitrarily slowly. In the static case, the resonance curve has the value

$$A(0) = \frac{F_0}{m\omega_0^2} = \frac{F_0}{c} = x_{\text{st}}.$$

A plausible explanation follows from comparing the three forces in the equation of motion $m\ddot{x} + b\dot{x} + cx = F(t)$. For the steady-state solution $x = A\sin(\omega t - \varphi)$, the inertial force $m\ddot{x}$, the damping force $b\dot{x}$, and the spring force cx have the amplitudes

$$mA\omega^2, \qquad bA\omega \qquad \text{and} \qquad cA.$$

For $\omega \to 0$, the first two become arbitrarily small compared with the spring force, and the equation of motion turns into Hooke's law $c\,x = F(t)$ (Theorem 2.7). The mass is moved so slowly that neither its inertia nor the damper plays a role; the spring alone balances the external force. This is why $A(0)$ depends neither on the mass m nor on the damping δ, and why all resonance curves in Figure 2.19 start at the same point. For the same reason, the displacement follows the force without delay; the phase shift is $\varphi(0) = 0$ (Section 2.7.7.2).

In the static case, the mass-spring oscillator looks as in Figure 2.2, with the force $F(t) = F_0$ pulling downward constantly. The mass rests in a new equilibrium position that lies $x_{\text{st}} = F_0/c$ below the original rest position $x = 0$. The spring is stretched by this amount in addition, and the piston of the damper stands still. The system acts like a spring balance that indicates the force F_0 by the extension of the spring.

For the behavior, we distinguish two situations:

1. *Constant force.* If the force F_0 is switched on at time $t = 0$ while the mass rests at the rest position $x = 0$, then by Table 2.2 (polynomial of degree 0), $x_{\text{p}} = F_0/c$ is a particular solution of $\ddot{x} + 2\delta\dot{x} + \omega_0^2 x = F_0/m$. The general solution is $x(t) = x_{\text{h}}(t) + F_0/c$. The component x_{h} of the homogeneous ODE describes a transient motion that, depending on the damping (Table 2.3), dies out as a damped oscillation about x_{st} or as a creeping motion. Afterwards, the mass stays at the position x_{st}.
2. *Very slowly varying force.* If the force $F(t) = F_0 \sin(\omega t)$ changes with a very small excitation angular frequency $\omega \ll \omega_0$, the mass passes through the equilibrium positions $x(t) \approx F(t)/c$ one after another. It moves in step with the force between $-x_{\text{st}}$ and x_{st} without oscillating noticeably at its natural frequency. Such a motion is called *quasi-static*.

The static displacement serves as the reference quantity for the resonance curve: the ratio $A(\omega)/x_{\text{st}}$ states by which factor the periodic excitation increases the displacement compared with the static case. In Example 2.44, $x_{\text{st}} = 100\,\text{N}/(250\,\text{N/m}) = 0.4$ m. At $\omega = 3\,\text{s}^{-1}$, the amplitude is 0.5 m, so the displacement is larger than in the static case by the factor 1.25. □

The maximum of the resonance curve lies where the radicand in the denominator of Equation 2.26,

$$\Psi(\omega) = (\omega_0^2 - \omega^2)^2 + (2\delta\omega)^2,$$

becomes minimal. From the condition

$$\Psi'(\omega) = 4\omega(\omega^2 - \omega_0^2 + 2\delta^2) = 0$$

we obtain, for $\omega > 0$, the minimizer

$$\omega_{\text{r}} = \sqrt{\omega_0^2 - 2\delta^2},$$

which is real only for $\omega_0^2 > 2\delta^2$. The second derivative

$$\Psi''(\omega) = 4(3\omega^2 - \omega_0^2 + 2\delta^2)$$

is positive at this point:

$$\Psi''(\omega_{\text{r}}) = 4(3\omega_{\text{r}}^2 - \omega_0^2 + 2\delta^2) = 8(\omega_0^2 - 2\delta^2) > 0.$$

At ω_{r}, the resonance curve $A(\omega)$ therefore has a maximum. The angular frequency ω_{r} is called the *resonance frequency*, the amplitude $A_{\text{r}} = A(\omega_{\text{r}})$ the *resonance amplitude*. In the presence of damping, the resonance frequency is always smaller than ω_{d} and hence smaller than the natural angular frequency ω_0:

$$\omega_{\text{r}} = \sqrt{\omega_0^2 - 2\delta^2} < \omega_{\text{d}} = \sqrt{\omega_0^2 - \delta^2} < \omega_0.$$

If the excitation angular frequency equals the resonance frequency, $\omega = \omega_r$, the system oscillates with the largest possible amplitude. This *resonance case* yields the resonance amplitude

$$A_r = A_{max} = A(\omega_r) = \frac{F_0}{2m\delta\sqrt{\omega_0^2 - \delta^2}} = \frac{F_0}{2m\delta\omega_d}.$$

With increasing excitation angular frequency ω, the amplitude $A(\omega)$ decreases and tends to zero as $\omega \to \infty$. The system is then no longer able to follow the rapid changes of the excitation force. Figure 2.19 shows the resonance curve for three dampings.

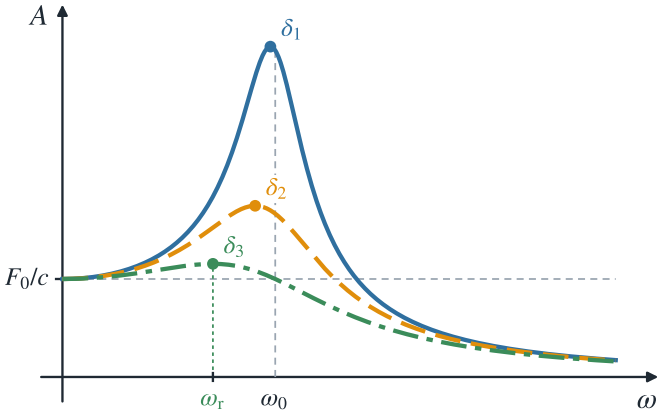


Figure 2.19.: Frequency response of the amplitude $A(\omega)$ (resonance curve) of the forced oscillation for the dampings $\delta_1 = 0.15\,\omega_0 < \delta_2 = 0.3\,\omega_0 < \delta_3 = 0.5\,\omega_0$. All curves start at the static value $A(0) = F_0/c$. The dots mark the resonance amplitudes at the resonance frequencies $\omega_r = \sqrt{\omega_0^2 - 2\delta^2} < \omega_0$ (drawn for δ_3). Adapted from Papula (2015).

In summary, we obtain the important result for the resonance case:

Theorem 2.13 (Resonance Case)**.** *The resonance case occurs when the excitation angular frequency ω equals the resonance frequency*

$$\omega_r = \sqrt{\omega_0^2 - 2\delta^2}$$

The system then oscillates with the largest possible amplitude, the resonance amplitude A_r. □

2.7.7.2. Frequency Response of the Phase Shift

By Equation 2.25, the forced oscillation always lags behind the excitation oscillation by the phase angle φ, where φ lies between 0 and π. Figure 2.20 shows the frequency response of the phase shift $\varphi(\omega)$.

In the static case ($\omega = 0$) the phase shift is $\varphi(0) = 0$, for $\omega = \omega_0$ it is $\varphi(\omega_0) = \pi/2$, and for $\omega \to \infty$ it is $\varphi(\infty) = \pi$. For high angular frequencies, the oscillation therefore lags behind the excitation oscillation by π: exciter and system oscillate in antiphase.

Example 2.45 (Frequency Response in the Absence of Damping and under Strong Damping)**.** We investigate the frequency response of the amplitude $A(\omega)$ and of the phase shift $\varphi(\omega)$ in the absence of damping ($\delta = 0$) and under strong damping ($\delta > \omega_0$). The basis is the resonance curve Equation 2.26 and the phase relation

$$\tan(\varphi) = \frac{2\delta\omega}{\omega_0^2 - \omega^2}, \qquad 0 \le \varphi \le \pi.$$

No damping ($\delta = 0$). The resonance curve simplifies to

$$A(\omega) = \frac{F_0}{m\,|\omega_0^2 - \omega^2|} \qquad (\omega \neq \omega_0).$$

It starts in the static case at $A(0) = F_0/(m\omega_0^2) = x_{st}$, increases monotonically for $0 < \omega < \omega_0$, and tends to infinity as $\omega \to \omega_0$. For $\omega > \omega_0$, it decreases monotonically and tends to zero as $\omega \to \infty$. The resonance frequency $\omega_r = \sqrt{\omega_0^2 - 2\delta^2}$

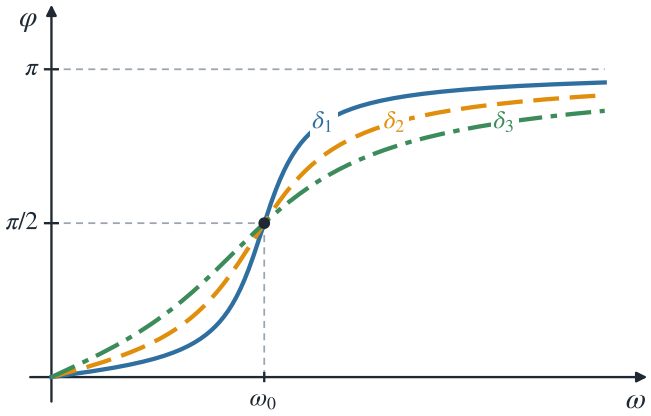


Figure 2.20.: Frequency response of the phase shift $\varphi(\omega)$ in a forced oscillation for the same dampings $\delta_1 < \delta_2 < \delta_3$ as in Figure 2.19. All curves pass through the point $(\omega_0, \pi/2)$ and approach the limit π as $\omega \to \infty$. Adapted from Papula (2015).

coincides with the natural angular frequency ω_0. A finite resonance amplitude does not exist. At $\omega = \omega_0$, the excitation is a solution of the homogeneous ODE. By Table 2.2, the ansatz with the factor t must then be chosen, which yields the particular solution

$$x_\mathrm{p}(t) = -\frac{F_0}{2m\omega_0}\, t \cos(\omega_0 t).$$

Its amplitude grows linearly with time beyond all bounds. This case is called a *resonance catastrophe*. A real system is destroyed by it unless damping prevents it.

The numerator $2\delta\omega$ of the phase relation vanishes, so $\tan(\varphi) = 0$ for all $\omega \neq \omega_0$. For $\omega < \omega_0$, the denominator is positive and $\varphi = 0$: the system oscillates in phase with the excitation. For $\omega > \omega_0$, the denominator is negative and $\varphi = \pi$: the system oscillates in antiphase. At $\omega = \omega_0$, the phase response jumps from 0 to π. It is the limit of the curves in Figure 2.20 as $\delta \to 0$.

Strong damping ($\delta > \omega_0$). We write the radicand in the denominator of Equation 2.26 as a polynomial in ω^2,

$$\Psi(\omega) = \omega^4 + 2\left(2\delta^2 - \omega_0^2\right)\omega^2 + \omega_0^4.$$

Since $\delta > \omega_0$, we have $2\delta^2 - \omega_0^2 > \delta^2 > 0$. All coefficients are positive, so Ψ increases strictly monotonically for $\omega > 0$ and the amplitude $A(\omega)$ decreases strictly monotonically. The resonance curve attains its maximum in the static case, $A(0) = x_\mathrm{st}$, and there is no resonance: $\omega_\mathrm{r} = \sqrt{\omega_0^2 - 2\delta^2}$ is not real. This already holds for $\delta \geq \omega_0/\sqrt{2}$. At the natural angular frequency, the amplitude is

$$A(\omega_0) = \frac{F_0}{2m\delta\omega_0} = \frac{\omega_0}{2\delta}\, x_\mathrm{st} < \frac{x_\mathrm{st}}{2},$$

and it tends to zero as $\omega \to \infty$.

As in the case of weak damping, the phase response runs from $\varphi(0) = 0$ via $\varphi(\omega_0) = \pi/2$ to $\varphi(\infty) = \pi$, but it rises noticeably already at small angular frequencies. At $\omega = \omega_0/2$,

$$\tan(\varphi) = \frac{4\delta}{3\omega_0}.$$

For $\delta > \omega_0$, it follows that $\varphi > \arctan(4/3) \approx 53°$, whereas the weak damping $\delta = 0.1\,\omega_0$ reaches only $\varphi \approx 7.6°$. Under strong damping, the transition from 0 to π is therefore spread over a wide frequency range. In Figure 2.20, this tendency is already visible in the curve with the largest damping δ_3. □

2.8. Simple Pendulum

Definition 2.13 (Simple Pendulum)**.** A *simple pendulum* is a physical model consisting of a mass m at the end of a massless, inextensible thread of length l. Its position is described by the angle of deflection $\varphi(t)$ from the vertical rest position. □

2.8.1. Equation of Motion

If the pendulum is displaced and released, it oscillates under the influence of gravity. The motion is described by an ODE based on Newton's second law and on the fact that the restoring force is proportional to the sine of the angle of deflection (Figure 2.21). With the weight force $F_G = mg$ and the gravitational acceleration g, the restoring force F_R is

$$F_R = F_G \sin(\varphi) = mg \sin(\varphi).$$

With the arc length $s(t)$, the equation of motion reads

$$F = m\ddot{s} = -mg \sin \varphi(t).$$

Because

$$\varphi(t) = \frac{s(t)}{l}$$

it can be formulated in terms of $s(t)$ alone,

$$F = m\ddot{s} = -mg \sin\left(\frac{s}{l}\right),$$

or in terms of the angle of deflection $\varphi(t)$ alone:

$$F = ml\ddot{\varphi} = -mg \sin(\varphi). \tag{2.28}$$

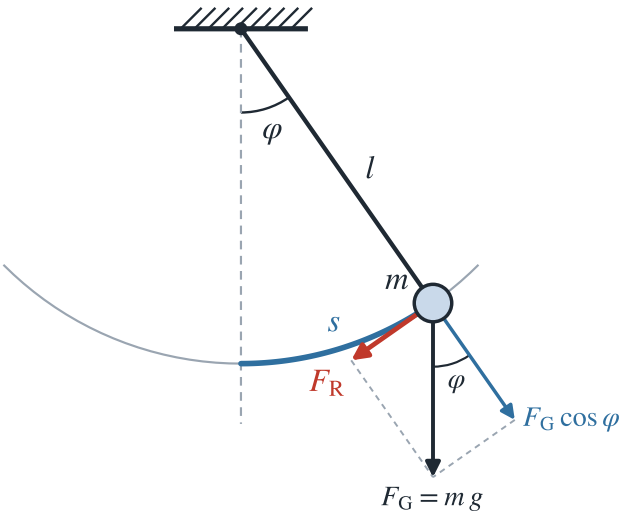


Figure 2.21.: Simple pendulum: mass m on a massless thread of length l, displaced from the rest position (dashed) by the angle φ. The arc length on the circular path with radius l is $s = l\varphi$. The weight force $F_G = mg$ is decomposed into the tangential restoring force $F_R = F_G \sin \varphi$ and the component $F_G \cos \varphi$ along the thread.

The simple pendulum sketched in Figure 2.21 oscillates almost harmonically for small displacements. Rearranging Equation 2.28 yields the ODE

$$\ddot{\varphi} + \frac{g}{l} \sin(\varphi) = 0. \tag{2.29}$$

In simplified form we obtain the representation in Figure 2.22.

For small displacements φ, $\sin(\varphi) \approx \varphi$, so that Equation 2.29 approximately takes the form

$$\ddot{\varphi} + \frac{g}{l}\varphi = 0 \tag{2.30}$$

With $\omega_0^2 = g/l$, the ODE reads

$$\ddot{\varphi} + \omega_0^2 \varphi = 0. \tag{2.31}$$

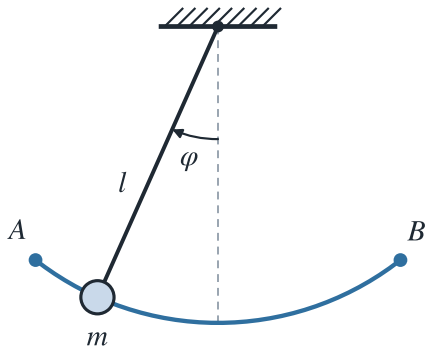


Figure 2.22.: Simple pendulum of length l with the mass m and the angle of deflection φ. The mass oscillates on a circular arc between the turning points A and B. Adapted from Papula (2015).

2.8.2. The General Solution of the Equation of Motion

The characteristic equation

$$\lambda^2 + \omega_0^2 = 0$$

has the solutions

$$\lambda_{1,2} = \pm \mathrm{j}\omega_0.$$

The general solution of Equation 2.31 is

$$\varphi(t) = C_1 \sin(\omega_0 t) + C_2 \cos(\omega_0 t)$$

with the time derivative

$$\dot{\varphi}(t) = \omega_0 \left[C_1 \cos(\omega_0 t) - C_2 \sin(\omega_0 t)\right].$$

The simple pendulum oscillates with the angular frequency

$$\omega_0 = \sqrt{\frac{g}{l}},$$

the frequency

$$\nu_0 = \frac{\omega_0}{2\pi} = \frac{1}{2\pi}\sqrt{\frac{g}{l}}$$

and the period

$$T_0 = 2\pi\sqrt{\frac{l}{g}}.$$

Example 2.46 (Simple Pendulum: Motion Starting from Rest)**.** For the initial conditions $\varphi(0) = \varphi_0$ and $\dot{\varphi}(0) = 0$, the pendulum starts from rest. Since $\varphi(0) = C_2 = \varphi_0$ and $\dot{\varphi}(0) = \omega_0 C_1 = 0$, i.e., $C_1 = 0$, the particular solution is

$$\varphi(t) = \varphi_0 \cos(\omega_0 t), \quad t \geq 0.$$ □

A pendulum that oscillates only at small angles is a rather uninteresting pendulum. Unfortunately, for the pendulum equation Equation 2.29 at large angles there is no explicit solution that is algebraically as simple. Since this system is much simpler than most engineering systems and still has no simple analytical solution, the need for numerical solution methods for ODEs becomes clear.

2.9. Reduction of Order

We consider a vector $S(t)$ that represents the state of a system as a function of time t. In general, the state comprises all dependent variables that are relevant to the behavior of the system. For an initial value problem of the ODE of order n

$$y^{(n)}(t) = f\left(t, y(t), y'(t), y''(t), \ldots, y^{(n-1)}(t)\right)$$

we define the state as

$$S(t) = \begin{bmatrix} y(t) \\ y'(t) \\ y''(t) \\ \vdots \\ y^{(n-1)}(t) \end{bmatrix}.$$

The derivative of the state is then

$$\frac{\mathrm{d}S(t)}{\mathrm{d}t} = \begin{bmatrix} y'(t) \\ y''(t) \\ \vdots \\ y^{(n-1)}(t) \\ y^{(n)}(t) \end{bmatrix} = \begin{bmatrix} y'(t) \\ y''(t) \\ \vdots \\ y^{(n-1)}(t) \\ f\left(t, y(t), y'(t), \dots, y^{(n-1)}(t)\right) \end{bmatrix} = \begin{bmatrix} S_2(t) \\ S_3(t) \\ \vdots \\ S_n(t) \\ f\left(t, S_1(t), S_2(t), \dots, S_n(t)\right) \end{bmatrix},$$

where $S_i(t)$ is the i-th component of $S(t)$. In this description, $\frac{\mathrm{d}S(t)}{\mathrm{d}t}$ can be expressed in terms of $S(t)$ alone, without $y(t)$ and its derivatives. If f now denotes the vector-valued right-hand side, then

$$\frac{\mathrm{d}S(t)}{\mathrm{d}t} = f(t, S(t)).$$

This equation is a first-order ODE in S: we have converted an ODE of order n into n coupled first-order ODEs.

Example 2.47 (Coupled Differential Equations)**.** A very simple model to describe the change in the populations of rabbits, $r(t)$, and wolves, $w(t)$, could be

$$\frac{\mathrm{d}r(t)}{\mathrm{d}t} = 4r(t) - 2w(t) \tag{2.32}$$

and

$$\frac{\mathrm{d}w(t)}{\mathrm{d}t} = r(t) + w(t). \tag{2.33}$$

Equation 2.32 states that at each time step the rabbit population quadruples, but each wolf eats two of the rabbits. Equation 2.33 states that at each time step the population of wolves increases by the number of rabbits and wolves in the system. We write this system of differential equations as an equivalent differential equation in $S(t)$.

To do so, we combine the two populations in the state vector

$$S(t) = \begin{bmatrix} r(t) \\ w(t) \end{bmatrix}.$$

We form its derivative componentwise and substitute Equation 2.32 and Equation 2.33:

$$\frac{\mathrm{d}S(t)}{\mathrm{d}t} = \begin{bmatrix} \frac{\mathrm{d}r(t)}{\mathrm{d}t} \\ \frac{\mathrm{d}w(t)}{\mathrm{d}t} \end{bmatrix} = \begin{bmatrix} 4r(t) - 2w(t) \\ r(t) + w(t) \end{bmatrix}.$$

The right-hand side is linear in $r(t)$ and $w(t)$ in both components. We thus write it as the product of a coefficient matrix and $S(t)$: the first row contains the coefficients 4 and -2 of Equation 2.32, the second row the coefficients 1 and 1 of Equation 2.33. The first-order differential equation

$$\frac{\mathrm{d}S(t)}{\mathrm{d}t} = \begin{bmatrix} 4 & -2 \\ 1 & 1 \end{bmatrix} S(t) \tag{2.34}$$

is therefore equivalent to the pair of differential equations. Multiplying out the right-hand side recovers Equation 2.32 and Equation 2.33. □

Equation 2.34 is a coupled ODE.

Definition 2.14 (Coupled ODE)**.** A system of ODEs in which one or more unknowns appear in more than one equation is called *coupled*. The equations then depend on each other. □

Many numerical methods for initial value problems are designed for first-order ODEs. To use them for higher-order ODEs, their order is reduced to one. To this end, new variables are introduced for the derivatives of the original function up to the $(n-1)$-th order, which converts an ODE of order n into a system of n first-order ODEs.

Theorem 2.14 (Reduction of Order)**.** *Consider an ODE of order n in the form*

$$y^{(n)} = f\left(t, y, y', y'', \dots, y^{(n-1)}\right).$$

With the n new variables

$$\begin{aligned} y_1 &= y, \\ y_2 &= y', \\ y_3 &= y'', \\ &\vdots \\ y_n &= y^{(n-1)}, \end{aligned}$$

each of which corresponds to a derivative of the original function y, it is equivalent to the system of n first-order ODEs consisting of the $n-1$ equations

$$\begin{aligned} y_1' &= y_2, \\ y_2' &= y_3, \\ &\vdots \\ y_{n-1}' &= y_n \end{aligned}$$

and the equation

$$y_n' = f(t, y_1, y_2, y_3, \dots, y_n) \tag{2.35}$$

□

Proof. The first $n-1$ equations follow from the definition of the y_i. The last equation is the original ODE expressed in terms of the y_i. □

The system contains only the y_i and can be solved with the usual numerical methods for systems of first-order ODEs, e.g., with the Runge-Kutta methods.

In vector form, the variables and their derivatives with respect to time t are collected into

$$S = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \quad \text{and} \quad S' = \begin{pmatrix} y_1' \\ y_2' \\ \vdots \\ y_n' \end{pmatrix}$$

A linear system can then be written in matrix form as

$$S' = AS + g(t),$$

where A is an $n \times n$ matrix of the coefficients and the vector function $g(t)$ describes external influences or inhomogeneous terms. If the right-hand side in Equation 2.35 is linear, $f = a_1 y_1 + a_2 y_2 + \cdots + a_n y_n$, and no external influences act, then $g(t) = 0$ is the zero vector. The matrix A then has the form

$$A = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ a_1 & a_2 & a_3 & \cdots & a_n \end{pmatrix}.$$

Example 2.48 (Reduction of Order)**.** Consider the second-order ODE

$$y'' = -\omega_0^2 y.$$

With the new variables

$$\begin{aligned} y_1 &= y, \\ y_2 &= y' \end{aligned}$$

we obtain, since the derivative of y_1 equals y_2, the system

$$\begin{aligned} y_1' &= y_2, \\ y_2' &= -\omega_0^2 y_1. \end{aligned}$$

It consists of two first-order ODEs that are equivalent to the original second-order ODE and can be solved with numerical methods. With

$$S = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}, \quad S' = \begin{pmatrix} y_1' \\ y_2' \end{pmatrix} = \begin{pmatrix} y_2 \\ -\omega_0^2 y_1 \end{pmatrix}, \quad A = \begin{pmatrix} 0 & 1 \\ -\omega_0^2 & 0 \end{pmatrix} \quad \text{and} \quad g(t) = 0$$

it reads in matrix form

$$\begin{pmatrix} y_1' \\ y_2' \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -\omega_0^2 & 0 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}.$$

□

Example 2.49 (Reduction of Order of the Differential Equation of the Simple Pendulum)**.** The second-order ODE of the simple pendulum Equation 2.30,

$$\ddot{\varphi} + \frac{g}{l}\varphi = 0,$$

reads, after rearranging,

$$\ddot{\varphi} = -\frac{g}{l}\varphi.$$

We set

$$y_1 = \varphi, \quad y_2 = \dot{\varphi}. \tag{2.36}$$

By Equation 2.36, differentiating y_1 yields $y_1' = y_2$. We thus obtain the first-order system

$$\begin{aligned} y_1' &= y_2, \\ y_2' &= -\frac{g}{l} y_1 \end{aligned}$$

or, in matrix form with $S = (y_1, y_2)^T$,

$$\frac{\mathrm{d}S(t)}{\mathrm{d}t} = \begin{pmatrix} 0 & 1 \\ -\frac{g}{l} & 0 \end{pmatrix} S(t). \tag{2.37}$$

□

Although the reduction of order leads to a system with several variables, all derivatives are still taken with respect to the same independent variable, e.g., t. Hence we again obtain a system of ordinary differential equations.

3. Numerical Integration of a Differential Equation

Chapter 6 (“Numerical Integration of a Differential Equation”) in Part IV of Papula (2015) gives an introduction to the numerical integration of ODEs. The symbols used in this chapter are listed in Table 3.1.

Table 3.1.: Symbols in Chapter 03

Symbol	Meaning	Type	Defined in
d	Differential (dx, $\frac{\mathrm{d}y}{\mathrm{d}x}$)	Operator	Ch. 02
e	Euler's number	Constant	Example 2.1
f	Function; right-hand side of the ODE $y' = f(x, y)$ or $u' = f(t, u)$	Function	Definition 3.1, Equation 3.6
g	Gravitational acceleration	Constant	Equation 2.30
h	Step size of an integration scheme; grid spacing $h = 1/(n + 1)$	Scalar	Definition 3.1
i	Running index (component, row, eigenvalue number, spring number)	Index	Ch. 02
k	Running index (time/iteration step, mode, eigenvalue number)	Index	Definition 3.1
$k_1, \ldots, k_4$	Increments of the Runge-Kutta method, $k_i = h\, s_i$	Scalar	Definition 3.7, Definition 3.8
l	Length of the simple pendulum	Constant	Equation 2.30
n	Order of an ODE; dimension of a matrix or system; number of steps	Index	Ch. 02, Definition 3.1
p	Order of consistency, $O(h^p)$	Index	Definition 3.4
$s_1, \ldots, s_4; \bar{s}$	Slope values of the Runge-Kutta method of order 4; mean slope	Scalar	Figure 3.7
t	Time	Scalar	Ch. 02
t_k	k-th time point of the grid; t_0 initial time, t_f final time	Scalar	Example 3.2
x	Independent variable (Ch. 02 §1–4, 03, 06, 08 eigenfunctions)	Scalar	Ch. 02
x_k, y_k	Grid point $x_k = x_0 + kh$ and approximation $y_k \approx y(x_k)$	Scalar	Definition 3.1
y	Unknown function $y(x)$ or vector of unknown functions; $y', y'', y^{(n)}$ derivatives	Function	Ch. 02
y_0	Initial value $y(x_0)$	Scalar	Ch. 02

Symbol	Meaning	Type	Defined in
$y_1, \dots, y_n$	Components of the state vector S in the reduction of order ($y_1 = y$, $y_2 = y'$, ...) or of the solution vector y of a system	Function	Theorem 2.14
$y_1(k), y_2(k)$	Approximation of the components y_1, y_2 of the state vector in step k	Scalar	Example 3.2
$O(h^p)$	Landau symbol (order of magnitude of the error)	Operator	Definition 3.4
P_0, P_k, Q, R, S	Points in the geometric interpretation of the Euler and Runge-Kutta methods	Point	Definition 3.1, Figure 3.7
S	State vector $S(t)$; S_i its i-th component, S_0 initial state	Vector	Ch. 02, Equation 3.4
γ_i	Coefficient of a linear combination (weights of the RK2 method; expansion of the starting vector in eigenvectors; basis representation in the eigenspace)	Scalar	Equation 3.7
Δy_k	Increment in step k	Scalar	Table 3.4
η_1, η_2	Node parameters of the RK2 ansatz (fractions of h in the time and state directions)	Scalar	Equation 3.7
$\varphi(t)$	Angle of deflection of the simple pendulum; φ_0 initial deflection	Function	Equation 2.30
$y', y'', y^{(n)}$	First, second, n-th derivative with respect to the independent variable	Operator	Ch. 02
$\frac{\partial}{\partial t}$	Partial derivative	Operator	Ch. 02
A^{-1}	Inverse	Operator	Example 3.2
local	Line elements a, b, c, d, straight line g, and coordinates $x_Q, y_Q, x_R, y_R, x_S, y_S$ of the construction in Figure 3.7; intermediate value $y_{h/2}$ of the midpoint method	local	Figure 3.7, Table 3.4

3.1. Explicit Euler Method (Euler's Polygon Method)

Definition 3.1 (Explicit Euler Method (Euler's Polygon Method))**.** Let a first-order ODE

$$y' = f(x, y)$$

with the initial point $P_0 = (x_0, y_0)$ be given. The *explicit Euler method* (*Euler's polygon method*) computes the solution curve $y = y(x)$ point by point at the positions $x_k = x_0 + kh$ with the step size h according to the iteration rule

$$y(x_k) \approx y_k = y_{k-1} + hf(x_{k-1}, y_{k-1}) \quad (k = 1, 2, \dots, n), \tag{3.1}$$

see Figure 3.1. □

For practical computation, the computational scheme in Table 3.2 is used.

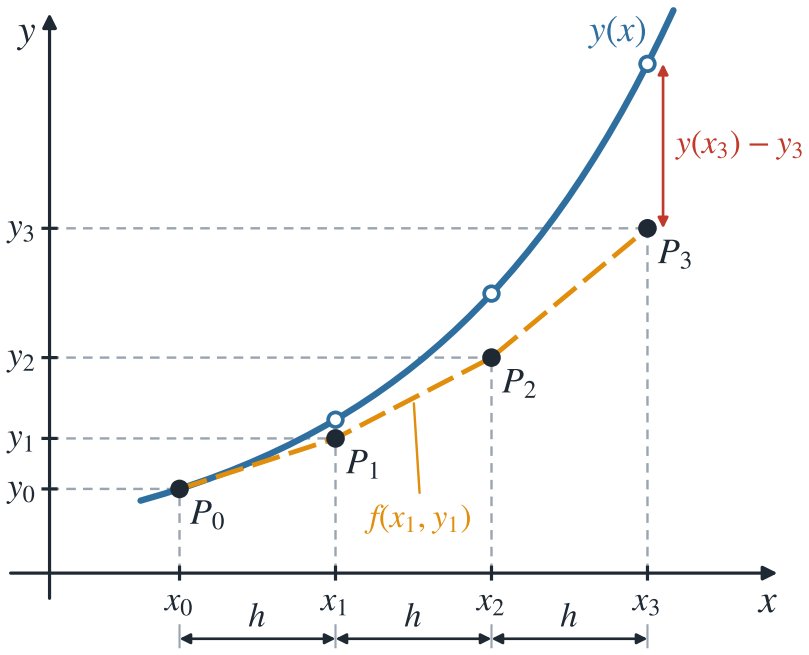


Figure 3.1.: Euler's polygon method, computed for $y' = y$ with $h = 0.6$: the exact solution curve $y(x)$ (blue, solid) and the Euler polygon (orange, dashed) through the points $P_k = (x_k, y_k)$ with $x_k = x_0 + kh$. Each segment has the slope $f(x_{k-1}, y_{k-1})$ of its left endpoint. The open circles mark the exact values $y(x_k)$, the red arrow marks the deviation $y(x_3) - y_3$ after three steps. Adapted from Papula (2015).

Table 3.2.: Computational scheme for Euler's method

k	x	y	$h \times f(x, y)$
0	x_0	y_0	$h \times f(x_0, y_0)$
1	$x_1 = x_0 + h$	$y_1 = y_0 + h \times f(x_0, y_0)$	$h \times f(x_1, y_1)$
2	$x_2 = x_0 + 2h$	$y_2 = y_1 + h \times f(x_1, y_1)$	$h \times f(x_2, y_2)$
⋮	⋮	⋮	⋮
n	$x_n = x_0 + nh$	$y_n = y_{n-1} + h \times f(x_{n-1}, y_{n-1})$	$h \times f(x_n, y_n)$

Example 3.1 (Euler's Method)**.** We solve the ODE

$$y' = \frac{\mathrm{d}y}{\mathrm{d}x} = \mathrm{e}^{-x}$$

with the initial condition $y(0) = -1$ numerically using Euler's method and compare the result with the exact solution $y(x) = -\mathrm{e}^{-x}$.

We use the step size $h = 0.1$ and compute the approximate solution at the points $x_k = 0.1\,k$ for $k = 0, 1, \ldots, 10$, that is, for $x = 0,\ 0.1,\ 0.2,\ \ldots,\ 1$. The right-hand side $f(x, y) = \mathrm{e}^{-x}$ depends on x only. The iteration rule Equation 3.1 therefore reads

$$y_k = y_{k-1} + h\,\mathrm{e}^{-x_{k-1}} \quad (k = 1, 2, \ldots, 10).$$

We start with $x_0 = 0$ and $y_0 = -1$:

1. $k = 1$: $hf(x_0, y_0) = 0.1 \cdot \mathrm{e}^{0} = 0.1$, so $y_1 = -1 + 0.1 = -0.9$.
2. $k = 2$: $hf(x_1, y_1) = 0.1 \cdot \mathrm{e}^{-0.1} = 0.090484$, so $y_2 = -0.9 + 0.090484 = -0.809516$.
3. $k = 3$: $hf(x_2, y_2) = 0.1 \cdot \mathrm{e}^{-0.2} = 0.081873$, so $y_3 = -0.809516 + 0.081873 = -0.727643$.

The remaining steps proceed in the same way. Table 3.3 contains the complete computational scheme from Table 3.2, extended by the exact values $y(x_k) = -\mathrm{e}^{-x_k}$ and the error $y_k - y(x_k)$.

Table 3.3.: Euler's method for $y' = \mathrm{e}^{-x}$, $y(0) = -1$, with $h = 0.1$

k	x_k	y_k	$h \times f(x_k, y_k)$	$y(x_k)$	$y_k - y(x_k)$
0	0	−1	0.1	−1	0
1	0.1	−0.9	0.090484	−0.904837	0.004837
2	0.2	−0.809516	0.081873	−0.818731	0.009214
3	0.3	−0.727643	0.074082	−0.740818	0.013175
4	0.4	−0.653561	0.067032	−0.670320	0.016759
5	0.5	−0.586529	0.060653	−0.606531	0.020001
6	0.6	−0.525876	0.054881	−0.548812	0.022935
7	0.7	−0.470995	0.049659	−0.496585	0.025590
8	0.8	−0.421337	0.044933	−0.449329	0.027992
9	0.9	−0.376404	0.040657	−0.406570	0.030166
10	1.0	−0.335747	0.036788	−0.367879	0.032133

The approximation always lies above the exact solution, and the error grows from step to step. The reason is that the exact solution is concave because $y'' = -\mathrm{e}^{-x} < 0$, and each segment of Euler's method takes the slope at its left endpoint, which exceeds the mean slope over the interval. At $x = 1$ the error is 0.032, roughly $8.7\,\%$ of the exact value -0.367879. □

```
import numpy as np

ydot = lambda x, y: np.exp(-x) # ODE
h = 0.1 # step size
x = np.arange(0, 1 + h, h) # computational grid
y0 = -1 # initial value
n = len(x) # number of steps

# Explicit Euler method
y = np.zeros(n)
y[0] = y0
for k in range(0, n - 1):
    y[k + 1] = y[k] + h * ydot(x[k], y[k])
    print(f" k: {k:<2} x: {x[k]:<5.6f} y: {y[k]:<5.6f} h*f: {h*ydot(x[k], y[k]):<5.6f}")
k = n - 1
print(f" k: {k:<2} x: {x[k]:<5.6f} y: {y[k]:<5.6f} h*f: {h*ydot(x[k], y[k]):<5.6f}")
```

```
 k: 0  x: 0.000000 y: -1.000000 h*f: 0.100000
 k: 1  x: 0.100000 y: -0.900000 h*f: 0.090484
 k: 2  x: 0.200000 y: -0.809516 h*f: 0.081873
 k: 3  x: 0.300000 y: -0.727643 h*f: 0.074082
 k: 4  x: 0.400000 y: -0.653561 h*f: 0.067032
 k: 5  x: 0.500000 y: -0.586529 h*f: 0.060653
 k: 6  x: 0.600000 y: -0.525876 h*f: 0.054881
 k: 7  x: 0.700000 y: -0.470995 h*f: 0.049659
 k: 8  x: 0.800000 y: -0.421337 h*f: 0.044933
 k: 9  x: 0.900000 y: -0.376404 h*f: 0.040657
 k: 10 x: 1.000000 y: -0.335747 h*f: 0.036788
```

```
import matplotlib.pyplot as plt
%matplotlib inline
plt.figure(figsize = (9, 6))
plt.plot(x, y, 'bo--', label='Approximation')
plt.plot(x, -np.exp(-x), 'g', label='Exact')
plt.xlabel('x')
plt.ylabel('y(x)')
plt.grid()
plt.legend(loc='lower right')
plt.show()
```

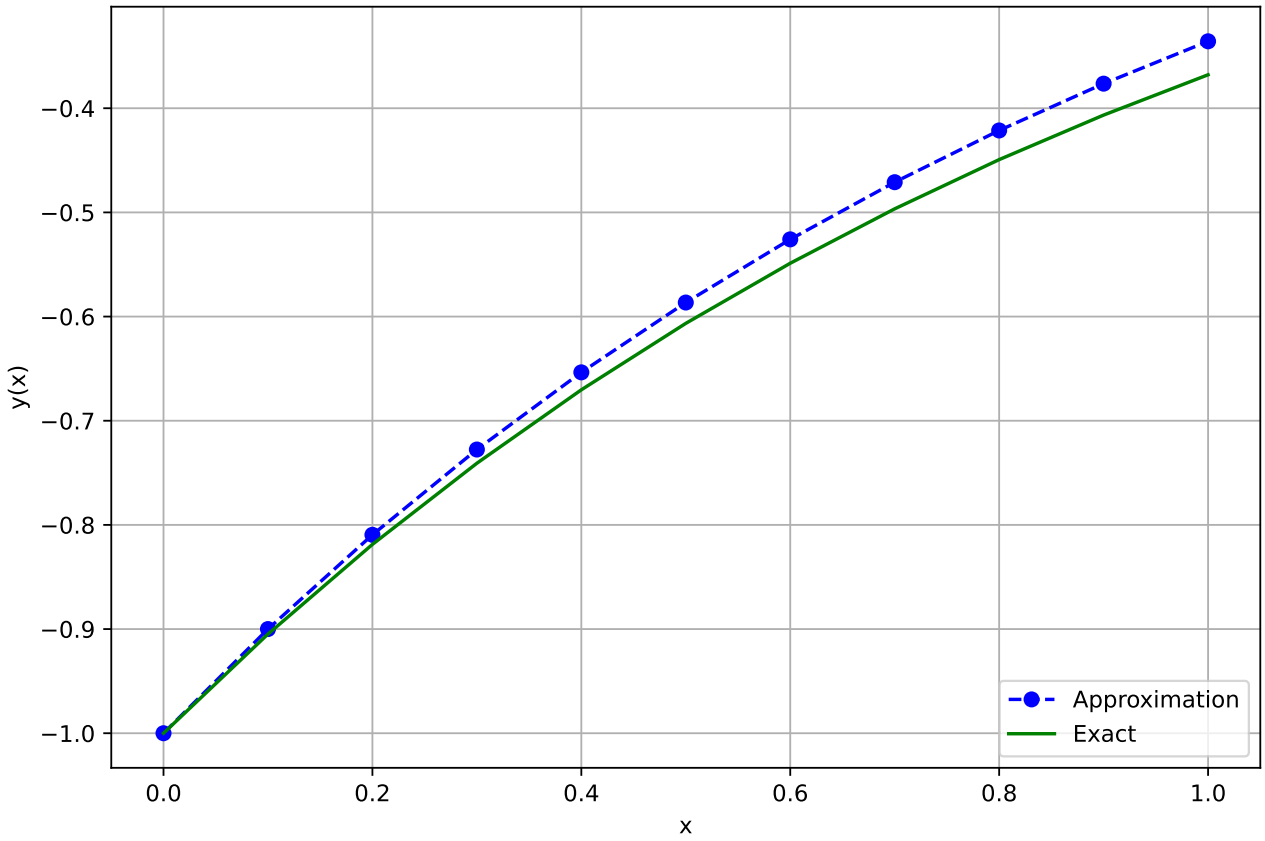


Figure 3.2.: Approximate solution of the differential equation with Euler's method. Step size $h = 0.1$.

In Figure 3.2 each point is a linear extrapolation of the previous point. Starting from the initial value, we thus obtain an approximation of the solution on the numerical grid. With $h = 0.01$ the approximation improves, see Figure 3.3.

```
import numpy as np

ydot = lambda x, y: np.exp(-x) # ODE
h = 0.01 # step size
x = np.arange(0, 1 + h, h) # computational grid
y0 = -1 # initial value
n = len(x) # number of steps

# Explicit Euler method
y = np.zeros(n)
y[0] = y0
for k in range(0, n - 1):
    y[k + 1] = y[k] + h * ydot(x[k], y[k])

import matplotlib.pyplot as plt
%matplotlib inline
plt.figure(figsize = (9, 6))
plt.plot(x, y, '--', label='Approximation')
plt.plot(x, -np.exp(-x), 'g', label='Exact')
plt.xlabel('x')
plt.ylabel('y(x)')
plt.grid()
plt.legend(loc='lower right')
plt.show()
```

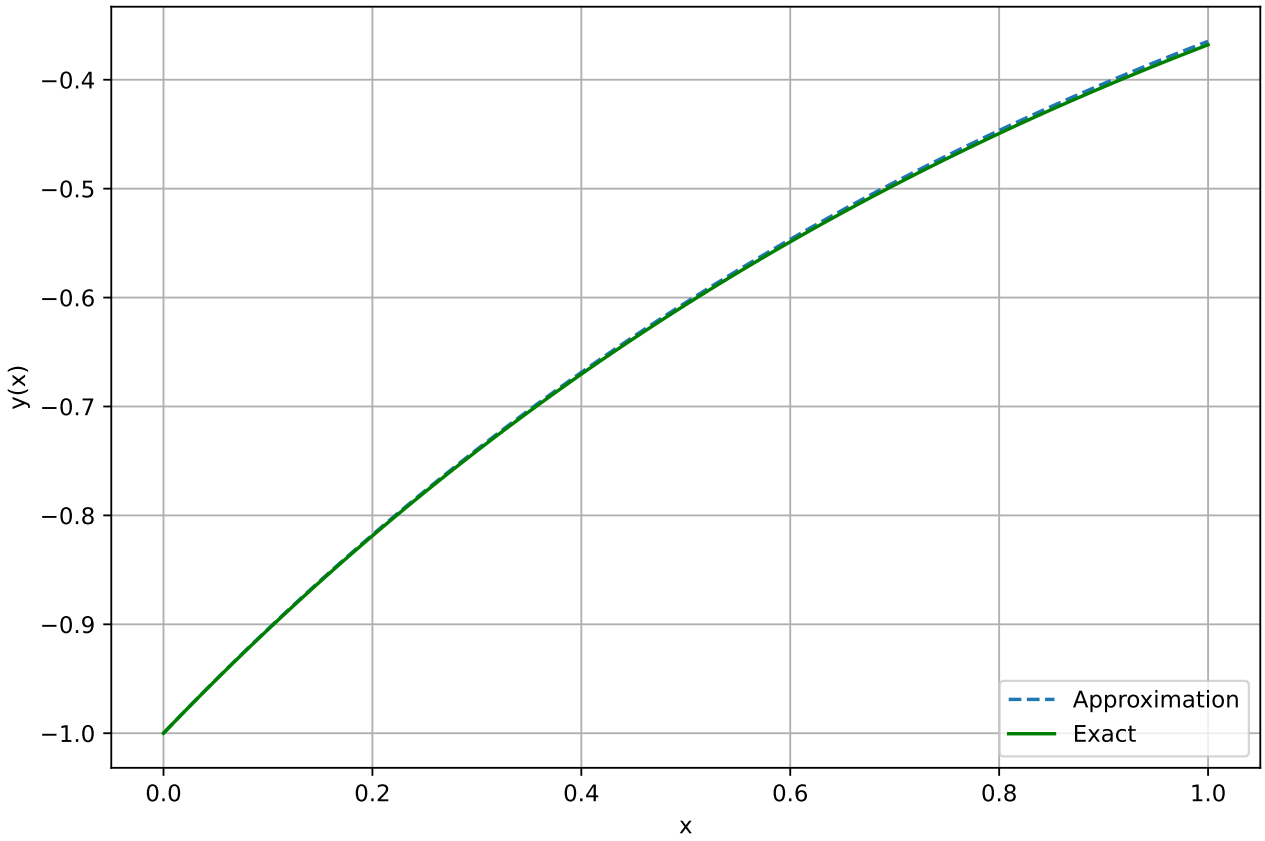


Figure 3.3.: Approximate solution of the differential equation with Euler's method. Step size $h = 0.01$.

The method is called explicit because y_{k+1} is computed solely from the already known values x_k and y_k.

3.2. Implicit Euler Method

The implicit Euler method differs from the explicit (forward) Euler method in that the function f is evaluated at the end point of the step rather than at the starting point.

