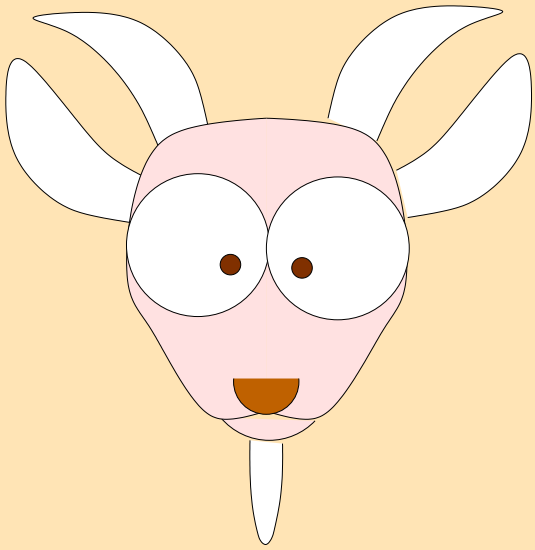
Some Engineering Mathematics

C. Pozrikidis



VERSION 26.02B

CHESTER & BENNINGTON

Some Engineering Mathematics

C. Pozrikidis

Chester & Bennington
2026

Chester & Bennington

Pozrikidis, C.
Some Engineering Mathematics / C. Pozrikidis. – Verstion 26.02B

Contents

Preface

A good way of learning, teaching, and refamiliarizing oneself with methods of applied mathematics and scientific computing is to study a selected set of topics related to applied science and engineering that are a blend of the old, the new, and the emerging.

This book discusses a pertinent collection of such carefully selected topics, problems, algorithms, and complete computer codes, including mainstream and original material. The discussion of the individual topics and accompanying mathematical and computational analysis provide opportunities for recalling, clarifying, and learning methods and developing skills.

One main goal of this book is to provide insights into a fair question often asked: *what are the applications of a particular method or topic in applied science and engineering?* The particular method or topic can be *tensor*, *separation of variables*, *curvilinear coordinates*, *algorithms*, *numerical methods*, *complex variables*, *fractional calculus*, and others. This book helps answer this question by providing motivation and by making cases by example. Regarded in hindsight, the discussion illustrates the usefulness of applied mathematics, scientific computing, and computer coding to problems encountered in a broad range of applications. A junior or even lower level background on linear algebra and calculus, and elementary familiarity with computer programming are assumed for most of the topics covered.

This book is suitable as a text for instruction in a mainstream, special topics, or special projects advanced undergraduate course or introductory graduate course in applied mathematics, engineering mathematics, computer programming, and scientific computing in the applied sciences and engineering. More broadly, the book offers a point of departure for exploring mathematical and computational concepts and ideas, performing online research from different points of view, and pursuing individual study. The intended audience includes students, course instructors, researchers, and anyone interested in science, technology, and mathematics.

In line with the intended character of this book, the notion of a physical or conceptual vector or tensor is developed from the ground up in a way that can be understood by the non-expert, the concept of volume and surface curvilinear coordinates is discussed in detail from a theoretical and computational standpoint with applications in physical transport and mechanics, the solution of differential equations by the method of separation of variables is exemplified with emphasis on engineering applications, and the Moore–Penrose inverse of a matrix is explained in the context of data regression and based on the solution of the Poisson equation with Neumann boundary conditions around the entire boundary of a solution domain.

Elementary and advanced *Matlab/Octave* computer codes performing miscellaneous computations, simulations, and graphics visualizations are listed and discussed in the text.[1][2] The codes illustrate explicitly how mathematical equations translate into computer code instructions. The computer programs are arranged in a software library named SEMLIB, which is available freely on the Internet under the terms of the current GNU license agreement. The library can be downloaded from the Internet book site: http://dehesa.freeshell.org/SEM

The book complements mainstream textbooks, online resources, and lecture notes on applied mathematics in a way that circumvents the need for referencing background material and invoking the work of others. Consistent with this goal, references and citations are not provided with an understanding that, if needed, they can be obtained readily by searching the Internet.

C. Pozrikidis

[1] Matlab is a proprietary computing environment for numerical computation and data visualization. Matlab® and Simulink are registered trademarks of The MathWorks, Inc. For product information, please contact: The MathWorks, Inc., 3 Apple Hill Drive, Natick, MA 01760-2098, USA, Tel: 508-647-7000, Fax: 508-647-7001, E-mail: info@mathworks.com, Web: www.mathworks.com

[2] *GNU octave* is a public-domain software application incorporating a Matlab-like high-level programming language for scientific computing and data visualization.

Notation

a	Scalars are set in italic
$\mathbf{A}$	Vectors and matrices are set in bold face
$\widehat{\circ}$	A caret (hat) over a symbol indicates an object
$\mathbf{A}^{\mathrm{T}}$	Matrix transpose
$\mathbf{A}^{-1}$	Matrix inverse
$\cdot$	Vector inner product or matrix product
$\times$	Outer product of two vectors
$\otimes$	Tensor product of two vectors
$\boldsymbol{\nabla}$	Gradient (nabla) operator
∇^2	Laplacian operator
$\nabla^2 f = 0$	Laplace equation
$\nabla^2 f + g = 0$	Poisson equation
$\nabla^2 f + \kappa f = 0$	Helmholtz equation
δ_{ij}	Kronecker's delta
ϵ_{ijk}	Alternating tensor

Chapter 1

Separation of variables

A linear partial differential equation (pdf) specifies that a linear combination of a function and its partial derivatives is zero over the entire solution domain. Linear partial differential equations can sometimes be solved by separating the dependencies of the unknown function on the independent variables, typically consisting of spatial coordinates and time. The methodology essentially leads to expansions in Fourier series involving trigonometric functions or other Sturm–Liouville eigensolutions.

Not all linear partial differential equations and not all coordinate systems admit separable solutions. Moreover, in practice, the method of separations of variables is restricted to solution domains with simple geometry. In spite of these limitations, the process of separation of variables provides us with mathematical insights and expressions that serve as computational benchmarks.

1.1 Flow through a rectangular tube

Consider unidirectional fluid flow through a tube whose cross-section is a rectangle with side-lengths equal to $2a$ and $2b$, as illustrated in Figure 1.1.1. The origin of the working Cartesian coordinates is located at the tube center. In the absence of entrance effects, exit effects, and turbulent motion, the axial velocity, u_x, satisfies a Poisson equation with a constant right-hand side,

$$\nabla^2 u_x = \frac{\chi + \rho g_x}{\mu}, \tag{1.1.1}$$

where

$$\nabla^2 = \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} \tag{1.1.2}$$

is the Laplacian operator in the yz plane, χ is a constant expressing the negative of the axial pressure gradient, g_x is the x component of the gravitational acceleration, ρ is the fluid density, and μ is the fluid viscosity.

The no-slip boundary condition stipulates that the velocity is zero along the four walls, $u_x = 0$ at $y = \pm a$ and $z = \pm b$.

1.1.1 Particular and homogeneous solutions

A standard method of solving the Poisson equation (1.1.1) involves expressing the solution for the velocity as the sum of two constituents:

1. A *chosen* particular solution that satisfies the Poisson equation (1.1.1) but not all boundary conditions, denoted as $u_x^{\rm p}(y, z)$.

2. A homogeneous solution, $u_x^{\rm h}(y, z)$, that satisfies Laplace's equation,

$$\nabla^2 u_x^{\rm h} = 0, \tag{1.1.3}$$

   subject to derived boundary conditions.

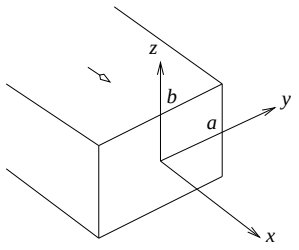


FIGURE 1.1.1 Illustration of steady pressure- or gravity-driven unidirectional flow through a rectangular tube or duct.

Boundary conditions for the homogeneous solution arise by requiring that the physical velocity,

$$u_x = u_x^{\mathrm{p}} + u_x^{\mathrm{h}}, \tag{1.1.4}$$

satisfies the no-slip boundary condition along the four walls, $u_x = 0$ at $y = \pm a$ and $z = \pm b$.

1.1.2 Choice of particular solution

A convenient particular solution representing Hagen–Poiseuille flow in a two-dimensional channel bounded by two parallel walls is

$$u_x^{\mathrm{p}}(z) = \tfrac{1}{2}\,\frac{\chi + \rho g_x}{\mu}\,(b^2 - z^2). \tag{1.1.5}$$

Requiring that $u_x^{\mathrm{p}} = -u_x^{\mathrm{h}}$ along the four channel walls, we derive boundary conditions for the homogeneous solution,

$$u_x^{\mathrm{h}}(y = \pm a, z) = -u_x^{\mathrm{p}}(y = \pm a, z) \tag{1.1.6}$$

and

$$u_x^{\mathrm{h}}(y, z = \pm b) = -u_x^{\mathrm{p}}(y, z = \pm b) = 0. \tag{1.1.7}$$

Equation (1.1.6) yields

$$u_x^{\mathrm{h}}(y = \pm a, z) = -\tfrac{1}{2}\,\frac{\chi + \rho g_x}{\mu}\,(b^2 - z^2). \tag{1.1.8}$$

1.1.3 Separation of variables

Because the homogeneous solution cannot be found in closed form, we resort to a Fourier series representation mediated by separating the y and z dependencies. To develop a solution by separation of variables, we set

$$u_x^{\mathrm{h}}(y, z) = \frac{\chi + \rho g_x}{\mu}\,b^2 \sum_{n=1}^{\infty} f_n(y)\, q_n(z), \tag{1.1.9}$$

where (f_n, q_n) are corresponding pairs of dimensionless functions to be found as part of the solution for $n = 1, \ldots, \infty$. Straightforward differentiation provides us with the Laplacian

$$\nabla^2 u_x^{\rm h}(y,z) = \frac{\chi + \rho g_x}{\mu}\, b^2 \sum_{n=1}^{\infty} \Big(\frac{{\rm d}^2 f_n}{{\rm d}y^2}\, q_n + \frac{{\rm d}^2 q_n}{{\rm d}z^2}\, f_n \Big). \tag{1.1.10}$$

A key observation is that the Laplacian is zero only if the general term enclosed by the tall parentheses is zero. Setting this term to zero and rearranging, we find that

$$\frac{1}{f_n}\frac{{\rm d}f_n}{{\rm d}y^2} = -\frac{1}{q_n}\frac{{\rm d}q_n}{{\rm d}z^2} = \frac{\alpha_n^2}{b^2}, \tag{1.1.11}$$

where α_n is a constant. The reason for this double equality is that the left-hand side is a function of y, while the middle side is a function of z; both must be equal to a constant that is presently assumed to be positive without loss of generality for reasons that will become apparent in hindsight.

The double equality in (1.1.11) provides us with two ordinary differential equations,

$$\frac{{\rm d}^2 f_n}{{\rm d}y^2} - \frac{\alpha_n^2}{b^2}\, f_n = 0, \qquad \frac{{\rm d}^2 q_n}{{\rm d}z^2} + \frac{\alpha_n^2}{b^2}\, q_n = 0. \tag{1.1.12}$$

The general solutions are

$$f_n(y) = A_n \cosh(\alpha_n \frac{y}{b}) + B_n \sinh(\alpha_n \frac{y}{b}) \tag{1.1.13}$$

and

$$q_n(z) = C_n \cos(\alpha_n \frac{z}{b}) + D_n \sin(\alpha_n \frac{z}{b}), \tag{1.1.14}$$

where A_n, B_n, C_n, D_n, are constant coefficients to be found as part of the solution.

To ensure that the functions $f_n(y)$ and $q_n(z)$ are even, and therefore the velocity profile is symmetric with respect to the vertical and horizontal mid-planes $y = 0$ and $z = 0$, we discard the hyperbolic sines and sines by setting $B_n = 0$ and $D_n = 0$.

Substituting the result into (1.1.9), we obtain

$$u_x^{\mathrm{h}}(y,z) = \frac{\chi + \rho g_x}{\mu}\, b^2 \sum_{n=1}^{\infty} E_n \cosh(\alpha_n \frac{y}{b}) \cos(\alpha_n \frac{z}{b}), \tag{1.1.15}$$

where $E_n = A_n C_n$. It can be confirmed by straightforward differentiation that this expression satisfies Laplace's equation for any coefficients, E_n.

If a minus sign was inserted in the right-hand side of (1.1.11), we would have obtained trigonometric functions in y and hyperbolic sines and cosines in z, which is certainly acceptable. We will see that the choice of the plus sign considerably facilitates the implementation of the boundary conditions.

1.1.4 Boundary conditions

To satisfy the boundary conditions (1.1.7), we set

$$\alpha_n = (n - \tfrac{1}{2})\,\pi, \tag{1.1.16}$$

where n is a natural number. To satisfy the remaining boundary conditions (1.1.8), we require that

$$\sum_{n=1}^{\infty} E_n \cosh(\alpha_n \frac{a}{b}) \cos(\alpha_n \frac{z}{b}) = -\tfrac{1}{2}\,(1 - \frac{z^2}{b^2}). \tag{1.1.17}$$

In fact, the left-hand side of (1.1.17) is the Fourier series with respect to z of the quadratic function on the right-hand side.