Definition 3.2 (Implicit Euler Method)**.** Let the initial value problem

$$y' = f(x, y), \quad y(x_0) = y_0,$$

be given with a step size $h > 0$ and the nodes $x_k = x_0 + kh$, $k = 0, 1, ...$ The *implicit Euler method* computes the approximations y_k according to the iteration rule

$$y_{k+1} = y_k + hf(x_{k+1}, y_{k+1}), \quad k = 0, 1, ... \tag{3.2}$$

□

The value $f(x_{k+1}, y_{k+1})$ is not given explicitly, because y_{k+1} appears on both sides of the equation. The equation therefore has to be solved for y_{k+1} in every step. The explicit method does not need this computation, because its rule is already solved for y_{k+1}. The values y_k approximate the values $y(x_k)$ of the exact solution of the initial value problem. The smaller the step size h, the higher the computational cost and the better the approximation.

3.3. Trapezoidal Method

Before showing how to solve the implicit Euler formula, we introduce another implicit formula, the trapezoidal formula. It is the average of the explicit and the implicit Euler formulas.

Definition 3.3 (Trapezoidal Method)**.** The *trapezoidal method* computes the next point in each case according to the iteration rule

$$y_{k+1} = y_k + \frac{h}{2}\left(f(x_k, y_k) + f(x_{k+1}, y_{k+1})\right). \tag{3.3}$$

□

The method can be read as a predictor-corrector scheme. First, the explicit Euler method yields a preliminary value for y_{k+1} (predictor step). This value is used to compute the slope at the end of the interval. The average of the slopes at the beginning and at the end of the interval then determines the final value y_{k+1}.

3.4. Simple Pendulum: Euler Methods in Matrix Form

To show how the implicit schemes (Equation 3.2 and Equation 3.3) are solved, we consider the pendulum equation, which was reduced to a first-order system in Equation 2.37. With the state vector $S(t) = (y_1(t), y_2(t))^T = (\varphi(t), \dot{\varphi}(t))^T$ it reads

$$\frac{\mathrm{d}S(t)}{\mathrm{d}t} = f(t, S(t)) = \begin{pmatrix} 0 & 1 \\ -\frac{g}{l} & 0 \end{pmatrix} S(t). \tag{3.4}$$

Example 3.2 (Euler and Trapezoidal Methods for the Simple Pendulum)**.** We consider the time points $t_k = t_0 + kh$ and write $y_1(k)$ and $y_2(k)$ for the approximations of the components $y_1(t_k)$ and $y_2(t_k)$ in step k. Substituting Equation 3.4 into the explicit Euler formula Equation 3.1

$$y_{k+1} = y_k + hf(x_k, y_k)$$

we obtain

$$\begin{pmatrix} y_1(k+1) \\ y_2(k+1) \end{pmatrix} = \begin{pmatrix} y_1(k) \\ y_2(k) \end{pmatrix} + h \begin{pmatrix} 0 & 1 \\ -\frac{g}{l} & 0 \end{pmatrix} \begin{pmatrix} y_1(k) \\ y_2(k) \end{pmatrix} = \begin{pmatrix} 1 & h \\ -\frac{gh}{l} & 1 \end{pmatrix} \begin{pmatrix} y_1(k) \\ y_2(k) \end{pmatrix}.$$

Substituting Equation 3.4 into the implicit Euler formula Equation 3.2

$$y_{k+1} = y_k + hf(x_{k+1}, y_{k+1})$$

yields

$$\begin{pmatrix} y_1(k+1) \\ y_2(k+1) \end{pmatrix} = \begin{pmatrix} y_1(k) \\ y_2(k) \end{pmatrix} + h \begin{pmatrix} 0 & 1 \\ -\frac{g}{l} & 0 \end{pmatrix} \begin{pmatrix} y_1(k+1) \\ y_2(k+1) \end{pmatrix}$$

and hence

$$\begin{pmatrix} 1 & -h \\ \frac{gh}{l} & 1 \end{pmatrix} \begin{pmatrix} y_1(k+1) \\ y_2(k+1) \end{pmatrix} = \begin{pmatrix} y_1(k) \\ y_2(k) \end{pmatrix}.$$

Rearranging (matrix inversion) yields

$$\begin{pmatrix} y_1(k+1) \\ y_2(k+1) \end{pmatrix} = \begin{pmatrix} 1 & -h \\ \frac{gh}{l} & 1 \end{pmatrix}^{-1} \begin{pmatrix} y_1(k) \\ y_2(k) \end{pmatrix}.$$

Substituting into the trapezoidal formula Equation 3.3

$$y_{k+1} = y_k + \frac{h}{2}\left(f(x_k, y_k) + f(x_{k+1}, y_{k+1})\right)$$

we obtain

$$\begin{pmatrix} y_1(k+1) \\ y_2(k+1) \end{pmatrix} = \begin{pmatrix} y_1(k) \\ y_2(k) \end{pmatrix} + \frac{h}{2}\left\{ \begin{pmatrix} 0 & 1 \\ -\frac{g}{l} & 0 \end{pmatrix} \begin{pmatrix} y_1(k) \\ y_2(k) \end{pmatrix} + \begin{pmatrix} 0 & 1 \\ -\frac{g}{l} & 0 \end{pmatrix} \begin{pmatrix} y_1(k+1) \\ y_2(k+1) \end{pmatrix} \right\}$$

and from this

$$\begin{pmatrix} 1 & -\frac{h}{2} \\ \frac{gh}{2l} & 1 \end{pmatrix} \begin{pmatrix} y_1(k+1) \\ y_2(k+1) \end{pmatrix} = \begin{pmatrix} 1 & \frac{h}{2} \\ -\frac{gh}{2l} & 1 \end{pmatrix} \begin{pmatrix} y_1(k) \\ y_2(k) \end{pmatrix}.$$

Solving for y_{k+1} yields for the trapezoidal formula

$$\begin{pmatrix} y_1(k+1) \\ y_2(k+1) \end{pmatrix} = \begin{pmatrix} 1 & -\frac{h}{2} \\ \frac{gh}{2l} & 1 \end{pmatrix}^{-1} \begin{pmatrix} 1 & \frac{h}{2} \\ -\frac{gh}{2l} & 1 \end{pmatrix} \begin{pmatrix} y_1(k) \\ y_2(k) \end{pmatrix}.$$

□

These equations solve the initial value problem, because the next state $S(t_{k+1})$ can be computed from every state $S(t_k)$. In general, this is possible whenever the ODE is linear.

3.5. Numerical Error and Instability

For integration schemes for ODEs, two questions are central: accuracy and stability.

Definition 3.4 (Accuracy)**.** The *accuracy* of an integration scheme describes how well it approximates the (usually unknown) exact solution as a function of the step size h. It is stated in O-notation: a scheme has order p if its error behaves like $O(h^p)$ for $h \to 0$. □

Definition 3.5 (Stability)**.** The *stability* of an integration scheme is its ability to keep the error from growing while integrating forward in time. If the error does not grow, the scheme is stable; otherwise it is unstable. A scheme that is stable only for certain step sizes h is also referred to as unstable. □

To illustrate stability problems, in Example 3.3 we solve the pendulum equation numerically using the explicit Euler method, the implicit Euler method, and the trapezoidal rule.

Example 3.3 (Programming the Euler Methods)**.** We solve the pendulum equation Equation 3.4 over the time interval $[0,\ 5]$ with the step size $h = 0.1$ and the initial state

$$S_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

using the explicit Euler method, the implicit Euler method, and the trapezoidal rule. For the model parameters we use $\sqrt{g/l} = 4$ (`w` in the code). □

```
import numpy as np
from numpy.linalg import inv
import matplotlib.pyplot as plt

%matplotlib inline

# define step size
h = 0.1
# define numerical grid
t = np.arange(0, 5.1, h)
# oscillation freq. of pendulum
w = 4
s0 = np.array([[1], [0]])

m_e = np.array([[1, h],
                [-w**2*h, 1]])
m_i = inv(np.array([[1, -h],
                [w**2*h, 1]]))
m_t = np.dot(inv(np.array([[1, -h/2],
    [w**2*h/2,1]])), np.array(
      [[1,h/2], [-w**2*h/2, 1]]))

s_e = np.zeros((len(t), 2))
s_i = np.zeros((len(t), 2))
s_t = np.zeros((len(t), 2))

# do integrations
s_e[0, :] = s0.T
s_i[0, :] = s0.T
s_t[0, :] = s0.T

for j in range(0, len(t)-1):
    s_e[j+1, :] = np.dot(m_e,s_e[j, :])
    s_i[j+1, :] = np.dot(m_i,s_i[j, :])
    s_t[j+1, :] = np.dot(m_t,s_t[j, :])
```

```
plt.figure(figsize = (12, 8))
plt.plot(t,s_e[:,0],'b-')
plt.plot(t,s_i[:,0],'g:')
plt.plot(t,s_t[:,0],'r--')
plt.plot(t, np.cos(w*t), 'k')
plt.ylim([-3, 3])
plt.xlabel('t')
plt.ylabel(r'$\varphi(t)$')
plt.legend(['Explicit', 'Implicit', \
            'Trapezoidal', 'Exact'])
plt.show()
```

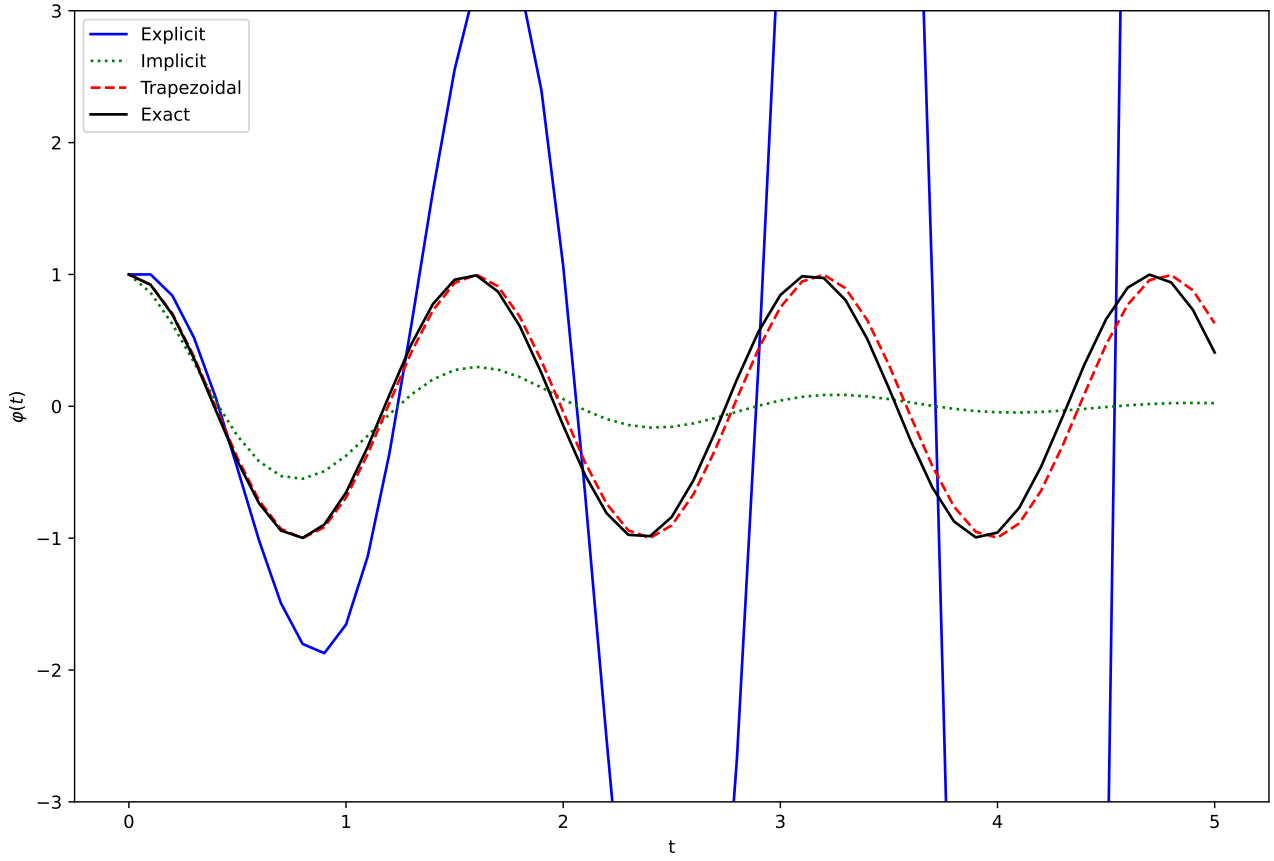


Figure 3.4.: Comparison of the numerical solutions for the angle of deflection $\varphi(t)$ of the pendulum with the explicit, implicit, and trapezoidal rules.

Figure 3.4 shows the angle φ over the time t. The exact solution is a pure cosine wave. The explicit Euler method is clearly unstable. The solution of the implicit Euler method decays exponentially, which is not correct. The trapezoidal rule captures the solution correctly, although with a small phase shift that increases with time. Figure 3.5 shows the angular velocity $\dot{\varphi}(t)$ of the pendulum.

```
# define step size
h = 0.1
# define numerical grid
t = np.arange(0, 5.1, h)
# oscillation freq. of pendulum
w = 4
s0 = np.array([[1], [0]])

m_e = np.array([[1, h],
                [-w**2*h, 1]])
m_i = inv(np.array([[1, -h],
                [w**2*h, 1]]))
```

```
m_t = np.dot(inv(np.array([[1, -h/2],
    [w**2*h/2,1]])), np.array(
      [[1,h/2], [-w**2*h/2, 1]]))

s_e = np.zeros((len(t), 2))
s_i = np.zeros((len(t), 2))
s_t = np.zeros((len(t), 2))

s_e[0, :] = s0.T
s_i[0, :] = s0.T
s_t[0, :] = s0.T


for j in range(0, len(t)-1):
    s_e[j+1, :] = np.dot(m_e,s_e[j, :])
    s_i[j+1, :] = np.dot(m_i,s_i[j, :])
    s_t[j+1, :] = np.dot(m_t,s_t[j, :])

plt.figure(figsize = (12, 8))
plt.plot(t,s_e[:,1],'b-')
plt.plot(t,s_i[:,1],'g:')
plt.plot(t,s_t[:,1],'r--')
plt.plot(t, -w*np.sin(w*t), 'k')
plt.ylim([-4*np.pi, 4*np.pi])
plt.xlabel('t')
plt.ylabel(r'$\dot{\varphi}(t)$')
plt.legend(['Explicit', 'Implicit', \
            'Trapezoidal', 'Exact'])
plt.show()
```

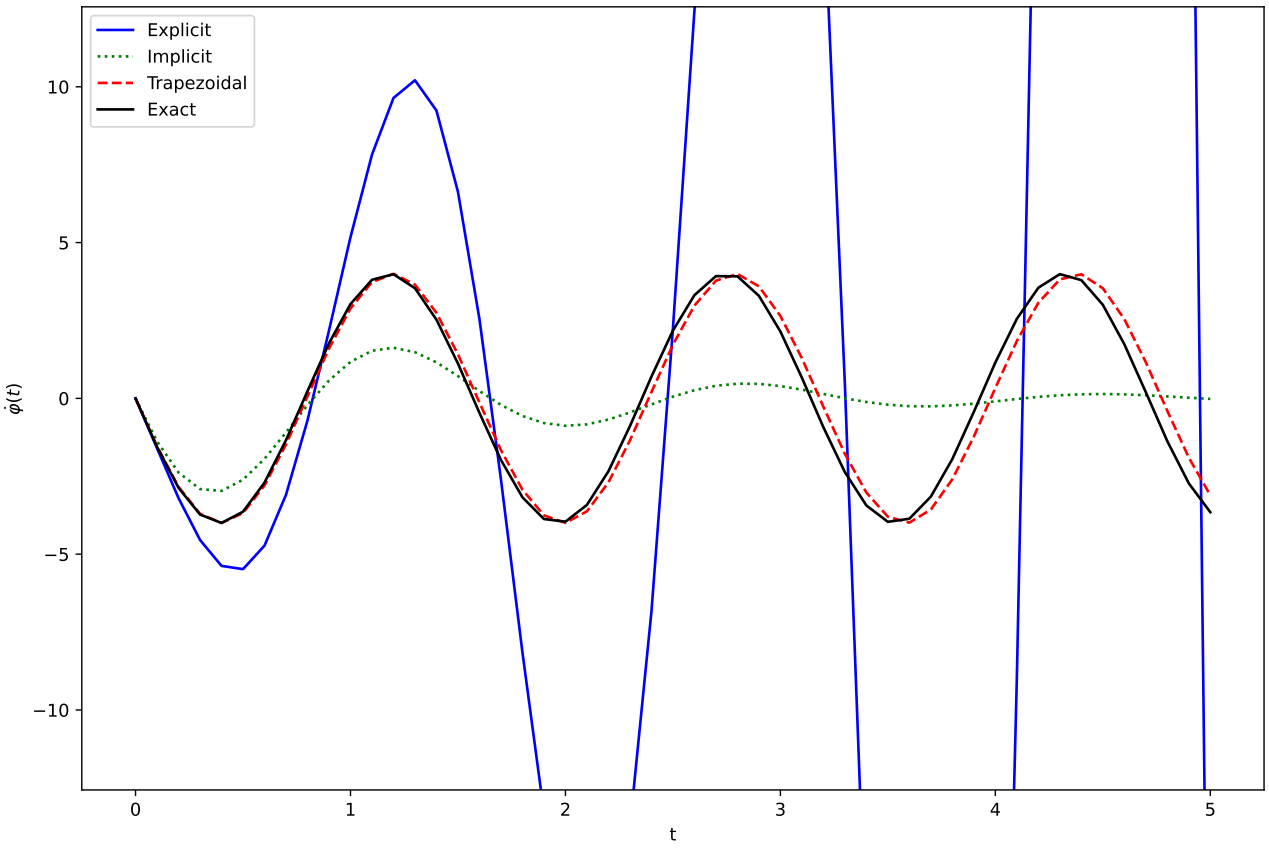


Figure 3.5.: Comparison of the numerical solutions for the angular velocity $\dot{\varphi}(t)$ of the pendulum with the explicit, implicit, and trapezoidal rules.

3.6. Runge-Kutta Methods

Runge-Kutta (RK) methods are among the most widely used methods for solving ODEs. They estimate the slope of the solution given by the ODE at several points within one time step. These estimates yield the next value without an explicit correction phase.

3.6.1. The Midpoint Method

More sophisticated methods achieve a higher order and thus more accuracy. One possibility is additional function evaluations, as the midpoint method in Definition 3.6 shows.

Definition 3.6 (The Midpoint Method)**.** The *midpoint method* computes the next point in each case according to the iteration rule

$$y_{k+1} = y_k + hf\left(x_k + \frac{1}{2}h, y_k + \frac{1}{2}hf(x_k, y_k)\right). \qquad \square$$

The midpoint method improves on the explicit Euler method by evaluating the slope not only at the starting point but also at the middle of the interval. In its first step, it uses the slope at the starting point to predict the solution value halfway between x_k and x_{k+1}:

$$y\left(x_k + \frac{h}{2}\right) = y(x_k) + \frac{h}{2}f(x_k, y(x_k)).$$

The slope at this point then determines the value at the end of the interval:

$$y(x_{k+1}) = y(x_k) + hf\left(x_k + \frac{h}{2}, y\left(x_k + \frac{h}{2}\right)\right).$$

Example 3.4 (Midpoint Method)**.** For the initial value problem

$$y' = f(x, y) = y + \mathrm{e}^x, \quad y(0) = 1,$$

with the exact solution $y(x) = (x + 1)\,\mathrm{e}^x$, the midpoint method with step size $h = 0.1$ can be carried out using the computational scheme in Table 3.4.

Table 3.4.: Computational scheme for the midpoint method

k	x	y	$y_{\frac{h}{2}} = y + \frac{h}{2}f(x, y)$	$\Delta y = h \times f(x + \frac{h}{2}, y_{\frac{h}{2}})$
0	x_0	y_0	$y_{\frac{h}{2},0} = y_0 + \frac{h}{2} \times f(x_0, y_0)$	$\Delta y_0 = h \times f(x_0 + \frac{h}{2}, y_{\frac{h}{2},0})$
1	$x_1 = x_0 + h$	$y_1 = y_0 + \Delta y_0$	$y_{\frac{h}{2},1} = y_1 + \frac{h}{2} \times f(x_1, y_1)$	$\Delta y_1 = h \times f(x_1 + \frac{h}{2}, y_{\frac{h}{2},1})$
2	$x_2 = x_1 + h$	$y_2 = y_1 + \Delta y_1$	$y_{\frac{h}{2},2} = y_2 + \frac{h}{2} \times f(x_2, y_2)$	$\Delta y_2 = h \times f(x_2 + \frac{h}{2}, y_{\frac{h}{2},2})$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
n	$x_n = x_0 + nh$	$y_n = y_{n-1} + \Delta y_{n-1}$		

$\square$

An implementation of the midpoint method for Example 3.4 in Python reads:

```
import numpy as np

ydot = lambda x, y: y + np.exp(x) # ODE
h = 0.1 # step size
h2 = h/2
x = np.arange(0, 1 + h, h) # computational grid
```

```
y0 = 1 # initial value
n = len(x) # number of steps

# Midpoint method
y = np.zeros(n)
y2 = np.zeros(n)
y[0] = y0
for k in range(0, n - 1):
    y2[k] = y[k] + h2 * ydot(x[k], y[k])
    y[k + 1] = y[k] + h * ydot(x[k] + h2, y2[k])
    print(f" k: {k:<2} x: {x[k]:<5.1f} y: {y[k]:<5.8f} y2: {y2[k]:<5.8f} h*f: {h*ydot(x[k] + h2, y2[k])
k = n - 1
print(f" k: {k:<2} x: {x[k]:<5.1f} y: {y[k]:<5.8f} y2: {y2[k]:<5.8f} h*f: {h*ydot(x[k] + h2, y2[k]):<5.
```

```
 k: 0  x: 0.0   y: 1.00000000 y2: 1.10000000 h*f: 0.21512711
 k: 1  x: 0.1   y: 1.21512711 y2: 1.33114201 h*f: 0.24929763
 k: 2  x: 0.2   y: 1.46442474 y2: 1.59871611 h*f: 0.28827415
 k: 3  x: 0.3   y: 1.75269889 y2: 1.90782677 h*f: 0.33268943
 k: 4  x: 0.4   y: 2.08538832 y2: 2.26424897 h*f: 0.38325612
 k: 5  x: 0.5   y: 2.46864444 y2: 2.67451272 h*f: 0.44077657
 k: 6  x: 0.6   y: 2.90942101 y2: 3.14599800 h*f: 0.50615388
 k: 7  x: 0.7   y: 3.41557489 y2: 3.68704127 h*f: 0.58040413
 k: 8  x: 0.8   y: 3.99597902 y2: 4.30705502 h*f: 0.66467019
 k: 9  x: 0.9   y: 4.66064921 y2: 5.01666182 h*f: 0.76023715
 k: 10 x: 1.0   y: 5.42088636 y2: 0.00000000 h*f: 0.28576511
```

```
import matplotlib.pyplot as plt
%matplotlib inline
plt.figure(figsize = (9, 6))
plt.plot(x, y, 'bo--', label='Approximation')
plt.plot(x, (x+1)*np.exp(x), 'g', label='Exact')
plt.xlabel('x')
plt.ylabel('y(x)')
plt.grid()
plt.legend(loc='lower right')
plt.show()
```

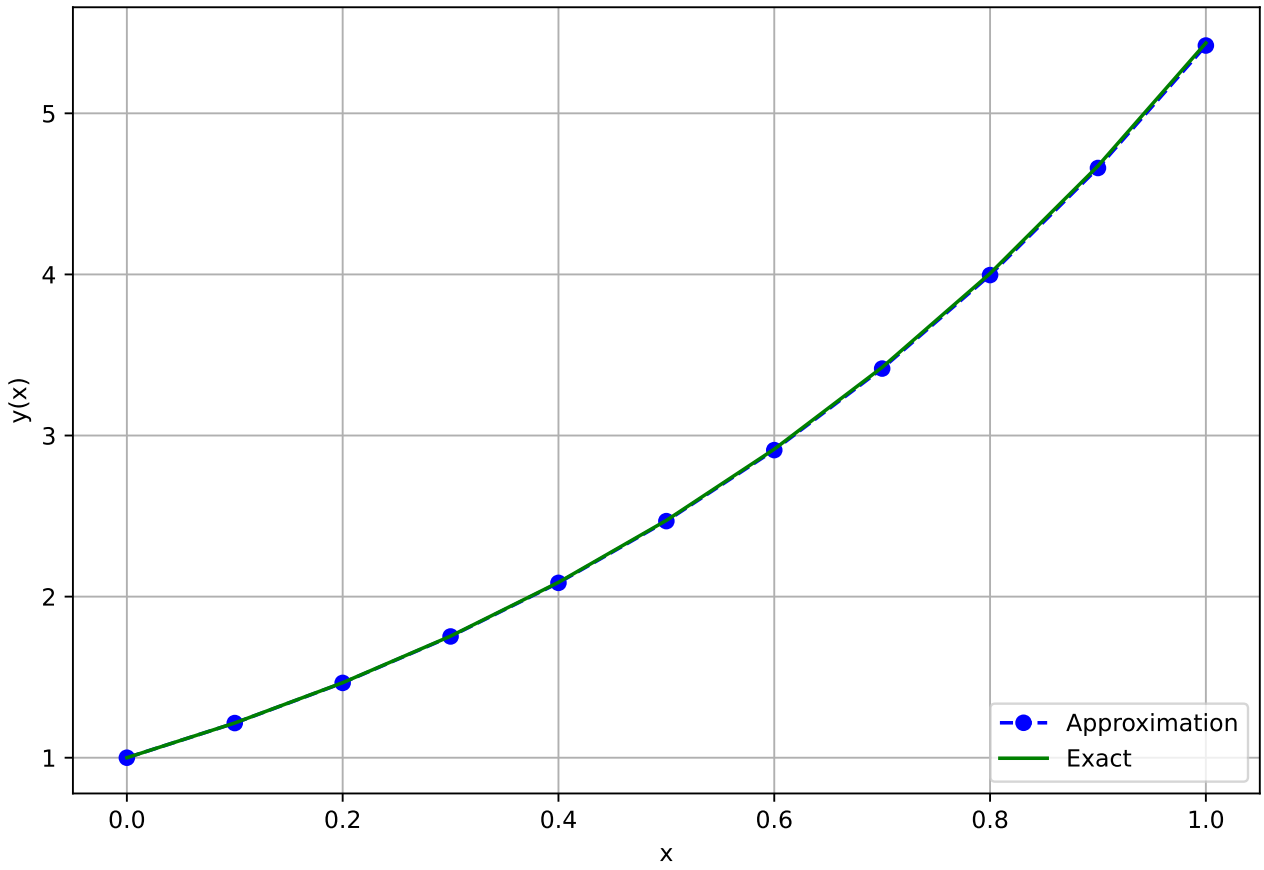


Figure 3.6.: Approximate solution of the differential equation $y' = y + \mathrm{e}^x$ with the midpoint method. Step size $h = 0.1$.

The midpoint method belongs to the family of Runge-Kutta methods; it is a second-order Runge-Kutta method.

3.6.2. Derivation of the Runge-Kutta Methods

Euler's method approximates the state S with the first two terms of the Taylor series, that is, linearly:

$$S(t_{k+1}) = S(t_k + h) = S(t_k) + h \cdot S'(t_k).$$

Further terms of the Taylor series improve the accuracy of the numerical integration considerably:

$$S(t_{k+1}) = S(t_k + h) = S(t_k) + S'(t_k)h + \frac{1}{2!}S''(t_k)h^2 + \cdots + \frac{1}{n!}S^{(n)}(t_k)h^n. \tag{3.5}$$

This would require deriving the expressions for $S''(t_k), S'''(t_k), \ldots, S^{(n)}(t_k)$. The RK methods avoid this additional work. They are based on truncated Taylor series but do not require higher derivatives.

3.6.2.1. Second-Order Runge-Kutta Method

We derive the second-order Runge-Kutta method. Let

$$\frac{\mathrm{d}S(t)}{\mathrm{d}t} = f(t, S(t)). \tag{3.6}$$

We assume an integration formula of the form

$$S(t + h) = S(t) + \gamma_1 f(t, S(t))h + \gamma_2 f(t + \eta_1 h, S(t) + \eta_2 h f(t, S(t)))h \tag{3.7}$$

that requires no derivatives of $S(t)$ beyond first order. Derivatives of second and higher order arise from combinations of $f(t, S(t))$ and $f(t + \eta_1 h, S(t) + \eta_2 h f(t, S(t)))$. We determine the parameters $\gamma_1, \gamma_2, \eta_1, \eta_2$ by comparing Equation 3.7 with the second-order Taylor series.

Proposition 3.1 (Order Conditions of the Second-Order Runge-Kutta Ansatz)**.** *For every right-hand side f, the integration formula Equation 3.7 agrees with the Taylor series Equation 3.5 up to terms of order h^3 if and only if*

$$\gamma_1 + \gamma_2 = 1, \quad \gamma_2\eta_1 = \frac{1}{2}, \quad \gamma_2\eta_2 = \frac{1}{2}. \tag{3.8}$$

□

Proof. The second-order Taylor series reads

$$S(t+h) = S(t) + S'(t)h + \frac{1}{2!}S''(t)h^2 = S(t) + f(t, S(t))h + \frac{1}{2!}f'(t, S(t))h^2. \tag{3.9}$$

Because of Equation 3.6, the time derivative of f is

$$f'(t, S(t)) = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial S}\frac{\mathrm{d}S}{\mathrm{d}t} = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial S}f.$$

Hence Equation 3.9 can be written as

$$S(t+h) = S + fh + \frac{1}{2!}\Big(\frac{\partial f}{\partial t} + \frac{\partial f}{\partial S}f\Big)h^2. \tag{3.10}$$

We expand the last term in Equation 3.7 with the Taylor series in several variables:

$$f(t + \eta_1 h, S + \eta_2 hf) \approx f + \frac{\partial f}{\partial t}\eta_1 h + \eta_2 h\frac{\partial f}{\partial S}f.$$

Hence Equation 3.7 becomes

$$S(t+h) = S + (\gamma_1 + \gamma_2)fh + (\gamma_2\eta_1)\frac{\partial f}{\partial t}h^2 + (\gamma_2\eta_2)\frac{\partial f}{\partial S}fh^2. \tag{3.11}$$

Comparing the coefficients of Equation 3.10 and Equation 3.11 yields Equation 3.8. □

Since Equation 3.8 has four unknowns and only three equations, we can choose one parameter freely and determine the remaining ones from it. A popular choice is

$$\gamma_1 = \frac{1}{2}, \quad \gamma_2 = \frac{1}{2}, \quad \eta_1 = 1, \quad \eta_2 = 1.$$

Definition 3.7 (Second-Order Runge-Kutta Method)**.** With parameters $\gamma_1, \gamma_2, \eta_1, \eta_2$ that satisfy Equation 3.8 and the increments

$$\begin{aligned} k_1 &= hf(t_k, S(t_k)), \\ k_2 &= hf\left(t_k + \eta_1 h, S(t_k) + \eta_2 k_1\right) \end{aligned}$$

the *second-order Runge-Kutta method* computes the next state as

$$S(t_{k+1}) \approx S(t_k) + \gamma_1 k_1 + \gamma_2 k_2.$$

For $\gamma_1 = \gamma_2 = \frac{1}{2}$ and $\eta_1 = \eta_2 = 1$, this gives $S(t_{k+1}) \approx S(t_k) + \frac{1}{2}(k_1 + k_2)$. □

3.6.2.2. Runge-Kutta Method of Order 4

A classical method for integrating ODEs with high accuracy is the fourth-order Runge-Kutta method (RK4). It can be derived from the Taylor series similarly to Section 3.6.2.1. Here we motivate it geometrically, see Figure 3.7.

Starting from the initial point P_0, we move along a straight line g with the mean slope $\bar{s}$ to the end of the interval $x_1 = x_0 + h$. The ordinate y_1 of the end point P_1 reached in this way is an approximation of the sought solution $y(x_1)$. The mean slope is determined from four slope values, one at each of the two boundary points of the interval and two at the middle of the interval. The values at the middle of the interval enter the mean slope $\bar{s}$ with a larger weight.

Summary: Runge-Kutta Method of Order 4

1. From P_0 with slope s_1 by $h/2$ to Q; determine the slope s_2 there.
2. From P_0 with slope s_2 by $h/2$ to R; determine the slope s_3 there.

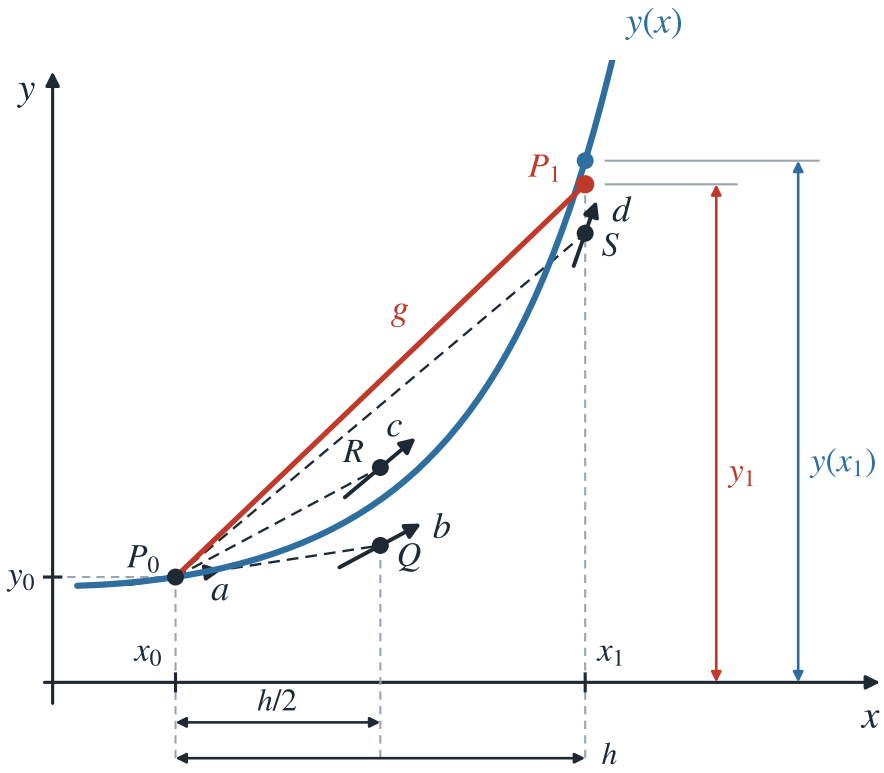


Figure 3.7.: Geometric interpretation of one step of the Runge-Kutta method of order 4 (computed for $y' = 2xy$, $P_0 = (0.3,\ 0.5)$, $h = 1$). From the initial point P_0, straight lines with the slopes s_1 (line element a), s_2 and s_3 lead to the auxiliary points Q and R at $x_0 + h/2$ and S at $x_1 = x_0 + h$. The line elements b, c, d at Q, R, S have the slopes s_2, s_3, s_4. The straight line g with the mean slope $\bar{s} = \frac{1}{6}(s_1 + 2s_2 + 2s_3 + s_4)$ leads to the end point P_1, whose ordinate y_1 approximates the exact solution $y(x_1)$. Adapted from Papula (2015).

3. From P_0 with slope s_3 by h to S; determine the slope s_4 there.
4. From P_0 with the mean slope $\bar{s} = \frac{1}{6}(s_1 + 2s_2 + 2s_3 + s_4)$ by h to P_1.

Papula (2015) computes with the increments $k_i = h\, s_i$ instead of the slopes s_i.

The first slope $s_1 = f(x_0, y_0)$ is the slope of the tangent to the solution curve at the initial point P_0 (line element a). Along this tangent we move from P_0 to the middle of the interval and reach the point Q with the coordinates

$$x_Q = x_0 + \frac{h}{2} \qquad \text{and} \qquad y_Q = y_0 + \frac{1}{2}h \times f(x_0, y_0) = y_0 + \frac{k_1}{2}, \qquad k_1 = h \times f(x_0, y_0).$$

The solution curve through Q has the line element b there with the slope

$$s_2 = f\left(x_0 + \frac{h}{2}, y_0 + \frac{k_1}{2}\right).$$

Parallel to the line element b, we again move from P_0 to the middle of the interval and reach the point R with the coordinates

$$x_R = x_0 + \frac{h}{2} \qquad \text{and} \qquad y_R = y_0 + \frac{1}{2}h \times s_2 = y_0 + \frac{k_2}{2}, \qquad k_2 = h \times s_2.$$

The solution curve through R has the line element c there with the slope

$$s_3 = f\left(x_0 + \frac{h}{2}, y_0 + \frac{k_2}{2}\right).$$

Parallel to the line element c, we move from P_0 to the end of the interval and reach the point S with the coordinates

$$x_S = x_0 + h \qquad \text{and} \qquad y_S = y_0 + h \times s_3 = y_0 + k_3, \qquad k_3 = h \times s_3.$$

The solution curve through S has the line element d there with the slope

$$s_4 = f(x_0 + h, y_0 + k_3).$$

The slope of the straight line g from the initial point P_0 to the end point P_1 is the weighted average of the four slope values:

$$\bar{s} = \frac{1}{6}(s_1 + 2s_2 + 2s_3 + s_4).$$

Definition 3.8 (Runge-Kutta Method of Order 4)**.** Let a first-order ODE

$$y' = f(x, y)$$

be given with the initial point $P_0 = (x_0, y_0)$, the step size $h = (x_n - x_0)/n$, and the nodes $x_i = x_{i-1} + h$. The *Runge-Kutta method of order 4* computes the approximations $y_i \approx y(x_i)$ according to the rule

$$\begin{aligned}
k_1 &= hf(x_{i-1}, y_{i-1}),\\
k_2 &= hf\left(x_{i-1} + \frac{h}{2}, y_{i-1} + \frac{k_1}{2}\right),\\
k_3 &= hf\left(x_{i-1} + \frac{h}{2}, y_{i-1} + \frac{k_2}{2}\right),\\
k_4 &= hf(x_{i-1} + h, y_{i-1} + k_3),\\
y_i &= y_{i-1} + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4).
\end{aligned}$$

□

Unlike Euler's polygon method, the RK4 method uses a mean slope value $\bar{s}$ for the slope of the substitute straight line. It takes into account the slope behavior of the solution curve at the two boundary points and at the middle of the interval with different weights. The polygon method, in contrast, uses only the slope $s_1 = f(x_0, y_0)$ at the initial point P_0.

4. Differential Equations in Python

Simple ODEs can be solved analytically, but nonlinear ODEs in general cannot. For them we need numerical methods. In this chapter we program general numerical solvers that can be applied to any ODE. We first consider scalar ODEs, i.e., ODEs with a single equation and a single unknown. In Section 4.5 we extend the ideas to systems of coupled ODEs.

This chapter is based on Sundnes (2023) and Kong et al. (2020). Table 4.1 summarizes the symbols used.

Table 4.1.: Symbols in Chapter 04

Symbol	Meaning	Type	Defined in
b	Damping constant of the mass-spring oscillator; locally the drag coefficient in the falling-body model of Equation 4.2	Constant	Equation 2.10
c	Spring constant	Constant	Theorem 2.7
d	Differential (dx, $\frac{\mathrm{d}y}{\mathrm{d}x}$)	Operator	Example 2.1
e	Euler's number	Constant	Example 2.1
f	Function; right-hand side of the ODE $y' = f(x, y)$ or $u' = f(t, u)$	Function	Equation 4.1
$f^{(i)}$	Component i of the vector-valued right-hand side f	Function	Definition 4.1
g	Gravitational acceleration	Constant	Equation 2.29
i	Running index (component, row, eigenvalue number, spring number)	Index	Definition 4.1
k	Running index (time/iteration step, mode, eigenvalue number)	Index	Equation 4.4
l	Length of the simple pendulum	Constant	Equation 2.29
m	Mass	Constant	Theorem 2.6
n	Order of an ODE; dimension of a matrix or system; number of steps	Index	Equation 4.4
t	Time	Scalar	Equation 4.1
t_k	k-th time point of the grid; t_0 initial time, t_f final time	Scalar	Equation 4.4
u	Unknown solution $u(t)$ of the ODE $u' = f(t, u)$; $u_k \approx u(t_k)$ approximation in step k; $u^{(i)}$ component i	Function/Vector	Equation 4.1
u_0	Initial value $u(t_0)$	Scalar/Vector	Equation 4.3
u_{max}	Carrying capacity in the logistic growth model	Constant	Equation 4.5
v	Velocity, $v = \dot{x}$; v_0 initial velocity	Scalar	Theorem 2.7
$x(t)$	Displacement from the rest position; x_i displacement of mass i; x_0 initial displacement	Function	Theorem 2.7

Symbol	Meaning	Type	Defined in
$x(t), y(t)$	Components of S (local to Exercise 4.3)	Function	Exercise 4.3
$y_1, \dots, y_n$	Components of the state vector S in the reduction of order ($y_1 = y$, $y_2 = y'$, …) or of the solution vector y of a system	Function	Theorem 2.14
C	Constant of integration; $C_1, \dots, C_n$ constants of integration or coefficients of the general solution	Constant	Example 2.1
S	State vector $S(t)$; S_i its i-th component, S_0 initial state	Vector	Theorem 2.14
Δt	Time step ($\Delta t \equiv h$ with t as the independent variable)	Scalar	Equation 4.4
λ	Eigenvalue or root of the characteristic equation (exponent in the ansatz $e^{\lambda t}$); here the growth rate in $u' = \lambda u$	Scalar	Equation 4.2
π	The number pi (italic, KaTeX exception)	Constant	–
φ	Phase angle (initial phase angle) of an oscillation; φ_d of the damped oscillation; $\varphi(\omega)$ phase response	Scalar	Equation 2.6
$\varphi(t)$	Angle of deflection of the simple pendulum; φ_0 initial deflection	Function	Equation 2.29
ω_0	Natural angular frequency of the undamped single-mass oscillator, $\omega_0 = \sqrt{c/m}$ (simple pendulum: $\sqrt{g/l}$)	Constant	Equation 2.8
$\dot{x}, \ddot{x}$	First and second derivative with respect to time t ($\dot{\varphi}$: angular velocity of the simple pendulum)	Operator	Theorem 2.7
$y', y'', y^{(n)}$	First, second, n-th derivative with respect to the independent variable	Operator	Example 2.1

The phase angle φ (a constant) and the pendulum angle $\varphi(t)$ (always written as a function of time) are both angles and share the letter. The superscript component index in $u^{(i)}$ and $f^{(i)}$ is not a derivative index.

Software

The Python files `pendulum.py` and `forward_euler_class_v1.py` are required to run the program code in this chapter. These files can be downloaded from the project repository.

4.1. Creating a General ODE Solver

Analytical solution methods typically consider a specific ODE or a class of ODEs and derive a formula for the solution. In contrast, we want to implement numerical solvers that can be applied to any ODE and are not restricted to a single example or a particular class of equations. For this purpose we need a general notation for an arbitrary ODE. We write it in the form

$$u'(t) = f(t, u(t)). \tag{4.1}$$

The ODE is thus completely specified by the right-hand side function $f(t, u)$. Examples are

$$\begin{aligned} f(t,u) &= \lambda u && \text{(exponential growth)},\\ f(t,u) &= \lambda u\left(1 - \frac{u}{u_{\max}}\right) && \text{(logistic growth)},\\ f(t,u) &= -b|u|u + g && \text{(falling body in a fluid)} \end{aligned} \tag{4.2}$$

with the growth rate λ, the carrying capacity $u_{\max}$, the drag coefficient b (in this falling-body model only), and the gravitational acceleration g.

For generality we write all right-hand sides as functions of t and u, although the examples depend only on u. Mathematically this is not necessary. For programming, however, it is very convenient, because the same solver then works for a wide range of ODE models. Our goal is to write functions and classes that take f as input, solve the corresponding ODE, and return u as output.

For Equation 4.1 to have a unique solution, we must specify the *initial condition*, i.e., the value of the solution at time $t = t_0$. The resulting problem

$$\begin{aligned} u'(t) &= f(t, u(t)),\\ u(t_0) &= u_0 \end{aligned} \tag{4.3}$$

is called an *initial value problem*. This is where the SciPy function `solve_ivp` gets its name. All problems in this chapter are initial value problems.

Example 4.1 (Initial Value Problem for $u' = u$)**.** The ODE $u' = u$ has the general solution $u = C\mathrm{e}^t$ with an arbitrary constant C, i.e., infinitely many solutions. The initial condition $u(0) = u_0$ gives $C = u_0$ and thus the unique solution $u = u_0\mathrm{e}^t$. □

When solving numerically, too, we must specify u_0 in order to start the method at all.

4.2. A Simple and General Solver: The Forward Euler Method

A numerical method for Equation 4.1 is obtained by approximating the derivative in $u' = f(t, u)$ by a finite difference. To this end, we assume that u has already been computed at discrete time points $t_0, t_1, \ldots, t_k$. At time t_k the ODE reads

$$u'(t_k) = f(t_k, u(t_k)),$$

and we approximate $u'(t_k)$ by the forward difference

$$u'(t_k) \approx \frac{u(t_{k+1}) - u(t_k)}{\Delta t}$$

with the time step Δt. Inserting this into the ODE at time $t = t_k$ gives

$$\frac{u(t_{k+1}) - u(t_k)}{\Delta t} = f(t_k, u(t_k)),$$

and rearranging yields the explicit formula

$$u(t_{k+1}) = u(t_k) + \Delta t\, f(t_k, u(t_k)).$$

This method is called the *forward Euler method* (FE) or the *explicit Euler method*. It was already introduced in Definition 3.1. It is called *forward* or *explicit* because the update formula for $u(t_{k+1})$ involves only known quantities at time t_k. In an *implicit* solver, by contrast, the update formula contains terms of the form $f(t_{k+1}, u(t_{k+1}))$. A generally nonlinear equation for the unknown $u(t_{k+1})$ must then be solved in every step.

With the notation $u_k \approx u(t_k)$ for the numerical approximation of the exact solution $u(t)$ at time $t = t_k$, the update formula reads

$$u_{k+1} = u_k + \Delta t\, f(t_k, u_k), \quad k = 0, 1, \ldots, n-1, \tag{4.4}$$

for n time steps. Starting from u_0 at time t_0, it successively yields u_1, u_2, u_3, and so on. For the ODE $u' = u$ we obtain

$$\begin{aligned} u_1 &= u_0 + \Delta t\, u_0, \\ u_2 &= u_1 + \Delta t\, u_1, \\ u_3 &= u_2 + \ldots . \end{aligned}$$

In a Python program, this repeated application of the same formula can conveniently be implemented in a for loop. The solution is stored in a list or a NumPy array.

In the code, the final time t_f is called `T`, the number of steps n is `N`, and the step index k is `n`. For a given final time t_f and number of steps n, we carry out the following steps:

1. Create arrays `t` and `u` of length $n + 1$
2. Set the initial condition: `u[0]` $= u_0$, `t[0]` $= 0$
3. Compute the time step Δt (`dt`) $= t_f/n$
4. For $k = 0, 1, 2, \ldots, n-1$:
    - `t[n+1] = t[n] + dt`
    - `u[n+1] = (1 + dt)*u[n]`

A complete Python implementation of this algorithm could look as follows:

```
%matplotlib inline

import numpy as np
import matplotlib.pyplot as plt

# Solution for N=20
N20 = 20
T = 4
dt20 = T/N20
u0 = 1

t20 = np.zeros(N20+1)
u20 = np.zeros(N20+1)

u20[0] = u0
for n in range(N20):
    t20[n+1] = t20[n] + dt20
    u20[n+1] = (1 + dt20)*u20[n]

# Solution for N=10
N10 = 10
dt10 = T/N10

t10 = np.zeros(N10+1)
u10 = np.zeros(N10+1)

u10[0] = u0
for n in range(N10):
    t10[n+1] = t10[n] + dt10
    u10[n+1] = (1 + dt10)*u10[n]

# Exact solution
t_exact = np.linspace(0, T, 100)  # 100 points for a smooth curve
u_exact = np.exp(t_exact)
```

```
plt.plot(t20, u20, label='$n=20$', linestyle=':')
plt.plot(t10, u10, label='$n=10$', linestyle='--')
plt.plot(t_exact, u_exact, label='Exact')
plt.xlabel('t')
plt.ylabel('u(t)')
plt.legend()
plt.title('Solution of $u\' = u, u(0) = 1$')
plt.show()
```

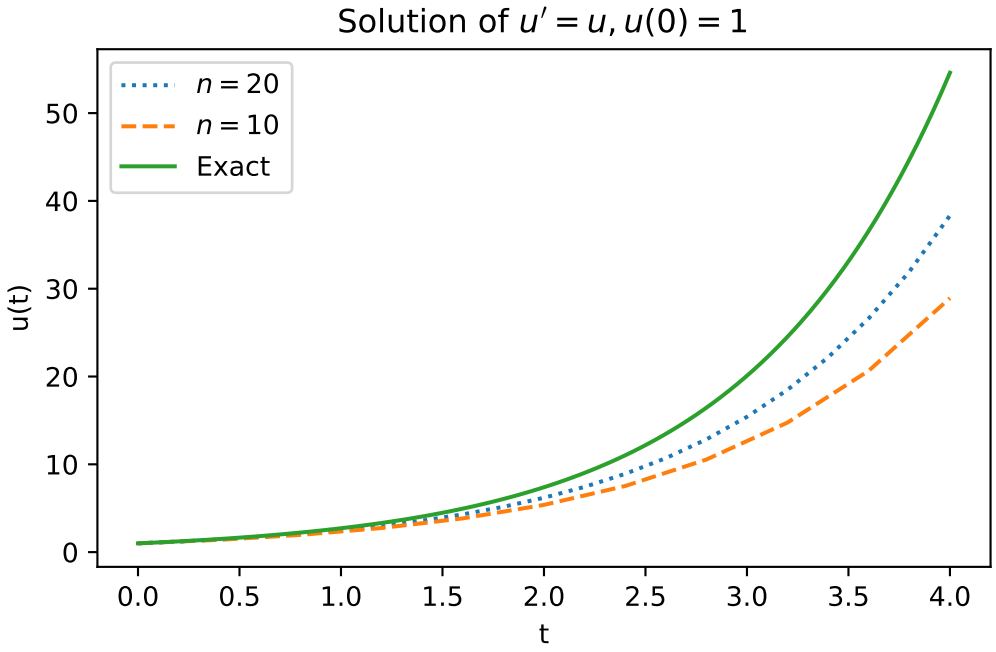


Figure 4.1.: Solution of $u' = u$, $u(0) = 1$ with $\Delta t = 0.4$ ($n = 10$) and $\Delta t = 0.2$ ($n = 20$).

If `t` is created with `np.zeros`, `t[0] = 0` need not be set explicitly. Setting `u[0]`, however, is essential: forgetting the line `u[0] = u0` is a very common error when programming ODE solvers. Figure 4.1 shows the solution for two time steps Δt. A smaller Δt improves the approximation, although both solutions are quite inaccurate. With an even smaller time step, the approximation becomes indistinguishable from the exact solution.

4.3. Extending the Solver to General ODEs

As mentioned in Section 4.1, we want to create general solvers that solve any ODE of the form $u' = f(t, u)$. This requires only a small change to the algorithm above:

1. Create arrays `t` and `u` of length $n + 1$
2. Set the initial condition: `u[0]` $= u_0$, `t[0]` $= 0$
3. For $k = 0, 1, 2, \ldots, n - 1$:
    - `t[n+1] = t[n] + dt`
    - `u[n+1] = u[n] + dt*f(t[n], u[n])`

Only the formula that computes `u[n+1]` from `u[n]` has changed. In the previous case we had $f(t, u) = u$. For a general solver we replace `u[n]` by the more general `f(t[n],u[n])`. The following Python function implements this generic version of the forward Euler method:

```
def forward_euler(f, u0, T, N):
    """Solve u'=f(t,u), u(0)=u0, with n steps until t=T."""
    import numpy as np
    t = np.zeros(N+1)
    u = np.zeros(N+1)  # u[n] is the solution at time t[n]
```

```
    u[0] = u0
    dt = T/N

    for n in range(N):
        t[n+1] = t[n] + dt
        u[n+1] = u[n] + dt*f(t[n], u[n])

    return t, u
```

This simple function solves any ODE of the form Equation 4.1. The right-hand side function $f(t, u)$ is implemented as a Python function and passed to `forward_euler` together with the initial condition `u0`, the final time `T`, and the number of time steps `N`. From the last two arguments the function computes the time step `dt`.

To illustrate this, we solve the same problem as above, $u' = u$, $u(0) = 1$, for $t \in [0, 3]$. The following code uses the function `forward_euler` for this purpose:

```
def f(t, u):
    return u

u0 = 1
T = 3
N = 30
t, u = forward_euler(f, u0, T, N)
```

The function `forward_euler` returns the two arrays `u` and `t`, which we can then plot or process further.

The definition of the right-hand side function `f` is important. It is always written with the two arguments `t` and `u`, even if, as here, only `u` is used. The solver is meant to work for all ODEs of the form $u' = f(t, u)$ and therefore calls the function as `f(t[n], u[n])`. If the right-hand side function were defined as a function of `u` only, i.e., with `def f(u):`, this call would fail with an error message. The remedy is simple: write `def f(t,u):`, even if `t` never appears inside the function.

For just 15 lines of Python code, `forward_euler` is remarkably capable. The function solves any linear or nonlinear ODE, including those that cannot be solved analytically. The procedure is always the same: identify $f(t, u)$ in the ODE, specify the initial condition u_0, implement $f(t, u)$ as a Python function `f(t, u)`, choose the number of time steps `N`, call `t, u = forward_euler(f, u0, T, N)`, and plot the solution.

The forward Euler method is the simplest of all ODE solvers. For challenging ODEs there are more accurate and more stable methods. For most ODEs it is nevertheless well suited: if the accuracy is insufficient, one reduces the time step and thereby usually obtains the required accuracy at a negligible increase in computing time.

4.4. The ODE Solver as a Python Class

Implementing the function `forward_euler` as a Python class makes the solver more flexible. There are many ways to do this. Its use could look like this:

```
method = ForwardEuler_v0(f)
method.set_initial_condition(u0)
t, u = method.solve(t_span=(0,10),N=100)
plot(t, u)
```

The advantages of a class over a function only become apparent once we introduce different solvers. First, we look at what such a solver class needs for the use case above. A constructor (`__init__`) takes the right-hand side function `f` as its only argument and stores it as an attribute. The method `set_initial_condition` takes the initial condition and stores it. The method `solve` receives the time interval `t_span` and the number of time steps `N`, implements the for loop, and returns the solution, similar to the function `forward_euler`. The time step Δt and the sequences t_k, u_k must be initialized in one of the methods and can be stored as attributes. Since the time interval and the number of steps are arguments of `solve`, this is most naturally done there.

In addition, it is convenient to implement advancing the solution by one step as a separate method `advance`. A new numerical method then typically requires changing only `advance`. A first version of the solver class could look like this:

```
import numpy as np

class ForwardEuler_v0:
    def __init__(self, f):
        self.f = f

    def set_initial_condition(self,u0):
        self.u0 = u0

    def solve(self,t_span,N):
        """Compute solution for
        t_span[0] <= t <= t_span[1],
        using N steps."""
        t0,T = t_span
        self.dt = T/N
        self.t = np.zeros(N+1)
        self.u = np.zeros(N+1)

        self.t[0] = t0
        self.u[0] = self.u0

        for n in range(N):
            self.n = n
            self.t[n+1] = self.t[n] + self.dt
            self.u[n+1] = self.advance()
        return self.t, self.u

    def advance(self):
        """Advance the solution one time step."""
        u, dt, f, n, t = self.u, self.dt, self.f, self.n, self.t

        unew = u[n] + dt*f(t[n],u[n])
        return unew
```

This class performs essentially the same tasks as the function `forward_euler`. Its main advantage is the flexibility provided by the method `advance`: for a different numerical method, a new version of `advance` typically suffices, and the remaining code stays unchanged.

A class is also suitable for the right-hand side $f(t, u)$, particularly when it contains parameters. An example is the model for logistic growth

$$u'(t) = \lambda u(t)\left(1 - \frac{u(t)}{u_{\max}}\right), \quad u(0) = u_0, \quad t \in [0, 40]. \tag{4.5}$$

It describes the self-limiting growth of a biological population, i.e., growth under limited resources. Initially the population grows approximately exponentially with the growth rate λ. As the population size approaches the *carrying capacity* $u_{\max}$, the curve flattens out (Figure 4.2). In the code, λ is called `alpha` and $u_{\max}$ is `R`.

The right-hand side contains the two parameters λ and $u_{\max}$. For our FE function or class, however, it must be implemented as a function of t and u alone. A convenient approach is a class with a call method: the parameters are set as attributes in the constructor and used in the method `__call__`:

```
class Logistic:
    def __init__(self, alpha, R, u0):
        self.alpha = alpha
        self.R = R
        self.u0 = u0

    def __call__(self, t, u):    # f(t,u)
        return self.alpha*u*(1 - u/self.R)
```

The main program for logistic growth could look like this:

```
problem = Logistic(alpha=0.2, R=1.0, u0=0.1)
solver = ForwardEuler_v0(problem)
solver.set_initial_condition(problem.u0)

T = 40
t, u = solver.solve(t_span=(0,T),N=400)

import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
plt.plot(t, u, label='u(t)')
plt.title('Solution of the logistic equation')
plt.xlabel('Time (t)')
plt.ylabel('u(t)')
plt.legend()
plt.grid(True)
plt.show()
```

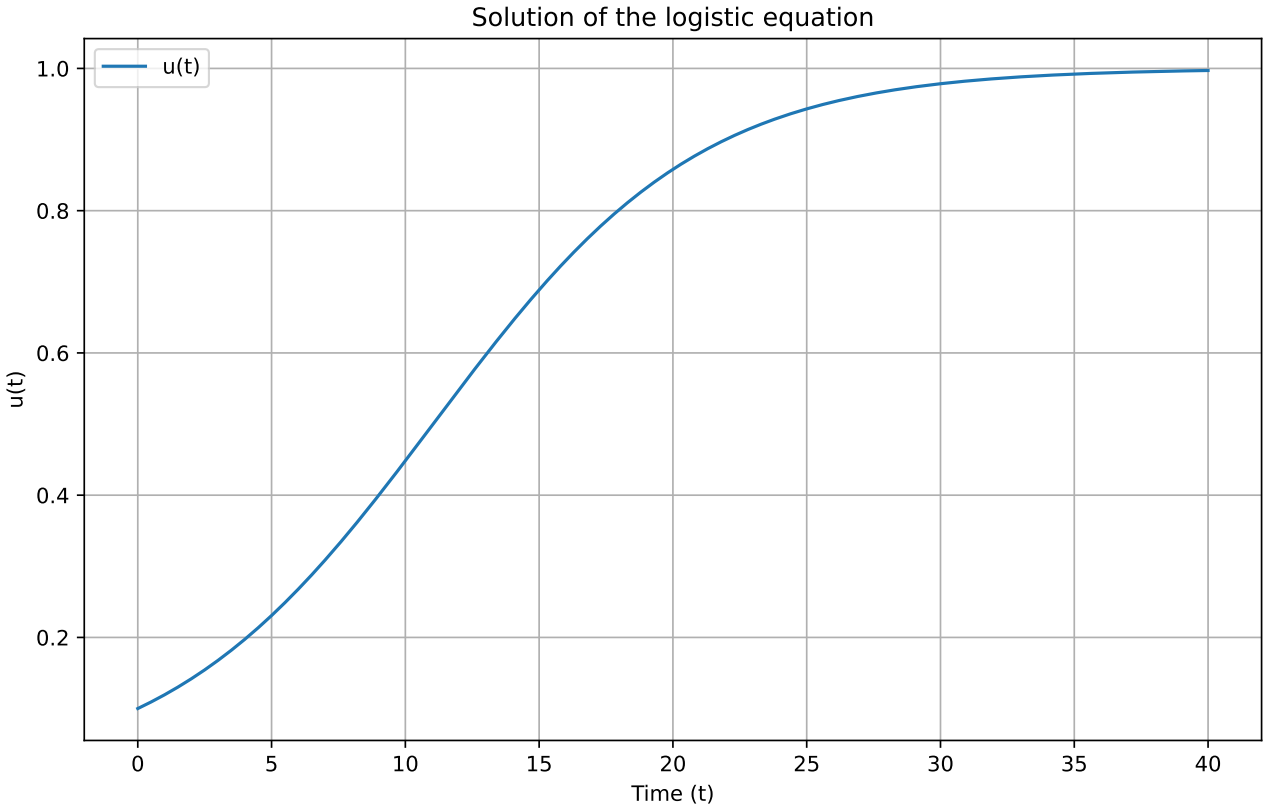


Figure 4.2.: Solution of the logistic growth model.

4.5. ODE Systems

So far our ODEs have had a single solution component. They are called scalar ODEs. Many interesting processes are described by systems of ODEs, i.e., by several ODEs in which the right-hand side of one equation depends on the solution of the others. Such systems are also called vector ODEs.

Example 4.2 (System with a Cosine Solution)**.** The system

$$\begin{aligned} u' &= v, & u(0) &= 1, \\ v' &= -u, & v(0) &= 0, \end{aligned}$$

has the solution $u = \cos t$, $v = -\sin t$. Inserting it into the equations and the initial conditions confirms this. □

General ODE systems are usually even harder to solve analytically than scalar ODEs, so numerical methods are needed. In this section we extend the Python programs from Section 4.1 to Section 4.4 to systems of ODEs. Relatively small changes to the code suffice.

The programs should be applicable to any vector ODE and any scalar ODE. To this end, we introduce a general notation.

Definition 4.1 (Vector Form of an ODE System)**.** A system of n ODEs for the n unknowns $u^{(0)}(t), u^{(1)}(t), \ldots, u^{(n-1)}(t)$ has the form

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}u^{(0)} &= f^{(0)}(t, u^{(0)}, u^{(1)}, \ldots, u^{(n-1)}),\\
\frac{\mathrm{d}}{\mathrm{d}t}u^{(1)} &= f^{(1)}(t, u^{(0)}, u^{(1)}, \ldots, u^{(n-1)}),\\
&\vdots\\
\frac{\mathrm{d}}{\mathrm{d}t}u^{(n-1)} &= f^{(n-1)}(t, u^{(0)}, u^{(1)}, \ldots, u^{(n-1)}).
\end{aligned}$$

With the solution vector $u = (u^{(0)}, u^{(1)}, \ldots, u^{(n-1)})$ and the vector-valued right-hand side $f = (f^{(0)}, f^{(1)}, \ldots, f^{(n-1)})$, the initial value problem in *vector form* reads

$$u' = f(t, u), \quad u(t_0) = u_0, \tag{4.6}$$

where u_0 is the vector of initial conditions. The superscript $i = 0, \ldots, n-1$ in $u^{(i)}$ and $f^{(i)}$ numbers components, not derivatives. □

The function f has $n + 1$ input arguments (t and the n components of u) and returns a vector of n values. Equation 4.6 has exactly the same form as Equation 4.3 for scalar ODEs. Whether a scalar ODE or a system is solved is determined solely by f and the initial condition u_0. This notation is standard in textbooks on ODEs, and the Python implementation can be just as general.

4.6. A `ForwardEuler` Class for Systems of ODEs

The class `ForwardEuler_v0` was written for scalar ODEs. It should now work for a system $u' = f$, $u(0) = u_0$, in which u, f, and u_0 are vectors (arrays). To see how the code must be changed, we first consider the numerical method. In the notation of Definition 4.1, the update formula of the forward Euler method looks exactly as in the scalar case, except that all terms are vectors:

$$\underbrace{u_{k+1}}_{\text{vector}} = \underbrace{u_k}_{\text{vector}} + \Delta t\, \underbrace{f(t_k, u_k)}_{\text{vector}}.$$

Component-wise it reads

$$u^{(i)}_{k+1} = u^{(i)}_k + \Delta t\, f^{(i)}(t_k, u_k), \quad i = 0, \ldots, n-1.$$

The compact vector notation, however, is much easier to read. It also corresponds exactly to computing with NumPy arrays. The Python code for the formula can therefore be identical to the version for scalar ODEs,

```
u[k+1] = u[k] + dt*f(t[k], u[k])
```

with the important difference that `u[k]` and `u[k+1]` are now arrays.

The solution `u` must therefore be a two-dimensional array whose rows are `u[k]`, `u[k+1]`, etc. The function `f` expects an array as its second argument and returns a one-dimensional array containing all right-hand sides $f^{(0)}, \ldots, f^{(n-1)}$.

What these arrays look like is shown by comparing the solution of a scalar ODE with that of a system of two ODEs.

For a scalar equation, `t` and `u` are one-dimensional NumPy arrays. Indexing into `u` yields numbers, the solution at each time step, for example:

```
t = array([0. ,  0.4, 0.8, 1.2, (...) ])
u = array([1. , 1.4, 1.96, 2.744, (...)])

u[0] = 1.0
u[1] = 1.4

(...)
```

For a system of two ODEs, `t` is still one-dimensional. The solution array `u`, however, is two-dimensional, with one column per solution component. The same indexing now yields a one-dimensional array of length two containing both solution components at one time step:

```
u = array([[1.0, 0.8],
           [1.4, 1.1],
           [1.9, 2.7],
              (...)])

u[0] = array([1.0, 0.8])
u[1] = array([1.4, 1.1])

(...)
```

The same notation for scalar and vector ODEs and the convenient algebra of NumPy arrays suggest that the solvers for both cases are also very similar. Indeed, after two small changes the class `ForwardEuler_v0` from Section 4.4 also works for ODE systems. First, `f(t,u)` must always return an array. Second, the solver checks whether u_0 is a single number or a list, an array, or a tuple, and creates `u` as a one- or two-dimensional array accordingly. The second step is not strictly necessary, since a two-dimensional array of size `(N+1,1)` would also work for scalar ODEs. A one-dimensional array, however, gives simpler and more intuitive indexing. The remaining code from Section 4.4 then works without changes.

The extended class implementation could look as follows:

```
import numpy as np

class ForwardEuler:
    def __init__(self, f):
        self.f = lambda t,u: np.asarray(f(t,u),float)


    def set_initial_condition(self,u0):
        if isinstance(u0, (float,int)):  #scalar ODE
            self.neq = 1                 #number of eqs.
            u0 = float(u0)
        else:
            u0 = np.asarray(u0)
            self.neq = u0.size
        self.u0 = u0

    def solve(self,t_span,N):
        """Compute solution for
        t_span[0] <= t <= t_span[1],
        using N steps."""
        t0,T = t_span
        self.dt = (T-t0)/N
        self.t = np.zeros(N+1)
        if self.neq == 1:
            self.u = np.zeros(N+1)
        else:
            self.u = np.zeros((N+1,self.neq))

        self.t[0] = t0
        self.u[0] = self.u0
```

```
        for n in range(N):
            self.n = n
            self.t[n+1] = self.t[n] + self.dt
            self.u[n+1] = self.advance()
        return self.t, self.u

    def advance(self):
        """Advance the solution one time step."""
        u, dt, f, n, t = self.u, self.dt, self.f, self.n, self.t

        unew = u[n] + dt*f(t[n],u[n])
        return unew
```

Some parts of this code deserve a comment. The constructor looks almost the same as in the scalar case. However, it uses a lambda function and `np.asarray` to convert any `f` that returns a list or a tuple into a function that returns a NumPy array. If `f` already returns an array, `np.asarray` passes it through unchanged. This change is not strictly necessary, since one could require the user to return an array. It makes the class more robust and flexible, though. The method `set_initial_condition` checks whether `u0` is a single number (`float`) or a NumPy array and stores the number of equations in the attribute `self.neq`. Finally, the method `solve` checks `self.neq` and creates `u` as a one- or two-dimensional array of the appropriate size.

The for loop and the method `advance` are identical to the previous version of the class.

Example 4.3 (ODE Model for a Simple Pendulum)**.** To demonstrate the extended class `ForwardEuler`, we consider a system of ODEs for the motion of a simple pendulum. This nonlinear system is a classical physical problem and, despite its simplicity, has no exact analytical solution. We formulate it with two variables, the angle of deflection $\varphi(t)$ and the angular velocity $\dot{\varphi}(t)$ (Figure 4.3). For a pendulum without friction, we have

$$\frac{\mathrm{d}}{\mathrm{d}t}\varphi = \dot{\varphi}, \tag{4.7}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\dot{\varphi} = -\frac{g}{l}\sin(\varphi) \tag{4.8}$$

with the pendulum length l and the gravitational acceleration g. Equation 4.7 follows directly from the definition of the angular velocity. Equation 4.8 follows from Newton's second law: the left-hand side is the angular acceleration $\ddot{\varphi}$, the right-hand side the tangential component of the weight, divided by the mass and the length of the pendulum. To solve the system, we need initial conditions for both unknowns, i.e., the initial deflection and angular velocity of the pendulum.

The right-hand side of Equation 4.7 and Equation 4.8 contains the parameters l and g. As for the logistic growth model, we therefore implement it as a class. In the code, φ is called `theta`, $\dot{\varphi}$ is `omega`, and l is `L`. A possible implementation:

```
from math import sin

class Pendulum:
    def __init__(self, L, g = 9.81):
        self.L = L
        self.g = g

    def __call__(self, t, u):
        theta, omega = u
        dtheta = omega
        domega = -self.g/self.L * sin(theta)
        return [dtheta, domega]
```

The function returns a list. The constructor of the solver class automatically wraps it into a function that returns an array. The main program differs from the previous examples only in the initial condition with two components. The code for the simple pendulum could look like this:

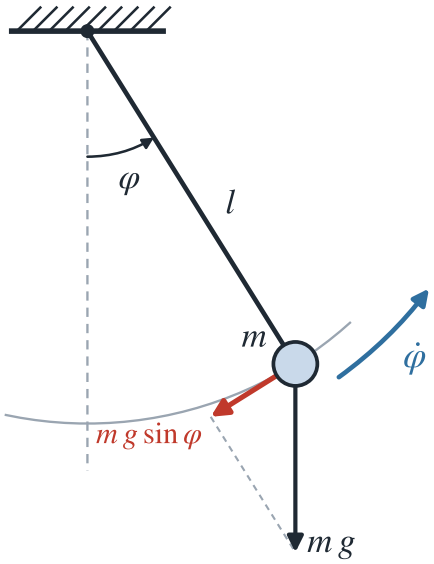


Figure 4.3.: Simple pendulum of length l with mass m, deflected by the angle φ from the rest position (dashed). The curved arrow shows the angular velocity $\dot{\varphi} = \mathrm{d}\varphi/\mathrm{d}t$ along the circular path. The weight $m\,g$ has the tangential component $m\,g \sin\varphi$, which drives the pendulum back toward the rest position and leads to $\ddot{\varphi} = -(g/l)\sin\varphi$. Adapted from Sundnes (2023).

```
import matplotlib.pyplot as plt

problem = Pendulum(L=1)
solver = ForwardEuler(problem)
theta0, omega0 = np.pi/4, 0
solver.set_initial_condition((theta0,omega0))
T = 10
N = 1000
t, u = solver.solve(t_span=(0,T),N=N)

plt.plot(t,u[:,0],label=r'$\varphi$')
plt.plot(t,u[:,1],label=r'$\dot{\varphi}$')
plt.legend()
plt.show()
```

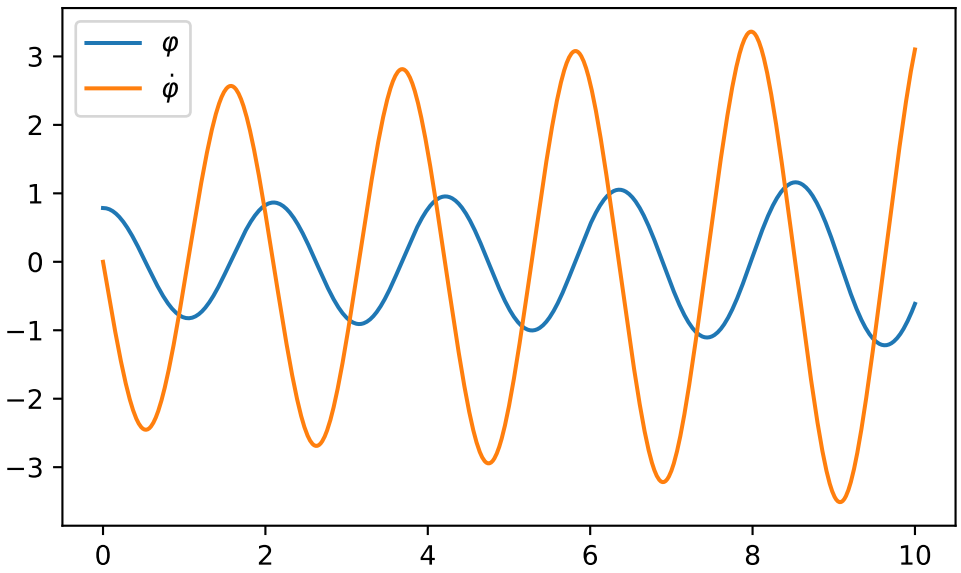


Figure 4.4.: Solution of the simple pendulum problem, computed with the forward Euler method.

Since `u` is a two-dimensional array, we extract the individual components by array slicing. In this example, `plt.plot(t,u)` would also work and plot both solution components. If only selected components are to be plotted, however, slicing is required.

Figure 4.4 shows the result. The labels use Python's raw string format, indicated by the `r` in front of the string. Raw strings treat the backslash (\) as a regular character and are therefore convenient for mathematical symbols in LaTeX notation.

The amplitude of the pendulum motion appears to increase with time, which is not physically correct. For the undamped pendulum according to Equation 4.7 and Equation 4.8, energy is conserved, so the amplitude should be constant. The increasing amplitude is a numerical artifact of the forward Euler method. A smaller time step or a different numerical method improves the solution. □

4.7. Using ODE Solvers from SciPy

There are numerous ready-made ODE solvers, so implementing our own may seem unnecessary. It is often very useful, however, to know how the solvers we apply work. The best way to learn this is to implement them ourselves.

If, on the other hand, a given ODE model is to be solved as efficiently as possible, several existing solvers are available. For Python programmers, the obvious choice is the solvers from *SciPy*, which have evolved into a robust and fairly efficient collection. SciPy is an extensive collection of scientific software in Python, with tools for linear algebra, optimization, integration, and other common tasks in scientific computing.

For initial value problems, the function `solve_ivp` from the module `integrate` is the tool of choice. The following code uses it to solve the pendulum problem of Equation 4.7 and Equation 4.8 with the class `Pendulum`, which is stored in the separate file `pendulum.py`.

```
from scipy.integrate import solve_ivp
import numpy as np
import matplotlib.pyplot as plt
from pendulum import Pendulum

problem = Pendulum(L=1)
t_span = (0,10.0)
u0 = (np.pi/4, 0)

solution = solve_ivp(problem, t_span, u0)



plt.plot(solution.t, solution.y[0,:])
plt.plot(solution.t, solution.y[1,:])
plt.legend([r'$\varphi$',r'$\dot{\varphi}$'])
plt.show()
```

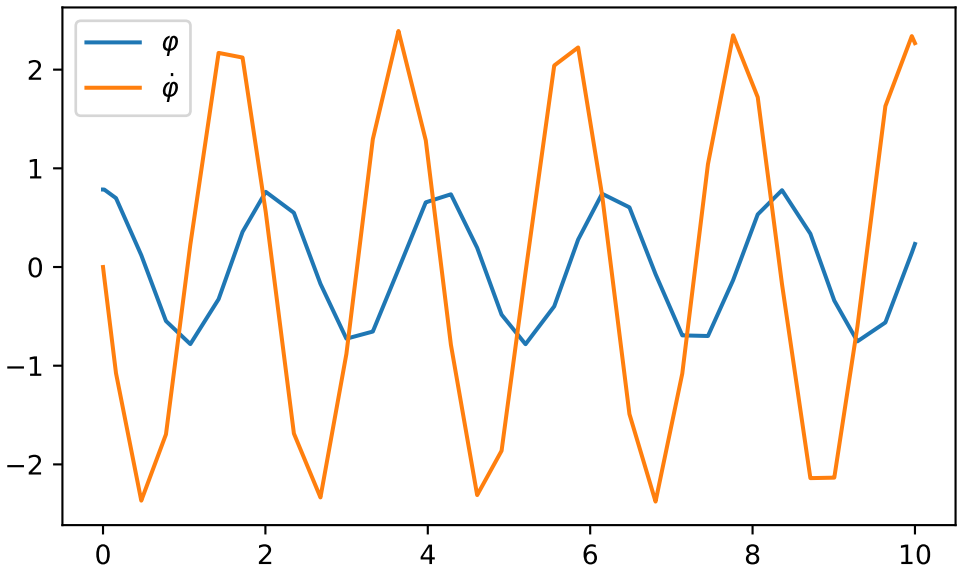


Figure 4.5.: Solution of the simple pendulum problem, computed with the SciPy function `solve_ivp` and the default tolerance.

Figure 4.5 shows the result. The solution looks far less smooth than that of the `ForwardEuler` solver.

The reason for this apparent error is that `solve_ivp` is an *adaptive* solver that chooses the time step automatically so that a given error tolerance is met. The default tolerance is relatively large. The solver therefore uses very few time steps, and the curves look jagged.

A comparison with the exact solution shows that the solution at the time points t_k is quite accurate. Only the linear interpolation between these time points spoils the picture. A smoother plot can be obtained in several ways.

For example, we pass the additional argument `t_eval`, a NumPy array containing the time points at which the solution is to be evaluated:

```
t_eval = np.linspace(0,10.0,1001)
solution = solve_ivp(problem, t_span, u0,t_eval=t_eval)
```

Alternatively, we reduce the error tolerance of the solver, for example as follows:

```
rtol = 1e-6
solution = solve_ivp(problem, t_span, u0,rtol=rtol)
```

This call reduces the relative tolerance `rtol` from its default value `1e-3` (0.001). The absolute tolerance can be adjusted with the parameter `atol`.

Of the many arguments and options of `solve_ivp`, we mention only the parameter `method`, which selects the numerical method. A call like

```
rtol = 1e-6
solution = solve_ivp(problem, t_span, u0,method='Radau')
```

replaces the default solver, the explicit embedded Runge-Kutta method of order 5(4) by Dormand and Prince (RK45, in SciPy `method='RK45'`), with an implicit Radau solver.

A complete description of all parameters of `solve_ivp` is given in the online documentation of SciPy. Here we present only the most important ones, which suffice for most applications.

> **Note 1: Using `solve_ivp`**
>
> The solver is called as
> `solve_ivp(fun, t_span, s0, method = 'RK45', t_eval=None)`
> with the arguments
>
> - `fun`: right-hand side function of the system,

- `t_span`: integration interval (t_0, t_f) with the initial time t_0 and the final time t_f,
- `s0`: initial state,
- `method`: numerical method, RK45 by default,
- `t_eval`: sorted time points in `t_span` at which the computed solution is stored.

Example 4.4 (Using `solve_ivp` for $S' = \cos(t)$)**.** Consider the ODE

$$\frac{\mathrm{d}S(t)}{\mathrm{d}t} = \cos(t)$$

with the initial value $S_0 = 0$. The exact solution is $S(t) = \sin(t)$. We approximate the solution of this initial value problem on the interval $[0, \pi]$ with `solve_ivp` and plot the approximation and the error (the difference to the exact solution) over time.

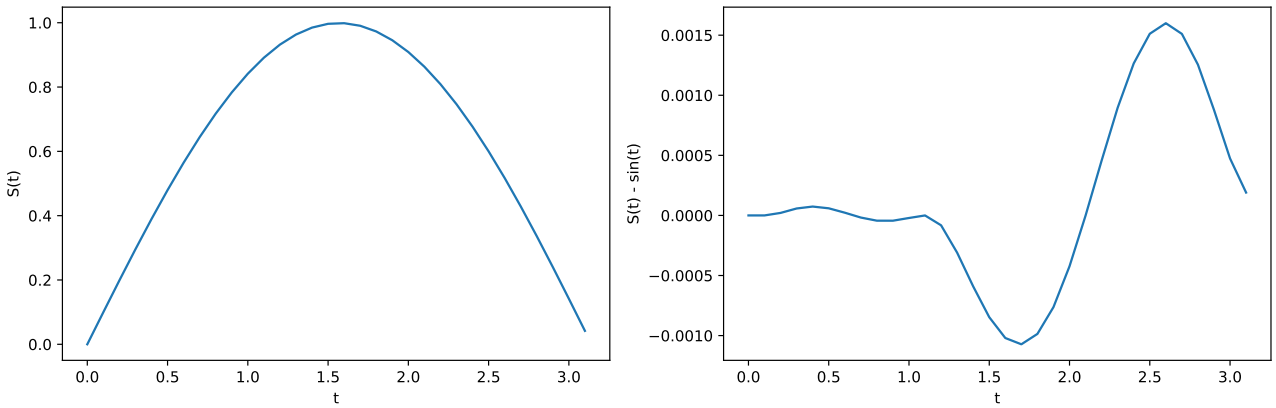
```
import matplotlib.pyplot as plt
import numpy as np
from scipy.integrate import solve_ivp

%matplotlib inline

F = lambda t, s: np.cos(t)

t_eval = np.arange(0, np.pi, 0.1)
sol = solve_ivp(F, [0, np.pi], [0], t_eval=t_eval)

plt.figure(figsize = (12, 4))
plt.subplot(121)
plt.plot(sol.t, sol.y[0])
plt.xlabel('t')
plt.ylabel('S(t)')
plt.subplot(122)
plt.plot(sol.t, sol.y[0] - np.sin(sol.t))
plt.xlabel('t')
plt.ylabel('S(t) - sin(t)')
plt.tight_layout()
plt.show()
```



The left figure shows the solution of $\mathrm{d}S(t)/\mathrm{d}t = \cos(t)$ computed with `solve_ivp`, the right figure its difference to the analytical solution. The difference is small. The relative and absolute tolerances can be controlled with the arguments `rtol` and `atol`. The solver keeps the local error estimates smaller than `atol + rtol * abs(S)`. The default values are `1e-3` for `rtol` and `1e-6` for `atol`. □

4.8. Exercises

Exercise 4.1 (Using `solve_ivp`)**.** Use `rtol` and `atol` to make the difference between the approximate and the exact solution in Example 4.4 smaller than 10^{-7}. □

Exercise 4.2 (Exponential Decay with `solve_ivp`)**.** Consider the ODE

$$\frac{\mathrm{d}S(t)}{\mathrm{d}t} = -S(t)$$

with the initial value $S_0 = 1$. The exact solution of this problem is $S(t) = \mathrm{e}^{-t}$. Use `solve_ivp` to approximate the solution of this initial value problem on the interval $[0, 1]$. Plot the approximate solution against the exact solution and the error (the difference to the exact solution) over time. □

Exercise 4.3 (Using `solve_ivp` for Coupled Systems)**.** Let the state of a system be defined by

$$S(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix},$$

and let the behavior of the system be defined by the ODE

$$\frac{\mathrm{d}S(t)}{\mathrm{d}t} = \begin{pmatrix} 0 & t^2 \\ -t & 0 \end{pmatrix} S(t).$$

Use `solve_ivp` to solve this ODE for the time interval $[0, 10]$ with the initial value $S_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. Plot the solution in $(x(t), y(t))$. □

Exercise 4.4 (Free Undamped Oscillation of a Spring Pendulum)**.** Use `solve_ivp` to model a spring pendulum that follows the equation of simple harmonic motion

$$\frac{\mathrm{d}^2x}{\mathrm{d}t^2} = -\frac{c}{m}x$$

Here x is the displacement of the spring from the equilibrium position (in meters), m the mass at the end of the spring (in kilograms), and c the spring constant (in newtons per meter). □

Exercise 4.5 (Damped Spring-Mass System)**.** Consider the model of a mass-spring-damper system in one dimension shown in Figure 4.6.

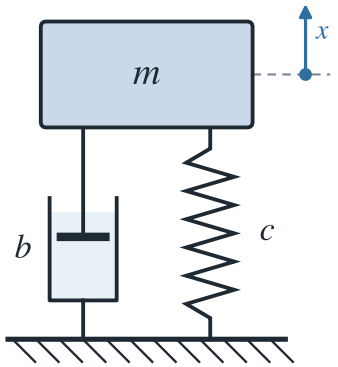


Figure 4.6.: Mass-spring-damper system in one dimension: a block of mass m rests on a damper with damping constant b and a spring with spring constant c, arranged in parallel between the block and the fixed ground. x denotes the displacement of the mass from its rest position (positive upward). Adapted from Kong et al. (2020).

In Figure 4.6, m denotes the mass of the block, b the damping constant, and c the spring constant. A damper dissipates energy by opposing the velocity with a resistance. The damped spring-mass system is a simplified model for engineering applications such as shock absorbers and structural systems.

Acceleration, velocity, and displacement are related by the ODE

$$m\ddot{x} + b\dot{x} + cx = 0$$

rewritten as

$$\ddot{x} = -\frac{b}{m}\dot{x} - \frac{c}{m}x.$$

The state of the system is described by the vector $S = [x, v]$, where x is the displacement of the mass from its rest position and v is its velocity. As a first-order ODE for the state S, the equation reads

$$\frac{\mathrm{d}S}{\mathrm{d}t} = f(t, S).$$

Solve the system with `solve_ivp`. To this end, define a function `my_msd(t, S, m, c, k)` with the damping constant b (`c`) and the spring constant c (`k`). Here t is a scalar denoting time and S is a $(2,)$ vector denoting the state of the system. The parameters m, b, and c are the mass, the damping constant, and the spring constant of the oscillation equation. □

Exercise 4.6 (Simple Pendulum: Runge-Kutta Method of Order 4)**.** The (nonlinear) ODE for the motion of a simple pendulum reads

$$\frac{\mathrm{d}^2\varphi}{\mathrm{d}t^2} + \frac{g}{l}\sin(\varphi) = 0,$$

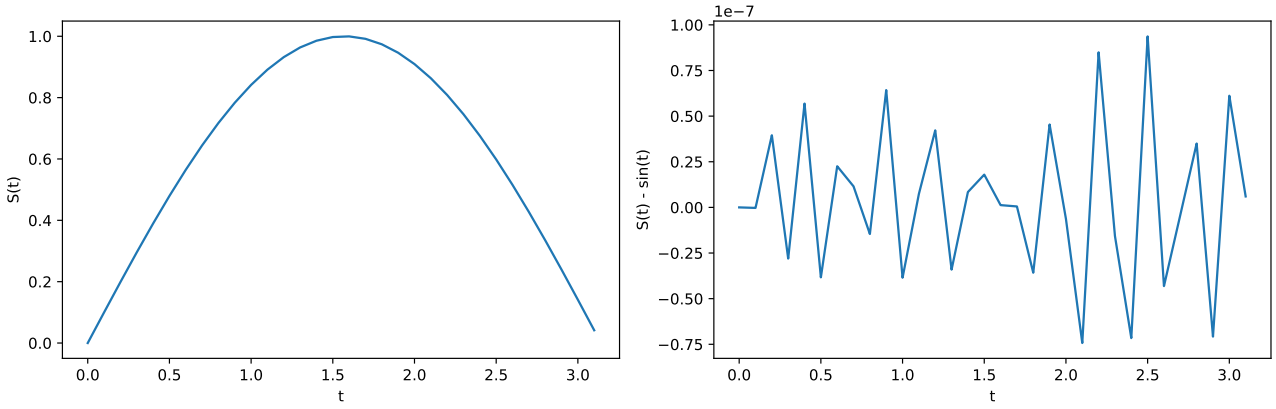
see Figure 2.21 and Equation 2.29. Here $\varphi = \varphi(t)$ is the angle of deflection of the pendulum from the vertical at time t, g the gravitational acceleration, and l the length of the pendulum string. The pendulum is to be deflected from its rest position ($\varphi(0) = 0$) and set in motion with an initial angular velocity $\dot{\varphi}(0) = 1$. For the special case $g = l$, compute the angle of deflection $\varphi(t)$ of the pendulum as well as the angular velocity $\dot{\varphi}$ at time $t = 0.1$ by using a Runge-Kutta method of order 4 from the Python library `scipy` (function `solve_ivp`). □

4.9. Solutions

Solution 4.1.

```
sol = solve_ivp(F, [0, np.pi], [0], t_eval=t_eval, \
                rtol = 1e-8, atol = 1e-8)

plt.figure(figsize = (12, 4))
plt.subplot(121)
plt.plot(sol.t, sol.y[0])
plt.xlabel('t')
plt.ylabel('S(t)')
plt.subplot(122)
plt.plot(sol.t, sol.y[0] - np.sin(sol.t))
plt.xlabel('t')
plt.ylabel('S(t) - sin(t)')
plt.tight_layout()
plt.show()
```



□

Solution 4.2.

```
F = lambda t, s: -s

t_eval = np.arange(0, 1.01, 0.01)
sol = solve_ivp(F, [0, 1], [1], t_eval=t_eval)



plt.figure(figsize = (12, 4))
plt.subplot(121)
plt.plot(sol.t, sol.y[0])
plt.xlabel('t')
plt.ylabel('S(t)')
plt.subplot(122)
plt.plot(sol.t, sol.y[0] - np.exp(-sol.t))
plt.xlabel('t')
plt.ylabel('S(t) - exp(-t)')
plt.tight_layout()
plt.show()
```

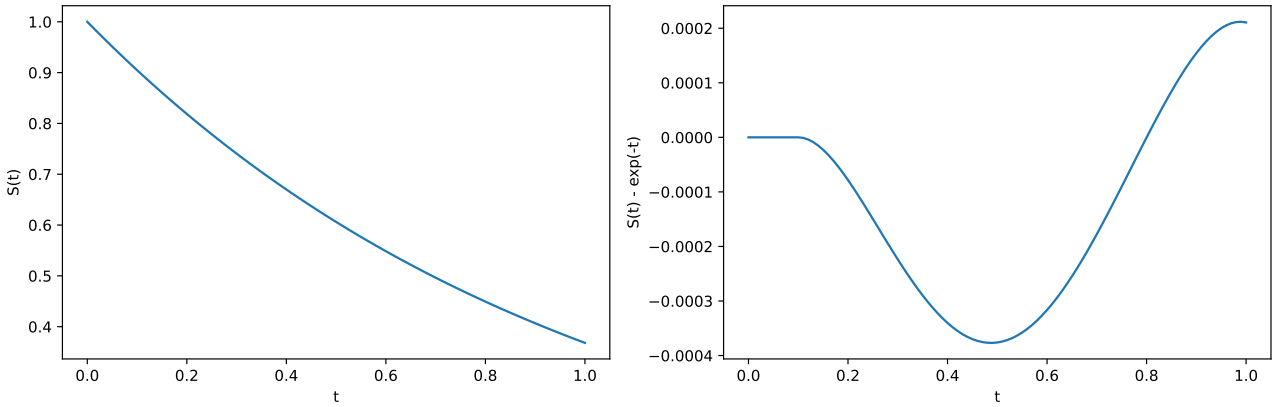


Figure 4.7.: Solution of the problem $\mathrm{d}S(t)/\mathrm{d}t = -S(t)$, computed with the SciPy function `solve_ivp`.

Figure 4.7 shows the corresponding numerical results.

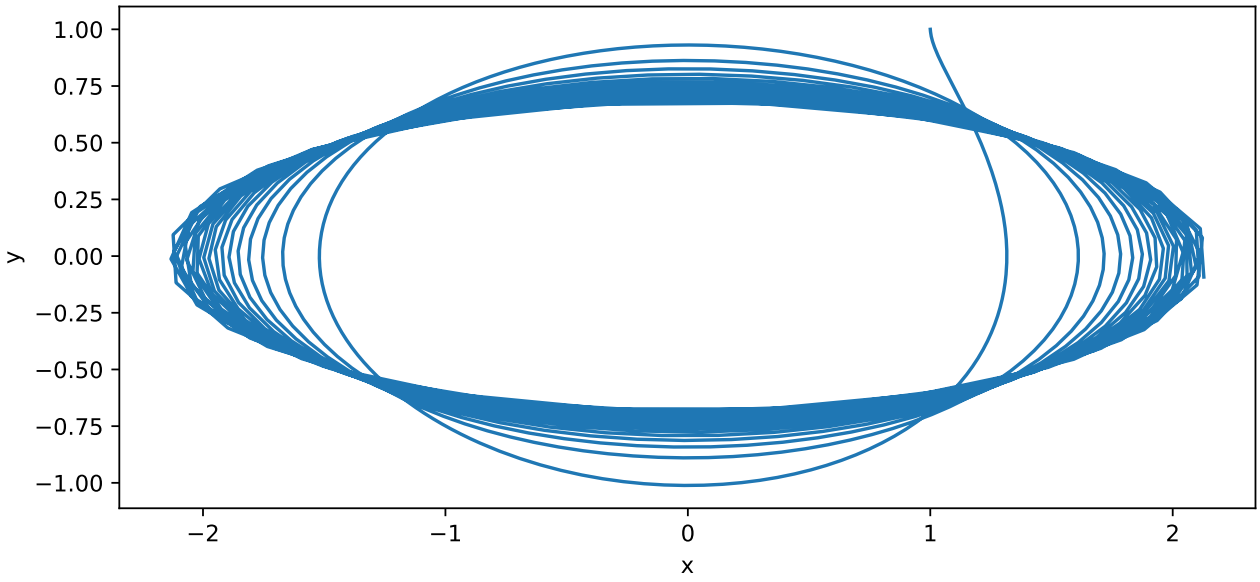
□

Solution 4.3.

```
F = lambda t, s: np.dot(np.array([[0, t**2], [-t, 0]]), s)

t_eval = np.arange(0, 10.01, 0.01)
sol = solve_ivp(F, [0, 10], [1, 1], t_eval=t_eval)

plt.figure(figsize = (9, 4))
plt.plot(sol.y.T[:, 0], sol.y.T[:, 1])
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



□

Solution 4.4. Since `solve_ivp` solves only first-order systems, we transform the second-order equation into a first-order system according to Theorem 2.14. With $y_1 = x$ and $y_2 = \mathrm{d}x/\mathrm{d}t$ (`y[0]` and `y[1]` in the code), we have

$$\begin{aligned} \frac{\mathrm{d}y_1}{\mathrm{d}t} &= y_2, \\ \frac{\mathrm{d}y_2}{\mathrm{d}t} &= -\frac{c}{m} y_1. \end{aligned}$$

We solve this system with `solve_ivp`. For comparison, we plot the exact (analytical) solution of the harmonic oscillation,

$$x(t) = x_0 \cos\left(\omega_0 t + \varphi\right), \quad \omega_0 = \sqrt{\frac{c}{m}},$$

with the amplitude x_0, the spring constant c (`k` in the code), the mass m, the time t, and the phase angle φ (cf. Equation 2.9). For the initial velocity 0, $\varphi = 0$, and x_0 equals the initial displacement. The following code implements and visualizes this solution (Figure 4.8).