1.1.5 Fourier orthogonality

The solution follows readily using the trigonometric identity

$$\int_{-b}^{b} \cos(\alpha_n \frac{z}{b}) \cos(\alpha_m \frac{z}{b})\, \mathrm{d}z = \begin{cases} b & \text{if } n = m, \\ 0 & \text{if } n \neq m. \end{cases} \tag{1.1.18}$$

Multiplying both sides of (1.1.17) by $\cos(\alpha_m\, z/b)$, where m is an arbitrary integer, integrating with respect to z from $-b$ to b, using (1.1.18)

to compute the integrals on the left-hand side, and then renaming m to n, we obtain

$$E_n = -\tfrac{1}{2}\,\frac{1}{\cosh(\alpha_n\,a/b)}\,\frac{1}{b}\int_{-b}^{b}(1-\frac{z^2}{b^2})\,\cos(\alpha_n\frac{z}{b})\,\mathrm{d}z. \qquad (1.1.19)$$

Defining $\eta \equiv \alpha_n z/b$, we obtain

$$E_n = -\tfrac{1}{2}\,\frac{1}{\cosh(\alpha_n\,a/b)}\,\frac{1}{\alpha_n}\int_{-\alpha_n}^{\alpha_n}(1-\frac{\eta^2}{\alpha_n^2})\,\cos\eta\,\mathrm{d}\eta. \qquad (1.1.20)$$

The definite integral on the right-hand side is given by

$$\int_{-\alpha_n}^{\alpha_n}(1-\frac{\eta^2}{\alpha_n^2})\,\cos\eta\,\mathrm{d}\eta = \Big[\,\sin\eta + \frac{(2-\eta^2)\sin\eta - 2\eta\cos\eta}{\alpha_n^2}\,\Big]_{-\alpha_n}^{\alpha_n}. \qquad (1.1.21)$$

Noting that $\cos\alpha_n = 0$ and $\sin\alpha_n = (-1)^{n+1}$, we obtain

$$E_n = (-1)^n\,\frac{2}{\cosh(\alpha_n\,\frac{a}{b})}\,\frac{1}{\alpha_n^3} \qquad (1.1.22)$$

for $n \geq 1$.

Finally, we substitute expression (1.1.22) into (1.1.15) and add the homogeneous to the particular solution expressed by (1.1.5) to obtain the velocity distribution in the form of a Fourier series,

$$u_x(y,z) = \tfrac{1}{2}\,\frac{\chi+\rho g_x}{\mu}\,b^2\,\Psi(y,z), \qquad (1.1.23)$$

where

$$\Psi(y,z) \equiv 1 - \frac{z^2}{b^2} + 4\sum_{n=1}^{\infty}(-1)^n\,\frac{1}{\alpha_n^3}\frac{\cosh(\alpha_n\,\frac{y}{b})}{\cosh(\alpha_n\,\frac{a}{b})}\,\cos(\alpha_n\frac{z}{b}) \qquad (1.1.24)$$

is a dimensionless shape function. The velocity distribution over the cross-section of a tube with aspect ratio $b/a = 2$ is shown in Figure 1.1.2(a). The graphics display was generated using a code located in directory TUBE_REC of SEMLIB.

(a)

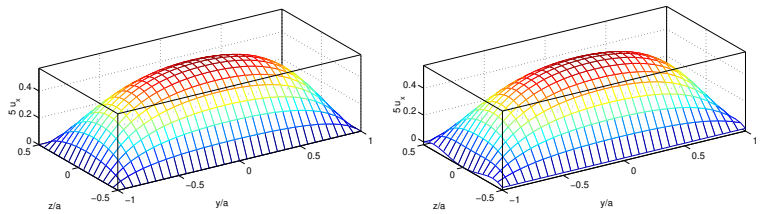


FIGURE 1.1.2 (a) Velocity distribution over a rectangular tube with aspect ratio $a/b = 2$. (b) Approximate distribution subject to the one-term Fourier expansion.

For comparison, the velocity distribution computed by retaining only one term Fourier expansion, $n = 1$, is shown in Figure 1.1.2(b). Only small differences between the exact and the approximate solution are observed around the edges and over the cross-section of the rectangle.

1.1.6 Flow rate

The flow rate through the tube arises by integrating the velocity distribution over the tube cross-section in the yz plane,

$$Q \equiv \int_{-a}^{a} \int_{-b}^{b} u_x(y,z)\, \mathrm{d}y\, \mathrm{d}z. \tag{1.1.25}$$

Substituting expression (1.1.23) and carrying out the integration, we obtain

$$Q = \tfrac{4}{3} \frac{\chi + \rho g_x}{\mu}\, ab^3\, F, \tag{1.1.26}$$

where

$$F = 1 - 6\,\frac{b}{a} \sum_{n=1}^{\infty} \frac{1}{\alpha_n^5}\, \tanh(\alpha_n \frac{a}{b}). \tag{1.1.27}$$

We note that the flow rate depends on the fourth power of the combined side half lengths, ab^3.

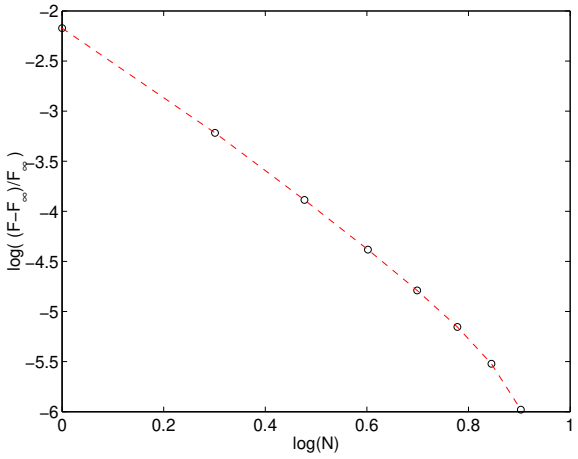


FIGURE 1.1.3 Dependence of the sum truncation level, N, on the dimensionless coefficient F determining the flow rate for a square tube, $a = b$.

1.1.7 Spectral convergence

A graph of the dimensionless coefficient F against the sum truncation level, N, is shown in Figure 1.1.3 on a log-log scale for a square tube, $a = b$. We observe that keeping only one term in the sum, $N = 1$, incurs a relatively small error on the order of 10^{-2}. The downward trend of the curve is typical of *spectral convergence* associated with Fourier expansions. A downward straight line would be typical of algebraic convergence.

1.1.8 Other representations

By symmetry, we also have another expression for the flow rate,

$$Q = \tfrac{4}{3}\,\frac{\chi + \rho g_x}{\mu}\,a^3 b\,G, \tag{1.1.28}$$

where

$$G \equiv 1 - 6\,\frac{a}{b}\sum_{n=1}^{\infty}\frac{1}{\alpha_n^5}\,\tanh(\alpha_n\frac{b}{a}). \tag{1.1.29}$$

Comparing expressions (1.1.27) and (1.1.29), we derive the identity

$$1 - 6\,\frac{1}{\gamma}\,\sum_{n=1}^{\infty}\frac{1}{\alpha_n^5}\,\tanh(\alpha_n\gamma) = \gamma^2\,\big(\,1 - 6\,\gamma\,\sum_{n=1}^{\infty}\frac{1}{\alpha_n^5}\,\tanh(\alpha_n\,\frac{1}{\gamma})\,\big), \tag{1.1.30}$$

where $\gamma = a/b$ is the aspect ratio.

1.1.9 Flow through a rectangular duct

Consider unidirectional gravity-driven flow through a tilted rectangular duct with side-lengths equal to $2a$ and $2b$, as illustrated in Figure 1.1.1. The top of the duct is open to the atmosphere and the free surface is assumed to be flat. The fluid velocity at the bottom, left, and right walls, and the shear stress at the top free surface are required to be zero.

Working as previously in this section in the absence of a streamwise pressure gradient, we express the solution in the deliberate form

$$u_x(y,z) = \tfrac{1}{2}\,\frac{\mu g_x}{\rho}\,(a^2 - y^2) + \sum_{n=1}^{\infty} A_n\,\cosh\big(\alpha_n\,\frac{b-z}{a}\big)\,\cos(\alpha_n\frac{y}{a}), \tag{1.1.31}$$

where $\alpha_n = (n - \frac{1}{2})\,\pi$ and A_n are Fourier coefficients. If the duct is inclined at an angle β with respect to the horizontal, $g_x = g\sin\beta$.

The Fourier representation satisfies the governing equation and three boundary conditions,

$$u_x(y = \pm a, z) = 0, \qquad \frac{\partial u_x}{\partial z}(y, z = b) = 0. \tag{1.1.32}$$

To satisfy the fourth boundary condition, $u_x(y, z = -b) = 0$, we require that

$$\sum_{n-1}^{\infty} A_n \cosh(\,2\,\frac{b}{a}\,\alpha_n)\,\cos(\alpha_n\frac{y}{a}) = \tfrac{1}{2}\,\frac{\mu g_x}{\rho}\,(y^2 - a^2). \tag{1.1.33}$$

Using the orthogonality property

$$\int_{-a}^{a}\cos(\alpha_n\frac{y}{a})\cos(\alpha_m\frac{y}{a})\,\mathrm{d}y = \begin{cases} a & \text{if } n = m, \\ 0 & \text{if } n \neq m, \end{cases} \tag{1.1.34}$$

we find that

$$A_n = (-1)^n \frac{2}{\cosh(2\frac{b}{a}\,\alpha_n)} \frac{\rho g_x}{\mu}\, a^2\, \frac{1}{\alpha_n^3} \tag{1.1.35}$$

for $n \geq 1$. The velocity profile is thus given by

$$u_x(y,z) = \tfrac{1}{2}\, \frac{\rho g_x}{\mu}\, a^2\, \Xi(y,z), \tag{1.1.36}$$

where

$$\Xi(y,z) \equiv 1 - \frac{y^2}{a^2} + 4 \sum_{n=1}^{\infty} (-1)^n\, \frac{1}{\alpha_n^3}\, \frac{\cosh[\alpha_n(b-z)/a]}{\cosh(2\frac{b}{a}\alpha_n)}\, \cos(\alpha_n \frac{y}{a}). \tag{1.1.37}$$

The flow rate is found by straightforward integration over the duct cross-section,

$$Q(a,b) = \tfrac{4}{3}\, \frac{\rho g_x}{\mu}\, a^3 b\, H, \tag{1.1.38}$$

where

$$H \equiv 1 - 3\, \frac{a}{b} \sum_{n=1}^{\infty} \frac{\tanh(2\frac{b}{a}\,\alpha_n)}{\alpha_n^5}. \tag{1.1.39}$$

Since the velocity profile for duct flow is the same as half the profile for tube flow with twice as large b, the flow rate for tube flow given in (1.1.29) arises from that for duct flow as $Q_{\rm tube} = 2\, Q_{\rm duct}(a, \frac{b}{2})$.

1.1.10 Flow driven by the translation of the top wall

Consider unidirectional viscous flow along the rectangular tube shown in Figure 1.1.1, driven exclusively by the streamwise translation of the top wall located at $z = b$ with velocity V, while the other three walls are stationary. The x velocity component satisfies Laplace's equation,

$$\frac{\partial^2 u_x}{\partial y^2} + \frac{\partial^2 u_x}{\partial z^2} = 0. \tag{1.1.40}$$

The boundary conditions specify that $u_x = 0$ at $y = \pm a$ and $z = -b$, and $u_x = V$ at $z = b$.

Working as previously in this section, we derive a Fourier representation in y,

$$u_x(y,z) = V \sum_{n=1}^{\infty} E_n \sinh\left(\alpha_n \frac{z+b}{a}\right) \cos(\alpha_n \frac{y}{a}), \tag{1.1.41}$$

where E_n are *a priori* unknown Fourier coefficients and $\alpha_n = (n - \frac{1}{2})\pi$. This representation satisfies the boundary conditions at the left, right, and bottom walls.

Enforcing the no-slip boundary condition at the top wall, we obtain

$$\sum_{n=1}^{\infty} E_n \sinh\left(2\alpha_n \frac{b}{a}\right) \cos(\alpha_n \frac{y}{a}) = 1. \tag{1.1.42}$$

Use the Fourier orthogonality property (1.1.34), we find that

$$E_n = (-1)^n \frac{2}{\alpha_n} \frac{1}{\sinh\left(2\alpha_n \frac{b}{a}\right)}, \tag{1.1.43}$$

thereby completing the solution.

Exercises

1.1.1 Derive expression (1.1.29).

1.1.2 Write a code to confirm identity (1.1.30).

1.1.3 Derive expression (1.1.35).

1.1.4 Derive the an expression for the flow rate of the flow driven by the translation of the top wall.

1.2 Flow through a rectangular tube with slip

The fluid may slip over the rectangular tube wall discussed in Section 1.1, so that the wall velocity is proportional to the wall shear rate or shear stress. At the right and left walls, we stipulate that

$$u_x(y = \pm a, z) = \overline{+}\ell \left(\frac{\partial u_x}{\partial y} \right)_{y=\pm a}, \tag{1.2.1}$$

where ℓ is a specified *slip length*. At the upper and lower walls, we stipulate that

$$u_x(y, z = \pm b) = \overline{+}\ell \left(\frac{\partial u_x}{\partial z} \right)_{z=\pm b}. \tag{1.2.2}$$

The signs are motivated by the expected negative or positive slope of the velocity profile at the wall. The no-slip boundary condition arises as $\ell \to 0$, and the zero stress boundary condition arises as $\ell \to \infty$.

1.2.1 Wall sleeve

To interpret the physics of the wall slip condition, we may assume that the wall is lined with a thin layer that slides over the wall. Balancing the shear stress is acting on the upper side of the sleeve with the frictional force, we obtain

$$\mu \left(\frac{\partial u_x}{\partial y} \right)_{y=\pm a} \pm \eta\, u_x = 0, \tag{1.2.3}$$

where μ is the fluid viscosity and η is a frictional coefficient. Rearranging, we obtain (1.2.1) with $\ell = \mu/\eta$.

1.2.2 Particular and homogeneous solutions

A convenient particular solution satisfying the boundary condition (1.2.2) is

$$u_x^{\rm p}(z) = \tfrac{1}{2} \frac{\chi + \rho g_x}{\mu} (b^2 + 2\, b\, \ell - z^2). \tag{1.2.4}$$

When $\ell = 0$, we obtain Hagen–Poiseuille flow in a two-dimensional channel whose walls are located at $z = \pm b$.

Repeating the Fourier analysis of Section 1.1, we find that the homogeneous solution to be added to the particular solution is given by (1.1.15), repeated below for convenience,

$$u_x^{\rm h}(y,z) = \frac{\chi + \rho g_x}{\mu}\, b^2 \sum_{n=1}^{\infty} E_n \, \cosh(\alpha_n \frac{y}{b}) \, \cos(\alpha_n \frac{z}{b}), \tag{1.2.5}$$

where E_n are dimensionless coefficients to be found as part of the solution along with the dimensionless wave numbers α_n.

Enforcing the bottom and top wall slip conditions for $u_x^{\rm h}(y,z)$,

$$u_x^{\rm h}(z=\pm b) = \mp \ell \left(\frac{\partial u_x^{\rm h}}{\partial z} \right)_{z=\pm b}, \tag{1.2.6}$$

we find that

$$\cot \alpha_n = \lambda \, \alpha_n, \tag{1.2.7}$$

where $\lambda = \ell/b$. The roots of this nonlinear equation for α_n lie at the intersection of the graph of the cotangent function, $\cot \alpha_n$, and a straight line described by $\lambda \alpha_n$, as shown in Figure 1.2.1 for $\lambda = 0.2$.

1.2.3 *Root location*

When $\lambda = 0$, the roots of (1.2.7) are given in (1.1.16),

$$\alpha_n = \alpha_n^0 \equiv (n - \tfrac{1}{2})\pi. \tag{1.2.8}$$

As $\lambda \to \infty$, the roots shift to lower values, $\alpha_n \to (n-1)\pi$.

When λ is small, the left-hand side of (1.2.7) can be expanded in a Taylor series about α_n^0 and obtain

$$\cot \alpha_n^0 - \frac{1}{\sin^2 \alpha_n^0} \, (\alpha_n - \alpha_n^0) \simeq \lambda \, \alpha_n^0, \tag{1.2.9}$$

yielding the linear approximation

$$\frac{\alpha_n}{\alpha_n^0} \simeq 1 - \lambda, \tag{1.2.10}$$

which confirms that the difference $\alpha_n - \alpha_n^0$ is of order λ.

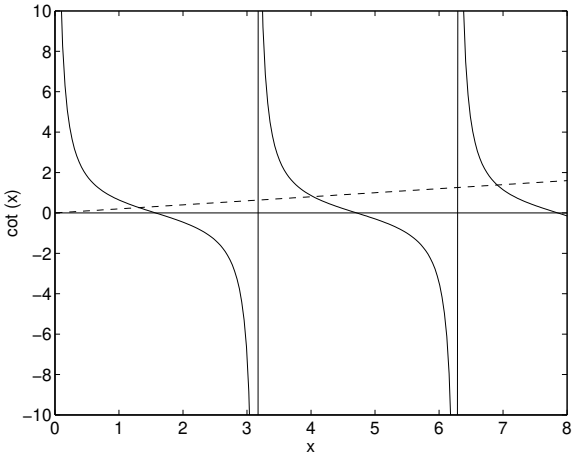


FIGURE 1.2.1 The roots of equation (1.2.7) lie at the intersection of the cotangent function, $\cot \alpha_n$, and a straight line described by λx. The dashed straight line shown in the graph corresponds to $\lambda = 0.2$.

1.2.4 Newton–Raphson method

The roots of equation (1.2.7) for arbitrary values of λ can be found by the Newton–Raphson method discussed in Section 6.5. The relevant code, not listed in the text, is located in directory TUBE_REC_SLIP of SEMLIB. The necessary evaluation function is listed below:

```
function f = newton_fun(x)
 global slipl
 f =  cot(x) - slipl * x;
return
```

Note that the slip parameter λ is passed to this function by way of a global statement, which is anathema to many.

Graphs of the first seven values of α_n scaled by $\alpha_n^0 = (n - \frac{1}{2})\,\pi$ against the dimensionless slip length $\lambda \equiv \ell/b$ are shown in Figure 1.2.2. All values of α_n decrease monotonically with increasing λ. As λ tends to infinity, the ratio α_n/α_n^0 tends to the limit of $2(n-1)/(2n-1)$.

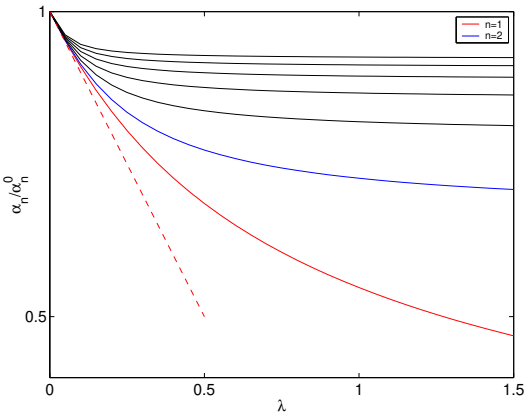


FIGURE 1.2.2 Graphs of the first seven roots of equation (1.2.7), α_1 (lowest solid line printed in red), ..., α_7 (highest solid line), scaled by $\alpha_n^0 = (n - \frac{1}{2})\,\pi$, plotted against the dimensionless slip length λ. The slope of the red broken line is -1.

The result for small values of λ shown in Figure 1.2.2 are consistent with the asymptotic estimate (1.2.10), represented by the red broken line in Figure 1.2.2. For low values of n, the approximation is accurate approximately up to $\lambda \simeq 0.1$. For $n = 1$, the approximation is accurate approximately up to $\lambda \simeq 0.2$.

1.2.5 Computation of the series expansion coefficients

The left and right wall slip condition requires that

$$\left(u_x^{\mathrm{h}} + u_x^{\mathrm{p}}\right)_{y=\pm a} = \mp\,\lambda\, b\,\Big(\,\frac{\partial(u_x^{\mathrm{h}} + u_x^{p})}{\partial y}\,\Big)_{y=\pm a}. \tag{1.2.11}$$

Substituting the particular solution given in (1.2.4) and the Fourier expansion for the homogeneous solution given in (1.2.5), we obtain

$$\sum_{n=1}^{\infty} E_n \left(\,\cosh(\alpha_n\,\frac{a}{b}) + \lambda\,\alpha_n\,\sinh(\alpha_n\,\frac{a}{b})\,\right)\,\cos(\alpha_n\frac{z}{b})$$
$$= -\tfrac{1}{2}\,(1 + 2\,\lambda - \frac{z^2}{b^2}). \tag{1.2.12}$$

In fact, the left-hand side is a Fourier-series like expansion of the right-hand side.

Unfortunately, the coefficients E_n may no longer be computed individually by Fourier projection. Instead, they must be found by solving a truncated system of linear equations that arise either by point collocation or by Galerkin projection.

An important drawback of these methods is that the accuracy of the computed coefficients $E_1, E_2, \ldots$, depends on how many terms in the sum are retained.

1.2.6 Collocation

Collocation involves enforcing equation (1.2.12) at a chosen number of M nodes, $z = z_i$ for $i = 1, \ldots M$, to generate a system of M linear equations. The highest accuracy is obtained by locating the nodes at the scaled roots of an Mth-degree orthogonal polynomial.

1.2.7 Galerkin projection

To illustrate the implementation of Galerkin's method, we retain only one term in the sum on the left-hand side of (1.2.12), multiply the simplified equation by $\cos(\alpha_1 \frac{z}{b})$, and integrate with respect to z to obtain the Galerkin equation

$$E_1 \left(\cosh(\alpha_1 \frac{a}{b}) + \lambda\, \alpha_1\, \sinh(\alpha_1 \frac{a}{b}) \right) \int_0^b \cos^2(\alpha_1 \frac{z}{b})\, \mathrm{d}z$$
$$= -\tfrac{1}{2} \int_0^b (1 + 2\,\lambda - \frac{z^2}{b^2})\, \cos(\alpha_1 \frac{z}{b})\, \mathrm{d}z, \qquad (1.2.13)$$

where α_1 is an implicit function of λ. Solving for E_1, we obtain

$$E_1 = -\frac{16}{\pi^3} \frac{\Phi(\lambda)}{\cosh(\alpha_1 \frac{a}{b}) + \lambda\, \alpha_1\, \sinh(\alpha_1 \frac{a}{b})}, \qquad (1.2.14)$$

where

$$\Phi(\lambda) \equiv \frac{\pi^3}{32} \frac{\int_0^1 (1 + 2\,\lambda - w^2)\, \cos(\alpha_1 w)\, \mathrm{d}w}{\int_0^1 \cos^2(\alpha_1 w)\, \mathrm{d}w}, \qquad (1.2.15)$$

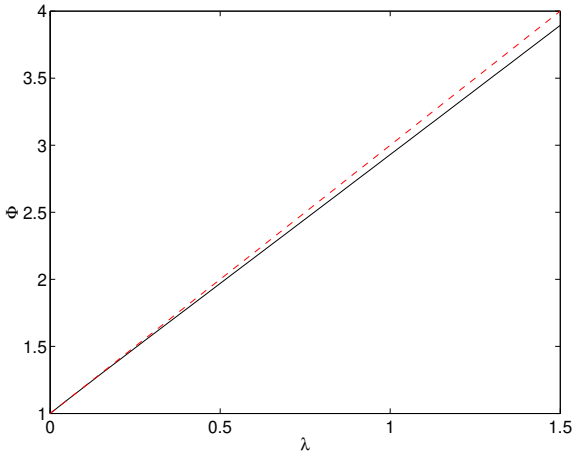


FIGURE 1.2.3 Dependence of the coefficient Φ on the scaled slip length λ for flow in a rectangular tube. The broken red line describes results obtained by the asymptotic estimate (1.2.30) for small λ.

and we have defined $w \equiv z/b$. A graph of Φ against λ is shown in Figure 1.2.3. Consistent with (1.1.22), as λ tends to zero, corresponding to no slip, Φ tends to unity, $\Phi(0) = 1$.

1.2.8 Velocity profile

The velocity profile subject to the one-term truncation is given by

$$u_x(y,z) \simeq \tfrac{1}{2}\,\frac{\chi+\rho g_x}{\mu}\,b^2\left(1+2\lambda-\frac{z^2}{b^2}\right. \\ \left.-\frac{32}{\pi^3}\,\Phi(\lambda)\,\frac{\cosh(\alpha_1\frac{y}{b})}{\cosh(\alpha_1\frac{a}{b})+\lambda\,\alpha_1\,\sinh(\alpha_1\frac{a}{b})}\,\cos(\alpha_1\frac{z}{b})\right), \tag{1.2.16}$$

where α_1 is an implicit function of λ. When $\lambda = 0$, we recover expression (1.4.38) with a single expansion term.

A typical velocity distribution over the tube cross-section is shown in Figure 1.2.4. Note that the slip velocity has low values at the four corners where the shear stress is small.

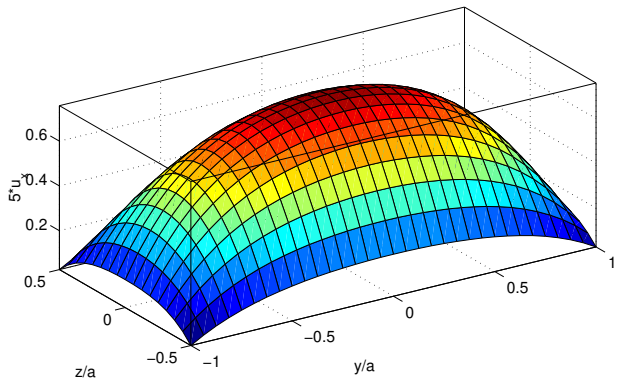


FIGURE 1.2.4 Velocity distribution over the cross-section of a rectangular tube with aspect ratio $a/b = 2$ for scaled slip coefficient $\lambda = 0.2$.

1.2.9 Flow rate

The flow rate through the tube arises by integrating the velocity distribution over the tube cross-section in the yz plane,

$$Q \equiv \int_{-a}^{a} \int_{-b}^{b} u_x(y,z)\, \mathrm{d}y\, \mathrm{d}z. \tag{1.2.17}$$

Substituting expression (1.2.16) and carrying out the integration, we obtain

$$Q = \tfrac{4}{3}\, \frac{\chi + \rho g_x}{\mu}\, ab^3\, F, \tag{1.2.18}$$

where

$$F = 1 + 3\lambda - \frac{48}{\pi^3}\, \Phi(\lambda)\, \frac{b}{a}\, \frac{1}{\alpha_1^2}\, \frac{\sinh(\alpha_1 \frac{a}{b}) \sin \alpha_1}{\cosh(\alpha_1 \frac{a}{b}) + \lambda\, \alpha_1\, \sinh(\alpha_1 \frac{a}{b})}. \tag{1.2.19}$$

We recall that α_1 is an implicit function of λ. When $\lambda = 0$, we recover expression (1.1.27) with one term in the sum.