```
# Parameters of the spring and the mass
k = 2.0  # spring constant in N/m
m = 0.5  # mass in kg

# The system of first-order differential equations
def federpendel(t, y):
    return [y[1], -k / m * y[0]]

# Initial conditions: initial displacement 1 m, initial velocity 0 m/s
y0 = [1.0, 0.0]

# Time span for the simulation
t_span = (0, 10)

# Solution of the system of differential equations
sol = solve_ivp(federpendel, t_span, y0, t_eval=np.linspace(t_span[0], t_span[1], 1000))

# Computation of the true solution
t_true = np.linspace(t_span[0], t_span[1], 1000)
x_true = y0[0] * np.cos(np.sqrt(k / m) * t_true)

plt.figure(figsize=(9, 6))
plt.plot(sol.t, sol.y[0], label='Numerical displacement (m)')
```

```
plt.plot(t_true, x_true, 'k--', label='True displacement (m)')
plt.plot(sol.t, sol.y[1], label='Numerical velocity (m/s)')
plt.title('Numerical vs. true solution of the spring pendulum')
plt.xlabel('Time (s)')
plt.ylabel('Displacement / velocity')
plt.legend()
plt.grid(True)
plt.show()
```

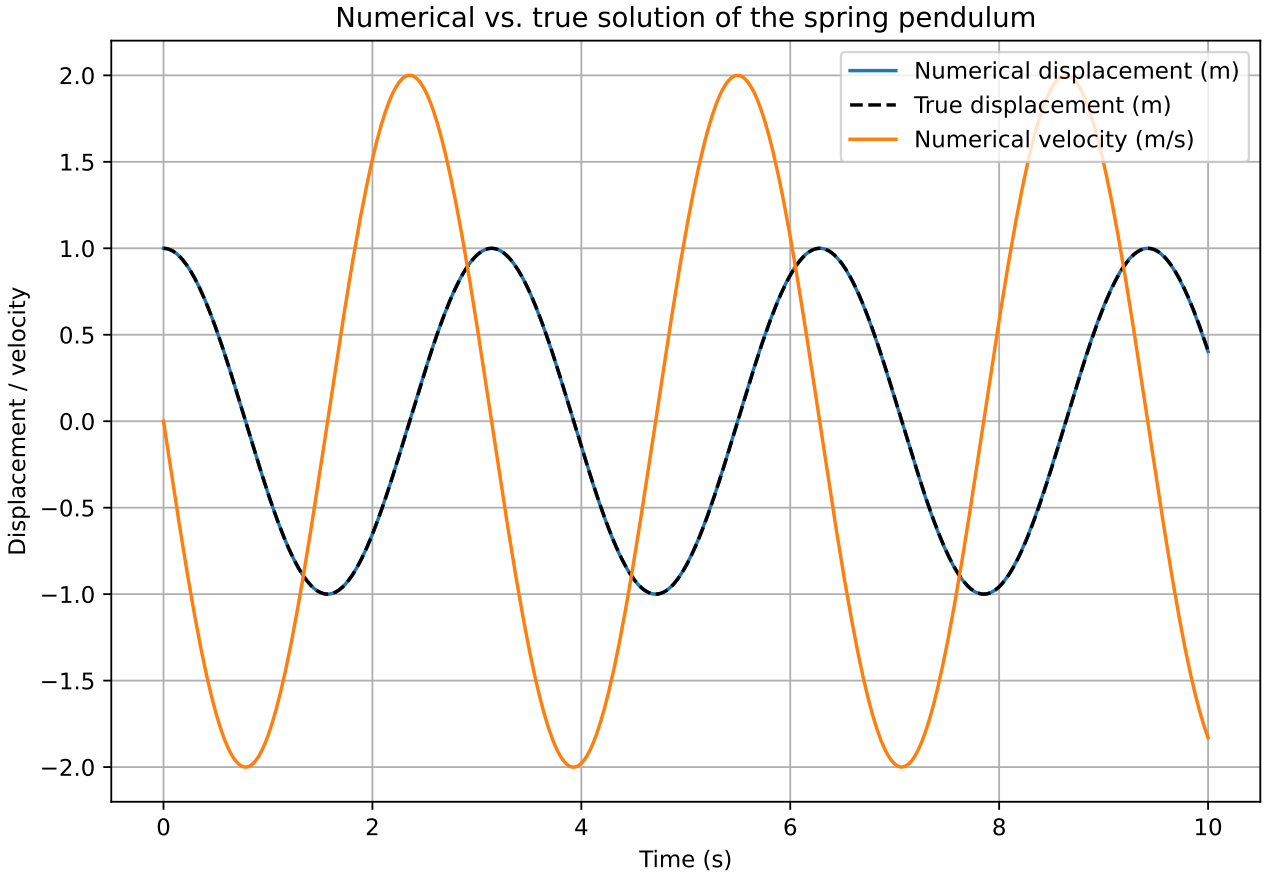


Figure 4.8.: Numerical and exact solution of the spring pendulum.

□

Solution 4.5.

```
def my_msd(t, S, m, c, k):
    x, v = S # Decompose the state vector S into displacement x and velocity v
    dxdt = v # The derivative of x with respect to time t is v
    dvdt = -(c*v + k*x) / m # The derivative of v with respect to time t
    return [dxdt, dvdt] # Return the derivatives as a list
```

Here the argument `c` stands for the damping constant b and `k` for the spring constant c. We test the function with $t = 0$, $S = [1, -1]$, $m = 10$, $b = 1$, and $c = 100$:

```
my_msd(0, [1, -1], 10, 1, 100)
```

```
[-1, -9.9]
```

Figure 4.9 shows the numerical solution of the spring-mass system for the damping constants $b = 0$, 1, and 10 with $m = 1$ and $c = 10$.

```
import numpy as np
from scipy.integrate import solve_ivp
import matplotlib.pyplot as plt
from functools import partial

m = 1
k = 10

# Case 1: no damping
f = partial(my_msd, m=m, c=0, k=k)
t_e = np.arange(0, 20, 0.1)
sol_1=solve_ivp(f,[0,20],[1,0],t_eval=t_e)

# Case 2: weakly damped
f = partial(my_msd, m=m, c=1, k=k)
sol_2=solve_ivp(f,[0,20],[1,0],t_eval=t_e)

# Case 3: strongly damped
f = partial(my_msd, m=m, c=10, k=k)
sol_3=solve_ivp(f,[0,20],[1,0],t_eval=t_e)
```

```
plt.figure(figsize = (10, 8))
plt.plot(sol_1.t, sol_1.y[0])
plt.plot(sol_2.t, sol_2.y[0])
plt.plot(sol_3.t, sol_3.y[0])
plt.xlabel('time')
plt.ylabel('displacement')
plt.legend(['No damping', 'Weak damping: $b=1$', \
            'Strong damping: $b=10$'], loc=1)
```

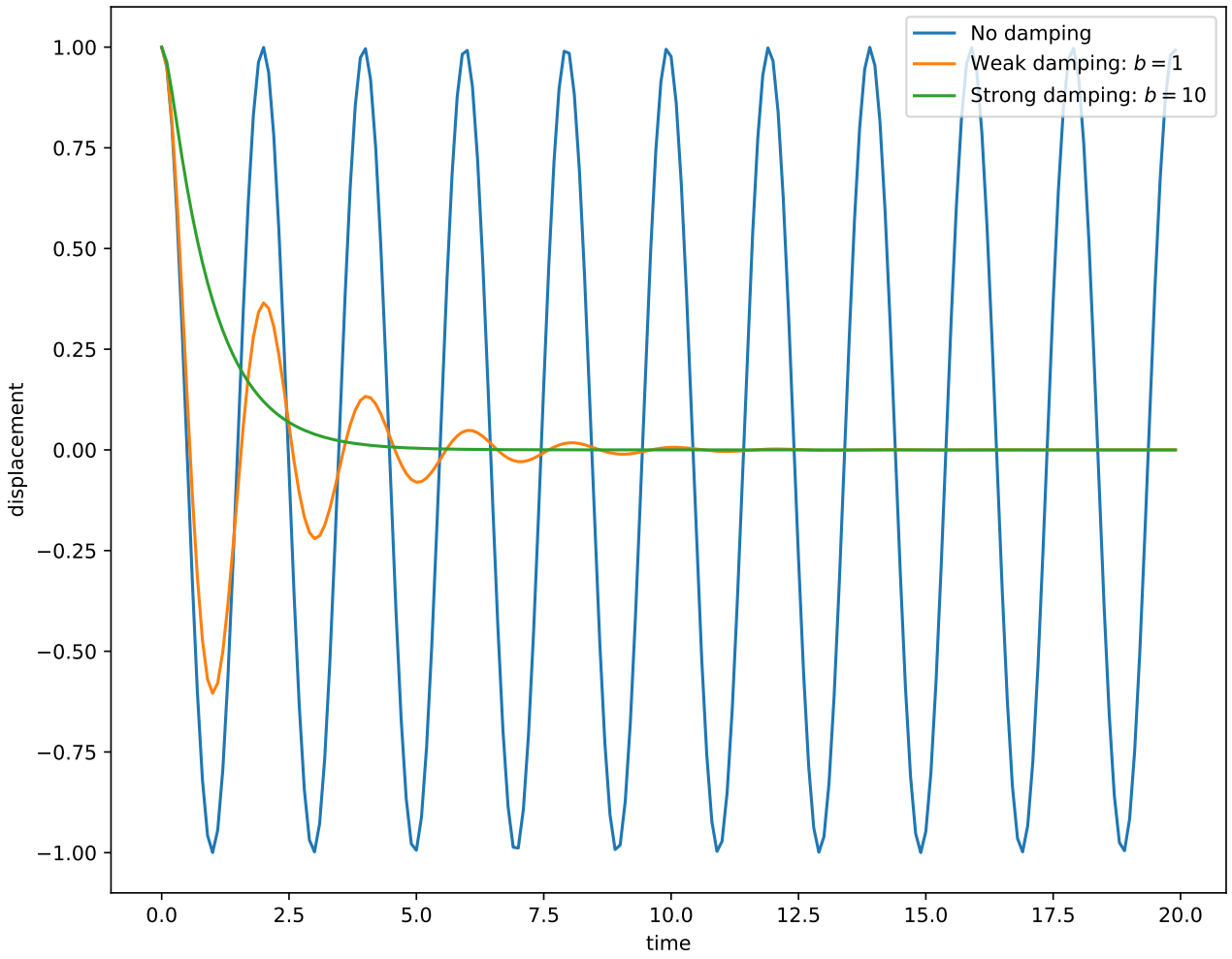


Figure 4.9.: Numerical solution of the spring-mass system with variable damping.

□

Solution 4.6. With $g = l$, the ODE for the simple pendulum reads

$$\frac{\mathrm{d}^2\varphi}{\mathrm{d}t^2} + \sin(\varphi) = 0$$

with the initial conditions $\varphi(0) = 0$ and $\dot{\varphi}(0) = 1$.

We solve it with `solve_ivp` from SciPy up to time $t = 0.1$. As its default method, `solve_ivp` already uses RK45.

First, the second-order ODE is converted into a first-order system. With the angular velocity $\dot{\varphi}$ as the second unknown (in the code, φ is called `phi` and $\dot{\varphi}$ is `omega`), we have

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\varphi &= \dot{\varphi},\\ \frac{\mathrm{d}}{\mathrm{d}t}\dot{\varphi} &= -\sin(\varphi).\end{aligned}$$

The following code solves this system, prints $\varphi(0.1)$ and $\dot{\varphi}(0.1)$, and plots $\varphi(t)$ and $\dot{\varphi}(t)$ for $0 \le t \le 0.1$ in Figure 4.10.

```
import numpy as np
from scipy.integrate import solve_ivp
import matplotlib.pyplot as plt

# Define the differential equations
def pendulum_equation(t, y):
    phi, omega = y
```

```
    dphi_dt = omega
    domega_dt = -np.sin(phi)
    return [dphi_dt, domega_dt]

# Initial conditions
phi_0 = 0
omega_0 = 1
y0 = [phi_0, omega_0]

# Time interval
t_span = [0, 0.1]
t_eval = np.linspace(t_span[0], t_span[1], 100)  # time points for the output

# Solve the differential equation
solution = solve_ivp(pendulum_equation, t_span, y0, method='RK45', t_eval=t_eval)

# Output the results
phi = solution.y[0]
omega = solution.y[1]

# Print the values for t=0.1
print("phi(t=0.1) =", phi[-1])
print("omega(t=0.1) =", omega[-1])
```

```
phi(t=0.1) = 0.09983349976270398
omega(t=0.1) = 0.9950083151697924
```

```
plt.figure(figsize=[12, 4])
plt.subplot(1, 2, 1)
plt.plot(solution.t, phi, label=r'$\varphi(t)$')
plt.xlabel('t')
plt.ylabel(r'$\varphi(t)$')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(solution.t, omega, label=r'$\dot{\varphi}(t)$')
plt.xlabel('t')
plt.ylabel(r'$\dot{\varphi}(t)$')
plt.legend()

plt.tight_layout()
plt.show()
```

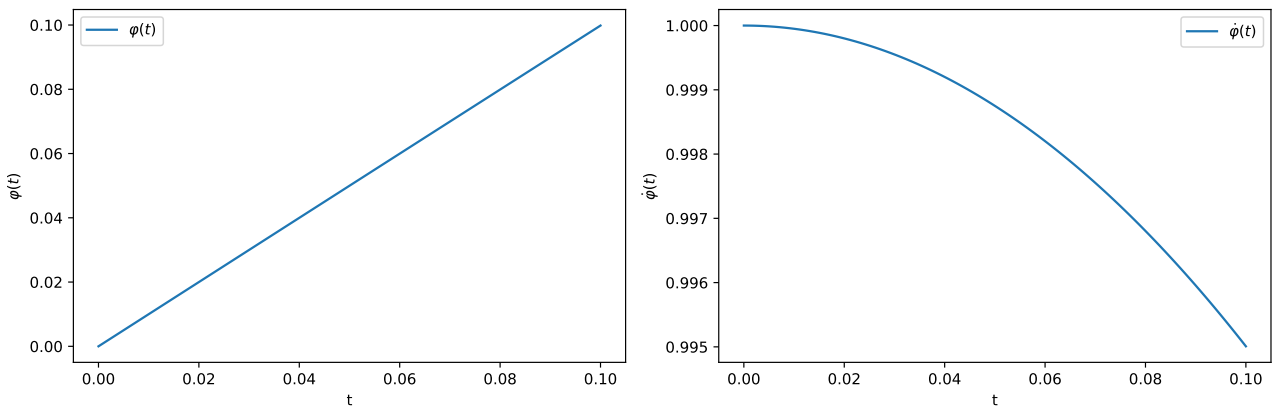


Figure 4.10.: Solution of the simple pendulum problem, computed with the `solve_ivp` function of SciPy.

□

Part II.

Systems of Linear Differential Equations

5. Eigenvalues and Eigenvectors

Table 5.1 summarizes the symbols of this chapter together with their type and the place where they are defined.

Table 5.1.: Symbols in Chapter 05

Symbol	Meaning	Type	Defined in
c	Spring constant	Constant	Theorem 2.7
c_{12}	Spring constant of spring i (c_{12}: coupling spring)	Constant	Example 5.1
i	Running index (component, row, eigenvalue number, spring number)	Index	Proposition 5.1
k	Running index (time/iteration step, mode, eigenvalue number)	Index	Proposition 5.2
m	Mass	Constant	Theorem 2.6
n	Order of an ODE; dimension of a matrix or system; number of steps	Index	Definition 5.1
t	Time	Scalar	Equation 2.8
v, v_i	Eigenvector; v_i eigenvector for the eigenvalue λ_i or μ_k	Vector	Definition 5.1
x_1, x_2	Displacement from the rest position; x_i displacement of mass i; x_0 initial displacement	Function	Theorem 2.7
z_k	Iteration vector of the power method at step k (z_0 starting vector)	Vector	Proposition 5.2
A	Matrix (coefficient, system, or difference matrix, $e = Au$)	Matrix	Definition 5.1
A_k, Q_k, R_k	Matrices in the k-th step of the QR method	Matrix	Definition 5.3
B	Further matrix (similar matrix $A = P^{-1}BP$; $B = A - \lambda E$ in the brute-force (“Hau-Ruck”) method)	Matrix	Definition 5.2
E	Identity matrix	Matrix	Definition 5.1
P	Invertible matrix of a similarity transformation, $A = P^{-1}BP$	Matrix	Definition 5.2
Q	Orthogonal matrix, $Q^TQ = E$; columns are orthonormal eigenvectors	Matrix	Definition 5.3
R	Upper triangular matrix of the QR decomposition	Matrix	Definition 5.3
T	Period of oscillation; T_0 of the simple pendulum, T_d of the damped oscillation	Scalar	Figure 2.3

Symbol	Meaning	Type	Defined in
γ_i	Coefficient of a linear combination (weights of the RK2 method; expansion of the starting vector in eigenvectors; basis representation in the eigenspace)	Scalar	Equation 3.7
κ	Coupling parameter $\kappa = c_{12}/m$ of two coupled oscillators	Constant	Example 5.1
λ, λ_i	Eigenvalue or root of the characteristic equation (exponent in the ansatz $e^{\lambda t}$); λ_i, λ_k, $\lambda_{1,2}$, $\lambda_{k\pm} = \pm j\omega_k$	Scalar	Definition 5.1
$\tilde{\lambda}_k$	Eigenvalue of the shifted matrix $A - \lambda_1 E$	Scalar	Proposition 5.4
ξ_1, ξ_2	Components of an unknown eigenvector	Scalar	Example 5.1
π	The number pi (italic, KaTeX exception)	Constant	Figure 2.3
ω	Angular frequency (excitation angular frequency; imaginary part of an eigenvalue; frequency of an eigenfunction)	Scalar	Example 5.1
ω_0	Natural angular frequency of the undamped single-mass oscillator, $\omega_0 = \sqrt{c/m}$ (simple pendulum: $\sqrt{g/l}$)	Constant	Equation 2.8
ω_1, ω_2	Natural angular frequency of the k-th normal mode (ω_1, ω_2 in 05)	Scalar	Remark 5.1
$\dot{x}, \ddot{x}$	First and second derivative with respect to time t	Operator	Equation 2.8
A^T	Transpose	Operator	Definition 5.3
A^{-1}	Inverse	Operator	Proposition 5.3
det	Determinant	Operator	Proposition 5.1
$\lvert\lambda\rvert$	Euclidean norm, absolute value	Operator	Proposition 5.2
local	ε_i (Example 5.3); placeholder $*$ for arbitrary entries (QR method)	local	where used

5.1. Definitions

Papula (2015) gives an introduction to the topic in Part I, Linear Algebra, Chapter 7, "Eigenvalues and Eigenvectors of a Square Matrix".

Definition 5.1 (Eigenvalues and Eigenvectors)**.** Let A be a (real or complex) matrix of order n and E the identity matrix of order n. The matrix equation

$$(A - \lambda E)\, v = 0$$

describes an n-dimensional *eigenvalue problem*. A number λ for which this equation has a solution $v \neq 0$ is called an *eigenvalue* of the matrix A. Every such solution v is called an *eigenvector* of A for the eigenvalue λ. The matrix $A - \lambda E$ is called the *characteristic matrix* of A. □

The eigenvalues and eigenvectors of the matrix A can be computed step by step.

Proposition 5.1 (Characteristic Equation)**.** *The eigenvalues of A are the solutions $\lambda_1, \lambda_2, \dots, \lambda_n$ of the characteristic equation*

$$\det(A - \lambda E) = 0,$$

an algebraic equation of degree n. An eigenvector v_i for the eigenvalue λ_i is a nontrivial solution of the homogeneous linear system of equations

$$(A - \lambda_i E)\, v_i = 0 \quad (i = 1, 2, \dots, n). \qquad \square$$

Proof. The homogeneous system $(A - \lambda E)\, v = 0$ has a solution $v \neq 0$ if and only if the matrix $A - \lambda E$ is singular, that is, if $\det(A - \lambda E) = 0$. $\square$

Eigenvectors are usually given in normalized form. A multiple eigenvalue may have several linearly independent eigenvectors.

Example 5.1 (Normal Modes of Coupled Mechanical Systems)**.** Figure 5.1 shows two identical oscillatory mechanical systems, each consisting of an oscillating mass m and an elastic spring with spring constant c, which are connected to each other, i.e., coupled, by a coupling spring with spring constant c_{12}.

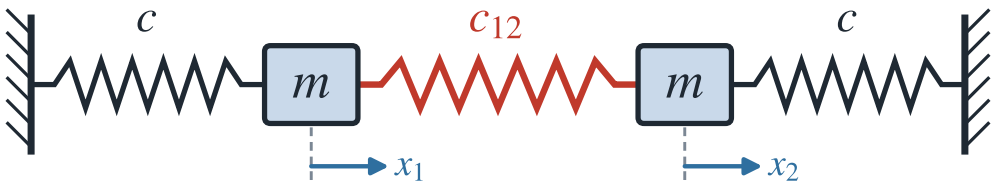


Figure 5.1.: Two identical spring-mass systems (mass m, spring constant c) attached to fixed walls and coupled to each other by a coupling spring (spring constant c_{12}, red). x_1 and x_2 are the displacements of the masses from their rest positions (dashed), positive to the right. Adapted from Papula (2015).

This system is capable of so-called normal modes in the direction of the system axis. Both masses then oscillate harmonically with the same angular frequency ω. Their position coordinates x_1 and x_2 are therefore periodic functions of time t with period $T = 2\pi/\omega$. The still unknown angular frequencies of these normal modes are the square roots of the (real) eigenvalues of the so-called system matrix A.

The system matrix can be set up from Newton's second law and Hooke's law for the spring forces (Theorem 2.6, Theorem 2.7). The first mass is acted on by the spring force $-cx_1$ and by the force $-c_{12}(x_1 - x_2)$ of the coupling spring. Its equation of motion is therefore

$$m\ddot{x}_1 = -cx_1 - c_{12}(x_1 - x_2).$$

For the second mass one obtains correspondingly

$$m\ddot{x}_2 = -c_{12}(x_2 - x_1) - cx_2.$$

Note the signs of the spring forces. Collecting the terms in x_1 and x_2 yields the equations

$$m\ddot{x}_1 = -(c + c_{12})x_1 + c_{12}x_2 \tag{5.1}$$

and

$$m\ddot{x}_2 = c_{12}x_1 - (c + c_{12})x_2. \tag{5.2}$$

Moving all terms to the left-hand side gives the system of equations

$$m\ddot{x}_1 + (c + c_{12})x_1 - c_{12}x_2 = 0, \tag{5.3}$$
$$m\ddot{x}_2 - c_{12}x_1 + (c + c_{12})x_2 = 0. \tag{5.4}$$

In matrix form it reads

$$m \begin{pmatrix} \ddot{x}_1 \\ \ddot{x}_2 \end{pmatrix} + \begin{pmatrix} c + c_{12} & -c_{12} \\ -c_{12} & c + c_{12} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = 0,$$

so that we obtain the system matrix

$$A = \begin{pmatrix} (c + c_{12})/m & -c_{12}/m \\ -c_{12}/m & (c + c_{12})/m \end{pmatrix}.$$

With the natural angular frequency $\omega_0 = \sqrt{c/m}$ of the uncoupled spring-mass systems (Equation 2.8) and the coupling parameter $\kappa = c_{12}/m$, the system matrix can be written as

$$A = \begin{pmatrix} \omega_0^2 + \kappa & -\kappa \\ -\kappa & \omega_0^2 + \kappa \end{pmatrix}.$$

The required eigenvalues follow from the characteristic equation

$$\det(A - \lambda E) = \begin{vmatrix} \omega_0^2 + \kappa - \lambda & -\kappa \\ -\kappa & \omega_0^2 + \kappa - \lambda \end{vmatrix} = (\omega_0^2 + \kappa - \lambda)^2 - \kappa^2 = 0.$$

We obtain

$$(\omega_0^2 + \kappa - \lambda)^2 = \kappa^2 \quad \Rightarrow \quad \omega_0^2 + \kappa - \lambda = \pm\kappa \quad \Rightarrow \quad \lambda_{1,2} = \omega_0^2 + \kappa \mp \kappa.$$

The eigenvalues of the system matrix A are therefore

$$\lambda_1 = \omega_0^2 = c/m \tag{5.5}$$

and

$$\lambda_2 = \omega_0^2 + 2\kappa = (c + 2c_{12})/m. \tag{5.6}$$

By Proposition 5.1, the eigenvectors v_1 and v_2 for the eigenvalues λ_1 and λ_2 are the nontrivial solutions of the homogeneous linear systems of equations $(A - \lambda_i E)\, v_i = 0$. We denote the components of the eigenvector to be determined by ξ_1 and ξ_2. For $\lambda_1 = \omega_0^2$, since $\omega_0^2 + \kappa - \lambda_1 = \kappa$, we obtain the system of equations

$$\begin{pmatrix} \kappa & -\kappa \\ -\kappa & \kappa \end{pmatrix} \begin{pmatrix} \xi_1 \\ \xi_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

Adding the first equation to the second and dividing the first equation by κ gives

$$\begin{pmatrix} 1 & -1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \xi_1 \\ \xi_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix},$$

hence $\xi_1 = \xi_2$ and thus the eigenvector $v_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$. Analogously, for $\lambda_2 = \omega_0^2 + 2\kappa$ we obtain the eigenvector $v_2 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$. □

Remark 5.1 (Interpretation of the Eigenvalues and Eigenvectors). Taking square roots in Equation 5.5 and Equation 5.6 yields the angular frequencies of the two normal modes in the direction of the system axis,

$$\omega_1 = \sqrt{c/m} \quad \text{and} \quad \omega_2 = \sqrt{(c + 2c_{12})/m}.$$

The first value corresponds to the natural angular frequency ω_0, i.e., the angular frequency of the uncoupled spring-mass systems:

$$\omega_1 = \omega_0 = \sqrt{c/m}.$$

In this normal mode the masses oscillate in phase (eigenvector $v_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$). The coupling spring is not loaded, see Figure 5.2.

In the second normal mode, with the larger angular frequency

$$\omega_2 = \sqrt{(c + 2c_{12})/m} > \omega_0 = \sqrt{c/m}$$

and the eigenvector $v_2 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$, the masses instead oscillate in antiphase. The coupling spring is then loaded to the maximum extent, see Figure 5.3.

□

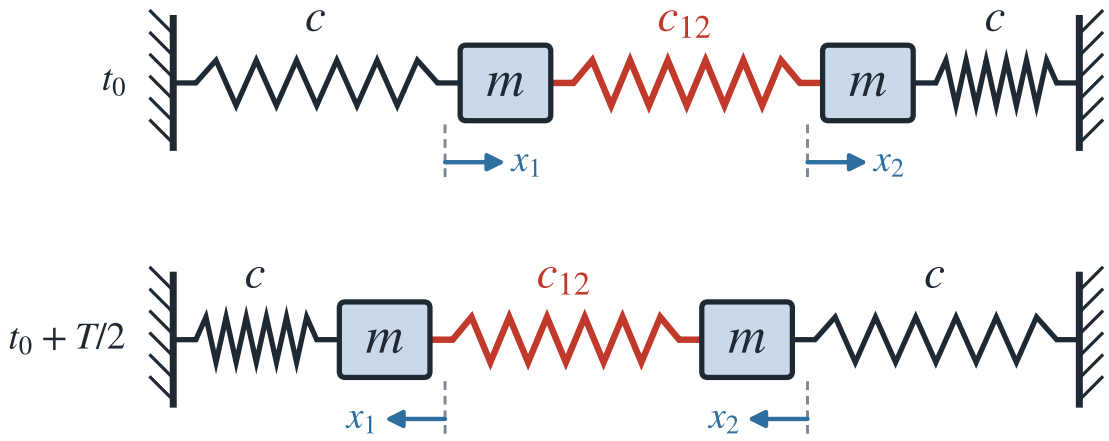


Figure 5.2.: In-phase normal mode with eigenvector $(1, 1)^T$ and angular frequency $\omega_1 = \sqrt{c/m}$: the masses oscillate synchronously ($x_1 = x_2$). Shown are two snapshots half a period $T/2$ apart, with $T = 2\pi/\omega_1$. The coupling spring c_{12} keeps its rest length and is not loaded. Adapted from Papula (2015).

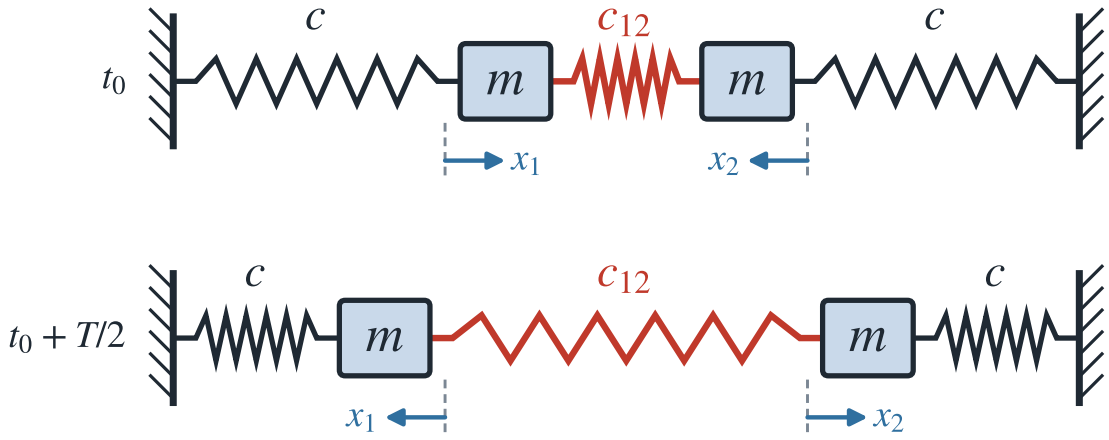


Figure 5.3.: Antiphase normal mode with eigenvector $(1, -1)^T$ and angular frequency $\omega_2 = \sqrt{(c + 2c_{12})/m}$: the masses oscillate against each other ($x_1 = -x_2$). Shown are two snapshots half a period $T/2$ apart, with $T = 2\pi/\omega_2$. The coupling spring c_{12} is alternately compressed and stretched to the maximum. Adapted from Papula (2015).

5.2. Numerical Eigenvalue Computations

5.2.1. The Power Method (Largest Eigenvalue)

5.2.1.1. Derivation

For some problems we only need the eigenvalue of largest magnitude (the dominant eigenvalue) and the corresponding eigenvector. In such cases the power method is suitable, an iterative method that converges to this eigenvalue.

Consider an $n \times n$ matrix A with n real eigenvalues $\lambda_1, \lambda_2, ..., \lambda_n$ and n linearly independent corresponding eigenvectors $v_1, v_2, ..., v_n$. Let the eigenvalues be ordered such that $|\lambda_1| > |\lambda_2| \geq ... \geq |\lambda_n|$. All that matters is that λ_1 is strictly larger in magnitude than all other eigenvalues. These may be equal to one another.

Because the eigenvectors are linearly independent, they form a basis. Every vector in the same space can therefore be written as a linear combination of the eigenvectors, in particular an arbitrary starting vector z_0:

$$z_0 = \gamma_1 v_1 + \gamma_2 v_2 + \cdots + \gamma_n v_n.$$

We assume $\gamma_1 \neq 0$. If $\gamma_1 = 0$, we choose a different starting vector.

Proposition 5.2 (Convergence of the Power Method)**.** *Under these assumptions, the vectors $z_k = A^k z_0/(\gamma_1 \lambda_1^k)$, $k = 1, 2, ...$, satisfy*

$$z_k = v_1 + \frac{\gamma_2}{\gamma_1}\left(\frac{\lambda_2}{\lambda_1}\right)^k v_2 + \cdots + \frac{\gamma_n}{\gamma_1}\left(\frac{\lambda_n}{\lambda_1}\right)^k v_n$$

and $Az_{k-1} = \lambda_1 z_k$ for $k \geq 2$. As $k \to \infty$, z_k tends to v_1, so $Az_{k-1} \approx \lambda_1 v_1$ holds for large k. □

Proof. We multiply z_0 by A. Since $Av_i = \lambda_i v_i$, we obtain

$$Az_0 = \gamma_1 Av_1 + \gamma_2 Av_2 + \cdots + \gamma_n Av_n = \gamma_1\lambda_1 v_1 + \gamma_2\lambda_2 v_2 + \cdots + \gamma_n\lambda_n v_n.$$

Factoring out $\gamma_1\lambda_1$ gives

$$Az_0 = \gamma_1\lambda_1\left[v_1 + \frac{\gamma_2}{\gamma_1}\frac{\lambda_2}{\lambda_1}v_2 + \cdots + \frac{\gamma_n}{\gamma_1}\frac{\lambda_n}{\lambda_1}v_n\right] = \gamma_1\lambda_1 z_1.$$

This completes the first iteration. Multiplying z_1 by A starts the second iteration:

$$Az_1 = \lambda_1 v_1 + \frac{\gamma_2}{\gamma_1}\frac{\lambda_2^2}{\lambda_1}v_2 + \cdots + \frac{\gamma_n}{\gamma_1}\frac{\lambda_n^2}{\lambda_1}v_n = \lambda_1\left[v_1 + \frac{\gamma_2}{\gamma_1}\frac{\lambda_2^2}{\lambda_1^2}v_2 + \cdots + \frac{\gamma_n}{\gamma_1}\frac{\lambda_n^2}{\lambda_1^2}v_n\right] = \lambda_1 z_2.$$

Continuing the multiplication with each new vector, one obtains in the k-th step

$$Az_{k-1} = \lambda_1\left[v_1 + \frac{\gamma_2}{\gamma_1}\frac{\lambda_2^k}{\lambda_1^k}v_2 + \cdots + \frac{\gamma_n}{\gamma_1}\frac{\lambda_n^k}{\lambda_1^k}v_n\right] = \lambda_1 z_k.$$

Since λ_1 is the eigenvalue of largest magnitude, $|\lambda_i/\lambda_1| < 1$ for all $i > 1$. For sufficiently large k, $(\lambda_i/\lambda_1)^k$ is therefore close to 0 for all $i > 1$. All terms containing this ratio can be neglected as k grows:

$$Az_{k-1} \approx \lambda_1 v_1.$$

□

For sufficiently large k, the power method thus yields the largest eigenvalue and the corresponding eigenvector. In the implementation we normalize the vector in every iteration by factoring out its component of largest magnitude, so that this component equals 1. The normalization factor then yields the largest eigenvalue, and the normalized vector the corresponding eigenvector.

The iteration stops as soon as one of three common criteria is met: the difference between the eigenvalues of successive iterations falls below a tolerance, the angle between successive eigenvectors falls below a threshold, or the norm of the residual vector falls below a prescribed value.

Example 5.2 (Power Method for a 2×2 Matrix)**.** The largest eigenvalue of the matrix

$$A = \begin{pmatrix} 0 & 2 \\ 2 & 3 \end{pmatrix}$$

is 4. We determine this eigenvalue and its eigenvector numerically with the power method, starting from the starting vector $z_0 = (1, 1)^T$. In Python the power method can be implemented as follows. The variable `x` holds the current iteration vector z_k.

```
import numpy as np
def normalize(x):
    # factor: component of largest magnitude, with its sign,
    # so that a negative dominant eigenvalue is also detected correctly
    fac = x[np.argmax(np.abs(x))]
    x_n = x / fac
    return fac, x_n

x = np.array([1, 1])
A = np.array([[0, 2],
              [2, 3]])

for i in range(8):
    x = np.dot(A, x)
    lambda_1, x = normalize(x)

print(f'Eigenvalue: {lambda_1}')
print(f'Eigenvector: {x}')

Eigenvalue: 3.999949137887188
Eigenvector: [0.50000636 1.        ]
```

□

Example 5.3 (Normalization in the Power Method)**.** We answer two questions about the implementation in Example 5.2: Why is the vector normalized in every iteration? And why does `lambda_1` hold the largest eigenvalue after the iteration?

Why normalize? Without normalization, $z_k = A^k z_0$ after k steps. By the derivation above, this vector behaves like $\gamma_1 \lambda_1^k v_1$, so its components grow like λ_1^k (for $|\lambda_1| > 1$) or decay like λ_1^k (for $|\lambda_1| < 1$). For the matrix from Example 5.2 with $\lambda_1 = 4$, eight steps without normalization already give

$$A^8 z_0 = \begin{pmatrix} 39322 \\ 78643 \end{pmatrix}.$$

Double-precision floating-point numbers only reach about $1.8 \cdot 10^{308}$. Since $4^{512} = 2^{1024} \approx 1.8 \cdot 10^{308}$, the components overflow to `inf` after roughly 500 steps, and from then on the iteration is worthless. For $|\lambda_1| < 1$ they underflow to zero accordingly. Normalization prevents both: after every step, the component of largest magnitude equals 1. The *direction* of the vector, which is all that matters for an eigenvector, does not change. Thus $A^8 z_0$, divided by 78643, also equals $(0.500006,\ 1)^T$.

Why does `lambda_1` hold the eigenvalue? The function `normalize` returns the component of largest magnitude of the vector Az_{k-1} as the factor `fac`, and this factor is stored in `lambda_1`. Once the iteration has converged, z_{k-1} is an eigenvector whose component of largest magnitude equals 1. For an eigenvector v of this form, for instance

$$v = \begin{pmatrix} 1 \\ \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_{n-1} \end{pmatrix} \quad \text{with} \quad |\varepsilon_i| \le 1,$$

we have

$$Av = \lambda v = \begin{pmatrix} \lambda \\ \lambda\varepsilon_1 \\ \lambda\varepsilon_2 \\ \vdots \\ \lambda\varepsilon_{n-1} \end{pmatrix}.$$

The component of largest magnitude of Av is therefore λ itself, and `normalize` returns exactly this value as the factor. In the example, z_k converges to $v_1 = (0.5,\ 1)^T$, and

$$Av_1 = \begin{pmatrix} 0 & 2 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} 0.5 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \end{pmatrix} = 4 \begin{pmatrix} 0.5 \\ 1 \end{pmatrix}.$$

The normalization factor is $4 = \lambda_1$, and the normalized vector is again v_1. Since `fac` keeps the sign of the component, a negative dominant eigenvalue is detected correctly as well. □

5.2.2. The Inverse Power Method

Proposition 5.3 (Eigenvalues of the Inverse Matrix)**.** *If A is invertible and λ is an eigenvalue of A with eigenvector v, then $1/\lambda$ is an eigenvalue of the inverse matrix A^{-1} with the same eigenvector v.* □

Proof. Since A is invertible, $\lambda \neq 0$. Multiplying $Av = \lambda v$ by A^{-1} gives $v = \lambda A^{-1} v$, hence $A^{-1} v = \lambda^{-1} v$. □

With Proposition 5.3 and the power method, the eigenvalue of A with the smallest magnitude can be determined. This is the basis of the inverse power method. Instead of multiplying by A, we multiply by A^{-1} in every iteration and thus find the eigenvalue $1/\lambda_n$ of A^{-1} with the largest magnitude. Its reciprocal λ_n is the eigenvalue of A with the smallest magnitude.

5.2.3. The Shifted Power Method

In some cases we need all eigenvalues and eigenvectors, not only the largest and the smallest. A simple but inefficient way is the shifted power method. The next section presents a more efficient method.

Proposition 5.4 (Eigenvalues of the Shifted Matrix)**.** *If A has the eigenvalues $\lambda_1, \dots, \lambda_n$ with the eigenvectors $v_1, \dots, v_n$, then the shifted matrix $A - \lambda_1 E$ has the same eigenvectors and the eigenvalues*

$$\tilde{\lambda}_k = \lambda_k - \lambda_1, \quad k = 1, \dots, n,$$

that is, $0, \lambda_2 - \lambda_1, \lambda_3 - \lambda_1, \dots, \lambda_n - \lambda_1$. □

Proof. From $Av_k = \lambda_k v_k$ it follows that $(A - \lambda_1 E)\, v_k = (\lambda_k - \lambda_1)\, v_k$. □

Applying the power method to the shifted matrix yields its eigenvalue $\tilde{\lambda}_k$ of largest magnitude. Since $\tilde{\lambda}_k = \lambda_k - \lambda_1$, this gives the eigenvalue $\lambda_k = \tilde{\lambda}_k + \lambda_1$ of A. Repeating this procedure several times finds all remaining eigenvalues, albeit at considerable cost. A better method for determining all eigenvalues is the QR method. It is named after the QR decomposition $A = QR$ of a matrix into an orthogonal matrix Q and an upper triangular matrix R (Definition 5.3).

5.2.4. The QR Decomposition

5.2.4.1. Derivation

The QR method is a preferred iterative method for finding all eigenvalues of a matrix (but not the eigenvectors at the same time). It rests on two concepts, the similarity of matrices and the QR decomposition.

Definition 5.2 (Similar Matrices)**.** Two square matrices A and B are called *similar* if there is an invertible matrix P with

$$A = P^{-1} B P.$$ □

Proposition 5.5 (Eigenvalues of Similar Matrices)**.** *Similar matrices have the same eigenvalues. If $A = P^{-1}BP$ and v is an eigenvector of B for the eigenvalue λ, then $P^{-1}v$ is an eigenvector of A for the same eigenvalue.* □

Proof. From $Bv = \lambda v$ it follows that $AP^{-1}v = P^{-1}BPP^{-1}v = P^{-1}Bv = \lambda P^{-1}v$. Here $P^{-1}v \neq 0$ because P^{-1} is invertible. Conversely, since $B = PAP^{-1}$, every eigenvalue of A is also an eigenvalue of B. □

Definition 5.3 (QR Decomposition)**.** A matrix Q is called *orthogonal* if $Q^{-1} = Q^T$, that is, $Q^{-1}Q = Q^TQ = E$. The *QR decomposition* of a matrix A is its representation $A = QR$ as the product of an orthogonal matrix Q and an upper triangular matrix R. □

How do we link these two concepts to find the eigenvalues? Let A_0 be a matrix whose eigenvalues are to be determined. In the k-th step (starting with $k = 0$) we decompose $A_k = Q_kR_k$, where Q_k is an orthogonal matrix and R_k is an upper triangular matrix. We then form $A_{k+1} = R_kQ_k$. Then

$$A_{k+1} = R_kQ_k = Q_k^{-1}Q_kR_kQ_k = Q_k^{-1}A_kQ_k.$$

By Definition 5.2 (with $P = Q_k$), all A_k are therefore similar and, by Proposition 5.5, have the same eigenvalues. As the iteration proceeds, the A_k eventually converge to an upper triangular matrix

$$A_{k+1} = R_kQ_k = \begin{pmatrix} \lambda_1 & * & \dots & * \\ 0 & \lambda_2 & \dots & * \\ & & \dots & \\ 0 & 0 & \dots & \lambda_n \end{pmatrix},$$

whose diagonal elements are the eigenvalues of the matrix. Here $*$ stands for arbitrary entries.

In every iteration of the QR method, the matrix can be factored into an orthogonal and an upper triangular matrix using a special matrix, the Householder matrix. We do not go into the mathematical details of how to obtain Q and R from the matrix. Instead, we directly use the Python function that returns both matrices.

5.2.4.2. The QR Method: Implementation in Python

Example 5.4 (QR Method for a 2×2 Matrix)**.** We use the function `qr` from `numpy.linalg` to decompose the matrix

$$A = \begin{pmatrix} 0 & 2 \\ 2 & 3 \end{pmatrix}.$$

We first import the required libraries and then perform the QR decomposition of A. The variables `q` and `r` hold the factors Q and R.

```
import numpy as np
from numpy.linalg import qr
A = np.array([[0, 2],
              [2, 3]])

q, r = qr(A)

print('Q:', q)
print('R:', r)

b = np.dot(q, r)
print('QR:', b)
```

```
Q: [[ 0. -1.]
 [-1.  0.]]
R: [[-2. -3.]
 [ 0. -2.]]
QR: [[0. 2.]
 [2. 3.]]
```

Next, we apply the QR method to find the eigenvalues of the matrix.

```
A = np.array([[0, 2],
              [2, 3]])
for i in range(20):
    q, r = qr(A)
    A = np.dot(r, q)
    if (i+1) % 5 == 0:
        print(f'Iteration {i+1}: {A}')
```

```
Iteration 5: [[ 3.99998093  0.00976559]
 [ 0.00976559 -0.99998093]]
Iteration 10: [[ 4.00000000e+00  9.53674316e-06]
 [ 9.53674316e-06 -1.00000000e+00]]
Iteration 15: [[ 4.00000000e+00  9.31322576e-09]
 [ 9.31322575e-09 -1.00000000e+00]]
Iteration 20: [[ 4.00000000e+00  9.09492929e-12]
 [ 9.09494702e-12 -1.00000000e+00]]
```

The diagonal elements are already close to the eigenvalues 4 and -1 after the fifth iteration and converge to the correct values after the tenth iteration. □

In Section 5.2.5 we look at how to obtain the eigenvalues and eigenvectors in Python easily with the built-in function.

5.2.5. Eigenvalues and Eigenvectors in `numpy.linalg` (Python)

The direct computation of eigenvalues and eigenvectors is quite simple in Python. The main function for solving the eigenvalue problem for a square array is the function `eig` in `numpy.linalg`.

Example 5.5 (Eigenvalues and Eigenvectors with `eig`)**.** For the matrix A from Example 5.4, `eig` returns the eigenvalues in the array `w` and the normalized eigenvectors v_i as the columns of the matrix `v`.