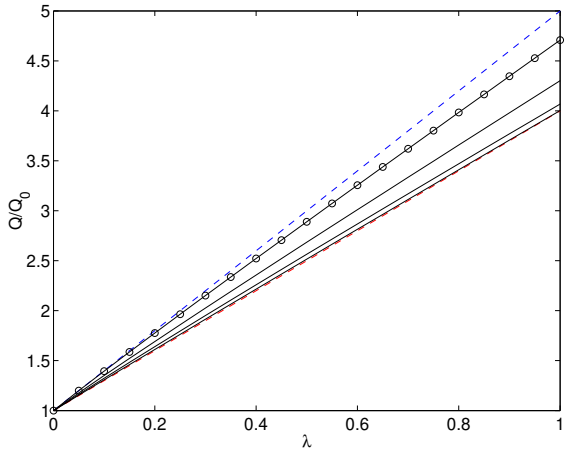


FIGURE 1.2.5 Ratio of the flow rate, Q, to that corresponding to the no-slip condition, Q_0, for $b/a = 1$ (circular symbols), 0.75, 0.50, and 0.20. The blue dashed line corresponds to a circle of radius a and the red dashed line corresponds to a two-dimensional channel with width $2b$.

A graph of the flow rate, Q, scaled by that when the no-slip boundary condition applies, Q_0, is shown in Figure 1.2.5. The blue dashed line corresponds to a circular tube of radius a where

$$\frac{Q}{Q_0} = 1 + 4\lambda. \tag{1.2.20}$$

The red dashed line corresponds to a two-dimensional channel of width $2b$ where

$$\frac{Q}{Q_0} = 1 + 3\lambda. \tag{1.2.21}$$

The perfect linear dependence on λ is due to the constant value of shear stress over the circular tube and two-dimensional channel. The flow rate for the rectangular tube lies between the two straight lines for the sphere and the two-dimensional channel.

1.2.10 Small slip

For small values of the scaled slip length, λ, we may use (1.2.10) to write $\alpha_n = \alpha_n^0(1-\lambda)$, and obtain as an approximation

$$\cos(\alpha_n \frac{z}{b}) \simeq \cos(\alpha_n^0(1-\lambda)\frac{z}{b}), \tag{1.2.22}$$

and then

$$\cos(\alpha_n \frac{z}{b}) \simeq \cos(\alpha_n^0 \frac{z}{b}) + \lambda\, \alpha_n^0\, \frac{z}{b}\, \sin(\alpha_n^0 \frac{z}{b}), \tag{1.2.23}$$

where we recall that $-b \leq z \leq b$.

Similar approximations can be written for the hyperbolic sine and cosine. Using (1.2.10) to write $\alpha_1 = \frac{\pi}{2}(1-\lambda)$, we obtain the approximation

$$\cosh(\alpha_1 \frac{y}{b}) \simeq \cosh(\tfrac{\pi}{2}(1-\lambda)\frac{y}{b}), \tag{1.2.24}$$

and then

$$\cosh(\alpha_1 \frac{y}{b}) \simeq \cosh(\tfrac{\pi}{2}\frac{y}{b}) - \lambda\, \tfrac{\pi}{2}\, \frac{y}{b}\, \sinh(\tfrac{\pi}{2}\frac{y}{b}), \tag{1.2.25}$$

and also

$$\cosh(\alpha_1 \frac{a}{b}) \simeq \cosh(\tfrac{\pi}{2}\frac{a}{b}) - \lambda\, \tfrac{\pi}{2}\, \frac{a}{b}\, \sinh(\tfrac{\pi}{2}\frac{a}{b}). \tag{1.2.26}$$

Substituting into (1.2.15)

$$\cos(\alpha_1 w) \simeq \cos(\tfrac{\pi}{2}\, w) + \lambda\, \tfrac{\pi}{2}\, w \sin(\tfrac{\pi}{2} w), \tag{1.2.27}$$

we obtain

$$\Phi(\lambda) \simeq \frac{\pi^3}{32}\, \frac{\mathcal{A} + \lambda \mathcal{B}}{\mathcal{C} + \lambda\, \mathcal{D}} \simeq \frac{\pi^3}{32}\, \frac{1}{\mathcal{C}}\, \big(\, \mathcal{A} + \lambda\, (\mathcal{B} - \frac{\mathcal{A}\mathcal{D}}{\mathcal{C}})\, \big), \tag{1.2.28}$$

where

$$\begin{aligned}
\mathcal{A} &= \int_0^1 (1-w^2)\,\cos(\tfrac{\pi}{2}w)\,\mathrm{d}w = \frac{16}{\pi^3},\\
\mathcal{B} &= 2\int_0^1 \cos(\tfrac{\pi}{2}w)\,\mathrm{d}w + \tfrac{\pi}{2}\int_0^1 (1-w^2)\,w\,\sin(\tfrac{\pi}{2}w)\,\mathrm{d}w = \frac{48}{\pi^3},\\
\mathcal{C} &= \int_0^1 \cos^2(\tfrac{\pi}{2}w)\,\mathrm{d}w = \tfrac{1}{2},\\
\mathcal{D} &= \tfrac{\pi}{2}\int_0^1 w\,\sin(\pi w)\,\mathrm{d}w = \tfrac{1}{2}.
\end{aligned} \tag{1.2.29}$$

Substituting the values for $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{D}$, we obtain

$$\Phi(\lambda) \simeq \frac{1+3\lambda}{1+\lambda} \simeq 1+2\lambda. \tag{1.2.30}$$

The last (linear) expression is represented by the broken line in Figure 1.2.3. The linear approximation for $\Phi(\lambda)$ is remarkably accurate even for high values of λ.

Making substitutions into (1.2.16), we obtain an approximation for the velocity field,

$$u_x(y,z) \simeq \tfrac{1}{2}\,\frac{\chi+\rho g_x}{\mu}\,b^2\,\big(1+2\lambda-\frac{z^2}{b^2}-\frac{32}{\pi^3}\,(1+2\lambda)\,\Phi\,\big), \tag{1.2.31}$$

accurate to first order in λ, where

$$\Phi = H(y)\,\big(\cos(\tfrac{\pi}{2}\frac{z}{b}) + \lambda\tfrac{\pi}{2}\frac{z}{b}\sin(\tfrac{\pi}{2}\frac{z}{b})\,\big). \tag{1.2.32}$$

where

$$H(y) = \frac{\cosh(\frac{\pi}{2}\frac{y}{b}) - \lambda\,\frac{\pi}{2}\,\frac{y}{b}\,\sinh(\frac{\pi}{2}\frac{y}{b})}{\cosh(\frac{\pi}{2}\frac{a}{b}) + \lambda\,\frac{\pi}{2}\,(1-\frac{a}{b})\,\sinh(\frac{\pi}{2}\frac{a}{b})}. \tag{1.2.33}$$

Making substitutions into (1.2.19), we obtain an approximation for the function F defining the flow rate,

$$F \simeq 1+3\lambda - \frac{192}{\pi^5}\,\frac{1+2\lambda}{1-2\lambda}\,\frac{b}{a}\,\frac{\sinh(\frac{\pi}{2}\frac{a}{b}) + \lambda\,\frac{\pi}{2}\,\cosh(\frac{\pi}{2}\frac{a}{b})}{\cosh(\frac{\pi}{2}\frac{a}{b}) + \lambda\,\frac{\pi}{2}\,(1-\frac{a}{b})\,\sinh(\frac{\pi}{2}\frac{a}{b})}. \tag{1.2.34}$$

1.2.11 General implementation of the Galerkin method

We have carried out the one-term Fourier solution. More generally, we retain N terms on the left-hand side of (1.2.12) and repeat the procedure to obtain a system of linear equations

$$A_{mn}\,\mathcal{E}_n = b_m, \tag{1.2.35}$$

where summation is implied over the repeated index, n,

$$\mathcal{E}_n \equiv E_n \left(\cosh(\alpha_n \frac{a}{b}) + \lambda\,\alpha_n\,\sinh(\alpha_n \frac{a}{b}) \right) \tag{1.2.36}$$

are the unknowns. The coefficient matrix and right-hand side are given by

$$\begin{aligned} A_{mn} &= \int_0^1 \cos(\alpha_m w)\,\cos(\alpha_n w)\,\mathrm{d}w, \\ b_m &= -\tfrac{1}{2}\int_0^1 (1 + 2\,\lambda - w^2)\,\cos(\alpha_m w)\,\mathrm{d}w, \end{aligned} \tag{1.2.37}$$

where $w \equiv z/b$.

Exercises

1.2.1 Confirm that the velocity distribution shown in (1.2.4) describes the velocity profiles in flow between two plates separated by distance $2b$.

1.2.1 Derive expressions for the velocity profile and flow rate of the flow discussed in Exercise 4.1.3 in the presence of slip.

1.3 Flow in a two-dimensional brush

Consider flow between two semi-infinite parallel plates that slide with velocity V against a stationary flat surface, as illustrated in Figure 1.3.1. The motion of the plates generates a unidirectional flow along the x axis, which can be regarded as a model of the flow occurring

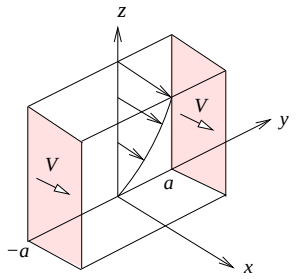


FIGURE 1.3.1 Illustration of steady unidirectional flow between two parallel plates sliding over a plane wall regarded as a painted surface.

between two hairs of an idealized liquid-loaded two-dimensional paint brush moving over a flat surface. Far from the surface, the fluid translates along the x axis with the plate velocity, V.

In the absence of pressure gradient and significant gravitational effects, the x component of the velocity parallel to the brush satisfies Laplace's equation,

$$\frac{\partial^2 u_x}{\partial y^2} + \frac{\partial^2 u_x}{\partial z^2} = 0. \tag{1.3.1}$$

The boundary conditions specify that $u_x \to V$ as $z \to \infty$ and u_x at $y = \pm a$.

1.3.1 *Fourier expansion*

Working as in Section 1.1 for flow in a rectangular tube in the absence of a streamwise pressure gradient, we express the solution in the deliberate form

$$u_x(y,z) = V\left(1 + \sum_{n=1}^{\infty} E_n \exp\left(-\alpha_n \frac{z}{a}\right) \cos(\alpha_n \frac{y}{a}\right), \tag{1.3.2}$$

where E_n are *a priori* unknown Fourier coefficients and α_n are *a priori* unknown positive wave numbers. The choice of the exponential term is motivated by the expected decay of the velocity toward the far-field value, V.

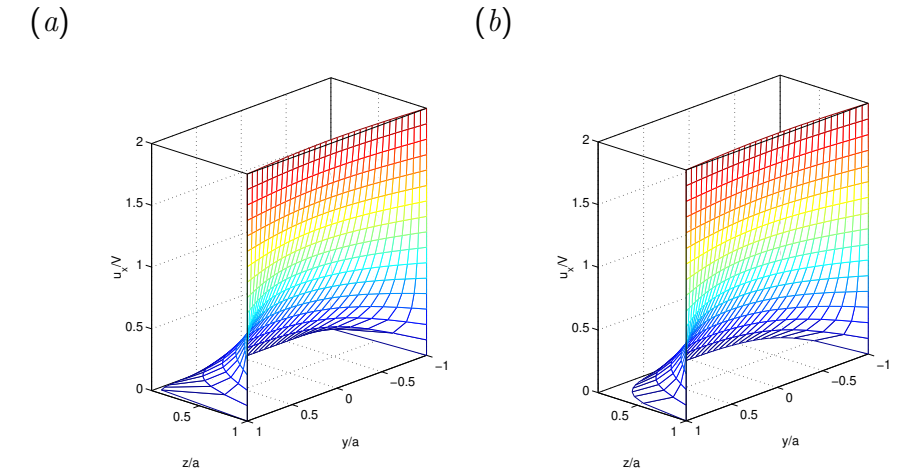


FIGURE 1.3.2 Velocity profile computed from the Fourier series solution subject to (a) the no-slip boundary condition and (b) the slip boundary condition at the wall with $\lambda_w = 0.1$.

Enforcing the no-slip boundary condition at the surface of the plates, $u_x = V$ at $y = \pm a$, we find that $\alpha_n = (n - \frac{1}{2})\pi$.

Enforcing the no-slip boundary condition at the wall, $u_x = 0$ at $z = 0$, and using the Fourier orthogonality property

$$\int_{-a}^{a} \cos(\alpha_n \frac{y}{a}) \cos(\alpha_m \frac{y}{a})\, \mathrm{d}y = \begin{cases} a & \text{if } n = m, \\ 0 & \text{if } n \neq m, \end{cases} \tag{1.3.3}$$

we derive the velocity profile

$$u_x(y,z) = V \left(1 + 2 \sum_{n=1}^{\infty} (-1)^n \frac{1}{\alpha_n} \exp(-\alpha_n \frac{z}{a}) \cos(\alpha_n \frac{y}{a}) \right). \tag{1.3.4}$$

The boundary conditions $u_x = V$ at $y = \pm a$ and $u_x = 0$ at $z = 0$ are satisfied. The velocity distribution computed with 64 terms in the sum is shown in Figure 1.3.2(a).

1.3.2 Amount of deposited liquid

The amount of fluid deposited on the wall is expressed by the flow rate

$$Q = \int_0^\infty \int_{-a}^{a} (V - u_x)\, \mathrm{d}y\, \mathrm{d}z. \tag{1.3.5}$$

Substituting the velocity profile and carrying out the integration, we find that

$$Q = 4Va^2 \sum_{n=1}^{\infty} \frac{1}{\alpha_n^3} \simeq 1.085\, Va^2. \tag{1.3.6}$$

Nearly a liquid layer of thickness half the distance between the plates is deposited on the wall. When only one term in the sum is retained, the coefficient 1.085 is reduced only moderately to 1.032.

1.3.3 Plate and wall slip

In the case of slippery plates, we find that the wave numbers α_n satisfy the equation

$$\cot \alpha_n = \lambda_p\, \alpha_n, \tag{1.3.7}$$

where $\lambda_p = \ell_p/a$ and ℓ_p is the slip length of the plates. The roots of this equation were discussed in Section 1.2.

The slip boundary condition at the wall requires that

$$u_x(z = 0) = \ell_w \left(\frac{\partial u_x}{\partial z}\right)_{z=0}, \tag{1.3.8}$$

where ℓ_p is the slip length of the wall. Substituting the Fourier series solution (1.3.2), we obtain

$$\sum_{n=1}^{\infty} E_n\, (1 + \lambda_w \alpha_n)\, \cos(\alpha_n \frac{y}{a}) = -1, \tag{1.3.9}$$

where $\lambda_w = \ell_w/a$.

1.3.4 No-slip plates

In the case of zero plate slip, we set $\alpha_n = (n - \frac{1}{2})\,\pi$ and obtain the velocity profile

$$u_x(y,z) = V\,\Big(1 + 2\sum_{n=1}^{\infty}(-1)^n \frac{1}{1+\lambda_w\alpha_n}\,\frac{1}{\alpha_n}\exp(-\alpha_n\frac{z}{a})\,\cos(\alpha_n\frac{y}{a})\Big). \tag{1.3.10}$$

The velocity distribution for $\lambda_w = 0.1$ computed with 64 terms in the sum is shown in Figure 1.3.2(b).

1.3.5 Slippery plates

To compute the coefficients E_n in the case of plate slip, we may work as discussed in Section 1.2. Keeping one term in the sum and performing the Galerkin projection, we obtain

$$E_1 = -\frac{1}{1+\lambda_w\alpha_n}\,\frac{\int_{-a}^{a}\cos(\alpha_1\frac{y}{a})\mathrm{d}\,y}{\int_{-a}^{a}\cos^2(\alpha_1\frac{y}{a})\mathrm{d}\,y}. \tag{1.3.11}$$

Rearranging, we obtain

$$E_1 = -\frac{4}{\pi}\,\frac{1}{1+\lambda_w\alpha_1}\,\Phi(\lambda_p), \tag{1.3.12}$$

where

$$\Phi(\lambda_p) = \frac{\pi}{4}\,\frac{\int_0^1\cos(\alpha_1 w)\,\mathrm{d}w}{\int_0^1\cos^2(\alpha_1 w)\,\mathrm{d}w}, \tag{1.3.13}$$

we have defined $w \equiv y/a$, and we recall that α_1 is an implicit function of λ_p. In the absence of plate slip, $\lambda_p = 0$, we find that $\alpha_1 = \frac{1}{2}\pi$ and $\Phi(0) = 1$.

A graph of Φ against λ_p is shown in Figure 1.3.3. The results confirm that, as λ_p tends to zero, corresponding to no slip, Φ tends to unity, $\Phi(0) = 1$. A local quadratic behavior prevails near $\lambda_p = 0$.

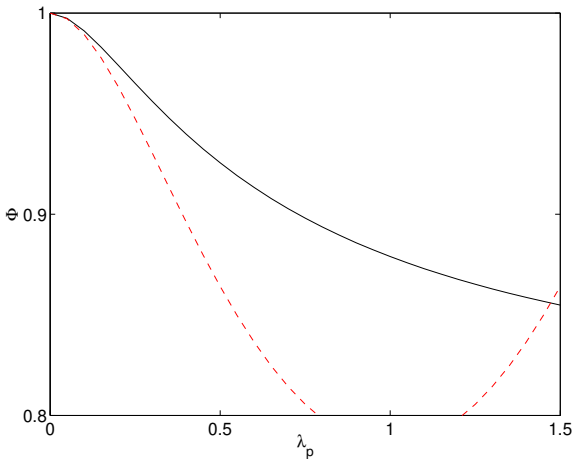


FIGURE 1.3.3 Dependence of the coefficient Φ on the scaled slip length λ for flow in a two-dimensional brush (solid line). The broken line represents results obtained by using the linear approximation $\alpha_1 \simeq \frac{\pi}{2}(1-\lambda_p)$.

1.3.6 *Velocity profile*

The velocity profile subject to the one-term truncation is given by

$$u_x(y,z) = V\left(1 - \frac{4}{\pi}\frac{1}{1+\lambda_w\,\alpha_1}\,\Phi(\lambda_p)\exp(-\alpha_1\frac{z}{a})\,\cos(\alpha_1\frac{y}{a})\right), \tag{1.3.14}$$

where α_1 is an implicit function of λ_w. When $\lambda_p = 0$ and $\lambda_w = 0$ we recover expression (1.3.6) with a single expansion term, $n = 1$.

The amount of liquid deposited on the wall is quantified by the flow rate

$$Q = \frac{8}{\pi}\,Va^2\frac{1}{1+\lambda_w\,\alpha_1}\,\frac{1}{\alpha_1^2}\,\Phi(\lambda_p), \tag{1.3.15}$$

where α_1 is an implicit function of λ_w. Increasing λ_p or λ_w decreases the amount of deposited liquid.

Exercise

1.3.1 Derive formula (1.3.15).

1.4 Transient flow through a tube

Consider transient unidirectional flow through a cylindrical tube of radius a due to the sudden application of a pressure gradient or tilting. The motion of the fluid will be described in cylindrical polar coordinates, (x, σ, φ), where the x axis coincides with the tube centerline.

The radial and azimuthal velocity components are identically zero, $u_\sigma = 0$ and $u_\varphi = 0$, while the axial component is a function of distance from the x axis and time, $u_x(\sigma, t)$. At the initial instant, the fluid is assumed to be quiescent, $u_x(\sigma, t = 0) = 0$.

1.4.1 Wall slip

The fluid may slip over the tube wall so that the wall velocity is proportional to the wall shear stress or wall shear rate,

$$u_x = -\ell \left(\frac{\partial u_x}{\partial \sigma} \right)_{\sigma=a}, \tag{1.4.1}$$

where ℓ is a specified slip length. The minus sign is motivated by the expected negative slope of the velocity profile at the wall. The no-slip boundary condition arises as $\ell \to 0$ and the zero stress boundary condition arises as $\ell \to \infty$.

1.4.2 Wall sleeve

To interpret the physics of the wall slip condition, we may assume that the wall is lined with a thin sleeve that slides over the wall. Balancing the shear stress is acting on the upper side of the sleeve with the frictional force, we obtain

$$\mu \frac{\partial u_x}{\partial \sigma} + \eta\, u_x = 0, \tag{1.4.2}$$

where μ is the fluid viscosity and η is a frictional coefficient. Rearranging, we obtain (1.4.1) with $\ell = \mu/\eta$.

A generalization of (1.4.2) accounts for the effect of sleeve inertia,

$$\varrho_s \frac{\partial u_x}{\partial t} = \mu \frac{\partial u_x}{\partial \sigma} + \eta\, u_x = 0, \tag{1.4.3}$$

where ϱ_s is an effective linear density. This effect is not considered in our analysis.

1.4.3 Governing equations

The pressure field is given by

$$p = -\chi x + \rho\,(g_y y + g_z z) + \pi_0, \tag{1.4.4}$$

with a constant axial pressure gradient, $\partial p/\partial x = -\chi$. Simplifying the x component of the equation of motion written in cylindrical polar coordinates, (x, σ, φ), we obtain a partial differential equation for the axial velocity,

$$\rho \frac{\partial u_x}{\partial t} = \chi + \mu \left(\frac{\partial^2 u_x}{\partial \sigma^2} + \frac{1}{\sigma} \frac{\partial u_x}{\partial \sigma} \right) + \rho g_x. \tag{1.4.5}$$

The fluid is assumed to be quiescent at the initial instant.

1.4.4 Steady state

When the flow reaches a steady state, the velocity profile is governed by the ordinary differential equation

$$\frac{1}{\sigma} \frac{\mathrm{d}}{\mathrm{d}\sigma} \Big(\sigma \frac{\mathrm{d}u_x}{\mathrm{d}\sigma} \Big) = \frac{\mathrm{d}^2 u_x}{\mathrm{d}\sigma^2} + \frac{1}{\sigma} \frac{\mathrm{d}u_x}{\mathrm{d}\sigma} = -\frac{\chi + \rho g_x}{\mu}. \tag{1.4.6}$$

Solving this equation subject to the slip boundary condition, we obtain a parabolic distribution described by a modified Poiseuille-flow solution,

$$u_x^{\text{steady}}(\sigma) = \tfrac{1}{4} \frac{\chi + \rho g_x}{\mu} \left(a^2 + 2\, a\ell - \sigma^2 \right). \tag{1.4.7}$$

We note that the wall shear stress at steady state is independent of the slip length, ℓ, so that the shear force balances the force exerted on a cylindrical body of fluid due to gravity or an imposed pressure gradient.

1.4.5 Separation of variables

To compute the solution of (1.4.5), we resolve the velocity profile into the steady Poiseuille profile and a transient profile that decays at long times, $v_x(\sigma, t)$,

$$u_x(\sigma, t) = u_x^{\text{steady}}(\sigma) + v_x(\sigma, t). \tag{1.4.8}$$

Substituting (1.4.8) into (1.4.5), we find that the transient profile satisfies a homogeneous partial differential equation,

$$\frac{\partial v_x}{\partial t} = \nu \left(\frac{\partial^2 v_x}{\partial \sigma^2} + \frac{1}{\sigma} \frac{\partial v_x}{\partial \sigma} \right), \tag{1.4.9}$$

where $\nu \equiv \mu/\rho$ is the kinematic viscosity of the fluid.

Equation (1.4.9) can be solved by the method of separation of variables implemented by the expansion

$$v_x(\sigma, t) = \sum_{n=1}^{\infty} c_n \, \phi_n(\sigma) \, \psi_n(t), \tag{1.4.10}$$

where $\phi_n(\sigma)$ and $\psi_n(t)$ are *a priori* unknown functions of their respective arguments, and c_n are constant coefficients. Substituting this expansion into (1.4.9), carrying out the differentiations, and rearranging, we obtain

$$\sum_{n=1}^{\infty} c_n \, \phi_n(\sigma) \, \psi_n(t) \left(\frac{1}{\psi_n} \frac{\mathrm{d}\psi_n}{\mathrm{d}t} - \frac{\nu}{\phi_n} \left(\frac{\mathrm{d}^2 \phi_n}{\mathrm{d}\sigma^2} + \frac{1}{\sigma} \frac{\mathrm{d}\phi_n}{\mathrm{d}\sigma} \right) \right) = 0. \tag{1.4.11}$$

For this equality to hold true at any radial position, σ, and any time, t, the expression enclosed by the tall parentheses on the right-hand side must be identically zero, yielding

$$\frac{1}{\psi_n} \frac{\mathrm{d}\psi_n}{\mathrm{d}t} = \frac{\nu}{\phi_n} \left(\frac{\mathrm{d}^2 \phi_n}{\mathrm{d}\sigma^2} + \frac{1}{\sigma} \frac{\mathrm{d}\phi_n}{\mathrm{d}\sigma} \right) = -\frac{\nu}{a^2} \alpha_n^2, \tag{1.4.12}$$

where α_n^2 is a positive dimensionless constant to be determined as part of the solution. The right-hand side of (1.4.12) was designed to facilitate forthcoming algebraic manipulations.

Rearranging the two equalities in (1.4.12), we derive two ordinary differential equations,

$$\frac{\mathrm{d}\psi_n}{\mathrm{d}t} = -\frac{\nu}{a^2}\,\alpha_n^2\,\psi_n \tag{1.4.13}$$

and

$$\frac{\mathrm{d}^2\phi_n}{\mathrm{d}\sigma^2} + \frac{1}{\sigma}\frac{\mathrm{d}\phi_n}{\mathrm{d}\sigma} + \frac{1}{a^2}\,\alpha_n^2\phi_n = 0. \tag{1.4.14}$$

The solution of (1.4.13) is found readily to be

$$\psi_n(t) = \exp(-\alpha_n^2\,\frac{\nu t}{a^2}), \tag{1.4.15}$$

which reveals that the transient flow decays exponentially in time.

1.4.6 Bessel functions

To derive the solution of (1.4.14), we introduce the Bessel function of the first kind, $\mathrm{J}_0(z)$, satisfying the zeroth-order Bessel equation

$$z\,\frac{\mathrm{d}}{\mathrm{d}z}\Big(z\,\frac{\mathrm{d}\mathrm{J}_0(z)}{\mathrm{d}z}\Big) + z^2\,\mathrm{J}_0(z) = 0, \tag{1.4.16}$$

which may also be written as

$$z^2\,\frac{\mathrm{d}^2\mathrm{J}_0(z)}{\mathrm{d}z^2} + z\,\frac{\mathrm{d}\mathrm{J}_0(z)}{\mathrm{d}z} + z^2\,\mathrm{J}_0(z) = 0. \tag{1.4.17}$$

A graph of $\mathrm{J}_0(z)$ is shown with the solid line in Figure 1.4.1.

Replacing z in equation (1.4.17) with $\alpha_n\sigma/a$ and simplifying, we find that the function

$$\phi_n(\sigma) = \mathrm{J}_0(\alpha_n\frac{\sigma}{a}) \tag{1.4.18}$$

satisfies equation (1.4.14), and is thus the desired solution.