```
import numpy as np
from numpy.linalg import eig
A = np.array([[0, 2],
              [2, 3]])
w,v = eig(A)
print(f"Eigenvalues: {w}")
print(f"Eigenvectors: {v}")
```

```
Eigenvalues: [-1.+0.j  4.+0.j]
Eigenvectors: [[-0.89442719+0.j -0.4472136 +0.j]
 [ 0.4472136 +0.j -0.89442719+0.j]]
```

□

6. Systems of Linear Differential Equations

This chapter treats systems of first-order linear ODEs with constant coefficients and their solution by means of the eigenvalues and eigenvectors of the coefficient matrix. Table 6.1 summarizes the symbols used.

Table 6.1.: Symbols in Chapter 06

Symbol	Meaning	Type	Defined in
a_{ik}	Element of the coefficient matrix A of an ODE system	Scalar	Definition 6.1
e	Euler's number	Constant	Example 2.1
$g(x)$, $g_i(x)$	Forcing function of a linear ODE; as a vector $g(x)$ or $g(t)$ for systems	Function	Definition 6.1
i	Running index (component, row, eigenvalue number, spring number)	Index	Definition 6.1
j	Imaginary unit, $\mathrm{j}^2 = -1$	Constant	Equation 2.5
k	Running index (time/iteration step, mode, eigenvalue number); here column index in a_{ik}	Index	Definition 6.1
k, k_0	Multiplicity of an eigenvalue ("root of multiplicity k"); k_0 smallest exponent with rank $B^{k_0} = n - k$	Index	alg. 6.1
ℓ	Running index (component of an eigenvector, solution number)	Index	Proposition 6.2
n	Order of an ODE; dimension of a matrix or system; number of steps	Index	Definition 6.2
$p(\lambda)$	Characteristic polynomial $\det(A - \lambda E)$	Function	Equation 6.17
v, v_i	Eigenvector; v_i eigenvector for the eigenvalue λ_i or μ_k	Vector	Definition 5.1
x	Independent variable (Ch. 02 §1–4, 03, 06, 08 eigenfunctions)	Scalar	Definition 6.1
y	Unknown function $y(x)$ or vector of unknown functions; y', y'', $y^{(n)}$ derivatives	Function	Definition 6.1
$y_1, \dots, y_n$	Components of the state vector S in the reduction of order ($y_1 = y$, $y_2 = y'$, …) or of the solution vector y of a system	Function	Definition 6.1
A	Matrix (coefficient, system, or difference matrix, $e = Au$)	Matrix	Definition 6.1

Symbol	Meaning	Type	Defined in
B	Further matrix (similar matrix $A = P^{-1}BP$; $B = A - \lambda E$ in the brute-force ("Hau-Ruck") method)	Matrix	alg. 6.1
$C_1, \ldots, C_n$	Constant of integration; $C_1, \ldots, C_n$ constants of integration or coefficients of the general solution	Constant	Theorem 6.2
E	Identity matrix	Matrix	Definition 5.1
Y_i, Y_ℓ	Basis solution (vector) of a homogeneous ODE system, $Y_i = \mathrm{e}^{\lambda_i x} v_i$	Vector	Equation 6.18
$\mathbb{R}$, $\mathbb{R}^n$	Set of real numbers or of n-tuples	Set	Section 2.1
α	Real part of a complex eigenvalue, $\lambda_{1,2} = \alpha \pm \mathrm{j}\omega$	Scalar	Theorem 6.1
λ	Eigenvalue or root of the characteristic equation (exponent in the ansatz $\mathrm{e}^{\lambda t}$); λ_i, λ_k, $\lambda_{1,2}$, $\lambda_{k\pm} = \pm\mathrm{j}\omega_k$	Scalar	Definition 5.1
ξ_1, ξ_2	Components of an unknown eigenvector	Scalar	Example 6.2
ω	Angular frequency (excitation angular frequency; imaginary part of an eigenvalue; frequency of an eigenfunction)	Scalar	Theorem 6.1
$y', y'', y^{(n)}$	First, second, n-th derivative with respect to the independent variable	Operator	Example 2.1
det	Determinant	Operator	Definition 5.1
tr	Trace of a matrix	Operator	Equation 6.4
Re, Im	Real and imaginary parts	Operator	Equation 2.19
$\overline{\lambda}$	Complex conjugate	Operator	Proposition 6.2

6.1. Systems of First-Order Linear Differential Equations with Constant Coefficients

We first consider *two* coupled first-order linear ODEs with constant coefficients. Systems of *more than two* such equations are treated in Section 6.1.3.

6.1.1. Two First-Order Linear Differential Equations with Constant Coefficients

The presentation in this section follows Chapter 7 ("Systems of First-Order Linear Differential Equations") in Papula (2015).

Definition 6.1 (System of Two First-Order Linear ODEs)**.** Two coupled first-order linear ODEs with constant coefficients of the type

$$y_1' = a_{11}y_1 + a_{12}y_2 + g_1(x), \tag{6.1}$$
$$y_2' = a_{21}y_1 + a_{22}y_2 + g_2(x) \tag{6.2}$$

with real, constant coefficients a_{ik} $(i, k = 1, 2)$ and forcing functions $g_i(x)$ form a *linear system of differential equations of order 2* (the order of the system is the number of its equations). In matrix form it reads

$$\begin{pmatrix} y_1' \\ y_2' \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} + \begin{pmatrix} g_1(x) \\ g_2(x) \end{pmatrix}$$

or

$$y' = Ay + g(x)$$

with the vector $y = (y_1, y_2)^T$ of the unknown functions, the vector $y' = (y_1', y_2')^T$ of the derivatives, the coefficient matrix $A = (a_{ik})$, and the vector of forcing functions $g(x) = (g_1(x), g_2(x))^T$.

The system is called *homogeneous* if $g(x) = 0$, that is,

$$y' = Ay.$$

An *initial value problem* is present when an initial value is given for each of the two unknown functions y_1 and y_2. □

The general solution $y_1 = y_1(x)$, $y_2 = y_2(x)$ contains two constants, which are determined by the initial conditions.

6.1.2. Solution of the Homogeneous System for $n = 2$

The homogeneous system

$$y' = Ay \tag{6.3}$$

can be solved by means of the eigenvalues λ_i and eigenvectors v_i of the coefficient matrix A. To this end we use the exponential ansatz $y = v\,\mathrm{e}^{\lambda x}$ with a constant vector v.

Proposition 6.1 (Exponential Ansatz for Homogeneous Systems)**.** *Let A be a constant matrix and $v \neq 0$ a constant vector. The function $y = v\,\mathrm{e}^{\lambda x}$ solves Equation 6.3 if and only if*

$$(A - \lambda E)v = 0$$

holds, i.e., if λ is an eigenvalue of A and v an associated eigenvector. Here E is the identity matrix. □

Proof. With $y' = \lambda v\,\mathrm{e}^{\lambda x}$, substitution into Equation 6.3 gives

$$\lambda v\,\mathrm{e}^{\lambda x} = Av\,\mathrm{e}^{\lambda x}.$$

Division by $\mathrm{e}^{\lambda x} \neq 0$ yields the eigenvalue problem $\lambda v = Av$, i.e., $(A - \lambda E)v = 0$. □

The eigenvalues λ_i are the roots of the characteristic equation $\det(A - \lambda E) = 0$. For $n = 2$ this is the quadratic equation

$$\lambda^2 - (a_{11} + a_{22})\lambda + (a_{11}a_{22} - a_{12}a_{21}) = 0$$

or, in short,

$$\lambda^2 - \mathrm{tr}(A)\,\lambda + \det(A) = 0. \tag{6.4}$$

The solutions of this equation are the eigenvalues of the coefficient matrix A. As for the homogeneous second-order linear ODE with constant coefficients (Theorem 2.4), three cases have to be distinguished.

Theorem 6.1 (General Solution of the Homogeneous System for $n = 2$)**.** *Let $a_{12} \neq 0$. With $C_1, C_2 \in \mathbb{R}$, the first component of the general solution of Equation 6.3 is*

1. *for two distinct real eigenvalues $\lambda_1 \neq \lambda_2$*

$$y_1 = C_1\mathrm{e}^{\lambda_1 x} + C_2\mathrm{e}^{\lambda_2 x},$$

2. *for one real eigenvalue $\lambda_1 = \lambda_2 = \lambda$ with algebraic multiplicity 2*

$$y_1 = C_1\mathrm{e}^{\lambda x} + C_2 x\,\mathrm{e}^{\lambda x},$$

3. *for two complex conjugate eigenvalues $\lambda_{1,2} = \alpha \pm \mathrm{j}\omega$ with $\alpha, \omega \in \mathbb{R}$ and $\omega \neq 0$*

$$y_1 = \mathrm{e}^{\alpha x}(C_1 \sin(\omega x) + C_2 \cos(\omega x)).$$

In all three cases the second component is

$$y_2 = \frac{1}{a_{12}}(y_1' - a_{11}y_1). \tag{6.5}$$

□

Proof. Solving the first equation $y_1' = a_{11}y_1 + a_{12}y_2$ of the system for y_2 gives Equation 6.5. Substituting Equation 6.5 into the second equation $y_2' = a_{21}y_1 + a_{22}y_2$ (substitution method) and multiplying by a_{12} yields

$$y_1'' - (a_{11} + a_{22})\, y_1' + (a_{11}a_{22} - a_{12}a_{21})\, y_1 = 0.$$

This is a homogeneous second-order linear ODE with constant coefficients whose characteristic equation is Equation 6.4. Its basis solutions in the three cases (Theorem 2.4, Equation 2.4, Equation 2.5) yield y_1. □

If $a_{12} = 0$, the first equation contains only y_1 and can be solved directly. As an alternative to the substitution method, the solution follows from the eigenvectors of the coefficient matrix A (Theorem 6.2). If a double eigenvalue has only one linearly independent eigenvector, alg. 6.1 supplies the missing solution.

Example 6.1 (System with Complex Eigenvalues (Real Solution Path))**.** Given is the system of first-order linear ODEs with constant coefficients

$$y_1' = y_1 + y_2, \tag{6.6}$$
$$y_2' = -y_1 + y_2, \tag{6.7}$$

in matrix form

$$\begin{pmatrix} y_1' \\ y_2' \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}.$$

With the exponential ansatz $y = v\,\mathrm{e}^{\lambda x}$ and the derivative $y' = \lambda v\,\mathrm{e}^{\lambda x}$, substitution into the homogeneous system gives

$$\lambda v\,\mathrm{e}^{\lambda x} = \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} v\,\mathrm{e}^{\lambda x}$$

and, after division by $\mathrm{e}^{\lambda x}$,

$$\lambda v = \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} v$$

the eigenvalue problem

$$\begin{pmatrix} 1-\lambda & 1 \\ -1 & 1-\lambda \end{pmatrix} v = 0. \tag{6.8}$$

The characteristic equation of the coefficient matrix A is

$$\lambda^2 - 2\lambda + 2 = 0,$$

so that we obtain the complex eigenvalues $\lambda_{1,2} = 1 \pm \mathrm{j}$.

With $\alpha = 1$ and $\omega = 1$, Theorem 6.1 gives the first solution function

$$y_1 = \mathrm{e}^x(C_1 \sin(x) + C_2 \cos(x)).$$

Its derivative is

$$y_1' = \mathrm{e}^x(C_1 \sin x + C_2 \cos x) + \mathrm{e}^x(C_1 \cos x - C_2 \sin x).$$

With Equation 6.5 and $a_{11} = 1$ as well as $a_{12} = 1$ we obtain the second solution function

$$y_2 = y_1' - y_1 \tag{6.9}$$
$$= \mathrm{e}^x(C_1 \sin x + C_2 \cos x) + \mathrm{e}^x(C_1 \cos x - C_2 \sin x) - \mathrm{e}^x(C_1 \sin(x) + C_2 \cos(x)) \tag{6.10}$$
$$= \mathrm{e}^x(C_1 \cos(x) - C_2 \sin(x)). \tag{6.11}$$

The solution functions are

$$y_1 = \mathrm{e}^x(C_1 \sin(x) + C_2 \cos(x)), \tag{6.12}$$
$$y_2 = \mathrm{e}^x(C_1 \cos(x) - C_2 \sin(x)), \tag{6.13}$$

in vector form

$$y = \mathrm{e}^x \left[C_1 \begin{pmatrix} \sin(x) \\ \cos(x) \end{pmatrix} + C_2 \begin{pmatrix} \cos(x) \\ -\sin(x) \end{pmatrix} \right] \qquad (C_1, C_2 \in \mathbb{R}). \tag{6.14}$$

□

Example 6.2 (System with Complex Eigenvalues (Complex Solution Path))**.** We now solve the system from Example 6.1 with

$$A = \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}$$

in the complex domain by means of the eigenvectors v_1 and v_2 of the coefficient matrix A. We obtain them by substituting the eigenvalues into the eigenvalue problem (Equation 6.8). For the first eigenvalue $\lambda_1 = 1 + \mathrm{j}$ the coefficient matrix reads

$$\begin{pmatrix} 1-(1+\mathrm{j}) & 1 \\ -1 & 1-(1+\mathrm{j}) \end{pmatrix} = \begin{pmatrix} -\mathrm{j} & 1 \\ -1 & -\mathrm{j} \end{pmatrix}.$$

Multiplying the first equation by j and adding the two equations yields

$$\begin{pmatrix} 1 & \mathrm{j} \\ 0 & 0 \end{pmatrix}.$$

For the eigenvector $v_1 = (\xi_1, \xi_2)^T$ we choose $\xi_1 = 1$ (arbitrarily) and obtain from

$$\xi_1 + \mathrm{j}\xi_2 = 0$$

the second component $\xi_2 = \mathrm{j}$. The eigenvector for the eigenvalue $\lambda_1 = 1 + \mathrm{j}$ is thus

$$v_1 = \begin{pmatrix} 1 \\ \mathrm{j} \end{pmatrix}.$$

The eigenvector v_2 for the eigenvalue $\lambda_2 = 1 - \mathrm{j}$ is computed correspondingly. This leads to

$$v_1 = \begin{pmatrix} 1 \\ \mathrm{j} \end{pmatrix}, \quad v_2 = \begin{pmatrix} \mathrm{j} \\ 1 \end{pmatrix}.$$

For the real solution we use the first eigenvalue $\lambda_1 = 1 + \mathrm{j}$ and the associated eigenvector v_1. Using Euler's formula

$$\mathrm{e}^{(\alpha+\mathrm{j}\omega)x} = \mathrm{e}^{\alpha x}(\cos(\omega x) + \mathrm{j}\sin(\omega x)) \tag{6.15}$$

we first set up the complex solution y:

$$y = \mathrm{e}^{(1+\mathrm{j})x} \begin{pmatrix} 1 \\ \mathrm{j} \end{pmatrix} = \mathrm{e}^{x}(\cos(x) + \mathrm{j}\sin(x)) \begin{pmatrix} 1 \\ \mathrm{j} \end{pmatrix} = \mathrm{e}^{x} \begin{pmatrix} \cos(x) + \mathrm{j}\sin(x) \\ \mathrm{j}\cos(x) - \sin(x) \end{pmatrix}.$$

Sorting the real and imaginary parts yields

$$y = \mathrm{e}^{x} \left[\begin{pmatrix} \cos(x) \\ -\sin(x) \end{pmatrix} + \mathrm{j} \begin{pmatrix} \sin(x) \\ \cos(x) \end{pmatrix} \right].$$

The real basis solutions are

$$\mathrm{Re}(y) = \mathrm{e}^{x} \begin{pmatrix} \cos(x) \\ -\sin(x) \end{pmatrix}, \quad \mathrm{Im}(y) = \mathrm{e}^{x} \begin{pmatrix} \sin(x) \\ \cos(x) \end{pmatrix}.$$

Their linear combination is the general solution of the homogeneous system

$$y = \mathrm{e}^{x} \left[C_1 \begin{pmatrix} \cos(x) \\ -\sin(x) \end{pmatrix} + C_2 \begin{pmatrix} \sin(x) \\ \cos(x) \end{pmatrix} \right] \qquad (C_1, C_2 \in \mathbb{R}). \tag{6.16}$$

□

Equation 6.14 and Equation 6.16 describe the same solution set. They transform into each other by interchanging the constants C_1 and C_2.

Example 6.3 (Computation with the Second Eigenvalue)**.** Equation 6.14 and Equation 6.16 show that the solutions of the homogeneous system of first-order linear ODEs with constant coefficients can be determined by the eigenvalues and eigenvectors of the coefficient matrix A. For the computation of the solution Equation 6.16 we used the first eigenvalue $\lambda_1 = 1 + \mathrm{j}$ and the associated eigenvector $v_1 = \begin{pmatrix} 1 \\ \mathrm{j} \end{pmatrix}$. We show that the same result is also obtained with the eigenvalue $\lambda_2 = 1 - \mathrm{j}$ and the associated eigenvector $v_2 = \begin{pmatrix} \mathrm{j} \\ 1 \end{pmatrix}$.

For the real solution we use the second eigenvalue $\lambda_2 = 1 - \mathrm{j}$ and the associated eigenvector $v_2 = \begin{pmatrix} \mathrm{j} \\ 1 \end{pmatrix}$. Using Euler's formula (Equation 6.15) we first set up the complex solution y (note that $\cos(-x) = \cos(x)$ and $\sin(-x) = -\sin(x)$):

$$y = \mathrm{e}^{(1-\mathrm{j})x} \begin{pmatrix} \mathrm{j} \\ 1 \end{pmatrix} = \mathrm{e}^x (\cos(x) - \mathrm{j}\sin(x)) \begin{pmatrix} \mathrm{j} \\ 1 \end{pmatrix} = \mathrm{e}^x \begin{pmatrix} \sin(x) + \mathrm{j}\cos(x) \\ \cos(x) - \mathrm{j}\sin(x) \end{pmatrix}.$$

Sorting the real and imaginary parts yields

$$y = \mathrm{e}^x \left[\begin{pmatrix} \sin(x) \\ \cos(x) \end{pmatrix} + \mathrm{j} \begin{pmatrix} \cos(x) \\ -\sin(x) \end{pmatrix} \right].$$

The real basis solutions are

$$\mathrm{Re}(y) = \mathrm{e}^x \begin{pmatrix} \sin(x) \\ \cos(x) \end{pmatrix}, \quad \mathrm{Im}(y) = \mathrm{e}^x \begin{pmatrix} \cos(x) \\ -\sin(x) \end{pmatrix}.$$

These are the same basis solutions as in the computation with λ_1, only interchanged: $\mathrm{Re}(y)$ here is the former basis solution $\mathrm{Im}(y)$ and vice versa. Assigning the constant C_1 to the imaginary part and C_2 to the real part, the general solution of the homogeneous system is

$$y = \mathrm{e}^x \left[C_1 \begin{pmatrix} \cos(x) \\ -\sin(x) \end{pmatrix} + C_2 \begin{pmatrix} \sin(x) \\ \cos(x) \end{pmatrix} \right] \qquad (C_1, C_2 \in \mathbb{R}),$$

that is, exactly Equation 6.16. The second eigenvalue therefore yields the same result. This is no coincidence: $\lambda_2 = \overline{\lambda_1}$, and the associated complex solution is, up to a constant factor, the complex conjugate of the first one (Proposition 6.2). □

6.1.3. Homogeneous Systems of First-Order Linear Differential Equations with Constant Coefficients of Arbitrary Order

The solution path described for $n = 2$ generalizes to systems of arbitrary order.

Definition 6.2 (Homogeneous System of Order n)**.** An $n \times n$ system of differential equations of the form

$$y' = Ay$$

with $y = (y_1, \dots, y_n)^T$ and a constant matrix $A \in \mathbb{R}^{n \times n}$ is called a *homogeneous system of first-order linear ODEs with constant coefficients*. □

Theorem 6.2 (Solutions of the Homogeneous System of First-Order Linear ODEs with Constant Coefficients)**.** *Let $y' = Ay$ be a homogeneous system according to Definition 6.2. The eigenvalues λ_i of A are the roots of the characteristic polynomial*

$$p(\lambda) = \det(A - \lambda E) = 0. \tag{6.17}$$

For each eigenvalue λ_i with eigenvector v_i, i.e., $(A - \lambda_i E)v_i = 0$,

$$Y_i = \mathrm{e}^{\lambda_i x} v_i \qquad (i = 1, \dots, n) \tag{6.18}$$

is a solution of the system. If A has n linearly independent eigenvectors, the general solution of the homogeneous system is

$$y = C_1 \mathrm{e}^{\lambda_1 x} v_1 + \dots + C_n \mathrm{e}^{\lambda_n x} v_n \qquad (C_1, \dots, C_n \in \mathbb{R}).$$ □

Proof. That Y_i is a solution follows from Proposition 6.1, whose proof holds verbatim for $n \times n$ matrices. Since the system is linear, every linear combination of solutions is again a solution. If $v_1, \dots, v_n$ are linearly independent, then so are $Y_1(x), \dots, Y_n(x)$ at every point x. They therefore form a fundamental system of the n-dimensional solution space. □

> **i** Eigenvalues and Eigenvectors
>
> A summary of the computation of eigenvalues and eigenvectors can be found in Chapter 5.

If the eigenvalue λ is a root of multiplicity k of the characteristic polynomial (Equation 6.17), then k linearly independent solutions belong to λ. Two cases require additional steps: complex eigenvalues (Proposition 6.2) and multiple eigenvalues with too few eigenvectors (alg. 6.1).

Proposition 6.2 (Real Solutions for Complex Eigenvalues)**.** *Let A be real and λ a complex eigenvalue of A with multiplicity k. Then $\overline{\lambda}$ is also an eigenvalue of A. If $Y_1, \dots, Y_k$ are the linearly independent solutions belonging to λ, then the $2k$ real functions*

$$\mathrm{Re}(Y_\ell), \quad \mathrm{Im}(Y_\ell) \qquad (\ell = 1, \dots, k)$$

are linearly independent solutions of the system. They belong to the two eigenvalues λ and $\overline{\lambda}$. The real and imaginary parts are computed with Euler's formula (Equation 6.15). □

Proof. Proof sketch: Since A is real, complex conjugation of $Av = \lambda v$ gives $A\overline{v} = \overline{\lambda}\,\overline{v}$. Hence, together with Y_ℓ, $\overline{Y_\ell}$ is also a solution. As linear combinations of solutions, the real and imaginary parts $\mathrm{Re}(Y_\ell) = \frac{1}{2}(Y_\ell + \overline{Y_\ell})$ and $\mathrm{Im}(Y_\ell) = \frac{1}{2\mathrm{j}}(Y_\ell - \overline{Y_\ell})$ are again solutions, and they span the same space as the $2k$ complex solutions Y_ℓ, $\overline{Y_\ell}$. □

If λ is a root of multiplicity k but there are fewer than k linearly independent eigenvectors, the missing solutions are found with the following method.

Algorithm 6.1 (Brute-Force ("Hau-Ruck") Method)**.** *Let λ be a root of multiplicity k of the characteristic polynomial of the $n \times n$ matrix A.*

1. *Form the matrix $B = A - \lambda E$.*
2. *Form $B^2, B^3, \dots$ until some power has rank $n - k$. Let this be the case for B^{k_0}.*
3. *Determine a basis $v_1, \dots, v_k$ of the solution space of $B^{k_0} v = 0$.*
4. *A system of solutions belonging to the eigenvalue λ of multiplicity k is then, for $\ell = 1, \dots, k$,*

$$Y_\ell = \mathrm{e}^{\lambda x} \left(v_\ell + xBv_\ell + \frac{x^2}{2} B^2 v_\ell + \dots + \frac{x^{k_0 - 1}}{(k_0 - 1)!} B^{k_0 - 1} v_\ell \right). \tag{6.19}$$

The method is taken from Furlan (1995). □

Example 6.4 (Real Eigenvalues)**.** Let

$$y' = \begin{pmatrix} 3 & -2 \\ -1 & 2 \end{pmatrix} y.$$

The characteristic polynomial of the coefficient matrix A is

$$p(\lambda) = \det(A - \lambda E) = \det \begin{pmatrix} 3 - \lambda & -2 \\ -1 & 2 - \lambda \end{pmatrix} = \lambda^2 - 5\lambda + 4 = 0,$$

and the eigenvalues are $\lambda_1 = 1$ and $\lambda_2 = 4$. We obtain the eigenvectors v_1 and v_2 by substituting the eigenvalues into the eigenvalue problem

$$\begin{pmatrix} 2 & -2 \\ -1 & 1 \end{pmatrix} v_1 = 0, \quad \begin{pmatrix} -1 & -2 \\ -1 & -2 \end{pmatrix} v_2 = 0,$$

so that we obtain the eigenvectors $v_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ and $v_2 = \begin{pmatrix} -2 \\ 1 \end{pmatrix}$. By Theorem 6.2 the general solution of the homogeneous system is

$$y = C_1 \mathrm{e}^x \begin{pmatrix} 1 \\ 1 \end{pmatrix} + C_2 \mathrm{e}^{4x} \begin{pmatrix} -2 \\ 1 \end{pmatrix} \qquad (C_1, C_2 \in \mathbb{R}).$$ □

Example 6.5 (Double Eigenvalues)**.** We determine the general solution of the system

$$y' = \begin{pmatrix} 3 & 2 \\ -2 & -1 \end{pmatrix} y.$$

The characteristic polynomial of the coefficient matrix A is

$$p(\lambda) = \det(A - \lambda E) = \det \begin{pmatrix} 3 - \lambda & 2 \\ -2 & -1 - \lambda \end{pmatrix} = \lambda^2 - 2\lambda + 1 = 0,$$

which yields the double eigenvalue $\lambda = 1$. We obtain the eigenvector v by substituting the eigenvalue into the eigenvalue problem

$$\begin{pmatrix} 2 & 2 \\ -2 & -2 \end{pmatrix} v = 0,$$

so $v = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$. We computed it only as a check. We determine the solution with the brute-force method (alg. 6.1) and set

$$B = A - E = \begin{pmatrix} 2 & 2 \\ -2 & -2 \end{pmatrix}.$$

The matrix B is raised to successive powers until a power B^{k_0} reaches the rank $n - k$. Here $n = 2$ and $k = 2$. B^1 has rank 1 and B^2 has rank 0, so that $k_0 = 2$. Thus $B^2 v = 0$ has $k = 2$ linearly independent solutions, which we can choose freely. We set

$$v_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad v_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

We substitute these vectors into Equation 6.19 and obtain for the first solution (v_1)

$$Y_1 = \mathrm{e}^x \left[\begin{pmatrix} 1 \\ 0 \end{pmatrix} + x \begin{pmatrix} 2 & 2 \\ -2 & -2 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right],$$

that is,

$$Y_1 = \mathrm{e}^x \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \mathrm{e}^x \begin{pmatrix} 2 \\ -2 \end{pmatrix} x = \mathrm{e}^x \begin{pmatrix} 1 + 2x \\ -2x \end{pmatrix}.$$

For the second solution (v_2) we obtain

$$Y_2 = \mathrm{e}^x \left[\begin{pmatrix} 0 \\ 1 \end{pmatrix} + x \begin{pmatrix} 2 & 2 \\ -2 & -2 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right],$$

that is,

$$Y_2 = \mathrm{e}^x \begin{pmatrix} 0 \\ 1 \end{pmatrix} + \mathrm{e}^x \begin{pmatrix} 2 \\ -2 \end{pmatrix} x = \mathrm{e}^x \begin{pmatrix} 2x \\ 1 - 2x \end{pmatrix}.$$

The general solution of the homogeneous system is the linear combination of the two solutions:

$$y = C_1 \mathrm{e}^x \begin{pmatrix} 1 + 2x \\ -2x \end{pmatrix} + C_2 \mathrm{e}^x \begin{pmatrix} 2x \\ 1 - 2x \end{pmatrix} \qquad (C_1, C_2 \in \mathbb{R}).$$

Every linear combination of the basis solutions Y_1 and Y_2 is again a solution of the homogeneous system. Their difference gives the basis solution

$$Y_1 - Y_2 = \mathrm{e}^x \begin{pmatrix} 1 \\ -1 \end{pmatrix}.$$

This is the solution belonging to the eigenvector v, which we computed at the beginning. The general solution of the homogeneous system can therefore also be written as

$$y = C_1 \mathrm{e}^x \begin{pmatrix} 1 \\ -1 \end{pmatrix} + C_2 \mathrm{e}^x \begin{pmatrix} 1 + 2x \\ -2x \end{pmatrix} = \mathrm{e}^x \left[C_1 \begin{pmatrix} 1 \\ -1 \end{pmatrix} + C_2 \begin{pmatrix} 1 + 2x \\ -2x \end{pmatrix} \right] \qquad (C_1, C_2 \in \mathbb{R}) \qquad \square$$

Example 6.6 (Complex Eigenvalues)**.** We determine the general solution of the system

$$y' = \begin{pmatrix} 3 & 2 \\ -2 & 1 \end{pmatrix} y.$$

The characteristic polynomial of the coefficient matrix A is

$$p(\lambda) = \det(A - \lambda E) = \det \begin{pmatrix} 3 - \lambda & 2 \\ -2 & 1 - \lambda \end{pmatrix} = \lambda^2 - 4\lambda + 7 = 0,$$

which yields the complex eigenvalues $\lambda_{1,2} = 2 \pm \mathrm{j}\sqrt{3}$. For the first eigenvalue $\lambda_1 = 2 + \mathrm{j}\sqrt{3}$, solving the eigenvalue problem

$$\begin{pmatrix} 1 - \mathrm{j}\sqrt{3} & 2 \\ -2 & -(1 + \mathrm{j}\sqrt{3}) \end{pmatrix} v_1 = 0$$

gives the eigenvector $v_1 = \begin{pmatrix} -2 \\ 1 - \mathrm{j}\sqrt{3} \end{pmatrix}$. We do not need the second eigenvalue for the real solution. It can serve as a check. We substitute the first eigenvalue and the associated eigenvector into Equation 6.18:

$$y = \mathrm{e}^{(2 + \mathrm{j}\sqrt{3})x} \begin{pmatrix} -2 \\ 1 - \mathrm{j}\sqrt{3} \end{pmatrix}.$$

Euler's formula (Equation 6.15) gives

$$y = \mathrm{e}^{2x}\left[\cos(\sqrt{3}x) + \mathrm{j}\sin(\sqrt{3}x)\right]\begin{pmatrix}-2\\1-\mathrm{j}\sqrt{3}\end{pmatrix}.$$

Multiplying out with the eigenvector yields

$$y = \mathrm{e}^{2x}\begin{pmatrix}-2\cos(\sqrt{3}x) - 2\mathrm{j}\sin(\sqrt{3}x)\\ \cos(\sqrt{3}x) - \mathrm{j}\sqrt{3}\cos(\sqrt{3}x) + \mathrm{j}\sin(\sqrt{3}x) + \sqrt{3}\sin(\sqrt{3}x)\end{pmatrix},$$

sorted by real and imaginary parts,

$$y = \mathrm{e}^{2x}\left[\begin{pmatrix}-2\cos(\sqrt{3}x)\\ \cos(\sqrt{3}x) + \sqrt{3}\sin(\sqrt{3}x)\end{pmatrix} + \mathrm{j}\begin{pmatrix}-2\sin(\sqrt{3}x)\\ -\sqrt{3}\cos(\sqrt{3}x) + \sin(\sqrt{3}x)\end{pmatrix}\right].$$

A real fundamental system is thus

$$\mathrm{Re}(y) = \mathrm{e}^{2x}\begin{pmatrix}-2\cos(\sqrt{3}x)\\ \cos(\sqrt{3}x) + \sqrt{3}\sin(\sqrt{3}x)\end{pmatrix}, \quad \mathrm{Im}(y) = \mathrm{e}^{2x}\begin{pmatrix}-2\sin(\sqrt{3}x)\\ -\sqrt{3}\cos(\sqrt{3}x) + \sin(\sqrt{3}x)\end{pmatrix}.$$

The general solution of the homogeneous system is their linear combination

$$y = \mathrm{e}^{2x}\left[C_1\begin{pmatrix}-2\cos(\sqrt{3}x)\\ \cos(\sqrt{3}x) + \sqrt{3}\sin(\sqrt{3}x)\end{pmatrix} + C_2\begin{pmatrix}-2\sin(\sqrt{3}x)\\ -\sqrt{3}\cos(\sqrt{3}x) + \sin(\sqrt{3}x)\end{pmatrix}\right] \qquad (C_1, C_2 \in \mathbb{R}). \qquad \square$$

7. Systems of Second-Order Linear Differential Equations in Python

Table 7.1 summarizes the symbols of this chapter together with their type and the place where they are defined.

Table 7.1.: Symbols in Chapter 07

Symbol	Meaning	Type	Defined in
c	Spring constant	Constant	Theorem 2.7
c_i	Spring constant of spring i (c_{12}: coupling spring)	Constant	Example 5.1, Section 7.2
d	Differential (dx, $\frac{\mathrm{d}y}{\mathrm{d}x}$)	Operator	Section 7.1
f_1, f_2	Right-hand sides of a system of two second-order ODEs	Function	Section 7.1
m	Mass	Constant	Theorem 2.6
m_i	Mass i	Constant	Section 7.2
t	Time	Scalar	Equation 2.8
v	Velocity, $v = \dot{x}$; v_0 initial velocity	Scalar	Figure 2.2
x_1, x_2	Displacement from the rest position; x_i displacement of mass i; x_0 initial displacement	Function	Theorem 2.7
$y_1, \dots, y_4$	Components of the state vector S in the reduction of order ($y_1 = y$, $y_2 = y'$, …) or of the solution vector y of a system	Function	Theorem 2.14
local	Unknown functions $x(t)$, $y(t)$ and derivative $w = \mathrm{d}y/\mathrm{d}t$ in the introductory example	local	Section 7.1
$\dot{x}, \ddot{x}$	First and second derivative with respect to time t	Operator	Equation 2.8

7.1. Introduction

A system of second-order linear ODEs can be solved in Python with the function `scipy.integrate.solve_ivp` from the `scipy` library, which was developed specifically for initial value problems of ODE systems. To this end, a second-order system first has to be converted into a first-order system, because `solve_ivp` and similar functions operate on first-order ODEs.

As an example, consider a system of two second-order ODEs for the unknown functions $x(t)$ and $y(t)$:

$$\frac{\mathrm{d}^2x}{\mathrm{d}t^2} = f_1\left(x, y, \frac{\mathrm{d}x}{\mathrm{d}t}, \frac{\mathrm{d}y}{\mathrm{d}t}, t\right),$$
$$\frac{\mathrm{d}^2y}{\mathrm{d}t^2} = f_2\left(x, y, \frac{\mathrm{d}x}{\mathrm{d}t}, \frac{\mathrm{d}y}{\mathrm{d}t}, t\right).$$

New variables for the first derivatives, $v = \frac{\mathrm{d}x}{\mathrm{d}t}$ and $w = \frac{\mathrm{d}y}{\mathrm{d}t}$, turn it into a system of four first-order ODEs:

$$\begin{aligned}
\frac{\mathrm{d}x}{\mathrm{d}t} &= v, \\
\frac{\mathrm{d}v}{\mathrm{d}t} &= f_1(x, y, v, w, t), \\
\frac{\mathrm{d}y}{\mathrm{d}t} &= w, \\
\frac{\mathrm{d}w}{\mathrm{d}t} &= f_2(x, y, v, w, t).
\end{aligned}$$

Example 7.1 (Two Harmonic Oscillators in Python)**.** For $f_1 = -x$ and $f_2 = -y$ (two uncoupled harmonic oscillators), the system can be solved in Python as follows:

```
import numpy as np
from scipy.integrate import solve_ivp
import matplotlib.pyplot as plt

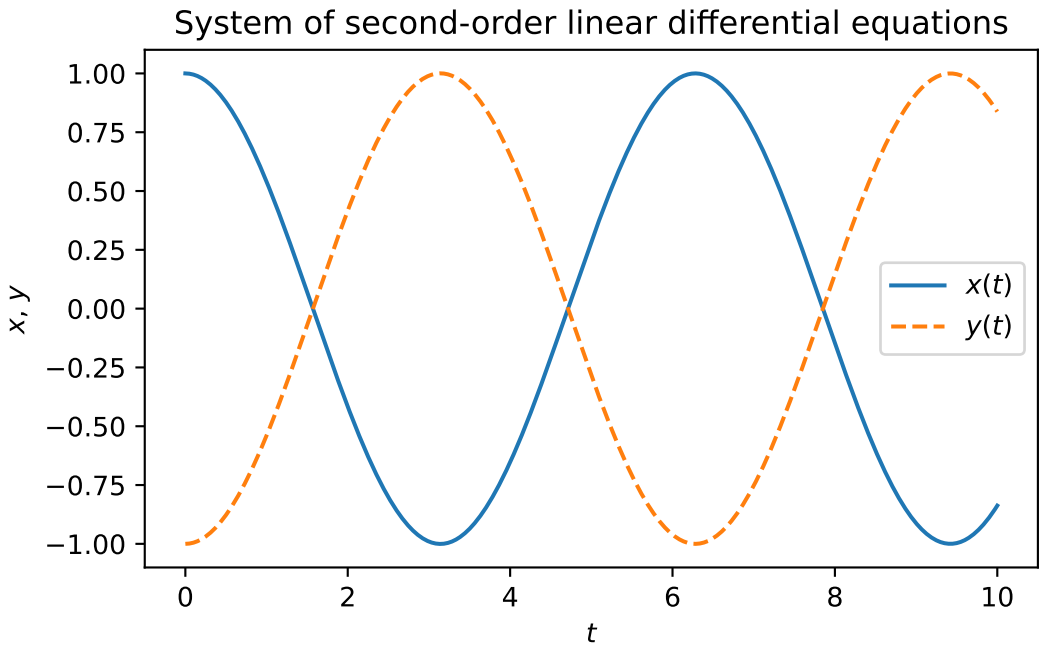
# Define the differential equations
def system(t, z):
    x, v, y, w = z
    dxdt = v
    dvdt = -x  # Example for f1: -x
    dydt = w
    dwdt = -y  # Example for f2: -y, a simple harmonic oscillator system
    return [dxdt, dvdt, dydt, dwdt]

# Initial conditions: x(0) = 1, dx/dt(0) = 0, y(0) = -1, dy/dt(0) = 0
z0 = [1.0, 0.0, -1.0, 0.0]

# Time span over which the solution is computed
t_span = (0, 10)
t_eval = np.linspace(*t_span, 1000)

# Solve the ODE system
sol = solve_ivp(system, t_span, z0, t_eval=t_eval)

# Plot the results
plt.plot(sol.t, sol.y[0], label=r'$x(t)$')
plt.plot(sol.t, sol.y[2], label=r'$y(t)$', linestyle='--')
plt.legend()
plt.xlabel(r'$t$')
plt.ylabel(r'$x$, $y$')
plt.title('System of second-order linear differential equations')
plt.show()
```



At each point in time, the state vector `z` contains the quantities x, v, y, and w (in the code `x`, `v`, `y`, `w`). The function `system()` defines the ODE system. `solve_ivp` solves it over the time span `t_span` with the initial conditions `z0`. The figure shows the curves of $x(t)$ and $y(t)$. □

This template can be adapted to a specific system of second-order linear ODEs by changing the expressions for `dvdt` and `dwdt`, i.e., f_1 and f_2, in the function `system()`.

7.2. Coupled Springs

We now solve the equations of motion of the two coupled spring-mass systems from Example 5.1 numerically. They have identical masses m, identical spring constants c, and a coupling spring with spring constant c_{12}. By Equation 5.1 and Equation 5.2, their equations of motion form a system of two coupled second-order ODEs:

$$\begin{aligned} m\ddot{x}_1 &= -cx_1 - c_{12}(x_1 - x_2), \\ m\ddot{x}_2 &= -cx_2 - c_{12}(x_2 - x_1). \end{aligned}$$

We generalize it to different masses m_1 and m_2 and different spring constants c_1 and c_2 of the two outer springs:

$$\begin{aligned} m_1\ddot{x}_1 &= -c_1x_1 - c_{12}(x_1 - x_2), \\ m_2\ddot{x}_2 &= -c_2x_2 - c_{12}(x_2 - x_1). \end{aligned}$$

To convert it into a first-order system, we introduce, as in Theorem 2.14, the state variables $y_1 = x_1$, $y_2 = x_2$, $y_3 = \dot{x}_1$, and $y_4 = \dot{x}_2$ and obtain four first-order ODEs:

$$\begin{aligned} \dot{y}_1 &= y_3, \\ \dot{y}_2 &= y_4, \\ m_1\dot{y}_3 &= -c_1y_1 - c_{12}(y_1 - y_2), \\ m_2\dot{y}_4 &= -c_2y_2 - c_{12}(y_2 - y_1). \end{aligned}$$

Example 7.2 (Coupled Spring-Mass Systems in Python)**.** The following code solves this system for $m_1 = m_2 = 1$, $c_1 = c_2 = 1$, and $c_{12} = 4$ (in the code `m1`, `m2`, `c1`, `c2`, `c12`). The state vector `y` holds the state variables $y_1, \dots, y_4$ under the names `x1`, `x2`, `v1`, `v2`. The initial conditions $x_1(0) = 1$, $x_2(0) = -1$, and $\dot{x}_1(0) = \dot{x}_2(0) = 0$ excite the antiphase normal mode from Remark 5.1.

```
import numpy as np
from scipy.integrate import solve_ivp
import matplotlib.pyplot as plt
from numpy import pi

# Parameters
m1, m2 = 1.0, 1.0  # masses
c1, c2 = 1.0, 1.0  # spring constants of the individual springs
c12 = 4.0  # spring constant of the coupling spring

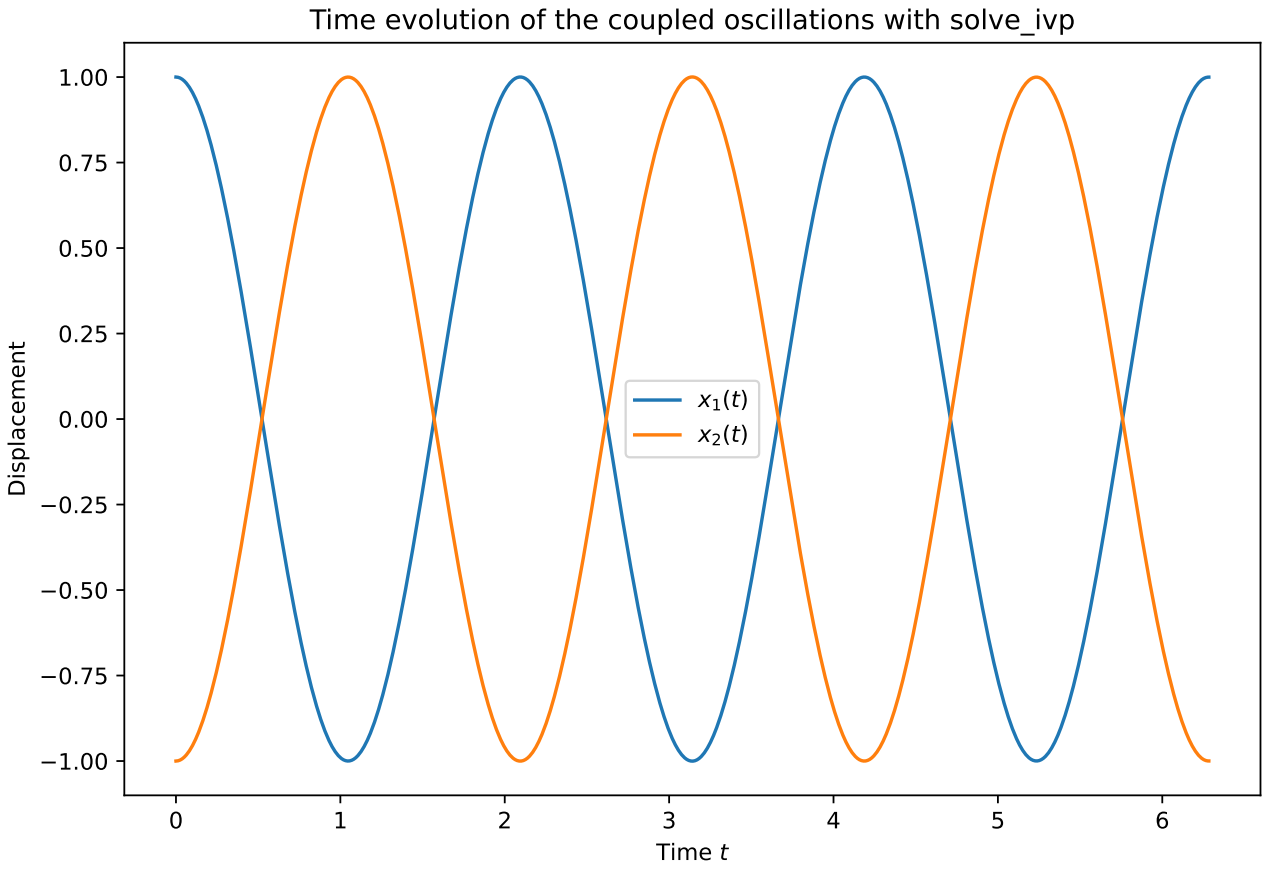
def dgl_system(t, y, m1, m2, c1, c2, c12):
    x1, x2, v1, v2 = y
    dx1dt = v1
    dx2dt = v2
    dv1dt = -(c1*x1 + c12*(x1-x2)) / m1
    dv2dt = -(c2*x2 + c12*(x2-x1)) / m2
    return [dx1dt, dx2dt, dv1dt, dv2dt]

# Initial conditions: x1(0) = 1, x2(0) = -1, v1(0) = v2(0) = 0
y0 = [1.0, - 1.0, 0.0, 0.0]

# Time span over which the solution is computed
t_span = (0, 2*pi)  # start and end
t_eval = np.linspace(*t_span, 1000)  # time points at which the solution is reported

# Compute the solution
sol = solve_ivp(dgl_system, t_span, y0, args=(m1, m2, c1, c2, c12), t_eval=t_eval)

# Plot the solution
plt.figure(figsize=(9, 6))
plt.plot(sol.t, sol.y[0], label=r'$x_1(t)$')
plt.plot(sol.t, sol.y[1], label=r'$x_2(t)$')
plt.xlabel(r'Time $t$')
plt.ylabel('Displacement')
plt.title('Time evolution of the coupled oscillations with solve_ivp')
plt.legend()
plt.show()
```



□

Exercise 7.1 (Changing the Initial Conditions)**.**

- Write a function to which different initial conditions can be passed, and use it to investigate the effect of different initial conditions on the system.
- Investigate the behavior of the system for different masses and spring constants of the two springs (e.g., $m_1 \neq m_2$, $c_1 \neq c_2$). □

8. Four Special Matrices K_n, C_n, T_n, and B_n

This chapter is based on Strang (2010). It treats four special families of matrices: K_n, C_n, T_n, and B_n. The family C_n in particular is important for the solution of Exercise 11.1. The symbols of this chapter are listed in Table 8.1.

Table 8.1.: Symbols in Chapter 08

Symbol	Meaning	Type	Defined in
a	Constant coefficient of a linear ODE ($y' + ay = g(x)$; growth rate in $y' = ay$)	Constant	Section 8.5.2
c	Spring constant	Constant	Theorem 2.7
c_i	Spring constant of spring i (c_{12}: coupling spring)	Constant	Section 8.6
d	Differential (dx, $\frac{\mathrm{d}y}{\mathrm{d}x}$)	Operator	Example 2.1
e	Euler's number	Constant	Example 2.1
e, e_i	Elongation (strain) of spring i; vector $e = Au$	Vector	Section 8.6
f, f_j	Vector of external forces (right-hand side of $Ku = f$); f_j force on mass j, f_0 reaction force at the upper end	Vector	Section 8.1
g	Gravitational acceleration	Constant	Equation 2.28
h	Step size of an integration scheme; grid spacing $h = 1/(n+1)$	Scalar	Definition 3.1, Section 8.5.5
i	Running index (component, row, eigenvalue number, spring number)	Index	Section 8.1
j	Running index (row/column, mass, blade); typographically distinct from j	Index	Section 8.1
j	Imaginary unit, $\mathrm{j}^2 = -1$	Constant	Theorem 2.4
k	Running index (time/iteration step, mode, eigenvalue number)	Index	Definition 3.1
l_0	Rest length of a spring	Constant	Figure 8.5
ℓ	Running index (component of an eigenvector, solution number)	Index	Section 8.5.5
m	Mass	Constant	Theorem 2.6
m_i	Mass i	Constant	Section 8.6
n	Order of an ODE; dimension of a matrix or system; number of steps	Index	Section 8.1
r	Number of springs (rows of the difference matrix A)	Index	Section 8.6
t	Time	Scalar	Section 8.5.2

Symbol	Meaning	Type	Defined in
u, u_i	Displacement vector of the masses (unknown in $Ku = f$); u_i displacement of mass i	Vector	Section 8.1
v, v_i	Eigenvector; v_i eigenvector for the eigenvalue λ_i or μ_k	Vector	Definition 5.1
w, w_i	Internal spring force (tension) of spring i, vector $w = Ce$	Vector	Section 8.6
x	Independent variable (Ch. 02 §1–4, 03, 06, 08 eigenfunctions)	Scalar	Theorem 8.11
y	Unknown function $y(x)$ or vector of unknown functions; $y', y'', y^{(n)}$ derivatives	Function	Section 8.5.2
$y_k(x)$	Eigenfunction of $-y'' = \lambda y$ for the boundary condition	Function	Theorem 8.11
z	Complex number	Scalar	Figure 8.2
A	Matrix (coefficient, system, or difference matrix, $e = Au$)	Matrix	Definition 5.1
A, B, A_i, B_i	Coefficients of the real modal solution, $q(t) = \sum(A_k \cos\omega_k t + B_k \sin\omega_k t)\hat{q}_k$ (A_i, B_i in 08)	Scalar	Theorem 8.19, Equation 8.3
B	Further matrix (similar matrix $A = P^{-1}BP$; $B = A - \lambda E$ in the brute-force ("Hau-Ruck") method)	Matrix	Definition 5.2
B_n	Family of matrices with free ends: tridiagonal $-1, 2, -1$, first and last diagonal entry 1	Matrix	Section 8.3
C	Constant of integration; $C_1, \ldots, C_n$ constants of integration or coefficients of the general solution	Constant	Example 2.1
C	Diagonal matrix of the spring constants c_i in $K = A^T CA$	Matrix	Section 8.6
C_n	Circulant matrix ($-1, 2, -1$ cyclic); C_N for the ring of N masses	Matrix	Definition 8.2
D	Diagonal matrix of the pivots in LDL^T (only in the name of the factorization)	Matrix	Theorem 8.1
E	Identity matrix	Matrix	Definition 5.1
F	Force; $F(t)$ external force, F_0 its amplitude, F_1, F_2, F_3 component forces, F_R restoring force, F_G weight	Scalar	Theorem 2.6
H_3	"Hanging matrix" (fixed-free, 3×3)	Matrix	Section 8.6
J	Jordan matrix	Matrix	Table 8.2

Symbol	Meaning	Type	Defined in
K	Stiffness matrix, $K = A^T C A$	Matrix	Definition 8.5
K_n	Family of second difference matrices: tridiagonal $-1, 2, -1$ (fixed-fixed)	Matrix	Section 8.1
K_{ij}	Element of the matrix K in row i, column j	Scalar	Section 8.1
L	Lower triangular matrix in LDL^T or LL^T (Cholesky)	Matrix	Theorem 8.1
M	Mass matrix, $M = \text{diag}(m_1, \dots, m_n)$	Matrix	Definition 8.5
P	Invertible matrix of a similarity transformation, $A = P^{-1} B P$	Matrix	Definition 5.2
Q	Orthogonal matrix, $Q^T Q = E$; columns are orthonormal eigenvectors	Matrix	Theorem 8.10
T_n	Fixed-free family of matrices: tridiagonal $-1, 2, -1$, first diagonal entry 1	Matrix	Section 8.2
U	Upper triangular matrix from elimination	Matrix	Section 8.2
U, Σ, V	Factors of the singular value decomposition $A = U \Sigma V^T$ (only in Table 8.2)	Matrix	Table 8.2
V	Eigenvector matrix (columns $v_1, \dots, v_n$), $V^{-1} A V = \Lambda$	Matrix	Theorem 8.6
$\mathbb{R}$, $\mathbb{R}^n$	Set of real numbers or of n-tuples	Set	Figure 8.4
α	Rigid-body displacement: all masses by the same amount, $u = (\alpha, \dots, \alpha)$	Scalar	Section 8.1
γ_i	Coefficient of a linear combination (weights of the RK2 method; expansion of the starting vector in eigenvectors; basis representation in the eigenspace)	Scalar	Example 8.2
$\Delta t, \Delta x$	Time step; spatial grid spacing	Scalar	Section 8.7, Section 8.5.5
θ	Angle: $\theta = k\pi/(n+1)$ or $2\pi k/n$ (discrete sine/exponential function)	Scalar	Section 8.5.5
λ	Eigenvalue or root of the characteristic equation (exponent in the ansatz $e^{\lambda t}$); λ_i, λ_k	Scalar	Definition 5.1
Λ	Diagonal matrix of the eigenvalues	Matrix	Theorem 8.6
π	The number pi (italic, KaTeX exception)	Constant	–

Symbol	Meaning	Type	Defined in
ω	Angular frequency (excitation angular frequency; imaginary part of an eigenvalue; frequency of an eigenfunction)	Scalar	Theorem 8.11
ω_0	Natural angular frequency of the undamped single-mass oscillator, $\omega_0 = \sqrt{c/m}$ (simple pendulum: $\sqrt{g/l}$)	Constant	Equation 2.8
ζ	n-th root of unity, $\zeta = \mathrm{e}^{2\pi\mathrm{j}/n}$; $\overline{\zeta}$ conjugate	Scalar	Theorem 8.15
$y', y'', y^{(n)}$	First, second, n-th derivative with respect to the independent variable	Operator	Example 2.1
$\dot{x}, \ddot{x}$	First and second derivative with respect to time t	Operator	Equation 2.8
$\sum$	Sum	Operator	Equation 8.3
A^T	Transpose	Operator	Section 8.1
A^{-1}	Inverse	Operator	Section 8.1
det	Determinant	Operator	Theorem 8.5
Re, Im	Real and imaginary parts	Operator	Table 8.2
diag	Diagonal matrix formed from entries	Operator	Section 8.7
DST	Matrix of the discrete sine transform	Matrix	Definition 8.6
$\overline{\lambda}$	Complex conjugate	Operator	Theorem 8.16
$\|v\|$, $\|z\|$	Euclidean norm, absolute value	Operator	Section 8.5.6
local	A, B (example matrices in Example 8.1 and Section 8.5.6.1), y_ℓ (samples $\sin(\ell\theta)$, $\cos(\ell\theta)$ in Section 8.5.5), u, v (arbitrary vectors in the rows "reflection" and "rank 1" of Table 8.2)	local	where used

8.1. The Matrices K_n and C_n

> **Literature Note**
>
> This section refers to Chapter 1.1 in Strang (2010).

An $r \times n$ matrix has r rows, n columns, and rn entries. We operate on these rows and columns to solve systems of linear equations $Au = f$ and eigenvalue problems $Av = \lambda v$. From the inputs A and f we obtain (with the help of software such as Python) the outputs u, or λ and v. A fast and stable algorithm is extremely important here.

Matrices are used to store information. In applied mathematics, however, we often regard them as an "operator": the matrix A acts on vectors u and produces other vectors Au. The components of u have a meaning: they stand for displacements, pressures, voltages, prices, or concentrations. The operator A also has a meaning: it forms differences. So the components of Au are pressure differences, voltage drops, or price differences.

We consider four special families of matrices. They are simple, useful, and truly fundamental. We first look at the properties of these special matrices K_n, C_n, T_n, and B_n. Some properties are obvious, others are hidden. It is a pleasure to do linear algebra with matrices that really matter.

We begin by describing the matrices K_2, K_3, K_4 from the first family, whose diagonals carry the entries -1, 2, and -1:

$$K_2 = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix},$$

$$K_3 = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix},$$

$$K_4 = \begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{pmatrix}.$$

What characterizes the matrices K_2, K_3, K_4, and finally the $n \times n$ matrix K_n?

1. The matrices are *symmetric*. The entry in row i, column j also appears in row j, column i, so $K_{ij} = K_{ji}$. The entries are mirrored across the main diagonal. For the whole matrix, symmetry reads $K = K^T$.
2. The matrices K_n are *sparse*. For large n, the vast majority of the entries are zero. K_{1000} has a million entries, but only $1000 + 999 + 999$ of them are nonzero.
3. The nonzero entries lie on a "band" around the main diagonal. Hence every matrix K_n is a banded matrix. There is only one superdiagonal and one subdiagonal, so these matrices are *tridiagonal*.

Because K is a tridiagonal matrix, $Ku = f$ can be solved quickly. If the unknown vector u has a thousand components, we can determine them in a few thousand steps (which takes a small fraction of a second on a computer). For a full matrix of order $n = 1000$, solving $Ku = f$ would take hundreds of millions of steps. Of course, we must first ask whether the linear equations have a solution at all. We come to this question shortly.

4. The matrices have *constant diagonals*. This property practically calls for a Fourier transform. It indicates that some quantity does not change as we move in space or time. The problem is translation-invariant or time-invariant. The coefficients are constant. The tridiagonal matrix is completely determined by the three numbers -1, 2, -1. These matrices are really "second difference matrices".

The whole world of Fourier transforms is tied to matrices with constant diagonals. In signal processing, the matrix $K/4$ serves as a "high-pass filter". The product $(K/4)u$ selects the rapidly varying (high-frequency) component of a vector u. It is a convolution with $\frac{1}{4}(-1, 2, -1)$.

Mathematicians call the matrix K a Toeplitz matrix, and Python uses this name as well: the command `K4 = toeplitz([2, -1, 0, 0])` constructs K_4 from row 1.

```
import numpy as np
from scipy.linalg import toeplitz
K4 = toeplitz([2, -1, 0, 0])
print(K4)
```

```
[[ 2 -1  0  0]
 [-1  2 -1  0]
 [ 0 -1  2 -1]
 [ 0  0 -1  2]]
```

Definition 8.1 (Toeplitz Matrices)**.** A *Toeplitz matrix* is a matrix in which all diagonals running from top left to bottom right have the same value. For example, the following matrix is a Toeplitz matrix:

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 1 & 2 & 3 \\ 9 & 5 & 1 & 2 \\ 7 & 9 & 5 & 1 \end{pmatrix}.$$ □

If we make two small changes in K_n, we obtain the family of matrices C_n: we enter the element -1 in the top right and bottom left corners. This completes two (circulating) diagonals. Each row vector of C_4 is now shifted one entry to the right relative to the row vector above it.

Definition 8.2 (Circulant Matrix)**.** A *circulant matrix* (also "periodic matrix" or "cyclic convolution") is a Toeplitz matrix whose diagonals wrap around cyclically: each row arises from the row above it by a shift of one entry to the right, and the entry pushed out at the right edge re-enters at the left edge. The matrices C_n are circulant. For $n = 4$,

$$C_4 = \begin{pmatrix} 2 & -1 & 0 & -1 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ -1 & 0 & -1 & 2 \end{pmatrix}. \qquad \square$$

```
import numpy as np
from scipy.linalg import toeplitz
C4 = toeplitz([2, -1, 0, -1])
print(C4)
```

```
[[ 2 -1  0 -1]
 [-1  2 -1  0]
 [ 0 -1  2 -1]
 [-1  0 -1  2]]
```

The matrix C_4 is singular. It is not invertible. Its determinant is zero. Instead of computing this determinant, it is better to find a nonzero vector u that solves $C_4u = 0$. (If C_4 had an inverse, the only solution to $C_4u = 0$ would be the zero vector. We could multiply by C_4^{-1} to find that $u = 0$.) For this matrix, it is the column vector of all ones (that is, $u = (1, 1, 1, 1)$) that solves the equation $C_4u = 0$.

The sum of the columns of the matrix C_4 is the zero column. The vector $u = (1, \quad 1, \quad 1, \quad 1)^T$ lies in the nullspace of C_4. The nullspace contains all solutions to $C_4u = 0$.

If the entries in every row of a matrix sum to zero, the matrix is certainly singular. The reason is again the column vector of all ones. In the matrix multiplication C_4u, all column vectors are added. Each sum is zero. The constant vector $u = (1, 1, 1, 1)$, or $u = (\alpha, \alpha, \alpha, \alpha)$, from the nullspace behaves like the constant C when integrating a function. In calculus, the "constant of integration" cannot be determined from the derivative. In algebra, the constant α in $u = (\alpha, \alpha, \alpha, \alpha)$ cannot be determined from the equation $C_4u = 0$.

5. All matrices $K = K_n$ are *invertible*. Unlike the matrices C_n, they are not singular. There exists a square matrix K^{-1} such that $K^{-1}K = E$. The matrix E is the identity matrix. If a square matrix has a left inverse, then $KK^{-1} = E$ holds as well. If the matrix K is symmetric, this "inverse matrix" is symmetric too. However, the matrix K^{-1} is not sparse.

The invertibility of a matrix is not easy to see at a glance. In theory, one could compute the determinant. Only if $\det K = 0$ does no inverse exist, because computing K^{-1} involves a division by $\det K$. In practice, however, the determinant is almost never computed. It is a cumbersome way to determine $u = K^{-1}f$.

Instead, we proceed with the elimination steps that solve $Ku = f$. These steps simplify the matrix so that it becomes triangular. This gives us the pivots of the matrix.

Definition 8.3 (Pivots)**.** A *pivot* is the entry of a matrix that is used during an elimination step to simplify the system of equations. It is typically the first nonzero entry in the current row and lies on the main diagonal. Three operations are allowed to transform the matrix into an upper triangular matrix: adding a multiple of one row to another, multiplying a row by a nonzero scalar, and interchanging two rows. These operations use the pivots by making all entries below the pivots zero. The pivots are therefore the diagonal entries of the resulting upper triangular matrix. $\square$

The nonzero pivots on the main diagonal of the triangular matrix show that the original matrix K is invertible. (Important note: we do not want and do not need K^{-1} to determine $u = K^{-1}f$. The inverse would be a full matrix with positive entries. We compute only the solution vector u.)

6. The symmetric matrices K_n are *positive definite*. This term may be new to you. Among other things, this chapter is meant to explain what this important property means (K_4 has it, C_4 does not). Let me first contrast positive definiteness and invertibility in terms of "pivots" and "eigenvalues".

Definition 8.4 (Positive Definiteness)**.** A symmetric $n \times n$ matrix A is called *positive definite* if $u^TAu > 0$ holds for all vectors $u \neq 0$. It is called *positive semidefinite* if $u^TAu \geq 0$ holds for all vectors u. $\square$

Theorem 8.1 (Pivots)**.** *An invertible matrix has n nonzero pivots. A positive definite symmetric matrix has n positive pivots.* □

Proof. We sketch the argument for both parts. A matrix is invertible if and only if elimination transforms it into an upper triangular matrix without a zero row, that is, if and only if all n pivots are nonzero. A zero row would mean that the system of equations has no unique solution. For a symmetric matrix A, elimination without row exchanges yields the factorization $A = LDL^T$ with a lower triangular matrix L whose diagonal entries are 1 and the diagonal matrix D of the pivots. Then $u^T Au = (L^T u)^T D(L^T u)$, and because L^T is invertible, $L^T u$ runs through all nonzero vectors as u does. Hence A is positive definite if and only if all pivots in D are positive. □

Example 8.1 (Pivots and Elementary Row Operations)**.** The matrix

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{pmatrix}$$

has the pivots 1, 2, and 3 on the main diagonal, all of which are greater than zero. The matrix is positive definite and invertible. If we multiply the last row by -1 (an elementary row operation), we obtain the matrix

$$B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & -3 \end{pmatrix}.$$

The pivots of the first two rows are still 1 and 2, but the pivot of the last row is -3. The matrix B is not positive definite, but it is invertible, since all pivots are nonzero. □

Theorem 8.2 (Eigenvalues)**.** *An invertible matrix has n nonzero eigenvalues. A positive definite symmetric matrix has n positive eigenvalues.* □

Positive definiteness can be tested by means of the pivots and eigenvalues. The matrix C_4 is not positive definite because it is singular. In fact, the matrix C_4 has three positive pivots and eigenvalues, so it "almost" passes the test. But its fourth eigenvalue is zero (the matrix is singular). Since none of the eigenvalues is negative ($\lambda \geq 0$), C_4 is positive semidefinite (Definition 8.4).

When we solve $Ku = f$ by elimination, the pivots appear on the main diagonal. The eigenvalues appear in $Kv = \lambda v$. Positive definiteness can also be tested by means of the determinant (but not simply by $\det K > 0$).

8.2. From K_n to T_n

Besides the two families K_n and C_n, there are two further families you need to know. Matrices from these families are symmetric and tridiagonal, like those from the family K_n. For matrices from the family T_n, however, the entry in position $(1, 1)$ is not 2 but 1:

$$T_1 = (1)\,,$$

$$T_2 = \begin{pmatrix} 1 & -1 \\ -1 & 2 \end{pmatrix},$$

$$T_3 = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix},$$

$$T_4 = \begin{pmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{pmatrix}.$$

This first row (T stands for "top") has to do with a new boundary condition whose meaning we will soon understand. For now, we use the matrix T_3 as a perfect demonstration example for elimination. Row operations make the entries below the main diagonal zero.

In two elimination steps we turn the matrix T_3 into the upper triangular matrix U. In the first step we add row 1 to row 2 so that all entries below the first pivot are zero. In the second step we add the new row 2 to row 3 to obtain U:

$$T_3 = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix} \quad \rightarrow \quad \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & -1 & 2 \end{pmatrix} \quad \rightarrow \quad \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix} = U.$$

All pivots of the matrix T_3 are 1. We can apply our test for invertibility (all pivots must be nonzero). The matrix T_3 even passes the test for positive definiteness (three positive pivots). In fact, every matrix T_n of this family is positive definite, because all pivots are always 1.

The matrix U has an inverse (which is automatically an upper triangular matrix). What is remarkable about this particular inverse U^{-1} is that all entries above the main diagonal are 1:

$$U_3^{-1} = \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix}^{-1} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}.$$

From this we learn that the inverse of a 3×3 "difference matrix" is a 3×3 "sum matrix". The product $U^{-1}U$ is the identity matrix E. The matrix U forms differences, its inverse U^{-1} forms sums. Taking differences first and then sums returns the original vector (u_1, u_2, u_3).

U produces differences, that is,

$$\begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} u_1 - u_2 \\ u_2 - u_3 \\ u_3 - 0 \end{pmatrix},$$

and U^{-1} produces sums, that is,

$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u_1 - u_2 \\ u_2 - u_3 \\ u_3 - 0 \end{pmatrix} = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix}.$$

8.3. From T_n to B_n

In the fourth family B_n, the last entry is also 1 instead of 2. The new boundary condition applies to both ends (B stands for "both"). The matrices B_n are symmetric and tridiagonal. As you will see shortly, they are not invertible. The matrices B_n are positive semidefinite, but not positive definite:

$$B_1 = (0),$$

$$B_2 = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix},$$

$$B_3 = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}.$$

Again, elimination reveals the properties of the matrix. The first $n-1$ pivots are 1, because nothing has changed here compared with the matrix T_n. But since the last entry of the matrix B_3 is now 1, the last entry of the matrix U changes:

$$B_3 = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix} \quad \rightarrow \quad \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & -1 & 1 \end{pmatrix} \quad \rightarrow \quad \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{pmatrix} = U.$$

There are only two pivots. (A pivot must be nonzero.) The last matrix U is obviously not invertible. Its determinant is zero because the third row contains only zeros. The constant vector $(1, 1, 1)$ lies in the nullspace of the matrix U and hence also in the nullspace of the matrix B_3.

The purpose of elimination was to simplify a linear system such as $B_3 u = 0$ without changing its solutions. In this case, we could also have seen that the matrix B_3 is not invertible from the fact that the entries in every row sum to zero. Then the sum of the columns is the zero column. This is exactly what we obtain when we multiply the matrix B_3 by the vector $(1, 1, 1)$.

Let me summarize this section in four points (all matrices considered are symmetric):

1. The matrices K_n and T_n are positive definite and therefore invertible.
2. The matrices C_n and B_n are singular and positive semidefinite.
3. The nullspaces of the matrices C_n and B_n contain the constant vectors $u = (\alpha, \alpha, \dots, \alpha)$. Their columns are dependent.
4. The nullspaces of the matrices K_n and T_n contain only the zero vector $u = (0, 0, \dots, 0)$. Their columns are independent.

8.4. Matrices in Python

8.4.1. Elementary Matrix Functions `eye` and `ones`

As a first step, we want to construct the matrices K_n. For $n = 3$ we can enter the 3×3 matrix row by row in square brackets:

```
import numpy as np
K3 = np.array([[2, -1, 0], [-1, 2, -1], [0, -1, 2]])
print(K3)
```

```
[[ 2 -1  0]
 [-1  2 -1]
 [ 0 -1  2]]
```

For large matrices this approach is too time-consuming. We can also generate the matrix K_8 with the command `eye()`: the function `eye(N, k)` generates an identity matrix. The parameter `k` specifies the position of the diagonal: `k=0` denotes the main diagonal, `k=1` the first superdiagonal, and `k=-1` the first subdiagonal. The matrix K_8 is symmetric.

```
K8 = 2*np.eye(8) - np.eye(8, k=1) - np.eye(8, k=-1)
print(K8)
```

```
[[ 2. -1.  0.  0.  0.  0.  0.  0.]
 [-1.  2. -1.  0.  0.  0.  0.  0.]
 [ 0. -1.  2. -1.  0.  0.  0.  0.]
 [ 0.  0. -1.  2. -1.  0.  0.  0.]
 [ 0.  0.  0. -1.  2. -1.  0.  0.]
 [ 0.  0.  0.  0. -1.  2. -1.  0.]
 [ 0.  0.  0.  0.  0. -1.  2. -1.]
 [ 0.  0.  0.  0.  0.  0. -1.  2.]]
```

8.4.2. The Function `toeplitz`

The constant diagonal entries make K a Toeplitz matrix. The command `toeplitz()` generates the matrix K when each diagonal contains only the same entry 2, -1, or 0.

```
K8 = toeplitz([2, -1, 0, 0, 0, 0, 0, 0])
print(K8)
```

```
[[ 2 -1  0  0  0  0  0  0]
 [-1  2 -1  0  0  0  0  0]
 [ 0 -1  2 -1  0  0  0  0]
 [ 0  0 -1  2 -1  0  0  0]
 [ 0  0  0 -1  2 -1  0  0]
 [ 0  0  0  0 -1  2 -1  0]
 [ 0  0  0  0  0 -1  2 -1]
 [ 0  0  0  0  0  0 -1  2]]
```

Here we use the command `[0]*6` for a zero vector, with which we generate the six zeros in the first row of the matrix K_8:

```
K8 = toeplitz([2, -1] + [0]*6)
print(K8)
```

```
[[ 2 -1  0  0  0  0  0  0]
 [-1  2 -1  0  0  0  0  0]
 [ 0 -1  2 -1  0  0  0  0]
 [ 0  0 -1  2 -1  0  0  0]
 [ 0  0  0 -1  2 -1  0  0]
 [ 0  0  0  0 -1  2 -1  0]
 [ 0  0  0  0  0 -1  2 -1]
 [ 0  0  0  0  0  0 -1  2]]
```

For an unsymmetric matrix with constant diagonal entries, you need the command `toeplitz(col1, row1)`. With `col1 = [1, -1, 0, 0]` and `row1 = [1, 0, 0]`, this command generates a 4×3 matrix of backward differences that has nonzero entries on only two diagonals, namely the entries 1 on one diagonal and the entries -1 on the other.

8.4.3. The Function `KTBC(n)`

To generate the matrices T_n, B_n, and C_n from the matrix K_n, only individual entries need to be changed. The input is the size n, and four matrices are generated.

The function `K, T, B, C = KTBC(n)` generates the four special matrices under the assumption $n > 1$, using the following rules (in MATLAB notation after Strang (2010)):

1. `K = toeplitz([2 -1 zeros(1,n-2)]);`
2. `T = K; T(1,1) = 1;`
3. `B = K; B(1,1) = 1; B(n,n) = 1;`
4. `C = K; C(1,n) = -1; C(n,1) = -1;`

It can be implemented in Python as follows:

```
import numpy as np
from scipy.linalg import toeplitz
def KTBC(n):
    K = toeplitz([2, -1] + [0]*(n-2))
    T = K.copy()
    T[0,0] = 1
    B = K.copy()
    B[0,0] = 1
    B[n-1,n-1] = 1
    C = K.copy()
    C[0,n-1] = -1
    C[n-1,0] = -1
    return K, T, B, C

n= 4
K, T, B, C = KTBC(n)
print(f"K{n} = \n{K}")
print(f"T{n} = \n{T}")
print(f"B{n} = \n{B}")
print(f"C{n} = \n{C}")
```

```
K4 =
[[ 2 -1  0  0]
 [-1  2 -1  0]
 [ 0 -1  2 -1]
 [ 0  0 -1  2]]
T4 =
[[ 1 -1  0  0]
 [-1  2 -1  0]
 [ 0 -1  2 -1]
 [ 0  0 -1  2]]
B4 =
[[ 1 -1  0  0]
 [-1  2 -1  0]
 [ 0 -1  2 -1]
 [ 0  0 -1  1]]
C4 =
[[ 2 -1  0 -1]
 [-1  2 -1  0]
 [ 0 -1  2 -1]
 [-1  0 -1  2]]
```

Note

Strang (2010) treats further properties of the matrices K_n, T_n, B_n, and C_n: which boundary conditions these finite difference matrices correspond to, how elimination produces pivots in D and triangular factors in LDL^T, how point loads lead to the inverse matrices K^{-1} and T^{-1}, and why the eigenvalues and eigenvectors involve sines and cosines. We take up only some of these topics in order to develop solution ideas for Exercise 11.1, and refer to the book for a comprehensive treatment.

8.5. Eigenvalues and Eigenvectors of the Matrices K_n, T_n, B_n, and C_n

Literature Note

This section refers to Chapter 1.5 in Strang (2010).

This section begins with the equation $Av = \lambda v$ for an eigenvector v and its eigenvalue λ. For small matrices A we can solve $Av = \lambda v$ by starting from $\det(A - \lambda E) = 0$. You may already be familiar with this (for large matrices this method would be dreadful). There is no "elimination" that delivers the exact eigenvalue λ and the exact eigenvector v in finite time. Since λ is multiplied by v, the equation $Av = \lambda v$ is not linear.

A major success of numerical linear algebra is the development of fast and stable algorithms for computing eigenvalues (especially for symmetric matrices A). The Python command `eig(A)` does not return an equation but n numbers $\lambda_1, \dots, \lambda_n$. In this chapter, however, we deal with special matrices! For these we will therefore determine λ and v exactly.

8.5.1. Part I: $Av = \lambda v$, $A^k v = \lambda^k v$, and the Diagonalization of A

Almost every vector changes its direction when multiplied by a matrix A. Certain exceptional vectors v lie on the same line as Av. These are the eigenvectors. For an eigenvector, Av is the original vector v multiplied by a number λ.

The eigenvalue λ tells us whether the special vector v is stretched, shrunk, reversed, or left unchanged when multiplied by the matrix A. Possible cases are, for example, $\lambda = 2$ (stretching), $\lambda = \frac{1}{2}$ (shrinking), $\lambda = -1$ (reversal), or $\lambda = 1$ (steady state, since $Av = v$).

Theorem 8.3 (Eigenvalue Zero). *$\lambda = 0$ can also occur. If the nullspace contains nonzero vectors, then $Av = 0v$ holds for them. So the nullspace contains eigenvectors belonging to $\lambda = 0$.* □

Theorem 8.4 (The Zero Vector). *The zero vector is not an eigenvector. It is not changed by multiplication with a matrix. Although $A0 = \lambda 0$ holds for every λ, the zero vector does not count as an eigenvector by definition.* □

For our special matrices we will guess v and then determine λ. In the general case we determine λ first. To separate λ from v, we first rewrite the basic equation:

$$Av = \lambda v \text{ means } (A - \lambda E)v = 0.$$

The matrix $A - \lambda E$ must be singular. Its determinant must be zero. The eigenvector v lies in the nullspace of $A - \lambda E$. First we note that λ is an eigenvalue if and only if the shifted matrix $A - \lambda E$ is not invertible:

Theorem 8.5 (Eigenvalues: Characteristic Equation). *The number λ is an eigenvalue of A if and only if* $\det(A - \lambda E) = 0$. □

This "characteristic equation" $\det(A - \lambda E) = 0$ involves only the eigenvalue λ, not the eigenvector v. The determinant of $A - \lambda E$ is a polynomial of degree n in λ. By the fundamental theorem of algebra, this polynomial must have n roots $\lambda_1, \dots, \lambda_n$. Some of these eigenvalues may have a higher multiplicity or be complex. In these cases we have a little more work to do.

8.5.1.1. Diagonalization of a Matrix

When we deal with an eigenvector, multiplication by A reduces to multiplication by a number: $Av = \lambda v$. All the difficulties with the $n \times n$ multiplications disappear. Instead of having to treat a coupled system, we can now consider the eigenvectors one at a time. It is as if we had a diagonal matrix in which the couplings through off-diagonal entries are absent.

With the help of the eigenvalues, we can also easily determine the 100th power of a diagonal matrix. The matrix A turns into a diagonal matrix Λ when we use the eigenvectors correctly. This is the matrix form of our key idea. The only essential computation follows.

Theorem 8.6 (Diagonalization of a Matrix). *Suppose the $n \times n$ matrix A has n linearly independent eigenvectors $v_1, \dots, v_n$. These are the columns of an eigenvector matrix V. Then the matrix $V^{-1}AV = \Lambda$ is diagonal:*

$$V^{-1}AV = \Lambda = \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix} = \textit{eigenvalue matrix}. \tag{8.1}$$

□

With Equation 8.1, K_2^{100} can easily be determined using the eigenvalues and eigenvectors of the matrix K_2.

Example 8.2 (Eigenvalues and Eigenvectors of the Matrix K_2). We compute the eigenvalues and eigenvectors of the matrix K_2 numerically in Python.

```
import numpy as np
from scipy.linalg import eig
K2 = np.array([[2, -1], [-1, 2]])
eigenvalues, eigenvectors = eig(K2)
print(f"Eigenvalues of K2: {eigenvalues}")
print(f"Eigenvectors of K2: {eigenvectors}")
```

```
Eigenvalues of K2: [3.+0.j 1.+0.j]
Eigenvectors of K2: [[ 0.70710678  0.70710678]
 [-0.70710678  0.70710678]]
```

As expected, the matrices $K_2 - E$ and $K_2 - 3E$ are singular. Each nullspace is a line of eigenvectors. We choose v_1 and v_2 so that they have the simple components 1 and -1. Arbitrary (nonzero) multiples $\gamma_1 v_1$ and $\gamma_2 v_2$ would have been just as good as eigenvectors. Python chooses γ_1 and γ_2 with absolute value $\frac{1}{\sqrt{2}}$, because the eigenvectors then have length 1 (that is, they are unit vectors).

If we use SymPy, a free Python library for symbolic mathematics, we obtain the exact eigenvalues and eigenvectors:

```
import sympy as sp
K2 = sp.Matrix([[2, -1], [-1, 2]])
eigenvalues = K2.eigenvals()
eigenvectors = K2.eigenvects()
for i, (eigenvalue, multiplicity, eigenvector) in enumerate(eigenvectors):
    print(f"Eigenvalue {i+1}: {eigenvalue}")
    print(f"Eigenvector {i+1}: {eigenvector}\n")
```

```
Eigenvalue 1: 1
Eigenvector 1: [Matrix([
[1],
[1]])]