To satisfy the wall slip boundary condition, we require that

$$\phi_n(a) = -\ell\,\Big(\,\frac{\partial\phi_n}{\partial\sigma}\,\Big)_{\sigma=a}. \tag{1.4.19}$$

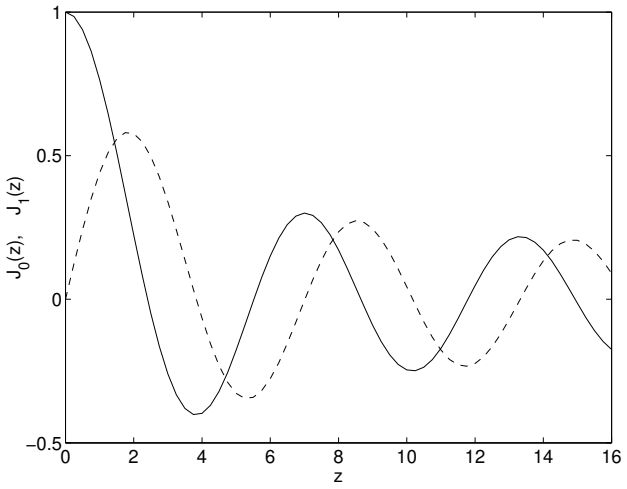


FIGURE 1.4.1 Graphs of the Bessel functions $\mathrm{J}_0(z)$ and $\mathrm{J}_1(z)$ drawn with the solid or broken line, respectively, necessary for the computation of transient tube flow.

Substituting the derived expression for ϕ_n, we find

$$\mathrm{J}_0(\alpha_n) = -\ell\,[\mathrm{J}_0'(\alpha_n \frac{\sigma}{a})]_{\sigma=a} = \lambda\,\alpha_n\,\mathrm{J}_1(\alpha_n), \tag{1.4.20}$$

$\lambda \equiv \ell/a$, $\mathrm{J}_1(z)$ is a Bessel function of the first kind.x, and we have noted that

$$\mathrm{J}_0'(z) = -\mathrm{J}_1(z). \tag{1.4.21}$$

By definition, $\mathrm{J}_1(z)$ satisfies the differential equation

$$z^2\,\frac{\mathrm{d}^2\mathrm{J}_1(z)}{\mathrm{d}z^2} + z\,\frac{\mathrm{d}\mathrm{J}_1(z)}{\mathrm{d}z} + (z^2-1)\,\mathrm{J}_1(z) = 0, \tag{1.4.22}$$

and is required to be finite for any value of z between, and including, zero and infinity. A graph of $\mathrm{J}_1(z)$ is shown with the dashed line in Figure 1.4.1.

Equation (1.4.20) shows that the dimensionless numbers α_n are roots of the nonlinear equation

$$\mathrm{J}_0(\alpha_n) = \lambda\,\alpha_n\,\mathrm{J}_1(\alpha_n). \tag{1.4.23}$$

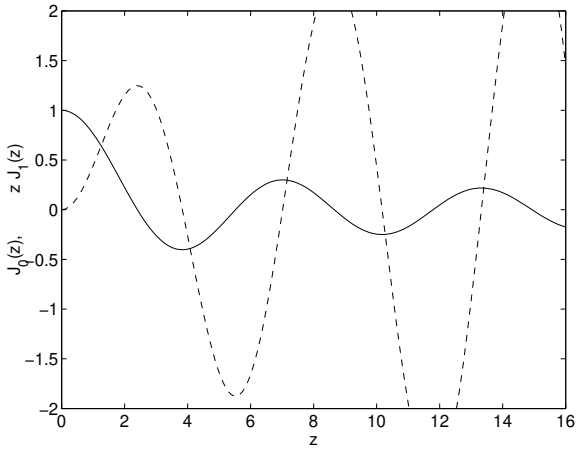


FIGURE 1.4.2 Graphs of the Bessel function $\mathrm{J}_0(z)$ and first moment $z\mathrm{J}_1(z)$, drawn with the solid or broken line, respectively, necessary for the computation of transient tube flow with wall slip.

Graphs of the functions $\mathrm{J}_0(z)$ and $z\mathrm{J}_1(z)$ are shown in Figure 1.4.2. The values of α_n lie at the intersection of the solid line and the broken line multiplied by $\lambda \equiv \ell/a$.

In the absence of wall slip, $\lambda = 0$, α_n is a zero of $\mathrm{J}_0(z)$. The first seven zeros of $\mathrm{J}_0(z)$ are

$$\alpha_1 = 2.4048, \quad \alpha_2 = 5.5201, \quad \alpha_3 = 8.6537, \quad \alpha_4 = 11.7915,$$
$$\alpha_5 = 14.9309, \quad \alpha_6 = 18.0711, \quad \alpha_7 = 21.2116, \qquad (1.4.24)$$

to shown accuracy. As $\lambda \to \infty$, the roots tend to the location of extrema, maxima or minima, of J_0, which are the zeros of J_1. The first six zeros of $\mathrm{J}_1(z)$ are

$$\alpha_1 = 0, \qquad \alpha_2 = 3.8317, \qquad \alpha_3 = 7.0156, \qquad \alpha_4 = 10.1735,$$
$$\alpha_5 = 13.3237, \qquad \alpha_6 = 16.4706, \qquad \alpha_7 = ..., \qquad (1.4.25)$$

to shown accuracy.

For nonzero values of λ, the roots of (1.4.23) can be computed by numerical methods, as discussed later in this section.

1.4.7 Newton–Raphson method

The roots of the nonlinear equation (1.4.23) can be found by Newton's method discussed in Section 8.1. The necessary function evaluation code, located in directory TUBE_TRANS of SEMLIB, is listed below:

```
function f = newton_fun(x)
 global slipl
 rj0 = besselj(0,x);
 rj1 = besselj(1,x);
 f = rj0 - slipl * x *rj1;
return
```

In this implementation, the Bessel functions are evaluated using an internal Matlab function.

The following Matlab code, located in directory TUBE_TRANS of SEMLIB, calls this function to compute the first seven roots with suitable initial guesses:

```
  global slipl

%---
% parameters
%---

  Niter = 10;
  eps = 0.00001;
  italk = 0;
  slipl = 0.10;

%---
% run
%---

  for n=1:7

  % initial guesses are the zeros of J0

  if(n==1) x =  2.4048; end
```

```
  if(n==2) x =  5.5201; end
  if(n==3) x =  8.6537; end
  if(n==4) x = 11.7915; end
  if(n==5) x = 14.9309; end
  if(n==6) x = 18.0711; end
  if(n==7) x = 21.2116; end

  [x,f,Iflag] = newton ...
     ...
    (Niter ...
    ,eps ...
    ,x ...
    ,italk ...
    );

   alpha(n) = x;

end

alpha'
```

Running the code generates the following output:

```
   2.1795
   5.0332
   7.9569
  10.9363
  13.9580
  17.0099
  20.0829
```

Graphs of the first seven values of α_n against the dimensionless slip length λ is shown in Figure 1.4.3. All values decrease monotonically with increasing slip length.

1.4.8 Bessel series representation

Now substituting $b_n = \alpha_n/a$ into (1.4.18) and (1.4.15), and the result into (1.4.10), we derive the expansion

$$v_x(\sigma,t) = \sum_{n=1}^{\infty} c_n \, \mathrm{J}_0(\alpha_n \, \frac{\sigma}{a}) \, \exp(-\alpha_n^2 \, \frac{\nu t}{a^2}). \qquad (1.4.26)$$

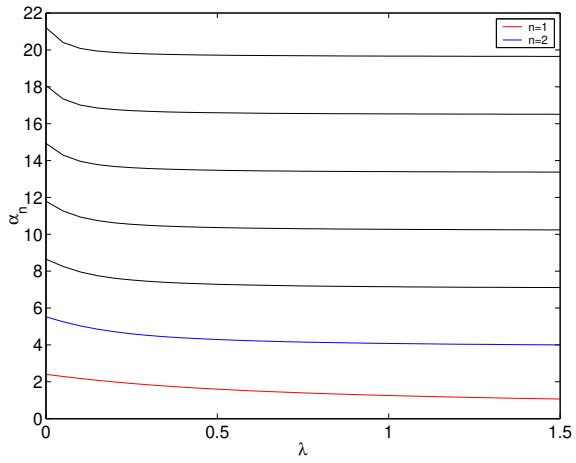


FIGURE 1.4.3 Graphs of the first seven roots, α_1 (lowest line), ..., α_7, plotted against the dimensionless slip length λ.

This representation clearly satisfies the requirement that the transient flow decays to zero at long times. The time scale of decay is determined by the first term in the series and is given by

$$\tau = \frac{1}{\alpha_1^2}\frac{a^2}{\nu} \tag{1.4.27}$$

Not surprisingly, as $\lambda \to \infty$, $\alpha_1 \to 0$ and the time scale tends to become infinite.

To also satisfy the initial condition, $u_x(\sigma, t = 0) = 0$, we evaluate the decomposition (1.4.8) at $t = 0$ and obtain

$$\sum_{n=1}^{\infty} c_n \, \mathrm{J}_0\left(\frac{\alpha_n \, \sigma}{a}\right) = -\tfrac{1}{4}\,\frac{\chi + \rho g_x}{\mu}\,(a^2 + 2a\ell - \sigma^2). \tag{1.4.28}$$

The coefficients c_n must be such that this representation holds true for any value of σ. In fact, the left-hand side is the Bessel series representation of the quadratic function on the right-hand side.

1.4.9 Orthogonality of Bessel functions

To extract the coefficients c_n from (1.4.28), we use the following orthogonality property of the Bessel functions,

$$\int_0^1 z\,\mathrm{J}_0(\alpha_n z)\,\mathrm{J}_0(\alpha_m z)\,\mathrm{d}z = \begin{cases} \frac{1}{2}\,\mathrm{J}_1^2(\alpha_n) & \text{if } n = m, \\ 0 & \text{if } n \neq m, \end{cases} \tag{1.4.29}$$

Replacing z in identity (1.4.29) with σ/a, and rearranging, we obtain the orthogonality property

$$\int_0^a \sigma\,\mathrm{J}_0(\alpha_n\,\frac{\sigma}{a})\,\mathrm{J}_0(\alpha_m\,\frac{\sigma}{a})\,\mathrm{d}\sigma = \begin{cases} \frac{1}{2}\,a^2\mathrm{J}_1^2(\alpha_n) & \text{if } n = m, \\ 0 & \text{if } n \neq m. \end{cases} \tag{1.4.30}$$

Inspired by this identity, we multiply both sides of (1.4.28) by $\sigma\,\mathrm{J}_0(\alpha_m\sigma/a)$, integrate with respect to σ from 0 to a, use the orthogonality property (1.4.30), and switch m to n to obtain

$$c_n = -\tfrac{1}{2}\,\frac{\chi+\rho g_x}{\mu a^2}\,\frac{1}{\mathrm{J}_1^2(\alpha_n)}\int_0^a \mathrm{J}_0(\frac{\alpha_n\,\sigma}{a})\,(a^2+2a\ell-\sigma^2)\,\sigma\,\mathrm{d}\sigma, \tag{1.4.31}$$

which can be rearranged into

$$c_n = -\tfrac{1}{2}\,\frac{\chi+\rho g_x}{\mu a^2}\,\frac{1}{\mathrm{J}_1^2(\alpha_n)}\int_0^1 \mathrm{J}_0(\alpha_n\,v)\,(1+2\,\frac{\ell}{a}-v^2)\,v\,\mathrm{d}v, \tag{1.4.32}$$

where $v \equiv \sigma/a$.

To compute the last integral, we replace z in Bessel's equation (1.4.17) with $\alpha_n v$, and simplify to obtain

$$\alpha_n^2\,v\,\mathrm{J}_0(\alpha_n\,v) = -\frac{\mathrm{d}}{\mathrm{d}v}\Big(v\,\frac{\mathrm{d}\mathrm{J}_0(\alpha_n v)}{\mathrm{d}v}\Big). \tag{1.4.33}$$

Note that the left- and thus the right-hand side of (1.4.33) vanishes when $v = 0$ or 1. Integrating (1.4.33) with respect to v from 0 to 1, we obtain

$$\alpha_n^2\int_0^1 \mathrm{J}_0(\alpha_n\,v)\,v\,\mathrm{d}v = -\Big[\,v\,\frac{\mathrm{d}\mathrm{J}_0(\alpha_n v)}{\mathrm{d}v})\,\Big]_{v=1} + \Big[\,v\,\frac{\mathrm{d}\mathrm{J}_0(\alpha_n v)}{\mathrm{d}v})\,\Big]_{v=0}. \tag{1.4.34}$$

Now using the aforementioned identity $\mathrm{d}\mathrm{J}_0(z)/\mathrm{d}z = -\mathrm{J}_1(z)$, we obtain

$$\int_0^1 \mathrm{J}_0(\alpha_n\, v)\, v\, \mathrm{d}v = \frac{1}{\alpha_n}\, \mathrm{J}_1(\alpha_n). \tag{1.4.35}$$

A similar computation yields

$$\begin{aligned}
\alpha_n^2 \int_0^1 \mathrm{J}_0(\alpha_n\, v)\, v^3\, \mathrm{d}v &= -\int_0^1 v^2\, \frac{\mathrm{d}}{\mathrm{d}v}(v\, \frac{\mathrm{d}\mathrm{J}_0(\alpha_n v)}{\mathrm{d}v})\, \mathrm{d}v \\
&= -(v^3\, \frac{\mathrm{d}\mathrm{J}_0(\alpha_n v)}{\mathrm{d}v})_{v=1} + (v^3\, \frac{\mathrm{d}\mathrm{J}_0(\alpha_n v)}{\mathrm{d}v})_{v=0} + 2\int_0^1 v^2\, \frac{\mathrm{d}\mathrm{J}_0(\alpha_n v)}{\mathrm{d}v}\, \mathrm{d}v \\
&= \alpha_n \mathrm{J}_1(\alpha_n) + 2\left[\, v^2\, \mathrm{J}_0(\alpha_n v)\, \right]_{v=1} - 2\left[\, v^2\, \mathrm{J}_0(\alpha_n v)\, \right]_{v=0} \\
&\quad -4\int_0^1 v\, \mathrm{J}_0(\alpha_n v)\, \mathrm{d}v = (\alpha_n - \frac{4}{\alpha_n})\, \mathrm{J}_1(\alpha_n).
\end{aligned} \tag{1.4.36}$$

Substituting (1.4.35) and (1.4.36) into (1.4.32), we obtain the desired result,

$$c_n = -\frac{1}{\mathrm{J}_1(\alpha_n)}\, \frac{2}{\alpha_n^3} \frac{\chi + \rho g_x}{\mu}\, a(a+\ell). \tag{1.4.37}$$

Finally, we substitute (1.4.37) into (1.4.26) and then into (1.4.8), and thus derive the velocity profile

$$\begin{aligned}
u_x(\sigma, t) = {} & \tfrac{1}{4}\, \frac{\chi + \rho g_x}{\mu} \Big(a^2 + 2a\ell - \sigma^2 \\
& -8\, a(a+\ell) \sum_{n=1}^{\infty} \frac{1}{\alpha_n^3} \frac{\mathrm{J}_0(\alpha_n\, \sigma/a)}{\mathrm{J}_1(\alpha_n)}\, \exp(-\alpha_n^2\, \frac{\nu\, t}{a^2}) \Big).
\end{aligned} \tag{1.4.38}$$

We recall that the dimensionless coefficients α_n must be computed in numerical form.

Exercises

1.4.1 Derive expression (1.4.37).

1.4.2 Compute the first seven zeros α_n for $\lambda = 1.00$.

1.4.3 Laplace's equation in cylindrical polar coordinates for a function $f(x, \sigma)$ that depends only on axial position, x, and radial position, σ, reads

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial \sigma^2} + \frac{1}{\sigma}\frac{\partial f}{\partial \sigma} = 0. \tag{1.4.39}$$

Consider the solution for $x \geq 0$ and $0 \leq \sigma \leq a$, subject to the boundary conditions $f = 0$ at $x = 0$, $f = 0$ at $\sigma = a$, and $f = f_b$ at $x = b$, where a and b are specified lengths and f_b is a specified boundary value.

Show that the solution is given by

$$f(x, \sigma) = f_b \sum_{n=1}^{\infty} E_n \sinh(\alpha_n \frac{x}{a})\, \mathrm{J}_0(\alpha_n \frac{\sigma}{a}), \tag{1.4.40}$$

where α_n are zeros of the Bessel function $\mathrm{J}_0(z)$ and E_n are expansion coefficients given by

$$E_n = \frac{2}{\alpha_n} \frac{1}{\sinh(\alpha_n \frac{b}{a}) \mathrm{J}_1(\alpha_n)}. \tag{1.4.41}$$

Discuss the solution in the limit as b/a tends to zero.

1.5 To beat a drum

Consider the axisymmetric vibration of a stretched membrane that is attached to a circular rim of radius a, as shown in Figure 1.5.1. In the absence of damping, the membrane displacement satisfies the wave equation

$$\rho \frac{\partial^2 u}{\partial t^2} = \frac{\tau}{r}\frac{\partial}{\partial r}\Big(r \frac{\partial u}{\partial r}\Big), \tag{1.5.1}$$

where t stands for time, r is the distance from the centerpoint, τ is the uniform tension of the stretched membrane, and ρ is the surface density of the membrane material with units of mass over area.

Since the membrane is attached to the rim at all times, $u = 0$ at $r = a$. A regularity condition requires that u is a continuous function with continuous derivatives over the entire membrane.

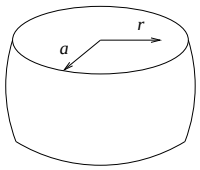


FIGURE 1.5.1 Beating a drum with proper hearing protection can be a lot of fun.

1.5.1 Periodic vibrations

To describe periodic vibrations with angular frequency ω, we separate the radial and the temporal dependencies by setting

$$u(r,t) = f(r)\,\cos(\omega t), \tag{1.5.2}$$

and find that the modulating function $f(r)$ satisfies the second-order linear ordinary differential equation

$$\frac{1}{r}\frac{\mathrm{d}}{\mathrm{d}r}\Big(r\,\frac{\mathrm{d}f}{\mathrm{d}r}\Big) + \frac{\rho\,\omega^2}{\tau}\,f = 0. \tag{1.5.3}$$

The rim boundary condition requires that $f(a) = 0$, while the regularity condition requires that $\mathrm{d}f/\mathrm{d}r \to 0$ as $r \to 0$.

Nontrivial eigensolutions expressing different modes of vibration are possible only for particular values of the group $\rho\,\omega^2/\tau$, representing specific frequencies of vibration.

1.5.2 Exact solution

Following the discussion in Section 1.4, we find that the exact solution of the eigenvalue problem is given by

$$f(r) = A\,\mathrm{J}_0(\mu\frac{r}{a}), \tag{1.5.4}$$

where A is a constant, J_0 is a Bessel function, and

$$\mu = \omega\,a\,\sqrt{\frac{\rho}{\tau}} \tag{1.5.5}$$

is a dimensionless vibration frequency.

To ensure that the membrane remains attached to the rim, we require that the eigenvalues, μ, are the zeros of J_0, satisfying $\mathrm{J}_0(\mu) = 0$.

1.5.3 Fourier expansion

Now deliberately disregarding the exact solution, we introduce the Fourier expansion

$$f(r) = \sum_{n=1}^{\infty} A_n \, \cos(\alpha_n \frac{r}{a}), \tag{1.5.6}$$

where

$$\alpha_n = (n - \tfrac{1}{2})\,\pi. \tag{1.5.7}$$

The regularity constraint at the center and the rim boundary condition are both satisfied.

Substituting the expansion (1.5.6) into the governing equation (1.5.3), carrying out the differentiations, and rearranging, we obtain a homogeneous algebraic equation,

$$\sum_{n=1}^{\infty} A_n \Big(-\alpha_n^2 \, \varrho \, \cos(\alpha_n \varrho) - \alpha_n \, \sin(\alpha_n \varrho) + \mu^2 \, \varrho \, \cos(\alpha_n \varrho) \Big) = 0, \tag{1.5.8}$$

where $\varrho \equiv r/a$. Nontrivial (non-zero) solutions for A_n are possible for specific eigenvalues, μ representing admissible frequencies of vibration.

1.5.4 Galerkin projection

In Galerkin's method, a system of linear equations for a truncated set of coefficients, A_n for $n = 1, \ldots, M$, is derived by multiplying (1.5.8) by $\varrho^\ell \cos(\alpha_m \varrho)$ and integrating the product with respect to ϱ from 0 to 1, where m is a free index and ℓ is a specified exponent.

1.5.5 One mode

When $M = 1$, we set $n = 1$ and $m = 1$ and obtain

$$A_1 \Big(-\alpha_1^2 \int_0^1 \varrho^{\ell+1} \cos^2(\alpha_1 \varrho)\, \mathrm{d}vr - \alpha_1 \int_0^1 (\varrho^{\ell} \sin(\alpha_1 \varrho) \cos(\alpha_1 \varrho)\, \mathrm{d}\varrho + \mu^2 \int_0^1 (\varrho^{\ell+1} \cos^2(\alpha_1 \varrho)\, \mathrm{d}\varrho \Big) = 0, \tag{1.5.9}$$

where $\alpha_1 = \frac{1}{2}\pi$. For A_1 to not be zero, the quantity enclosed by the tall parentheses must be zero, yielding the expression

$$\mu^2 = \frac{\alpha_1^2 \int_0^1 \varrho^{\ell+1} \cos^2(\alpha_1 \varrho)\, \mathrm{d}\varrho + \alpha_1 \int_0^1 (\varrho^{\ell} \sin(\alpha_1 \varrho) \cos(\alpha_1 \varrho)\, \mathrm{d}\varrho}{\int_0^1 (\varrho^{\ell+1} \cos^2(\alpha_1 \varrho)\, \mathrm{d}\varrho}, \tag{1.5.10}$$

which is equivalent to

$$\mu^2 = \alpha_1^2 \frac{\int_0^{\pi/2} \omega^{\ell+1} \cos^2 \omega\, \mathrm{d}\omega + \int_0^{\pi/2} \omega^{\ell} \sin\omega\, \cos\omega\, \mathrm{d}\omega}{\int_0^{\pi/2} \omega^{\ell+1} \cos^2 \omega\, \mathrm{d}\omega}, \tag{1.5.11}$$

where $\omega = \alpha_1 \varrho$.

1.5.6 First mode

For $M = 1$ and $\ell = 0$, we obtain

$$\mu^2 = \alpha_1^2 \frac{\int_0^{\pi/2} \omega \cos^2 \omega\, \mathrm{d}\omega + \int_0^{\pi/2} \sin\omega\, \cos\omega\, \mathrm{d}\omega}{\int_0^{\pi/2} \omega \cos^2 \omega\, \mathrm{d}\omega}. \tag{1.5.12}$$

Computing the required integrals

$$\int_0^{\pi/2} \omega \cos^2 \omega\, \mathrm{d}\omega = \tfrac{1}{16}\,(\pi^2 - 4) \tag{1.5.13}$$

and

$$\int_0^{\pi/2} \sin\omega\, \cos\omega\, \mathrm{d}\omega = \tfrac{1}{2}, \tag{1.5.14}$$

we obtain

$$\mu^2 = \frac{\pi^2}{4}\,\frac{\pi^2+4}{\pi^2-4} \simeq 5.830, \tag{1.5.15}$$

yielding $\mu = 2.415$, which is remarkably close to the first zero of J_0, given by 2.405.

For $M = 1$ and $\ell = -1$, we obtain

$$\mu^2 = \alpha_1^2\,\frac{\int_0^{\pi/2}\cos^2\omega\,\mathrm{d}\omega + \int_0^{\pi/2}\frac{\sin\omega}{\omega}\,\cos\omega\,\mathrm{d}\omega}{\int_0^{\pi/2}\cos^2\omega\,\mathrm{d}\omega}. \tag{1.5.16}$$

We compute the required integrals,

$$\int_0^{\pi/2}\cos^2\omega\,\mathrm{d}\omega = \int_0^{\pi/2}\frac{1+\cos(2\omega)}{2}\,\mathrm{d}\omega = \frac{\pi}{4} \tag{1.5.17}$$

and

$$\begin{aligned}\int_0^{\pi/2}\frac{\sin\omega}{\omega}\,\cos\omega\,\mathrm{d}\omega &= \int_0^{\pi/2}\frac{\sin(2\omega)}{2\omega}\,\mathrm{d}\omega \\ &\equiv \tfrac{1}{2}\int_0^{\pi}\mathrm{sinc}(\mathrm{z})\,\mathrm{d}z \equiv \tfrac{1}{2}\mathrm{Si}(\pi) \simeq \tfrac{1}{2}\,1.85194,\end{aligned} \tag{1.5.18}$$

where $z = 2\omega$. Substituting these values into (1.5.16), we obtain

$$\mu^2 = \frac{\pi^2}{4}\,\frac{\frac{\pi}{4}+1.85194}{\frac{\pi}{4}} \simeq 5.3764. \tag{1.5.19}$$

yielding $\mu = 2.3191$, which is also close to the first zero of J_0, given by 2.405.

Exercises

1.5.1 (a) Explain why it is not possible to set $\ell = -2$. (b) Compute and discuss the value of μ obtained for $M = 1$ and $\ell = 2$.

1.5.2 In plane polar coordinates, (r, θ), Laplace's equation for a function $f(r, \theta)$ takes the form

$$\frac{1}{r}\,\frac{\partial}{\partial r}\Big(r\,\frac{\mathrm{d}f}{\mathrm{d}r}\Big) + \frac{1}{r^2}\,\frac{\partial^2 f}{\partial\theta^2} = 0. \tag{1.5.20}$$

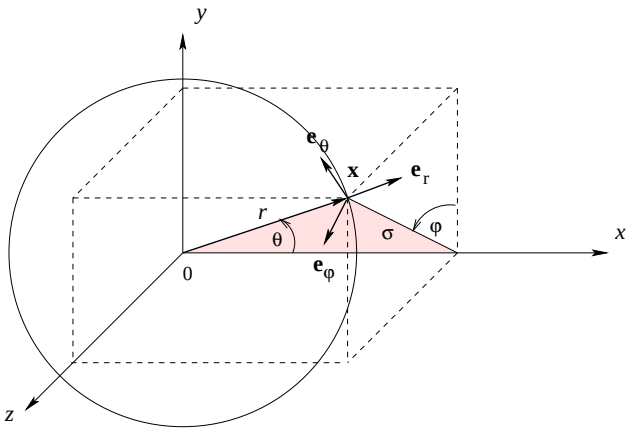


FIGURE 1.6.1 Illustration of spherical polar coordinates, (r, θ, φ), defined with respect to the Cartesian coordinates, (x, y, z), and cylindrical polar coordinates, (x, σ, φ), where r is the distance from the origin, θ is the meridional angle, φ is the azimuthal angle, and σ is the distance from the x axis.

Confirm that the general solution of this equation is given by the Fourier series expansion

$$f = A_0 + C_0 \ln r + \sum_{n=1}^{\infty} \left(A_n r^n + \frac{B_n}{r} \right) \cos(n\theta) + \sum_{n=1}^{\infty} \left(C_n r^n + \frac{D_n}{r} \right) \sin(n\theta), \tag{1.5.21}$$

where A_i, B_i, C_i, D_i are constants.