Eigenvalue 2: 3
Eigenvector 2: [Matrix([
[-1],
[ 1]])]
```

Thus we obtain the eigenvector $(1,1)$ for the eigenvalue $\lambda = 1$ and the eigenvector $(-1,1)$ for the eigenvalue $\lambda = 3$, that is, after normalization, $\frac{1}{\sqrt{2}}(1,1)$ and (after multiplication by -1) $\frac{1}{\sqrt{2}}(1,-1)$. These eigenvectors of the matrix

$$K_2 = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$$

are special (just like the matrix K_2 itself). □

Theorem 8.7 (Orthogonal Eigenvectors of Symmetric Matrices)**.** *Every symmetric $n \times n$ matrix has a complete set of n orthogonal unit eigenvectors.* □

8.5.2. Application to Systems of ODEs

A single ODE $\frac{dy}{dt} = ay$ has the general solution $y(t) = Ce^{at}$. The initial value $y(0)$ determines the constant C. The solution $y(t) = y(0)e^{at}$ decays for $a < 0$ and grows for $a > 0$. Decay stands for stability, growth for instability. If a is a complex number, its real part decides between growth and decay. The imaginary part ω provides an oscillation factor $e^{j\omega t} = \cos\omega t + j\sin\omega t$.

Now we consider two coupled ODEs for the components of the vector $u = (u_1, u_2)^T$, which together form the vector equation $\frac{du}{dt} = Au$:

$$\begin{aligned} \frac{du_1}{dt} &= 2u_1 - u_2, \\ \frac{du_2}{dt} &= -u_1 + 2u_2, \end{aligned}$$

or

$$\frac{d}{dt}\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$$

or equivalently

$$\frac{d}{dt}\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = K_2\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}.$$

The solution will still contain exponential functions $e^{\lambda t}$. But there is no longer a single growth rate as in e^{at}. The associated matrix $A = K_2$ has two eigenvalues $\lambda_1 = 1$ and $\lambda_2 = 3$. The solution therefore contains two exponential functions e^t and e^{3t}. They are multiplied by the eigenvectors $v_1 = (1,1)$ and $v_2 = (1,-1)$.

The correct way to find solutions is via the eigenvectors. The pure solutions $e^{\lambda t}v$ are eigenvectors that grow according to their own eigenvalue 1 or 3. We combine them:

$$u(t) = C_1 e^t v_1 + C_2 e^{3t} v_2$$

gives

$$u_1(t) = C_1 e^t + C_2 e^{3t}, \quad u_2(t) = C_1 e^t - C_2 e^{3t},$$

or in matrix notation

$$\begin{pmatrix} u_1(t) \\ u_2(t) \end{pmatrix} = C_1 \begin{pmatrix} 1 \\ 1 \end{pmatrix} e^t + C_2 \begin{pmatrix} 1 \\ -1 \end{pmatrix} e^{3t}.$$

This is the complete solution. The two constants C_1 and C_2 are determined by the two initial values $u_1(0)$ and $u_2(0)$. First convince yourself that each pure solution $e^{\lambda t} v$ solves the equation $\frac{du}{dt} = Au$:

Theorem 8.8 (Eigenvectors and Solutions)**.** *If v is an eigenvector of A for the eigenvalue λ, then $u(t) = e^{\lambda t} v$ solves the ODE $\frac{du}{dt} = Au$.*

□ □

Proof. We have $\frac{du}{dt} = \lambda e^{\lambda t} v = e^{\lambda t} Av = Au$. □

The number $e^{\lambda t}$ is just a factor by which all components of the eigenvector v are multiplied. This is the essential property of eigenvectors: they grow or decay at their own rate λ. The complete solution $u(t)$ is a combination of the pure *modes* $C_1 e^t v_1$ and $C_2 e^{3t} v_2$.

With Theorem 8.6, this computation can be carried out in general in three steps. One decomposes the initial value in terms of the eigenvectors, $u(0) = C_1 v_1 + \cdots + C_n v_n$, multiplies each C_i by $e^{\lambda_i t}$, and obtains

$$u(t) = C_1 e^{\lambda_1 t} v_1 + \cdots + C_n e^{\lambda_n t} v_n = V e^{\Lambda t} V^{-1} u(0),$$

where $e^{\Lambda t}$ is the diagonal matrix with the entries $e^{\lambda_i t}$ and the vector $V^{-1}u(0)$ contains the constants $C_1, \ldots, C_n$.

Example 8.3 (Initial Values and Eigenvectors of the Matrix K_2)**.** Suppose the initial values are $u_1(0) = 7$ and $u_2(0) = 3$. These determine the constants C_1 and C_2. At the starting time $t = 0$, both growth factors $e^{\lambda t}$ equal one:

$$u(0) = C_1 \begin{pmatrix} 1 \\ 1 \end{pmatrix} + C_2 \begin{pmatrix} 1 \\ -1 \end{pmatrix} \text{ is } \begin{pmatrix} 7 \\ 3 \end{pmatrix} = 5 \begin{pmatrix} 1 \\ 1 \end{pmatrix} + 2 \begin{pmatrix} 1 \\ -1 \end{pmatrix}.$$

We solve the two equations and obtain $C_1 = 5$ and $C_2 = 2$. The complete solution is therefore $u(t) = 5e^t v_1 + 2e^{3t} v_2$. It is a combination of a slower and a faster growth. For large t, the faster-growing e^{3t} dominates, so the solution points in the direction of v_2. □

In Section 8.7 we will discuss the important equation $M\ddot{u} + Ku = 0$ in more detail.

Newton's law involves acceleration (the second derivative of position with respect to time). We can imagine two masses connected by springs. They can oscillate in phase, which corresponds to the first eigenvector $(1, 1)$. But they can also oscillate completely in antiphase, moving in opposite directions, which corresponds to the second eigenvector $(1, -1)$. The eigenvectors provide the pure oscillations $e^{j\omega t} v$, which are called "normal modes". The initial conditions lead to a superposition.

Normal modes are the characteristic patterns of motion of a system capable of harmonic oscillation, in which all parts of the system oscillate at the same frequency. For a mass-spring chain or a polyatomic molecule, these are the specific ways in which the system can oscillate. Normal modes describe the dynamics of the system completely. Under ideal conditions they are decoupled, so each mode oscillates independently of the others.

Definition 8.5 (Normal Modes)**.** Let K be the stiffness matrix of a system (it describes the forces that pull the elements of the system back to the equilibrium position) and M its mass matrix (it describes the mass distribution). The motion of the system obeys the ODE

$$M\ddot{u} + Ku = 0,$$

where $\ddot{u}$ is the second derivative of the displacement $u(t)$ with respect to time. The *normal modes* are the solutions of the form $u(t) = e^{j\omega t} v$. Their vectors v and angular frequencies ω result from the associated eigenvalue problem $Kv = \omega^2 Mv$. □

In practice, this means that when a normal mode is excited, all masses of the system oscillate coherently with the same phase and frequency. The total motion of the system at any time can be expressed as a superposition of these normal modes.

8.5.3. Symmetric Matrices and Orthonormal Eigenvectors

Our special matrices K_n, T_n, B_n, and C_n are all symmetric. If A is a symmetric matrix, then its eigenvectors are orthogonal (and the eigenvalues λ are real):

Theorem 8.9 (Symmetric Matrices and Orthonormal Eigenvectors)**.** *Symmetric matrices have real eigenvalues and orthonormal eigenvectors.* □

The columns of V can then be chosen as orthonormal eigenvectors $v_1, \dots, v_n$ with $\|v_i\| = 1$. For the matrix with these columns we write Q instead of V. Orthonormal vectors are orthogonal unit vectors:

$$v_i^T v_j = \begin{cases} 0 & \text{for } i \neq j \text{ (orthogonal vectors),} \\ 1 & \text{for } i = j \text{ (unit vectors).} \end{cases}$$

The matrix Q is easy to work with because $Q^T Q = E$. The transpose is the inverse! This expresses once more, in matrix language, that the columns of Q are orthonormal. $Q^T Q = E$ contains all the inner products, which are either 0 or 1.

For two orthonormal columns in three-dimensional space, Q is a 3×2 matrix. In this orthogonal case, $Q^T Q = E$ still holds, but not $QQ^T = E$. For a complete basis of eigenvectors, Q is square and $Q^T = Q^{-1}$. When diagonalizing a real symmetric matrix, $V = Q$ and $V^{-1} = Q^T$:

Theorem 8.10 (Symmetric Diagonalization)**.** *A real symmetric matrix A can be diagonalized by an orthogonal matrix Q:*

$$A = V\Lambda V^{-1} = Q\Lambda Q^T \text{ with } Q^T = Q^{-1}.$$ □

Note that $Q\Lambda Q^T$ is automatically symmetric (like LDL^T). These factorizations perfectly reflect the symmetry of the matrix A. The eigenvalues $\lambda_1, \dots, \lambda_n$ form the "spectrum" of the matrix, and $A = Q\Lambda Q^T$ is a statement of the spectral theorem or the principal axis theorem.

8.5.4. Eigenvectors for Derivatives and Differences

A main theme in Strang (2010) is the analogy between discrete and continuous problems (matrix equations and ODEs). The special matrices introduced so far produce second differences. We therefore first turn to the ODE $-y'' = \lambda y$. The eigenfunctions $y(x)$ are sines and cosines.

Theorem 8.11 (Eigenfunctions of the ODE $-y'' = \lambda y$)**.**

$$-\frac{\mathrm{d}^2 y}{\mathrm{d}x^2} = \lambda y(x) \text{ is solved by } y = \cos \omega x, \quad y = \sin \omega x \text{ with } \lambda = \omega^2. \tag{8.2}$$ □

If we allow all frequencies ω, we obtain an abundance of eigenfunctions. The boundary conditions select particular frequencies ω and decide between cosine and sine.

8.5.4.1. The Fixed-Fixed Eigenfunctions

With the boundary conditions $y(0) = 0$ and $y(1) = 0$, the *fixed-fixed eigenfunctions* are $y(x) = \sin k\pi x$. The boundary condition $y(0) = 0$ reduces to $\sin 0 = 0$, the boundary condition $y(1) = 0$ to $\sin k\pi = 0$.

The zeros of the sine function lie at the integer multiples of π. Hence $k = 1, 2, 3, \dots$ (since $k = 0$ only gives $\sin 0x = 0$). We substitute $y(x) = \sin k\pi x$ into Equation 8.2 to determine the eigenvalues λ:

$$-\frac{\mathrm{d}^2}{\mathrm{d}x^2}(\sin k\pi x) = k^2\pi^2 \sin(k\pi x) \text{ so } \lambda = k^2\pi^2 = \left\{\pi^2, 4\pi^2, 9\pi^2, \dots\right\}.$$

Analogously, we will later offer a guess (discrete sine) for the discrete eigenvectors of the matrices K_n.

If we change the boundary conditions, we obtain different eigenfunctions and eigenvalues. The solutions of the equation $-y'' = \lambda y$ are still sines and cosines. Instead of the eigenfunctions $y = \sin(k\pi x)$, which vanish at the boundary points, we obtain for *free-free* (zero slope), *periodic*, and *free-fixed* boundary conditions the following eigenfunctions $y_k(x)$ with the associated eigenvalues λ_k:

- As in B_n: $y'(0) = 0$ and $y'(1) = 0$, $y(x) = \cos k\pi x$, $\lambda = k^2\pi^2$.
- As in C_n: $y(0) = y(1)$, $y'(0) = y'(1)$, $y(x) = \sin 2\pi kx, \cos 2\pi kx$, $\lambda = 4k^2\pi^2$.
- As in T_n: $y'(0) = 0$ and $y(1) = 0$, $y(x) = \cos\left(k + \frac{1}{2}\right)\pi x$, $\lambda = \left(k + \frac{1}{2}\right)^2 \pi^2$.

Recall that the matrices B_n and C_n are singular (one eigenvalue is $\lambda = 0$). Their continuous counterparts also have an eigenvalue $\lambda = 0$ with the eigenfunction $\cos 0x = 1$ (set $k = 0$). This constant eigenfunction $y(x) = 1$ is like the constant vector $(1, 1, \ldots, 1)$.

The free-fixed eigenfunctions $\cos\left(k + \frac{1}{2}\right)\pi x$ start with zero slope, since $\sin 0 = 0$. They end with zero height, because $\cos\left(k + \frac{1}{2}\right)\pi = 0$. Hence $y'(0) = 0$ and $y(1) = 0$. As we will see shortly, the eigenvectors of the matrices contain the same sines and cosines (but their eigenvalues λ are different).

8.5.5. Eigenvectors of the Matrices K_n: Discrete Sine Function

We now turn to the eigenvectors of the matrices with the entries $-1, 2, -1$. These are *discrete sines and cosines*: just try substituting them. In all interior rows, the sequences $y_\ell = \sin(\ell\theta)$ and $y_\ell = \cos(\ell\theta)$ satisfy the equation $-y_{\ell-1} + 2y_\ell - y_{\ell+1} = \lambda y_\ell$ with the eigenvalues $\lambda = 2 - 2\cos\theta \geq 0$:

$$-1\begin{Bmatrix} \sin(\ell-1)\theta \\ \cos(\ell-1)\theta \end{Bmatrix} + 2\begin{Bmatrix} \sin(\ell\theta) \\ \cos(\ell\theta) \end{Bmatrix} - 1\begin{Bmatrix} \sin(\ell+1)\theta \\ \cos(\ell+1)\theta \end{Bmatrix} = (2 - 2\cos\theta)\begin{Bmatrix} \sin(\ell\theta) \\ \cos(\ell\theta) \end{Bmatrix}.$$

These are the imaginary and real parts of the simpler identity

$$-e^{j(\ell-1)\theta} + 2e^{j\ell\theta} - e^{j(\ell+1)\theta} = (2 - e^{-j\theta} - e^{j\theta})e^{j\ell\theta}.$$

The boundary rows decide on θ and everything else! For the matrices K_n, the angles are $\theta = \frac{k\pi}{n+1}$. The first eigenvector v_1 samples the first eigenfunction $y(x) = \sin(\pi x)$ at the n grid points with $h = \frac{1}{n+1}$:

Theorem 8.12 (The First Eigenvector Is a Discrete Sine)**.**

$$v_1 = (\sin(\pi h), \sin(2\pi h), \ldots, \sin(n\pi h)). \qquad \square$$

The ℓth component is $\sin\left(\frac{\ell\pi}{n+1}\right)$. As required by the boundary conditions, it is zero for $\ell = 0$ and $\ell = n + 1$. The angle is $\theta = \frac{\pi}{n+1}$. The smallest eigenvalue is $\lambda = 2 - 2\cos(\theta) \approx \theta^2$.

The first eigenvalue of the matrix K_n is

$$\lambda_1 = 2 - 2\cos(\pi h) = 2 - 2\left(1 - \frac{1}{2}\pi^2 h^2 + \cdots\right) \approx \pi^2 h^2.$$

Compare this value with the first eigenvalue $\lambda = \pi^2$ of the ODE (when $y(x) = \sin(\pi x)$ and $-y'' = \pi^2 y$). Divide the matrix K by $h^2 = (\Delta x)^2$ to be able to compare differences with derivatives. The eigenvalues of the matrix K must also be divided by h^2:

$$\frac{\lambda_1(K)}{h^2} \approx \frac{\pi^2 h^2}{h^2} = \pi^2$$

is close to the first eigenvalue $\lambda_1 = \pi^2$.

The remaining continuous eigenfunctions are $\sin(2\pi x)$, $\sin(3\pi x)$, and in general $\sin(k\pi x)$. Clearly, the kth discrete eigenvector again samples $\sin(k\pi x)$ at the grid points $x = h, \ldots, nh$:

Theorem 8.13 (Discrete Sine Eigenvectors and Eigenvalues of the Matrix K_n)**.** *The eigenvectors (discrete sines) are*

$$v_k = (\sin(k\pi h), \ldots, \sin(nk\pi h)).$$

All eigenvalues of the matrix K_n are

$$\lambda_k = 2 - 2\cos(k\pi h), \quad k = 1, \ldots, n. \qquad \square$$

The sum $\lambda_1 + \cdots + \lambda_n$ must be $2n$, because that is the sum of all main diagonal entries 2 (the trace). The product of the λ must be $n + 1$.

Theorem 8.14 (Orthogonality of the Eigenvectors of Symmetric Matrices)**.** *The eigenvectors of a symmetric matrix are orthogonal.* □

The eigenvectors $(1, 1)$ and $(1, -1)$ in the two-dimensional case confirm Theorem 8.14 by example. The three eigenvectors (when $n = 3$ and $n + 1 = 4$) are the columns of the *sine matrix* (see Figure 8.1).

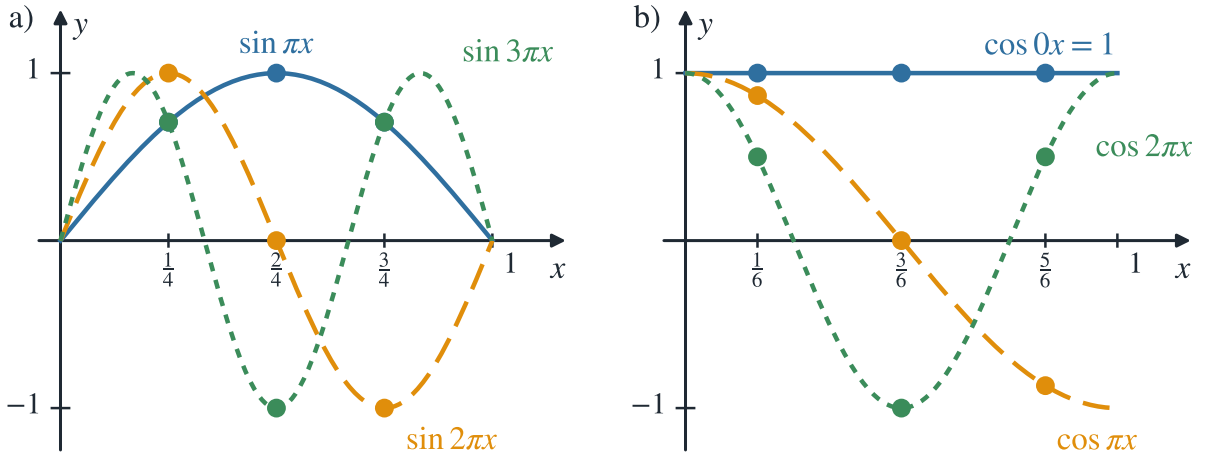


Figure 8.1.: The three discrete eigenvectors fall on three continuous eigenfunctions. a) The eigenvectors v_k of the matrix K_3 sample $\sin(k\pi x)$, $k = 1, 2, 3$, at the grid points $x = \frac{1}{4}, \frac{2}{4}, \frac{3}{4}$ ($h = \frac{1}{n+1} = \frac{1}{4}$). b) The eigenvectors of the matrix B_3 lie on the cosine curves $\cos(k\pi x)$, $k = 0, 1, 2$, sampled at $x = \frac{1}{6}, \frac{3}{6}, \frac{5}{6}$. Adapted from Strang (2010).

Definition 8.6 (Discrete Sine Transform)**.** The *discrete sine transform* (DST) is the multiplication by the $n \times n$ matrix DST whose columns are the discrete sine eigenvectors $v_1, \dots, v_n$ of the matrix K_n. Its entry in row ℓ and column k is $\sin\left(\frac{k\ell\pi}{n+1}\right)$. □

For $n = 3$ the matrix reads

$$\text{DST} = \begin{pmatrix} \sin\left(\frac{\pi}{4}\right) & \sin\left(\frac{2\pi}{4}\right) & \sin\left(\frac{3\pi}{4}\right) \\ \sin\left(\frac{2\pi}{4}\right) & \sin\left(\frac{4\pi}{4}\right) & \sin\left(\frac{6\pi}{4}\right) \\ \sin\left(\frac{3\pi}{4}\right) & \sin\left(\frac{6\pi}{4}\right) & \sin\left(\frac{9\pi}{4}\right) \end{pmatrix} = \begin{pmatrix} \frac{\sqrt{2}}{2} & 1 & \frac{\sqrt{2}}{2} \\ 1 & 0 & -1 \\ \frac{\sqrt{2}}{2} & -1 & \frac{\sqrt{2}}{2} \end{pmatrix}.$$

The columns of the DST matrix are orthogonal vectors of length $\sqrt{2}$. If we divide all components by $\sqrt{2}$, the three eigenvectors become orthonormal. Their components lie on sine curves. The DST matrix becomes an orthogonal matrix $Q = \text{DST}/\sqrt{2}$ with $Q^{-1} = Q^T$.

8.5.6. Eigenvectors of the Matrices C_n: Powers of $\zeta = \mathrm{e}^{2\pi\mathrm{j}/n}$

We consider the eigenvectors of the matrices C_n. These are sines and cosines, in other words: *complex exponentials*. The matrices C_n are even more important than the matrices of the sine and cosine transforms, because the eigenvectors now yield the discrete Fourier transform.

There are no better eigenvectors than these. Every circulant matrix (Definition 8.2) has such eigenvectors (Strang 2010, Ch. 4). Its constant diagonals continue with wrap-around into the next row (the -1 below the main diagonal of C_n is carried over into the top right corner). Our goal is to determine the eigenvalues and eigenvectors of the matrices C_n. The circulant matrix C_4 has the form

$$C_4 = \begin{pmatrix} 2 & -1 & 0 & -1 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ -1 & 0 & -1 & 2 \end{pmatrix}.$$

This real symmetric matrix has real, orthogonal eigenvectors (discrete sines and cosines). They have full periods like $\sin(2k\pi x)$ instead of half periods like $\sin(k\pi x)$.

Counting, however, becomes more difficult, because the cosines start with $k = 0$ and the sines with $k = 1$. It is better to work with complex exponentials $e^{j\theta}$. The kth eigenvector of the matrix C_n arises by sampling the function $y_k(x) = e^{j2\pi kx}$ at the n grid points $x = \ell/n$. This brings in the nth *root of unity* $\zeta = e^{2\pi j/n}$, a solution of $z^n = 1$ (Figure 8.2).

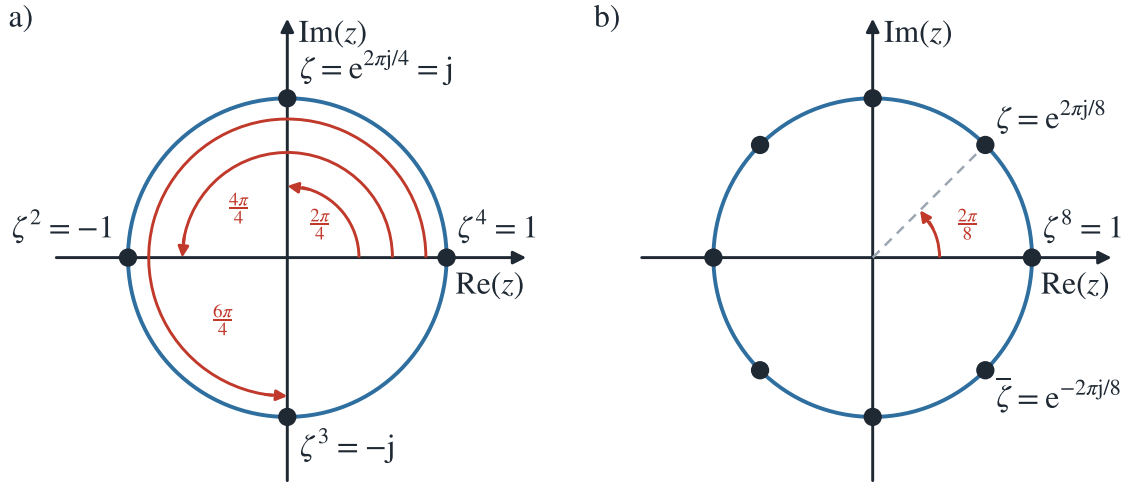


Figure 8.2.: Roots of unity on the unit circle $|z| = 1$ of the complex plane. a) The solutions to $z^4 = 1$ are the four roots $z = 1, -1, j, -j$. The powers of $\zeta = e^{2\pi j/4}$ are $1, j, -1, -j$. The arcs mark the angles $\frac{2\pi}{4}$, $\frac{4\pi}{4}$, and $\frac{6\pi}{4}$. b) For $n = 8$, $\zeta = e^{2\pi j/8}$ with the angle $\frac{2\pi}{8}$. The complex conjugate is $\overline{\zeta} = e^{-2\pi j/8} = \zeta^7$. Adapted from Strang (2010).

Theorem 8.15 (Eigenvectors of the Matrices C_n)**.** *The ℓth component of the vector v_k is*

$$e^{j2\pi k(\ell/n)} = \zeta^{\ell k}, \quad \zeta = e^{2\pi j/n} = \textit{nth root of unity}. \qquad \square$$

This special number ζ is the key to the discrete Fourier transform. Its angle is $2\pi/n$, which corresponds to the nth part of the way around the unit circle. The powers of ζ move around the unit circle and return to $\zeta^n = 1$. The eigenvectors and eigenvalues of C_n are

$$v_k = (1, \zeta^k, \zeta^{2k}, \dots, \zeta^{(n-1)k}), \qquad \lambda_k = 2 - \zeta^k - \zeta^{-k} = 2 - 2\cos\left(\frac{2\pi k}{n}\right).$$

Theorem 8.16 (Eigenvalues of the Matrix C_n)**.** *Let $\zeta = e^{2\pi j/n}$. For $k = 0, 1, \dots, n-1$, the vector v_k with the components $\zeta^{\ell k}$, $\ell = 0, 1, \dots, n-1$, is an eigenvector of C_n for the eigenvalue*

$$\lambda_k = 2 - \zeta^k - \zeta^{-k} = 2 - 2\cos\left(\frac{2\pi k}{n}\right) = 4\sin^2\left(\frac{\pi k}{n}\right). \qquad \square$$

Proof. Row ℓ of $C_n v_k$ reads $-\zeta^{(\ell-1)k} + 2\zeta^{\ell k} - \zeta^{(\ell+1)k} = (2 - \zeta^{-k} - \zeta^k)\zeta^{\ell k}$. Because $\zeta^n = 1$, this also holds in the first and the last row, where the indices $\ell - 1 = -1$ and $\ell + 1 = n$ are read cyclically as $n - 1$ and 0. With $\zeta^{-k} = \overline{\zeta^k}$ we have $\zeta^k + \zeta^{-k} = 2\cos(2\pi k/n)$, and $2 - 2\cos\varphi = 4\sin^2(\varphi/2)$ yields the last form. $\square$

The ordering is $k = 0, 1, \dots, n-1$. The eigenvector with $k = 0$ is the constant vector $v_0 = (1, 1, \dots, 1)$. The choice $k = n$ would give the same vector $(1, 1, \dots, 1)$, so nothing new. The smallest eigenvalue $2 - 2\cos 0$ is $\lambda_0 = 0$. The matrices C_n are singular.

The eigenvector with $k = 1$ is $v_1 = (1, \zeta, \dots, \zeta^{n-1})$. Its components are the n roots of unity, that is, the n solutions of $z^n = 1$.

Example 8.4 (Eigenvector v_1 of the Matrix C_4)**.** Figure 8.2 shows the unit circle $|z| = 1$ in the complex plane, where in part a) of the figure the $n = 4$ values are equally spaced around the circle. These numbers are e^0, $e^{2\pi j/4}$, $e^{4\pi j/4}$, $e^{6\pi j/4}$, and their fourth powers are 1. For $n = 4$, $\zeta = j$, and the following holds:

$$C_4 v_1 = \begin{pmatrix} 2 & -1 & 0 & -1 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ -1 & 0 & -1 & 2 \end{pmatrix} \begin{pmatrix} 1 \\ j \\ j^2 \\ j^3 \end{pmatrix} = (2 - j - j^3) \begin{pmatrix} 1 \\ j \\ j^2 \\ j^3 \end{pmatrix} = \lambda_1 v_1.$$

For arbitrary n, the first row gives $2-\zeta-\zeta^{n-1} = 2-\zeta-\overline{\zeta}$. Note that ζ^{n-1} is also the complex conjugate $\overline{\zeta} = \mathrm{e}^{-2\pi \mathrm{j}/n} = 1/\zeta$: multiplying it by ζ gives 1. Hence

$$\lambda_1 = 2 - \zeta - \overline{\zeta} = 2 - \mathrm{e}^{2\pi \mathrm{j}/n} - \mathrm{e}^{-2\pi \mathrm{j}/n} = 2 - 2\cos\left(\frac{2\pi}{n}\right).$$

Now we know the first two eigenvectors $v_0 = (1,1,1,1)$ and $v_1 = (1, \mathrm{j}, \mathrm{j}^2, \mathrm{j}^3)$ of C_4. The eigenvalues are 0 and 2. To complete the task, we need the eigenvectors $v_2 = (1, \mathrm{j}^2, \mathrm{j}^4, \mathrm{j}^6)$ and $v_3 = (1, \mathrm{j}^3, \mathrm{j}^6, \mathrm{j}^9)$. Their eigenvalues are 4 and 2, which come from $2 - 2\cos(\pi)$ and $2 - 2\cos(3\pi/2)$. Then the sum of the eigenvalues is $0 + 2 + 4 + 2 = 8$, which agrees with the sum of the diagonal entries (the trace $2 + 2 + 2 + 2$) of this matrix C_4. □

8.5.6.1. Positive Definite Matrices

Positive definite matrices (Definition 8.4) have many equivalent characterizations. In my view, we need three basic properties to make progress:

1. Every matrix $K = A^T A$ is symmetric and positive definite (or at least positive semidefinite).
2. If the matrices A and B are positive definite, then the matrix $A + B$ is also positive definite.
3. All pivots and eigenvalues of a positive definite matrix are positive.

8.5.6.2. Properties of Eigenvalues and Eigenvectors

Table 8.2 summarizes the properties of matrices and their eigenvectors. The table contains the most important properties discussed in this section. The eigenvectors of the matrices K_n, T_n, and B_n are sines and cosines, the eigenvectors of the matrices C_n are complex exponentials. The last row concerns the singular value decomposition (SVD) $A = U\Sigma V^T$ of an arbitrary matrix A. Its factors U and V are orthogonal, and Σ is diagonal and contains the singular values.

Table 8.2.: Properties of matrices and their eigenvectors

Matrix	Eigenvalues	Eigenvectors
symmetric	all λ are real	orthogonal $v_i^T v_j = 0$
orthogonal	all $\lvert\lambda\rvert = 1$	orthogonal $v_i^T v_j = 0$
skew-symmetric	all λ are imaginary	orthogonal $v_i^T v_j = 0$
complex Hermitian	all λ are real	orthogonal $v_i^T v_j = 0$
positive definite	all $\lambda > 0$	orthogonal
Markov	$\lambda_{\text{max}} = 1$	steady state $v > 0$
similar (Definition 5.2)	$\lambda(B) = \lambda(A)$	$v(B) = Pv(A)$
projection	$\lambda = 1; 0$	column space; nullspace
reflection $E - 2uu^T$, $\lVert u\rVert = 1$	$\lambda = -1; 1, \dots, 1$	u; orthogonal to u
rank 1, uv^T	$\lambda = v^T u; 0, \dots, 0$	u; orthogonal to v
inverse	$1/\lambda(A)$	eigenvectors of A
shift by a multiple of E	$\lambda(A)$ plus the shift	eigenvectors of A
stable powers	all $\lvert\lambda\rvert < 1$	-
stable exponential	all Re $\lambda < 0$	-
cyclic	$\lambda_k = \mathrm{e}^{2\pi \mathrm{j} k/n}$	$v_k = (1, \lambda_k, \dots, \lambda_k^{n-1})$
Toeplitz (K_n)	$\lambda_k = 2 - 2\cos\frac{k\pi}{n+1}$	$v_k = (\sin\frac{k\pi}{n+1}, \sin\frac{2k\pi}{n+1}, \dots)$
diagonalizable	diagonal of Λ	columns of V are independent
symmetric	diagonal of Λ (real)	columns of Q are orthonormal
Jordan	diagonal of J	each block gives $v = (0, \dots, 1, \dots, 0)$
SVD	singular values in Σ	eigenvectors of $A^T A$, AA^T in V, U

8.6. Equilibrium and the Stiffness Matrix

8.6.1. A Spring Chain

> **Literature Note**
>
> This section refers to Chapter 2.1 in Strang (2010).

Figure 8.3 shows three masses m_1, m_2, m_3 connected by springs. In one example it is a chain of four springs, fixed at the top and at the bottom. In the second example there are only three springs: the first spring is fixed at the top, and the lowest mass hangs free. The boundary conditions in the two examples are *fixed-fixed* and *fixed-free*, respectively. We seek equations for the essential variables, the displacements of the masses and the tensions in the springs:

$$u = (u_1, u_2, u_3) = \text{displacements of the masses,}$$
$$w = (w_1, w_2, w_3, w_4) \text{ or } (w_1, w_2, w_3) = \text{tension in the springs.}$$

The displacement of a mass has a positive sign when the mass is pulled downward. Under tension the spring force w_i has a positive sign, under compression a negative one ($w_i < 0$). Under tension the spring is stretched and then pulls the masses together. Under compression the masses are pushed apart by the spring.

We need the elastic properties (the spring constants c_i) of each spring. This is an essential change compared with Section 8.1! The matrices K_n and T_n treated there had "spring constants = 1". Here c_i relates the elongation e_i to the force w_i. Hooke's law reads $w_i = c_i e_i$, that is, force equals spring constant times elongation. With four springs there are four constants c_1, c_2, c_3, c_4.

Our task is to link these equations for the individual springs into a vector equation $f = Ku$ for the whole system. The vector f is the external force. Gravity pulls on each mass and gives $f = (m_1 g, m_2 g, m_3 g)$. At the surface of the earth, g is about $9.8\,\text{m/s}^2$. Multiplied by the mass, this gives a force.

Our real problem is the stiffness matrix K that links u to f. Two different structures, namely the springs with fixed-fixed and with fixed-free boundary conditions, also produce two different K.

It is best to construct K in three steps rather than in one. Now comes the dominant basic pattern of this chapter, which we first look at for a spring chain. Instead of linking the displacements u directly to the forces f, for n masses and r springs it is better to link each vector to its successor in the following list:

- $u = (u_1, u_2, \dots, u_n)$: displacements of the n masses,
- $e = (e_1, e_2, \dots, e_r)$: elongations of the r springs,
- $w = (w_1, w_2, \dots, w_r)$: internal forces in the r springs,
- $f = (f_1, f_2, \dots, f_n)$: external forces on the n masses.

The basic pattern that links u, e, w, and f is shown in Figure 8.4.

For both examples we will write down the matrices A, C, and A^T, first for the case with fixed ends and then for the case with a freely hanging lower end. Forgive the simplicity of the matrices; their significance lies in their form.

In the first step we determine the elongation e. It is a measure of the strain, that is, of how strongly the springs are stretched. Initially there is no displacement and no stretching (the system lies on a table). When it is set upright, gravity acts. The masses fall by u_1, u_2, u_3. If mass 2 falls by u_2 and mass 1 falls by u_1, then the spring stretches by $u_2 - u_1$ (Figure 8.5).

The lower end of spring i moves down by u_i. Its upper end moves down by u_{i-1}. If both ends move by the same amount, the spring is not stretched: $u_i = u_{i-1}$ and $e_i = 0$. Note that the equation $e_i = u_i - u_{i-1}$ (the difference of displacements) also holds at the ends if we fix them at the top and bottom by $u_0 = 0$ and $u_4 = 0$.

In the four equations, the matrix A is a 4×3 difference matrix. We write these four differences $u_i - u_{i-1}$ as a matrix A multiplied by a vector u. Our equation for the strain is $e = Au$:

$$\begin{pmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} u_1 \\ u_2 - u_1 \\ u_3 - u_2 \\ -u_3 \end{pmatrix}.$$

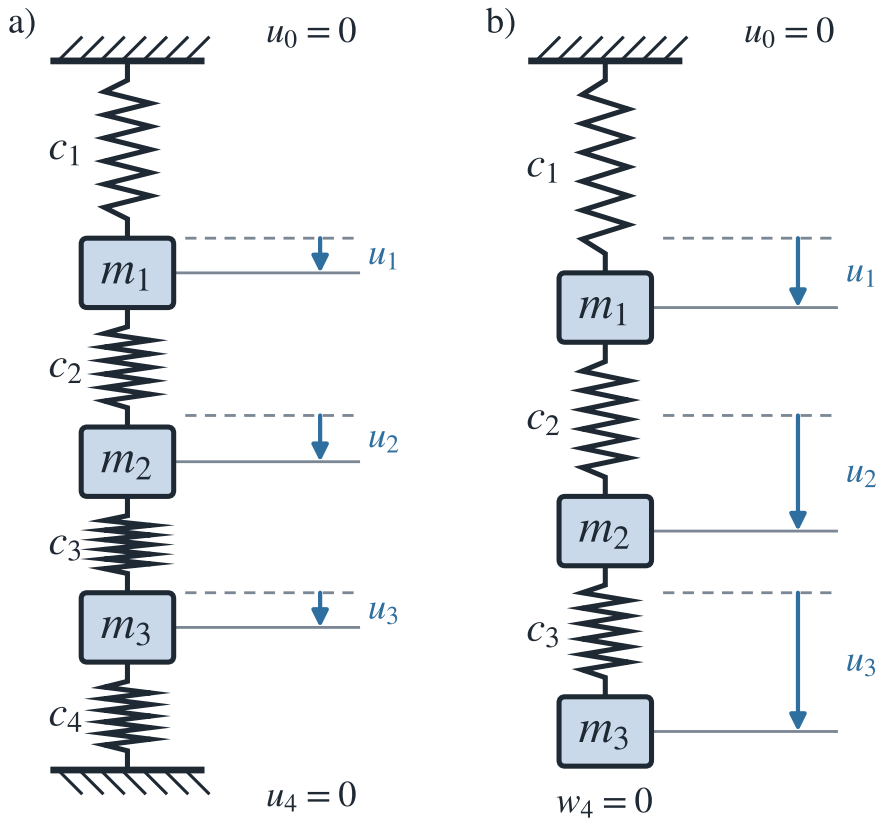


Figure 8.3.: Spring chain with three masses m_1, m_2, m_3 and spring constants c_i: a) fixed-fixed boundary conditions with four springs ($u_0 = u_4 = 0$), b) fixed-free boundary conditions with three springs ($u_0 = 0, w_4 = 0$). Dashed lines show the rest positions, solid lines the equilibrium positions under gravity for $c_i = c$ and $m_j = m$, that is, $u = \frac{mg}{c}(1.5,\ 2,\ 1.5)$ and $u = \frac{mg}{c}(3,\ 5,\ 6)$, respectively. The arrows show the displacements u_j (positive downward). Adapted from Strang (2010).

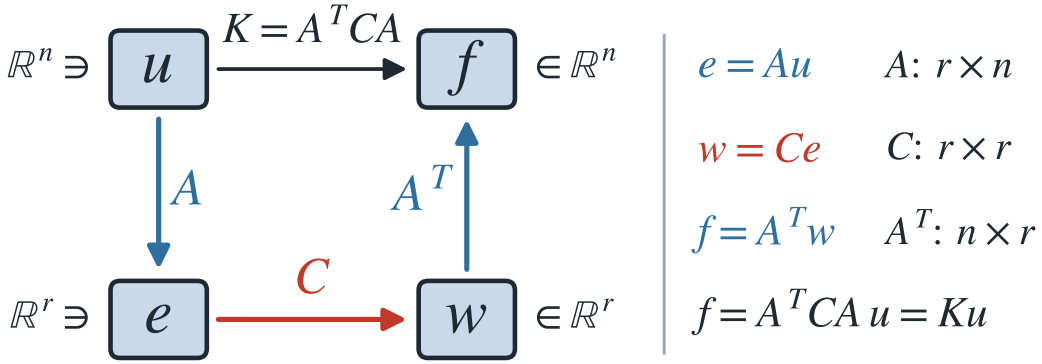


Figure 8.4.: Basic pattern $K = A^T CA$: the displacements $u \in \mathbb{R}^n$ of the masses yield the elongations $e = Au$ of the r springs (A is an $r \times n$ matrix), Hooke's law yields the internal forces $w = Ce$ (C is $r \times r$), and the force balance yields the external forces $f = A^T w$ (A^T is $n \times r$). Together, $f = A^T CAu = Ku$. Adapted from Strang (2010).

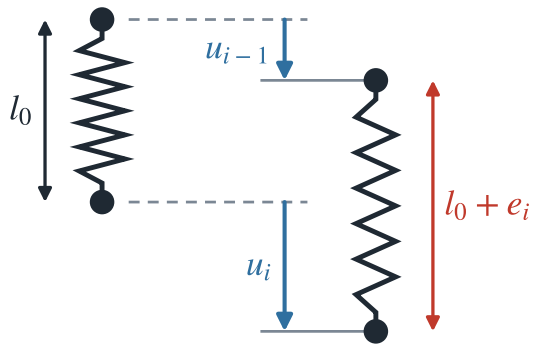


Figure 8.5.: Elongation of spring i: at rest it has the length l_0. If its upper end moves down by u_{i-1} and its lower end by u_i, it has the new length $l_0 + e_i$ with the strain $e_i = u_i - u_{i-1}$. Adapted from Strang (2010).

For a continuous problem, the differences $e = Au$ become $e = \frac{\mathrm{d}u}{\mathrm{d}x}$. The matrix becomes a derivative! Instead of a matrix equation we now obtain an ODE.

In the second step, the physical equation $w = Ce$ links the elongations e of the springs to the spring tensions w. This is Hooke's law $w_i = c_i e_i$ for each individual spring. It is the "constitutive law", which depends on the material of the spring and on its shape. A soft thin spring has a small c_i, so that a moderate force w_i can cause a large elongation e_i. Note again that Hooke's law is linear. This description is nearly exact over a wide range, until the spring is overstretched and the material deforms.

Each spring has its own law. Therefore, the matrix C in $w = Ce$ is a diagonal matrix. For the four springs, Hooke's law in matrix form reads

$$\begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix} = \begin{pmatrix} c_1 & 0 & 0 & 0 \\ 0 & c_2 & 0 & 0 \\ 0 & 0 & c_3 & 0 \\ 0 & 0 & 0 & c_4 \end{pmatrix} \begin{pmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{pmatrix} = Ce.$$

If we substitute $e = Au$ into $w = Ce$, we know the relation $w = CAu$ between the internal spring forces w and the mass displacements u.

In the third step we come to the *balance equation*. The internal forces w of the springs must balance the external forces f on the masses. Each mass is in equilibrium: it is pulled upward by a spring force w_j and at the same time pulled downward by a spring force w_{j+1} and the gravitational force f_j. Hence $w_j = w_{j+1} + f_j$, or $f_j = w_j - w_{j+1}$.

The force balance $A^T w = f$ reads

$$\begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix} = \begin{pmatrix} w_1 - w_2 \\ w_2 - w_3 \\ w_3 - w_4 \end{pmatrix} = \begin{pmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix}.$$

This is the matrix A^T! The balance equation of the forces is $f = A^T w$. In a natural way, the rows and columns of the e-u matrix are transposed so that the f-w matrix comes out. Herein lies the beauty of this basic pattern: A^T appears together with A.

8.6.2. Stiffness Matrix and Solution

All large finite element programs devote considerable effort to assembling the stiffness matrix $K = A^T CA$ from smaller pieces. For four springs we do this by matrix multiplication of A^T with CA. Here is the stiffness matrix K for fixed-fixed boundary conditions:

$$K = A^T CA = \begin{pmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \end{pmatrix} \begin{pmatrix} c_1 & 0 & 0 \\ -c_2 & c_2 & 0 \\ 0 & -c_3 & c_3 \\ 0 & 0 & -c_4 \end{pmatrix} = \begin{pmatrix} c_1 + c_2 & -c_2 & 0 \\ -c_2 & c_2 + c_3 & -c_3 \\ 0 & -c_3 & c_3 + c_4 \end{pmatrix}.$$

If all springs are identical with $c_1 = c_2 = c_3 = c_4 = 1$, then C is the identity matrix, $C = E$, and the stiffness matrix reduces to $K = A^T A$. With $c_i = 1$ we thus obtain the special matrix K_3:

Theorem 8.17 (Stiffness Matrix with $C = E$ (Fixed-Fixed))**.** *The stiffness matrix with $C = E$ is K_3:*

$$K_3 = A^T A = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix}. \qquad \square$$

A chain of four identical springs, fixed at both ends, has led us to our favorite matrix. A chain of $n + 1$ identical springs would give K_n. The matrix is tridiagonal because each spring is connected only to its two neighbors. It has constant diagonal entries because all springs are identical. The matrix K_n is symmetric positive definite because nature insists on this property (by producing A and A^T).

$K_3 = A^T A$ is something quite different from $K_3 = LL^T$. The matrix A from the problem with four springs is a 4×3 matrix. The matrix L from elimination (Cholesky factorization) is square. The matrix K is assembled from $A^T A$ and then factored into LL^T. One step belongs to applied mathematics, the other lies in the realm of scientific computing. Every matrix K_n is assembled from rectangular matrices and factored into square matrices.

Let me list a few properties of the matrix $K = A^T CA$. Almost all of them are familiar to you:

1. K is tridiagonal because mass 3 is not connected to mass 1.
2. K is symmetric because C is symmetric (and A^T appears together with A).
3. K is positive definite because A has independent columns.
4. K^{-1} is a full matrix with only positive entries.

This last property has an important consequence for $u = K^{-1} f$: if all forces act downward ($f_j > 0$), then the masses are displaced only downward ($u_j > 0$). Bear in mind that the "positivity" of a matrix is something different from its "positive definiteness". Here the matrix K is not positive (its entries include -1), while both matrices K and K^{-1} are positive definite.

Example 8.5 (Spring Chain with Fixed-Fixed Ends (Example 2.1 from Strang (2010)))**.** Suppose $c_i = c$ and $m_j = m$. Determine the displacements u and the forces w.

All springs and masses are identical. But the displacements, elongations, and spring forces will not all be identical. The inverse matrix K^{-1} contains the factor $\frac{1}{c}$, because the matrix $K = A^T CA$ contains the factor c. For the displacements,

$$u = K^{-1} f = \frac{1}{4c} \begin{pmatrix} 3 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 3 \end{pmatrix} \begin{pmatrix} mg \\ mg \\ mg \end{pmatrix} = \frac{mg}{c} \begin{pmatrix} 1.5 \\ 2.0 \\ 1.5 \end{pmatrix}.$$

The displacement u_2 of the middle mass is larger than u_1 and u_3. The units are consistent: the force mg divided by the force per unit length c gives a length u. The elongations of the springs are

$$e = Au = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \end{pmatrix} \frac{mg}{c} \begin{pmatrix} 1.5 \\ 2.0 \\ 1.5 \end{pmatrix} = \frac{mg}{c} \begin{pmatrix} 1.5 \\ 0.5 \\ -0.5 \\ -1.5 \end{pmatrix}.$$

The sum of these elongations should be zero because the ends of the chain are fixed. And indeed $u_1 + (u_2 - u_1) + (u_3 - u_2) + (-u_3) = 0$. To compute the spring force w, by Hooke's law only e needs to be multiplied by c. So w_1, w_2, w_3, w_4 equal $1.5\,mg$, $0.5\,mg$, $-0.5\,mg$, $-1.5\,mg$. The two upper springs are stretched, the two lower springs are compressed.

Note how u, e, w were computed in this order. We assembled $K = A^T CA$ from rectangular matrices. To determine $u = K^{-1} f$, we work with the whole matrix and not with its three parts. The rectangular matrices A and A^T have no (two-sided) inverses. In general, you therefore cannot write $K^{-1} = A^{-1} C^{-1} (A^T)^{-1}$.

The matrices are linked by the triple product $A^T CA$ and cannot simply be separated again. In general, $A^T w = f$ has many solutions. Four equations in three unknowns would usually have no solution. But $A^T CA$ delivers the correct solution to all three equations in the basic pattern. Only in the case $r = n$ with square matrices can we go from $w = (A^T)^{-1} f$ via $e = C^{-1} w$ to $u = A^{-1} e$. We will look at this now. $\square$

8.6.3. Fixed and Free End

Let us remove the fourth spring. All matrices become 3×3 matrices. The pattern does not change! The matrix A loses its fourth row (there is no e_4). Furthermore, the matrix A^T loses its fourth column (there is no w_4). The stiffness matrix becomes a product of square matrices:

$$A = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{pmatrix}, \quad C = \begin{pmatrix} c_1 & 0 & 0 \\ 0 & c_2 & 0 \\ 0 & 0 & c_3 \end{pmatrix}, \quad A^T = \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix}.$$

For fixed-free boundary conditions, this gives

$$K = A^T C A = \begin{pmatrix} c_1 + c_2 & -c_2 & 0 \\ -c_2 & c_2 + c_3 & -c_3 \\ 0 & -c_3 & c_3 \end{pmatrix}.$$

You recognize this matrix in the standard case $c_1 = c_2 = c_3 = 1$. It is the familiar tridiagonal matrix with the entries $-1, 2, -1$, but the last entry is not $c_3 + c_4 = 2$ but $c_3 = 1$. Compared with T_3 from Section 8.1, the ends are interchanged, so that the *hanging matrix* H_3 results. This corresponds to a second difference matrix with $u_0 = 0$ at one end and $u_4 = u_3$ at the other. The lowest mass hangs free.

Example 8.6 (Fixed-Free Spring Chain (Example 2.2 from Strang (2010)))**.** For the fixed-free hanging spring chain, $c_i = c$ and $m_j = m$ for all i and j. Then

$$A^T C A = c \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}, \quad (A^T C A)^{-1} = \frac{1}{c} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 2 & 3 \end{pmatrix}.$$

The displacements $u = K^{-1} f$ differ from those in the example with fixed-fixed boundary conditions because the matrix K has changed:

$$u = (A^T C A)^{-1} f = \frac{1}{c} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 2 & 3 \end{pmatrix} \begin{pmatrix} mg \\ mg \\ mg \end{pmatrix} = \frac{mg}{c} \begin{pmatrix} 3 \\ 5 \\ 6 \end{pmatrix}. \qquad \square$$

In this case with fixed-free boundary conditions, the displacements $(3,\ 5,\ 6)$ are larger than the displacements $(1.5,\ 2.0,\ 1.5)$ from before. The number 3 appears as the first displacement u_1 because all three masses pull the first spring downward. The next mass has an additional displacement $(3 + 2 = 5)$ due to the two masses below it. The third mass moves down even further, namely by $(3 + 2 + 1 = 6)$. The elongations $e = Au$ of the three springs show these three numbers $3, 2, 1$:

$$e = Au = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{pmatrix} \frac{mg}{c} \begin{pmatrix} 3 \\ 5 \\ 6 \end{pmatrix} = \frac{mg}{c} \begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix}.$$

After multiplication by c, the forces in the three springs are therefore $w_1 = 3mg$, $w_2 = 2mg$, and $w_3 = mg$. The first spring carries three masses, the second spring two, and the third spring one. All springs are now stretched.

The special feature of a square matrix A is that the internal forces w can be read off directly from the external forces f. The balance equation $A^T w = f$ determines w immediately and uniquely, because $r = n$ and A^T is square and invertible. For the spring forces,

$$w = (A^T)^{-1} f = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} mg \\ mg \\ mg \end{pmatrix} = \begin{pmatrix} 3mg \\ 2mg \\ mg \end{pmatrix}.$$

Then the elongations e follow from $C^{-1} w$ and the displacements u from $A^{-1} e$. In this "determinate" case $r = n$ we may write $(A^T C A)^{-1} = A^{-1} C^{-1} (A^T)^{-1}$.

If the displacement at the upper end is fixed by $u_0 = 0$, a force is needed to accomplish this. This is an external reaction force f_0 that supports the chain of springs. This reaction force is not given in advance. It is part of the result of the force balance at the upper end. The first spring pulls downward with an internal force $w_1 = 3mg$. The reaction force pulls upward with $f_0 = -3mg$ to balance this force. A structural engineer must know the reaction forces to make sure that the support holds and the building does not collapse.

Example 8.7 (Spring Chain with Free-Free Ends (Example 2.3 from Strang (2010)))**.** A spring chain with free-free boundary conditions has no support. This causes trouble with A and K (spring 1 has disappeared). The matrix A is a 2×3 matrix, short and wide. Here is $e = Au$:

$$e_2 = u_2 - u_1, \quad e_3 = u_3 - u_2,$$

$$\begin{pmatrix} e_2 \\ e_3 \end{pmatrix} = \begin{pmatrix} -1 & 1 & 0 \\ 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix}.$$

Now there is a nonzero solution to $Au = 0$. The masses can move without stretching the springs: the whole chain can move by $u = (1, 1, 1)$, and still $e = (0, 0)$. The columns of A are dependent, and the vector $(1, 1, 1)$ lies in the nullspace. For this rigid-body motion,

$$u = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \quad Au = \begin{pmatrix} 0 \\ 0 \end{pmatrix} = e.$$

In this case $A^T CA$ cannot be invertible. K must be singular, because $Au = 0$ obviously leads to $A^T CAu = 0$. The stiffness matrix is still square and symmetric, but it is only positive semidefinite (like B_n with two free ends):

$$A^T CA = \begin{pmatrix} c_2 & -c_2 & 0 \\ -c_2 & c_2 + c_3 & -c_3 \\ 0 & -c_3 & c_3 \end{pmatrix}.$$

The pivots are c_2 and c_3. There is no third pivot. Two eigenvalues are positive, but the vector $(1, 1, 1)$ is an eigenvector for $\lambda = 0$. The matrix is not invertible, and we can solve $A^T CAu = f$ only for special vectors f. The external forces must add up to zero: $f_1 + f_2 + f_3 = 0$. Otherwise, the whole spring chain (with two free ends) would take off like a rocket.

In Strang (2010, Problem 2.1.4, p. 126), the upper end is pulled upward and the lower end downward, with the forces canceling each other. We can solve $A^T CAu = (-1, 0, 1)$ even though the system is singular. A circle of springs (K is then the circulant matrix C_n from Section 8.1) is also singular. □

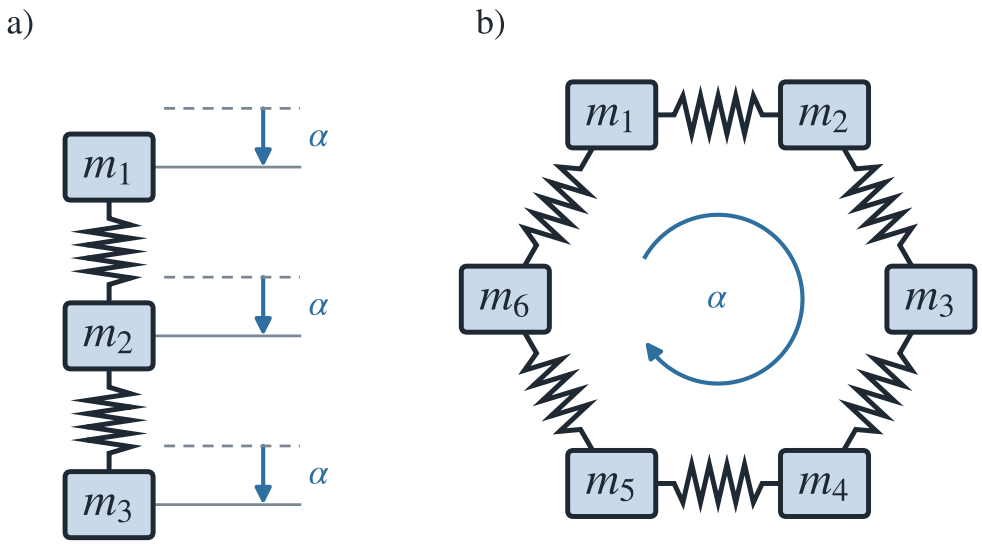


Figure 8.6.: Rigid-body motions without support. a) Free-free spring chain: all masses shift by the same amount α (translation), no spring is stretched, so $Au = 0$ has the nonzero solution $u = (\alpha, \alpha, \alpha)$ and $A^T CA$ is singular. b) Circle of six springs: an equal shift α of all masses along the circle (rotation) likewise stretches no spring, and this (circulant) stiffness matrix is singular as well. Adapted from Strang (2010).

8.7. Oscillations by Newton's Law

> **i** Literature Note
>
> This section refers to Chapter 2.2 in Strang (2010).

This section deals with the most important law of mechanics, Newton's law $F = m\ddot{u}$: force equals mass times acceleration. Let n masses $m_1, \dots, m_n$ be given that obey this law. Their displacements $u_1(t), \dots, u_n(t)$ change with time, but by no means at nearly the speed of light (we hope). The acceleration of each mass is the second derivative $\ddot{u} = \frac{\mathrm{d}^2 u}{\mathrm{d}t^2}$ (also written as u_{tt} or u''). We need the forces F.

Compare the statement of $F = m\ddot{u}$ with the statement about equilibrium in the previous section. There the masses did not move ($\ddot{u} = 0$). Each mass was in equilibrium between external forces f and spring forces Ku. Now the masses are out of this equilibrium: $F = f - Ku \neq 0$, the masses are in motion. In the case $f = 0$ there are only spring forces, and

$$M\ddot{u} = -Ku.$$

Friction and damping terms would involve the velocities $\dot{u} = \frac{\mathrm{d}u}{\mathrm{d}t}$. This is the case for flow problems (and for many applications in which flows play a role). In this section, only $\ddot{u}$ occurs and no $\dot{u}$. Without damping or external forces, the springs oscillate forever. The sum of the kinetic energies $\frac{1}{2}m\dot{u}^2$ of the masses and the stored energies $\frac{1}{2}ce^2$ of the springs must be constant.

When we treat coupled ODEs, we naturally use matrices. The vector of n displacements is $u(t)$, and we write the masses into a diagonal mass matrix $M = \mathrm{diag}(m_1, \dots, m_n)$:

$$u(t) = \begin{pmatrix} u_1(t) \\ \vdots \\ u_n(t) \end{pmatrix}, \qquad M = \begin{pmatrix} m_1 & & 0 \\ & \ddots & \\ 0 & & m_n \end{pmatrix}.$$

The vector of the n external forces on the masses is $f(t)$. If $Ku = f$ holds, the masses are in equilibrium: there are no oscillations. The force in Newton's law $F = M\ddot{u}$ is the difference $F = f - Ku$.

Theorem 8.18 (Newton's Law)**.**

$$F = M\ddot{u}, \quad \textit{so} \quad f - Ku = M\ddot{u} \quad \textit{or} \quad M\ddot{u} + Ku = f. \qquad \square$$

We do not have to repeat the same steps for every new application. That is the advantage of the basic pattern! Once the matrices A, C, $K = A^T C A$, and M are known, the system works by itself. The basic equation $M\ddot{u} + Ku = 0$ with $f = 0$ is conservative. We will solve it exactly using the eigenvalues and eigenvectors of $M^{-1}K$.

In real applications, large problems are solved using finite differences with time steps Δt. A key point in scientific computing is the choice between "explicit" and "implicit". With explicit methods we obtain the new $Mu(t + \Delta t)$ directly.

8.7.1. One Mass and One Spring

Let us begin with a mass m hanging from a spring (with spring constant c). The upper end is fixed. When the mass moves down by a displacement $u(t)$, the spring pulls it back up with the force $-cu(t)$. The spring force has a negative sign because it opposes the displacement. Newton's law $m\ddot{u} = -cu$ gives the equation for the one unknown $u(t)$,

$$m\frac{\mathrm{d}^2 u}{\mathrm{d}t^2} + cu = 0$$

with given $u(0)$ and $\dot{u}(0)$.

Keep in mind that we are talking here about motion about the equilibrium position. Gravity has already been accounted for. If you wanted to compute the total displacement from the rest position (without any stretching), you would have to add mg on the right-hand side and the value $\frac{mg}{c}$ to the solution, the constant value of u at equilibrium. We measure from the equilibrium point rather than from zero.

The solution $u(t)$ contains cosines and sines (a second-order equation has two solutions). For $m = 1$ and $c = 1$, $\cos(t)$ and $\sin(t)$ already satisfy the equation $\ddot{u} + u = 0$. Otherwise, we need a factor $\frac{c}{m}$ from the second derivative in the equation, so we multiply t by the natural angular frequency $\omega_0 = \sqrt{c/m}$:

Theorem 8.19 (Oscillation Solution)**.** *With $\omega_0 = \sqrt{c/m}$, the solution of $m\ddot{u} + cu = 0$ reads*

$$u(t) = A\cos(\omega_0 t) + B\sin(\omega_0 t). \qquad \square$$

At $t = 0$ the displacement is $u(0) = A$. Taking the first derivative of $u(t)$ gives the velocity. At $t = 0$ the velocity is $\dot{u}(0) = B\omega_0$. The initial conditions therefore determine $A = u(0)$ and

$$B = \frac{\dot{u}(0)}{\omega_0} = \sqrt{\frac{m}{c}}\,\dot{u}(0).$$

The larger ω_0, the faster the mass oscillates. A small mass on a stiff spring oscillates quickly, like an electron. A large mass hanging from a soft spring oscillates slowly, like a heavy ball on a rubber band.

The potential energy in the one spring is $\frac{1}{2}cu^2$. The kinetic energy of the one mass is $\frac{1}{2}m(\dot{u})^2$. With more springs and masses, the potential energy becomes $\frac{1}{2}e^TCe$. The kinetic energy is then $\frac{1}{2}\dot{u}^TM\dot{u}$. The total energy (sum of kinetic and potential energy) is constant and equal to its value at time $t = 0$:

Theorem 8.20 (Energy Conservation)**.**

$$\frac{1}{2}m\dot{u}^2(t) + \frac{1}{2}cu^2(t) = \frac{1}{2}m\dot{u}^2(0) + \frac{1}{2}cu^2(0). \qquad \square$$

Proof. For one spring, the derivative of the total energy is $m\dot{u}\ddot{u} + cu\dot{u} = \dot{u}\,(m\ddot{u} + cu)$. Because $m\ddot{u} + cu = 0$, this derivative is zero, so the energy does not change. $\square$

Example 8.8 (Fixed-Fixed (Example 2.4 from Strang (2010)))**.** We consider two equal masses $m_1 = m_2$ and three identical springs: both ends are fixed. The 2×2 mass matrix is simply $M = mE$. We are also already familiar with the 2×2 stiffness matrix:

$$\text{Fixed-fixed} \quad K = c\begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}, \quad M^{-1}K = \frac{c}{m}\begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}.$$

The eigenvalues are $\lambda_1 = \frac{c}{m}$ and $\lambda_2 = \frac{3c}{m}$. The eigenvectors are $v_1 = (1, 1)$ and $v_2 = (1, -1)$. They are orthogonal with respect to the M inner product. Here, however, $M = mE$ and $M^{-1}K$ are symmetric, so they are orthogonal in the usual sense and $v_1^Tv_2 = 0$. Substitute λ_i and v_i into the general solution

$$u(t) = \sum_{i=1}^{n}\left(A_i\cos(\sqrt{\lambda_i}t) + B_i\sin(\sqrt{\lambda_i}t)\right)v_i \tag{8.3}$$

The solution is a combination of the normal modes v_1 and v_2:

$$u(t) = \left(A_1\cos\left(\sqrt{\frac{c}{m}}\,t\right) + B_1\sin\left(\sqrt{\frac{c}{m}}\,t\right)\right)\begin{pmatrix}1\\1\end{pmatrix} + \left(A_2\cos\left(\sqrt{\frac{3c}{m}}\,t\right) + B_2\sin\left(\sqrt{\frac{3c}{m}}\,t\right)\right)\begin{pmatrix}1\\-1\end{pmatrix}.$$

Part a) of Figure 8.7 shows these two pure oscillations. In $v_1 = (1, 1)$ the masses oscillate in phase. If they start with the initial conditions $u_1(0) = u_2(0)$ and $\dot{u}_1(0) = \dot{u}_2(0)$, they oscillate in phase forever. In $v_2 = (1, -1)$ the masses oscillate in antiphase (they move in opposite directions). The oscillation is faster because λ_2 has the factor 3. $\square$

Example 8.9 (Lower End Free (Example 2.5 from Strang (2010)))**.** Masses $m_1 = 9, m_2 = 1$ with $c_1 = 75, c_2 = 6$: lower end free.

Now $c_1 = 75$ and $c_2 = 6$ are substituted into $K = A^TCA$. The entries of the last row sum to zero (free end):

$$\text{Fixed-free} \quad M = \begin{pmatrix} 9 & 0 \\ 0 & 1 \end{pmatrix}, \quad K = \begin{pmatrix} 75+6 & -6 \\ -6 & 6 \end{pmatrix}.$$

The eigenvalues $\lambda_1 = 5$ and $\lambda_2 = 10$ come from $M^{-1}K$ (note the missing symmetry):

$$M^{-1}K = \begin{pmatrix} 81/9 & -6/9 \\ -6 & 6 \end{pmatrix} \quad \text{leads to} \quad \det\left(M^{-1}K - \lambda E\right) = \lambda^2 - 15\lambda + 50 = 0.$$

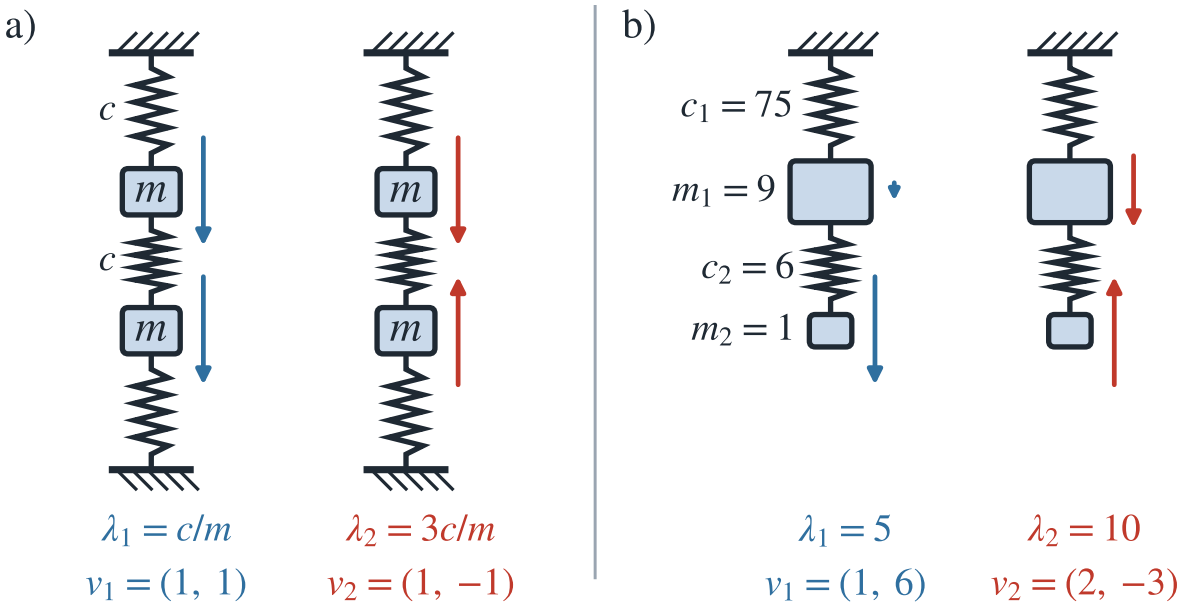


Figure 8.7.: Normal modes of two masses. The arrows are proportional to the components of the eigenvectors (positive downward). a) Fixed-fixed with equal masses m and springs c: in the slow mode $v_1 = (1, 1)$ with $\lambda_1 = c/m$ the masses oscillate in phase, in the fast mode $v_2 = (1, -1)$ with $\lambda_2 = 3c/m$ in opposite directions. b) Lower end free with $m_1 = 9$, $m_2 = 1$, $c_1 = 75$, $c_2 = 6$: eigenvectors $v_1 = (1, 6)$ for $\lambda_1 = 5$ and $v_2 = (2, -3)$ for $\lambda_2 = 10$. The smaller mass is displaced more strongly in both modes. Adapted from Strang (2010).

The eigenvectors $v_1 = (1, 6)$ and $v_2 = (2, -3)$ are not orthogonal because $M^{-1}K$ is not symmetric. We must have $v_1^T M v_2 = 0$. Here $(1)(9)(2) + (6)(1)(-3) = 0$. Now the smaller mass is displaced more strongly in both normal modes v_1 and v_2, as can be seen in part b) of Figure 8.7. Starting from rest at $u(0) = (1, 0)$ (there are no sine terms), the solution is again a combination of two normal modes:

$$u(t) = \left(\frac{1}{5}\cos(\sqrt{5}t) + 0\sin(\sqrt{5}t)\right)\begin{pmatrix}1\\6\end{pmatrix} + \left(\frac{2}{5}\cos(\sqrt{10}t) + 0\sin(\sqrt{10}t)\right)\begin{pmatrix}2\\-3\end{pmatrix}. \qquad \square$$

Part III.

Applications

9. Turbines and Turbochargers

9.1. Definitions

Definition 9.1 (Turbine)**.** A *turbine* is a rotating turbomachine that converts the drop in internal energy of a flowing fluid (liquid or gas) into mechanical rotational energy, which it delivers via its shaft. In an *axial turbine* the working fluid flows parallel to the shaft, in a *radial turbine* radially, i.e., perpendicular to the shaft. □

Axial turbines are used in large engines, radial turbines in small engines.

Definition 9.2 (Turbocharger)**.** A *turbocharger* is a component for compressing the air supplied to an internal combustion engine (engine supercharging). It uses part of the energy of the engine exhaust gas by means of a turbine to drive a compressor, usually a radial compressor. □

Figure 9.1 shows an exhaust gas turbocharger for a gasoline (Otto) engine.

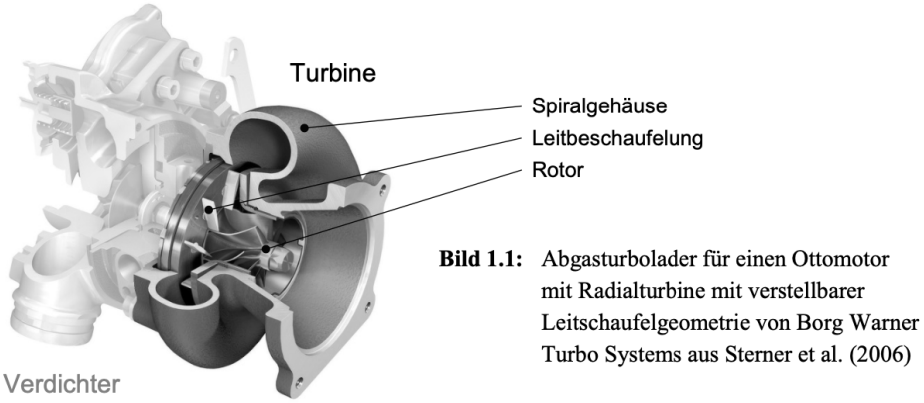


Figure 9.1.: Exhaust gas turbocharger. Taken from Schwitzke (2015)

Definition 9.3 (Impeller and Rotor)**.** An *impeller* consists of a ring of blades (vanes, rotor blades) that are held on the shaft by a suitable attachment. The totality of all impellers together with the connecting shaft is called the *rotor*. □

A distinction is made between main blades and splitter blades. A rotor, including the compressor wheel, is shown in Figure 9.2.

9.2. High-Cycle and Low-Cycle Fatigue

Definition 9.4 (Low-Cycle and High-Cycle Fatigue)**.** Material fatigue due to alternating loads is divided into *low-cycle fatigue* (LCF) and *high-cycle fatigue* (HCF). LCF describes the consequences of low-frequency, mostly thermal loads caused by changing operating states of the machine. HCF denotes damage caused by high-frequency dynamic loads resulting from vibrations. □

Failure due to rotor vibrations is prevented by balancing and by sufficient damping in the plain bearings. Damage caused by blade vibrations is critical and unpredictable.

The interaction of the rotor blades with components of the housing or of the guide vane assembly gives rise to pulsating aerodynamic forces that induce a wide variety of bending and torsional vibrations of the blades.

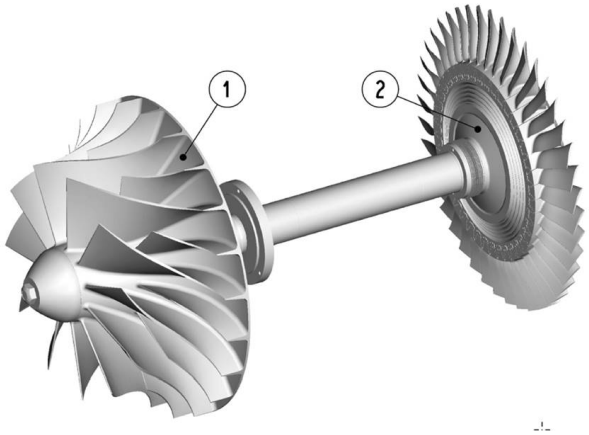


Figure 9.2.: Rotor of a turbocharger with turbine (left), compressor (right), steel shaft, and radial and axial bearings. Source: MAN Energy Solution SE (2023)

The fluid-mechanical interaction of the guide vanes with the blades of the rotor produces high force amplitudes, which favor vibration excitation. In addition, the high excitation frequencies, which result from the number of guide vanes as a multiple of the rotational frequency, prove to be problematic. They fall into the range of the high natural frequencies of the stiff rotor geometry with its blades, which are relatively short compared with those of axial rotors. In the relevant frequency range, a typical rotor geometry has a large number of natural frequencies. On the other hand, in conjunction with the wide speed range in which turbochargers are operated, a broad excitation spectrum results. The consequence is a large number of possible resonance states within the operating range. Resonances are therefore unavoidable. To ensure a fatigue-resistant design of the rotor structure, critical resonant vibrations must be identified and their amplitudes must be predictable.

9.3. Natural Vibration Behavior of Turbomachines

Vibration investigations of turbomachines begin with a modal analysis of the rotor.

Definition 9.5 (Modal Analysis)**.** *Modal analysis* determines the natural vibration behavior of a structure, i.e., its natural frequencies and the associated mode shapes, which are also referred to as *modes*. □

The number of existing modes depends on the number of degrees of freedom of the vibrating system. For a continuum, theoretically infinitely many modes therefore exist. In the relevant frequency range, which follows from the excitation spectrum, a rotor usually has a finite number of modes. Knowledge of the natural frequencies and the associated modal deformations is the basic prerequisite for all further HCF investigations.

9.3.1. Blade and Disk Mode Shapes

9.3.1.1. The Behavior of a Single Blade

As shown in Figure 9.3, a blisk (blade integrated disk) consists of a disk and several blades. Axial blades can approximately be regarded as bending plates clamped at one end.

Each mode shape has an associated natural frequency, of which the one for the first bending mode takes the smallest value. Typically, the first torsion and the second bending mode form the next higher natural frequencies. With increasing frequency, a vibration referred to here as stiff bending occurs in the direction with the higher second moment of area. A longitudinal stretching of the blade, which is not shown, plays a subordinate role because it is not excited by aerodynamic excitation forces during operation.

Higher modes are formed by the superposition of the shapes named first. The sequence of the natural frequencies depends on the dimensions. For the flat plate and also for a twisted plate, the natural frequencies and the modal shapes can be derived from an analytical solution of the vibration equation.

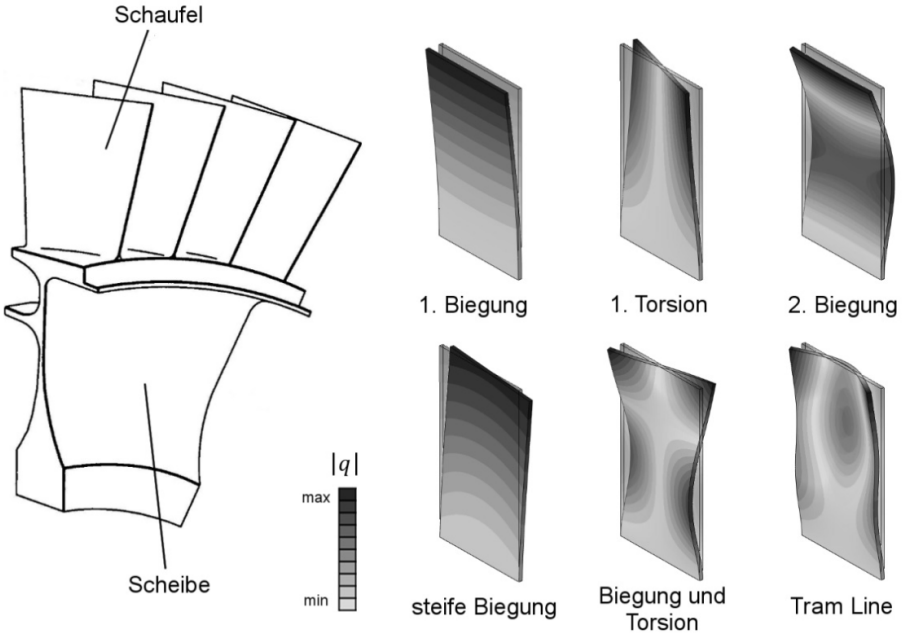


Figure 9.3.: Four blade-disk sectors of a blisk (left) and blade mode shapes of a flat plate clamped at one end (right). Regions of the plate with a high displacement amplitude are shaded, while the light zones contain nodal lines along which the blade shows no displacement from its initial state during the vibration. Taken from Schwitzke (2015)

9.3.1.2. The Behavior of a Disk

The disk serves to absorb centrifugal forces and to transmit the torque to the shaft. It is shown in idealized form in order to illustrate typical disk mode shapes by means of nodal diameters and nodal circles.

Definition 9.6 (Nodal Diameter and Nodal Circle)**.** A *nodal diameter* (KD, from the German "Knotendurchmesser"; usually abbreviated ND in English) is a nodal line through the center of the disk along which the disk is not deformed. A *nodal circle* (KK) is a circular nodal line. The mode shapes are designated KD0, KD1, KD2, … according to their number of nodal diameters. KD0 is the homogeneous, in-phase displacement without nodal diameters. □

These mode shapes are shown in Figure 9.4 for a disk of constant thickness clamped at its center. Mode shapes with nodal circles are relevant only for rotors with correspondingly slender disks. MacBain et al. (1979) compare the experimental generation of mode shapes with nodal diameters with the results of the finite element method (FEM).

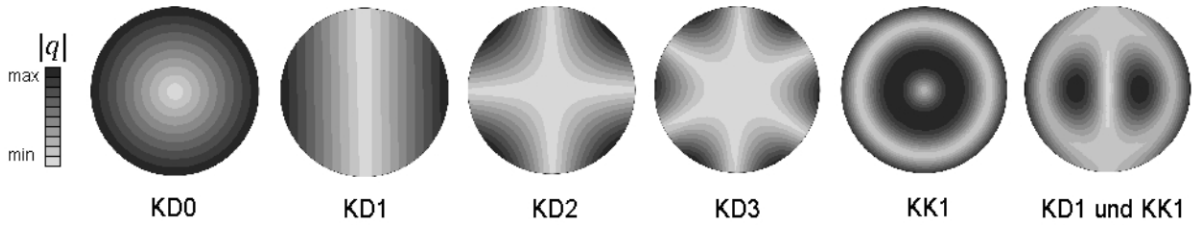


Figure 9.4.: Mode shapes of a flat disk clamped at its center with nodal diameters and nodal circles. The amplitude of the local deformation is again visualized by black shading for the first mode shapes. The first shape on the left describes a homogeneous, in-phase displacement along the entire circumference and can occur in the axial, radial, or circumferential direction. These mode shapes are frequently referred to as the umbrella mode. The next three modes show the characteristic nodal diameters. With increasing natural frequency, their number increases, with the first three shapes (KD1–KD3) shown as examples. The umbrella mode named first follows the nomenclature and is designated KD0. For the idealized structure, further shapes with nodal circles and their superpositions with nodal diameters result, as shown on the right. Taken from Schwitzke (2015)

10. Models for Vibration Analysis

Strehlau (2011) uses a model of a cyclically symmetric structure consisting of individual masses to understand the basic vibration behavior of blisks, see Figure 10.1.

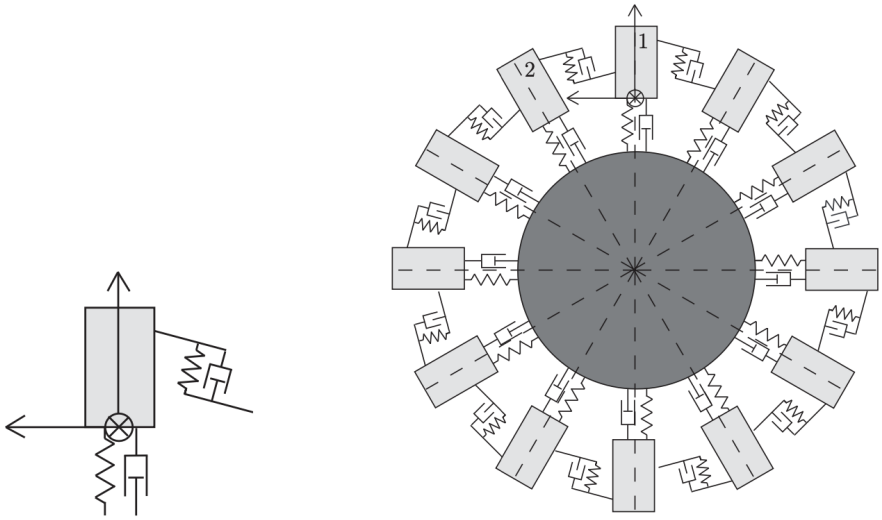


Figure 10.1.: Schematic structure of a cyclically symmetric model; single sector (left), assembled structure (right). Taken from Strehlau (2011)

Table 10.1 summarizes the symbols used in this chapter.

Table 10.1.: Symbols in Chapter 10 (q corresponds to the displacement vector u from Section 8.7; $K = c\,C_N$)

Symbol	Meaning	Type	Defined in
c	Spring constant	Constant	Theorem 2.7
e	Euler's number	Constant	Example 2.1
j	Running index (row/column, mass, blade); italic, as distinct from the imaginary unit j	Index	Definition 10.1
j	Imaginary unit, $\mathrm{j}^2 = -1$	Constant	Equation 2.5
k	Running index (time/iteration step, mode, eigenvalue number)	Index	Definition 3.1
m	Mass	Constant	Theorem 2.6
q, q_j	Displacement vector of the oscillator chain; q_j displacement of element j	Vector	Equation 10.1
$\hat{q}, \hat{q}_k$	Amplitude vector in the ansatz $q = \hat{q}\,\mathrm{e}^{\lambda t}$; $\hat{q}_k$ mode shape (eigenvector) of mode k	Vector	Proposition 10.2
t	Time	Scalar	Figure 2.2

Symbol	Meaning	Type	Defined in
A_k, B_k	Coefficients of the real modal solution, $q(t) = \sum(A_k \cos \omega_k t + B_k \sin \omega_k t)\hat{q}_k$; A_0, B_0 rigid-body component	Scalar	Equation 10.4
C_n	Circulant matrix $(-1, 2, -1$ cyclic); C_N for the ring of N masses	Matrix	Section 8.1
$\underline{C}_k$	Complex coefficient of the modal solution, $\underline{C}_k = \frac{1}{2}(A_k + \mathrm{j}B_k)$; conjugate $\overline{\underline{C}}_k$	Scalar	Equation 10.3
D	Damping matrix	Matrix	Section 10.1
E	Identity matrix	Matrix	Definition 5.1
G	Matrix of the gyroscopic forces (gyroscopic matrix)	Matrix	Section 10.1
K	Stiffness matrix, $K = A^T C A$; here $K = c\, C_N$	Matrix	Definition 8.5, Equation 10.2
K_{ij}	Element of the matrix K in row i, column j	Scalar	Equation 10.2
M	Mass matrix, $M = \mathrm{diag}(m_1, \ldots, m_n)$; here $M = mE$	Matrix	Definition 8.5
N	Number of masses, springs, or blades (ring)	Index	Definition 10.1
ζ	n-th root of unity, $\zeta = \mathrm{e}^{2\pi \mathrm{j}/n}$; $\overline{\zeta}$ conjugate	Scalar	Section 8.1
λ	Eigenvalue or root of the characteristic equation (exponent in the ansatz $\mathrm{e}^{\lambda t}$); $\lambda_{k\pm} = \pm \mathrm{j}\omega_k$	Scalar	Definition 5.1
μ_k	Eigenvalue of C_N, $\mu_k = 4 \sin^2(\pi k/N)$	Scalar	Proposition 10.1
π	The number pi	Constant	–
φ_{KD}	Inter-blade phase angle, $\varphi_{\mathrm{KD}} = 2\pi\, \mathrm{KD}/N$	Scalar	Definition 10.2
ω_k	Natural angular frequency of the k-th normal mode; it vanishes for $k = 0$ (never written as ω_0)	Scalar	Proposition 10.1
$\dot{q}, \ddot{q}$	First and second derivative with respect to time t	Operator	Equation 2.8
$\sum$	Sum	Operator	Equation 8.3
KD	Number of nodal diameters; $\mathrm{KD}_{\max}$ its maximum	Index	Definition 9.6, Equation 10.7

Schwitzke (2015) uses a simplified spring-mass system with N springs and N masses, see Figure 10.2 top left. Using this cyclically symmetric structure with six blades, he illustrates the eigenvalue pairs resulting from the solution of the equation of motion $M\ddot{q} + Kq = 0$ and the associated six natural frequencies and mode shapes.

10.1. Mathematical Derivation

The mathematical considerations in this section follow Strehlau (2011) and Schwitzke (2015). Further information can be found in Zurmühl and Falk (1992) (p. 296, “Zweites Beispiel” (second example), and p. 359, Equations 57 and 59).

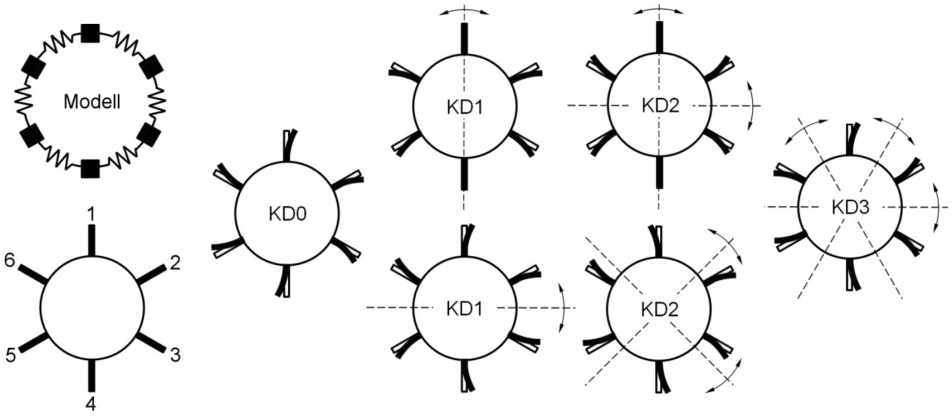


Figure 10.2.: Model of an ideal bladed disk and mode shapes KD0 to KD3 with rotating nodal diameters. For a cyclically symmetric structure with six blades, the solution of the equation of motion yields $N = 6$ eigenvalue pairs and thus six natural frequencies and mode shapes (KD0, twice KD1 and KD2, and once KD3). Taken from Schwitzke (2015)

The general homogeneous equation of motion

$$M\ddot{q} + (D + G)\dot{q} + Kq = 0$$

with mass matrix M, damping matrix D, stiffness matrix K, the matrix G of the gyroscopic forces, and the displacement vector q simplifies for $D = G = 0$ (undamped, without gyroscopic forces) to

$$M\ddot{q} + Kq = 0. \tag{10.1}$$

Definition 10.1 (Cyclic Oscillator Chain)**.** A *cyclic oscillator chain* consists of N identical masses m connected by N identical springs with spring constant c to form a closed ring. Element j obeys the equation of motion

$$m\ddot{q}_j = -c(q_j - q_{j-1}) - c(q_j - q_{j+1}), \qquad j = 1, \dots, N,$$

with cyclically read indices ($q_0 := q_N$, $q_{N+1} := q_1$). In matrix form, this is Equation 10.1 with the mass matrix $M = mE$ and the stiffness matrix

$$K = c\,C_N = c \begin{pmatrix} 2 & -1 & 0 & \dots & 0 & -1 \\ -1 & 2 & -1 & \dots & 0 & 0 \\ 0 & -1 & 2 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 2 & -1 \\ -1 & 0 & 0 & \dots & -1 & 2 \end{pmatrix}, \tag{10.2}$$

that is, $K_{j,j} = 2c$ and $K_{j,j-1} = K_{j,j+1} = -c$ (cyclically, i.e., $K_{1,N} = K_{N,1} = -c$). □

The j-th row of K thus contains the coefficients of the equation of motion of element j. Both M and K are symmetric, and the vector q has N degrees of freedom. With the ansatz for a harmonic oscillation

$$q(t) = \hat{q}\,\mathrm{e}^{\lambda t}$$

division by $\mathrm{e}^{\lambda t} \neq 0$ yields the eigenvalue problem

$$\lambda^2 M\hat{q}\,\mathrm{e}^{\lambda t} + K\hat{q}\,\mathrm{e}^{\lambda t} = 0 \quad \Leftrightarrow \quad (\lambda^2 M + K)\hat{q} = 0.$$

Proposition 10.1 (Natural Angular Frequencies of the Cyclic Oscillator Chain)**.** *The eigenvalue problem* $(\lambda^2 M + K)\hat{q} = 0$ *of the cyclic oscillator chain has* N *solutions for* λ^2 *and thus* $2N$ *eigenvalues* $\lambda_{k\pm} = \pm\mathrm{j}\omega_k$ *with the natural angular frequencies*

$$\omega_k = 2\sqrt{\frac{c}{m}}\left|\sin\frac{\pi k}{N}\right|, \qquad k = 0, 1, \dots, N-1.$$ □

Proof. Since $M = mE$ and $K = c\,C_N$, we have $\lambda^2 = -\frac{c}{m}\mu_k$, where μ_k are the eigenvalues of C_N. By Theorem 8.16 and its subsequent generalization to all eigenvectors of C_n, $\mu_k = 2 - 2\cos(2\pi k/N) = 4\sin^2(\pi k/N)$, $k = 0, \dots, N-1$. Hence $\lambda = \pm\mathrm{j}\,2\sqrt{c/m}\,|\sin(\pi k/N)|$. □

For $k = 0$ the natural angular frequency vanishes: the ring moves as a whole without stretching any spring. The eigenvalue $\lambda = 0$ is then double, and the associated component of the solution is $(A_0 + B_0 t)\,\hat{q}_0$ instead of an oscillation.

The solution is the superposition of the N partial solutions with the eigenvectors $\hat{q}_k$,

$$q(t) = (A_0 + B_0 t)\,\hat{q}_0 + \sum_{k=1}^{N-1} \hat{q}_k \left(\underline{C}_k\, \mathrm{e}^{-\mathrm{j}\omega_k t} + \overline{\underline{C}}_k\, \mathrm{e}^{+\mathrm{j}\omega_k t} \right) \tag{10.3}$$

with constant, mutually complex conjugate coefficients $\underline{C}_k = \frac{1}{2}(A_k + \mathrm{j}B_k)$ and $\overline{\underline{C}}_k = \frac{1}{2}(A_k - \mathrm{j}B_k)$. Euler's formula then yields the real solution

$$q(t) = (A_0 + B_0 t)\,\hat{q}_0 + \sum_{k=1}^{N-1} \left(A_k \cos(\omega_k t) + B_k \sin(\omega_k t) \right) \hat{q}_k. \tag{10.4}$$

The eigenvalues $\lambda_{k\pm}$ follow from the natural angular frequencies ω_k; the eigenvectors $\hat{q}_k$ are the associated mode shapes.

Proposition 10.2 (Mode Shapes of the Cyclic Oscillator Chain)**.** *For $0 < k < N/2$, the indices k and $N - k$ share the same natural angular frequency $\omega_k = \omega_{N-k}$. The associated eigenvectors*

$$\hat{q}_k = \begin{pmatrix} \cos(1 \times 2\pi k/N) \\ \cos(2 \times 2\pi k/N) \\ \vdots \\ \cos(j \times 2\pi k/N) \\ \vdots \\ \cos(N \times 2\pi k/N) \end{pmatrix} \tag{10.5}$$

and

$$\hat{q}_{N-k} = \begin{pmatrix} \sin(1 \times 2\pi k/N) \\ \sin(2 \times 2\pi k/N) \\ \vdots \\ \sin(j \times 2\pi k/N) \\ \vdots \\ \sin(N \times 2\pi k/N) \end{pmatrix} \tag{10.6}$$

are mutually orthogonal. The j-th row contains the displacement of the j-th element of the oscillator chain. For $k = 0$ and, for even N, for $k = N/2$, only the cosine vector Equation 10.5 is an eigenvector. □

Proof. As shown after Theorem 8.16, $(\zeta^k, \zeta^{2k}, \ldots, \zeta^{Nk})^T$ with $\zeta = \mathrm{e}^{2\pi\mathrm{j}/N}$ is an eigenvector of C_N for the real eigenvalue μ_k. Since C_N is real, the real and imaginary parts of this vector are also eigenvectors for μ_k. These are Equation 10.5 and Equation 10.6. For $k = 0$ and $k = N/2$ the imaginary part vanishes. Orthogonality follows from $\sum_{j=1}^{N} \cos(2\pi jk/N) \sin(2\pi jk/N) = \frac{1}{2} \sum_{j=1}^{N} \sin(4\pi jk/N) = 0$ for $2k \neq N$. □

10.2. Interpretation of the Mathematical Considerations

Section 10.1 yields the following relationships. For $k = 0$, the eigenvector shows that all blades are maximally displaced at the same time. This corresponds to the KD0 mode shape in Figure 10.2 (bottom left), for which a single eigenvalue occurs. The indices k and $N - k$ yield a cosine- or sine-shaped deformation along the circumference, that is, a mode shape with $k =$ KD nodal diameters, see Figure 10.3.

For an even number N of blade-disk sectors, $k = N/2$ also has only a single solution; in the case considered (Figure 10.2), this is the KD3 mode shape. All remaining eigenvalues, i.e., for odd N all those with $k \neq 0$, occur twice. Their two eigenvectors are mutually orthogonal, which, as in Figure 10.3, shows up as a position of the nodal diameter lines rotated by the angle $\pi/(2\,\mathrm{KD})$.

At the maximum number of nodal diameters, all adjacent blades vibrate in antiphase. For odd numbers of blades, one blade does not participate in the vibration at $\mathrm{KD}_{\max}$ if the nodal diameter line passes through it. Mode shapes with $\mathrm{KD} > \mathrm{KD}_{\max}$ are confined to the disk, since the cyclically symmetric structure, with its limited number of blade-disk sectors, cannot represent such vibrations. For example, the dashed KD5 mode shape in Figure 10.3 exhibits the same blade displacements as the one with a single nodal diameter. The maximum number of nodal diameters follows from the number N of blades:

$$\mathrm{KD}_{\max} = \begin{cases} N/2 & \text{for even } N, \\ (N-1)/2 & \text{for odd } N. \end{cases} \tag{10.7}$$

10.3. Phase Angle of the Blades

According to Equation 10.3 or Equation 10.4, the j-th blade, i.e., the j-th element of the oscillator chain, performs a harmonic oscillation that can be represented either in complex exponential form or with trigonometric functions. A single mode shape describes an in-phase vibration of all blades with nodal diameter lines fixed relative to the structure. Since the double eigenvalue pairs share the same natural frequency, both mode shapes occur superimposed under excitation. Their superposition yields a rotating mode shape. Since the nodal diameter lines have no preferred direction in the ideal system, they rotate counterclockwise or clockwise, as indicated in Figure 10.2. The blades perform a wave motion and consequently do not vibrate in phase.

Definition 10.2 (Inter-Blade Phase Angle)**.** The *inter-blade phase angle* (blade phase angle) is the phase difference between two adjacent blades of a rotating mode shape with KD nodal diameters,

$$\varphi_{\mathrm{KD}} = \frac{2\pi\,\mathrm{KD}}{N}. \qquad \square$$

Figure 10.3 shows the inter-blade phase angle for KD1. Each blade experiences its maximum displacement at a different time and leads or lags the adjacent blade by φ_{KD}.

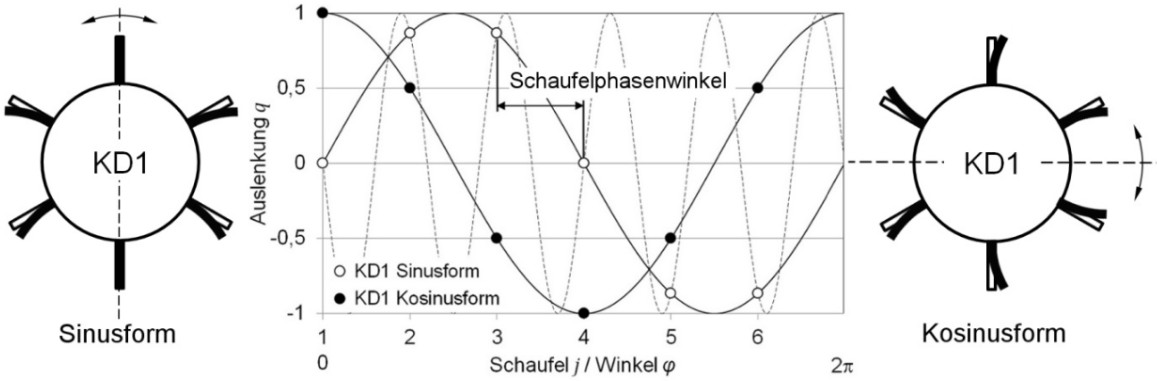


Figure 10.3.: Unrolled sine and cosine shapes for the KD1 vibration and a KD5 vibration (dashed) as well as the inter-blade phase angle for KD1. Taken from Schwitzke (2015)

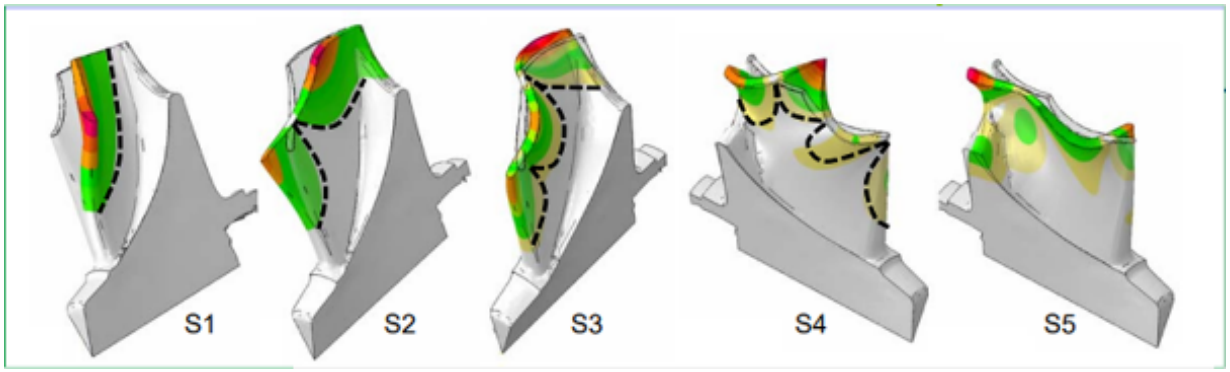


Figure 10.4.: Overview of the vibration modes of a free-standing blade; the deformation of the blade is shown. Source: MAN Energy Solution SE (2023)

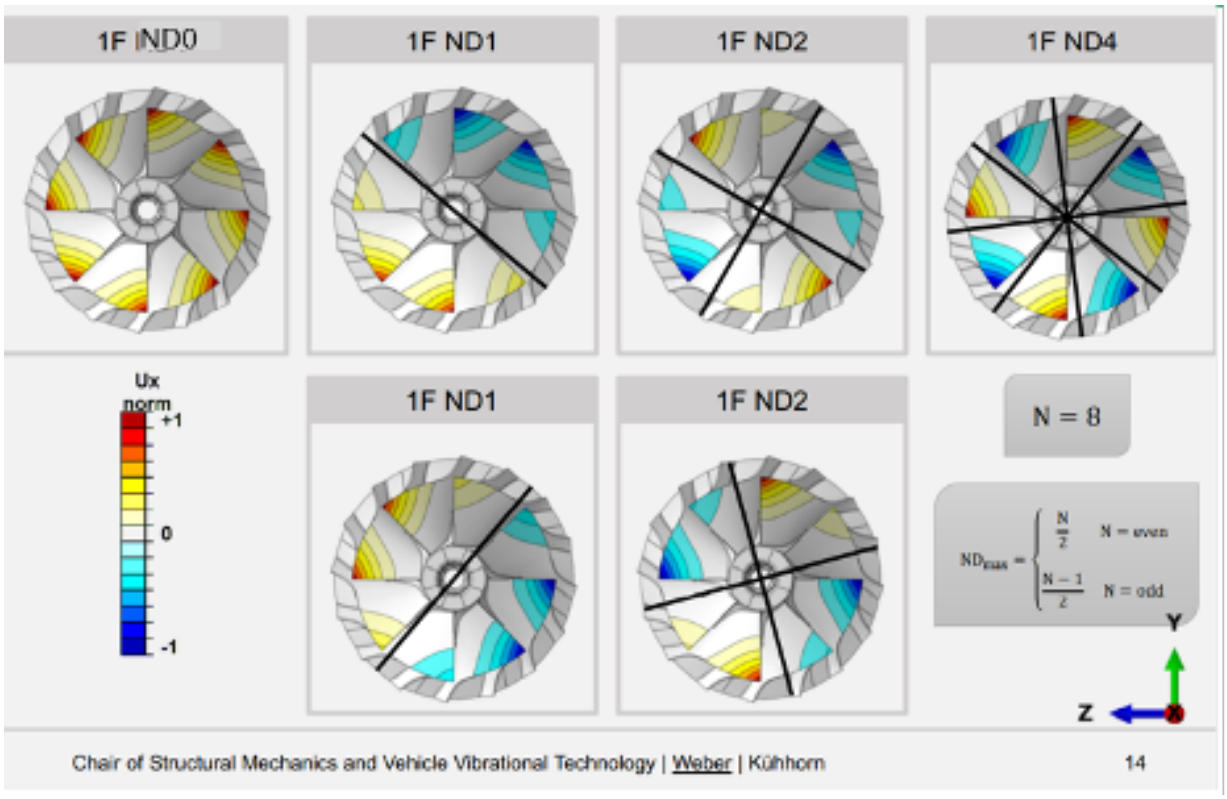


Figure 10.5.: Overview of the nodal diameters (ND0 to ND4) of the disk for the first blade vibration mode S1 from Figure 10.4. The deformation of the blades along the circumference is shown as a function of the nodal diameter of the disk; ND0 to ND4 correspond to KD0 to KD4. Source: MAN Energy Solution SE (2023)

Part IV.

Project

11. Project: Vibration Analysis of Turbochargers

Exercise 11.1 (Spring-Mass Systems)**.**

1. Implement a solution for the spring-mass system with N springs and N masses described in Chapter 10 (after Schwitzke (2015)) in order to analyze the vibrations of a turbocharger. The model you implement should be based on Equation 10.1 and should be able to compute the vibrations of the turbocharger as a function of the system parameters. Try to reproduce the steps described in Section 10.1. Where possible, create graphical representations of the results.
2. Compute the natural frequencies/eigenvalues and mode shapes/eigenvectors of the system, analytically where possible and additionally numerically (by programming). □

> **Deliverable Format**
>
> Submit a Jupyter notebook that runs on the IDE+A JupyterHub, additionally as a PDF (max. 10 pages).

References

Furlan, Peter. 1995. *Das Gelbe Rechenbuch*. Vol. 3. Martina Furlan.

Kong, Qingkai, Timmy Siauw, and Alexandre Bayen. 2020. *Python Programming and Numerical Methods. A Guide for Engineers and Scientists*. Elsevier.

MacBain, J. C., J. E. Horner, W. A. Stange, and J. S. Ogg. 1979. "Vibration Analysis of a Spinning Disk Using Image-Derotated Holographic Interferometry." *Experimental Mechanics* 19 (1): 17–22.

MAN Energy Solution SE. 2023. *Persönliche Mitteilung*.

Papula, Lothar. 2015. *Mathematik für Ingenieure und Naturwissenschaftler, Band 2*. 14. Springer.

Schwitzke, Martin. 2015. "Aerodynamische Anregung Resonanter Schaufelschwingungen in Radialturbinen." PhD thesis, Fakultät für Maschinenbau Karlsruher Institut für Technologie (KIT).

Strang, Gilbert. 2010. *Wissenschaftliches Rechnen*. Springer.

Strehlau, Ulrik. 2011. "Schaufelschwingungen Realer Integraler Verdichterlaufräder Bei Zusätzlicher Berücksichtigung höherer Moden." PhD thesis.

Sundnes, Joakim. 2023. *Solving Ordinary Differential Equations in Python*.

Zurmühl, Rudolf, and Sigurd Falk. 1992. *Matrizen 1: Grundlagen*. 6th ed. Springer.

A. Table of Integrals

Table A.1 lists the antiderivatives used in these lecture notes, in particular for separation of variables and variation of constants (Section 2.2.1). Constants of integration are omitted.

Table A.1.: Antiderivatives

$f(x)$	$\int f(x)\,\mathrm{d}x$
1	x
$x^n,\ n \neq -1$	$\frac{x^{n+1}}{n+1}$
$\frac{1}{x}$	$\ln\lvert x\rvert$
e^x	e^x
$\mathrm{e}^{ax},\ a \neq 0$	$\frac{1}{a}\,\mathrm{e}^{ax}$
$\sin x$	$-\cos x$
$\cos x$	$\sin x$
$x\,\mathrm{e}^x$	$(x-1)\,\mathrm{e}^x$
$x\cos x$	$x\sin x + \cos x$

The last two rows follow from integration by parts

$$\int u'v\,\mathrm{d}x = uv - \int uv'\,\mathrm{d}x,$$

see also Remark 2.1.