1.6 Laplace's equation in spherical coordinates

A point in space can be identified by the values of the ordered triplet (r, θ, φ), as illustrated in Figure 1.6.1, where:

- r is the distance from the designated origin taking values in the range $[0, \infty)$.

- θ is the meridional angle subtended between the x axis, the origin, and the chosen point, taking values in the range $[0, \pi]$.
- φ is the azimuthal angle measured around the x axis, taking values in the range $[0, 2\pi)$.

The Laplacian of a scalar function, $f(r, \theta, \varphi)$, is given by

$$\nabla^2 f = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial f}{\partial r}\right) + \frac{1}{r^2\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial f}{\partial\theta}\right) + \frac{1}{r^2\sin^2\theta}\frac{\partial^2 f}{\partial\varphi^2}. \tag{1.6.1}$$

Expanding the derivatives, we obtain terms involving first and second derivatives in r, θ, and φ.

1.6.1 *Axisymmetric solutions*

Assume that a function, f, is axisymmetric, that is, it depends on r and θ but not on φ. Laplace's equation for such a function becomes

$$\nabla^2 f = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial f}{\partial r}\right) + \frac{1}{r^2\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial f}{\partial\theta}\right) = 0. \tag{1.6.2}$$

To develop a solution by separation of variables, we set

$$f(r, \theta) = \sum_{n=1}^{\infty} p_n(r)\, q_n(\theta), \tag{1.6.3}$$

where (p_n, q_n) are corresponding pairs of dimensionless functions to be found as part of the solution. Straightforward differentiation provides us with an expression for the Laplacian,

$$\nabla^2 f = \frac{1}{r^2}\sum_{n=1}^{\infty}\left[q_n\frac{\mathrm{d}}{\mathrm{d}r}\left(r^2\frac{\mathrm{d}p_n}{\mathrm{d}r}\right) + \frac{p_n}{\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial q_n}{\partial\theta}\right)\right] = 0. \tag{1.6.4}$$

A key observation is that the Laplacian is zero only if the general term enclosed by the tall square brackets is zero. Setting this term to zero and rearranging, we find that

$$\frac{1}{p_n}\frac{\mathrm{d}}{\mathrm{d}r}\left(r^2\frac{\mathrm{d}p_n}{\mathrm{d}r}\right) = -\frac{1}{q_n}\frac{1}{\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial q_n}{\partial\theta}\right) = \alpha_n, \tag{1.6.5}$$

where α_n is a constant to be found as part of the solution. The reason for this double equality is that the left-hand side is a function of r, while the middle side is a function of θ; both must be equal to a constant.

The double equality in (1.1.11) provides us with two ordinary differential equations,

$$\frac{\mathrm{d}}{\mathrm{d}r}\left(r^2\frac{\mathrm{d}p_n}{\mathrm{d}r}\right)-\alpha_n p_n=0 \tag{1.6.6}$$

and

$$\frac{\partial}{\partial\theta}\left(\sin\theta\,\frac{\partial q_n}{\partial\theta}\right)+\alpha_n\,\sin\theta\,q_n=0. \tag{1.6.7}$$

Suitable regularity conditions will be imposed.

The general solution of (1.6.6) is

$$p_n(y)=A_n r^{\lambda_1}+B_n r^{\lambda_2}, \tag{1.6.8}$$

where A_n and B_n are two constants and λ_1, λ_2 are the roots of the quadratic equation

$$\lambda(\lambda+1)=\alpha_n. \tag{1.6.9}$$

To solve equation (1.6.7), we introduce the variable $\mu=\cos\varphi$ and express it in the form

$$(1-\mu^2)\,\frac{\mathrm{d}^2 q_n}{\mathrm{d}\mu^2}-2\mu\,\frac{\mathrm{d}q_n}{\mathrm{d}\mu}+\alpha_n q_n=0, \tag{1.6.10}$$

which can be rewritten as

$$\frac{\mathrm{d}}{\mathrm{d}\mu}\left((1-\mu^2)\,\frac{\mathrm{d}q_n}{\mathrm{d}\mu}\right)+\alpha_n q_n=0. \tag{1.6.11}$$

The solution is required to be regular at $\theta=0,\pi$, that is, $\mu=\pm1$. This regularity condition is satisfied when

$$\alpha_n=n(n-1) \tag{1.6.12}$$

for $n = 1, 2, \ldots$, and the solution is given by the Legendre polynomials, L_m,

$$q_n(\mu) = \mathrm{L}_{n-1}(\mu), \tag{1.6.13}$$

where

$$\mathrm{L}_0 = 1, \qquad \mathrm{L}_1 = \mu, \qquad \mathrm{L}_2 = \frac{1}{2}\,(3\mu^2 - 1), \quad \ldots\,. \tag{1.6.14}$$

Substituting (1.6.12) into (1.6.9) we obtain

$$\lambda\,(\lambda + 1) = n(n-1), \tag{1.6.15}$$

yielding

$$\lambda_1 = n - 1, \qquad \lambda_2 = -n. \tag{1.6.16}$$

The general solution of the differential equation is then

$$f(r,\theta) = \sum_{n=1}^{\infty} \left(A_n r^{n-1} + B_n \frac{1}{r^n} \right) \mathrm{L}_{n-1}(\mu), \tag{1.6.17}$$

where the constants A_n and B_n are chosen to satisfy boundary conditions.

When the solution domain includes the origin, we set $B_n = 0$ to prevent singularities. When the solution domain extends to infinity, we set $A_n = 0$ for $n > 1$ to prevent unbounded growth.

1.6.2 *Interior of a sphere*

For example, the solution inside a sphere of radius a subject to the boundary condition $f(\theta) = \phi(\theta)$ at $r = a$, where $\phi(\theta)$ is a given function, is given by

$$f(r,\theta) = \sum_{n=1}^{\infty} E_n \left(\frac{r}{a} \right)^{n-1} \mathrm{L}_{n-1}(\mu), \tag{1.6.18}$$

where the coefficients E_n satisfy

$$\sum_{n=1}^{\infty} E_n \mathrm{L}_{n-1}(\mu) = \phi(\theta) \tag{1.6.19}$$

To compute these coefficients, we introduce the orthogonality property

$$\int_{-1}^{1} \mathrm{L}_i(\mu)\,\mathrm{L}_j(\mu)\,\mathrm{d}\mu = \frac{2}{2i+1}\,\delta_{ij}, \tag{1.6.20}$$

where δ_{ij} is Kronecker's delta, and find that

$$E_n = \frac{2n-1}{2}\int_{-1}^{1} \mathrm{L}_{n-1}(\mu)\,\phi(\theta)\,\mathrm{d}\mu, \tag{1.6.21}$$

where we recall that $\mu = \cos\theta$.

Exercise

1.6.1 Evaluate the coefficients E_n given in (1.6.21) when $\phi(\theta)$ is a constant.

1.7 Sturm–Louiville equations

Previously in this chapter, we derived ordinary differential equation by separation of variables, including equations (1.1.12) satisfied by sinusoidal and exponential functions,

$$\frac{\mathrm{d}^2 f_n}{\mathrm{d}y^2} - \frac{\alpha_n^2}{b^2}\,f_n = 0, \qquad \frac{\mathrm{d}^2 q_n}{\mathrm{d}z^2} + \frac{\alpha_n^2}{b^2}\,q_n = 0, \tag{1.7.1}$$

equation (1.4.16) satisfied by a Bessel function,

$$z\,\frac{\mathrm{d}}{\mathrm{d}z}\Big(z\,\frac{\mathrm{d}\mathrm{J}_0(z)}{\mathrm{d}z}\Big) + z^2\,\mathrm{J}_0(z) = 0, \tag{1.7.2}$$

equation (1.6.6) satisfied by power functions,

$$\frac{\mathrm{d}}{\mathrm{d}r}\left(r^2\frac{\mathrm{d}p_n}{\mathrm{d}r}\right) - \alpha_n p_n = 0, \tag{1.7.3}$$

and equation (1.6.11) satisfied by the Legendre polynomials

$$\frac{\mathrm{d}}{\mathrm{d}\mu}\Big((1-\mu^2)\,\frac{\mathrm{d}q_n}{\mathrm{d}\mu}\Big) + \alpha_n q_n = 0. \tag{1.7.4}$$

All of these equations are instances of the general Sturm–Liouville equation for a function $f(z)$,

$$\frac{\mathrm{d}}{\mathrm{d}z}\Big(p(z)\,\frac{\mathrm{d}f(z)}{\mathrm{d}z}\Big) + q(x)\,f(z) = -\lambda\,w(z)\,f(z), \tag{1.7.5}$$

where $p(z), q(z), w(z)$ are specified functions and λ is a constant playing the role of an eigenvalue. In a typical Sturm–Liouville problem, λ is found as part of the solution.

The eigenvalues of a Sturm–Liouville problem comprise an infinite set. Any two solutions corresponding to different eigenvalues subject to suitable homogeneous boundary conditions satisfy the orthogonality condition

$$\int f_m(z)\,f_n(z)\,w(z)\,\mathrm{d}z = \delta_{mn}\,F_n, \tag{1.7.6}$$

where the integral is performed over the domain of definition of the solution, and F_n is a normalization constant.

Exercise

1.7.1 Identify the functions $p(z), q(z), w(z)$ corresponding to (1.7.4).

1.8 Biharmonic equation

The biharmonic equation for a function, $\psi(x, y)$, reads

$$\nabla^4\psi = 0, \tag{1.8.1}$$

where $\nabla^4 = \nabla^2\nabla^2$ is the biharmonic operator and ∇^2 is the Laplacian operator. Writing

$$\nabla^2\psi = \frac{\partial^2\psi}{\partial x^2} + \frac{\partial^2\psi}{\partial^2 y}, \tag{1.8.2}$$

we obtain

$$\nabla^4\psi = \frac{\partial^4\psi}{\partial^4 x} + 2\,\frac{\partial^4\psi}{\partial^2 x\partial^2 y} + \frac{\partial^4\psi}{\partial^4 y}. \tag{1.8.3}$$

Two scalar boundary conditions must be specified around the boundary of the solution domain.

To develop a solution by separation of variables, we set

$$\nabla^2\psi(x,y) = \sum_{n=1}^{\infty} f_n(x)\,q_n(y), \tag{1.8.4}$$

where $f_n(x)$ and $q_n(y)$ are requisite functions. Substituting this expansion into (1.8.3), we obtain

$$\frac{1}{f_n}\,\frac{\mathrm{d}f_n}{\mathrm{d}x^2} = -\frac{1}{q_n}\,\frac{\mathrm{d}q_n}{\mathrm{d}y^2} = \pm\frac{\alpha_n^2}{a^2}. \tag{1.8.5}$$

where α is a chosen length and α_n is a constant. The reason for this double equality is that the left-hand side is a function of x, while the middle side is a function of y; both must be equal to a constant.

The double equality in (1.8.5) provides us with two ordinary differential equations,

$$\frac{\mathrm{d}^2 f_n}{\mathrm{d}x^2} - \frac{\pm\alpha_n^2}{a^2}\,f_n = 0, \qquad \frac{\mathrm{d}^2 q_n}{\mathrm{d}y^2} + \frac{\pm\alpha_n^2}{a^2}\,q_n = 0. \tag{1.8.6}$$

whose solutions are either trigonometric or hyperbolic functions, depending on the choice of the $+$ or $-$ sign.

Motivated by these solutions, we set

$$\psi(x,y) = \sum_{n=1}^{\infty} s_n(y)\,\cos(\alpha_n x) + \sum_{n=1}^{\infty} r_n(y)\,\sin(\alpha_n x), \tag{1.8.7}$$

and find that the functions $s_n(y)$ and $r_n(y)$ satisfy the ordinary differential equation

$$\chi'''' - 2\,\alpha_n^2\,\chi'' + \alpha_n^4 = 0, \tag{1.8.8}$$

where a prime denotes a derivative with respect to y, and χ can be s_n or r_n. The general solution is

$$\chi(y) = A\,\mathrm{e}^{\alpha_n y} + B\,y\mathrm{e}^{\alpha_n y} + C\,y\mathrm{e}^{-\alpha_n y} + D\,y\mathrm{e}^{-\alpha_n y}, \tag{1.8.9}$$

where A, B, C and D are four constants. Altogether, the general solution involves eight constant for each n. The x and y coordinates can be interchanged in the general solution according to the nature of the problem under consideration.

Exercise

1.8.1 Confirm that (1.8.9) satisfies equation (1.8.8).

1.9 Viscous flow in a wedge-shaped domain

The velocity components of a two-dimensional flow in the xy plane, u_x and u_y, are related to the stream function, ψ, by the equations

$$u_x = \frac{\partial \psi}{\partial y}, \qquad u_y = -\frac{\partial \psi}{\partial x}, \tag{1.9.1}$$

ensuring that

$$\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} = 0. \tag{1.9.2}$$

according to the continuity equation for an incompressible fluid.

In the case of purely viscous flow devoid of inertial effects (Stokes flow), the stream function satisfies the biharmonic equation,

$$\nabla^4 \psi = 0, \tag{1.9.3}$$

where $\nabla^4 = \nabla^2\nabla^2$ as discussed in Section 1.9.

1.9.1 Solution in plane polar coordinates

In the case of Stokes flow in a wedge-shaped domain, it is appropriate to introduce plane polar coordinates with origin at the vertex, (r, θ). The biharmonic equation (1.9.3) takes the form

$$\nabla^4 \psi = \Big(\frac{1}{r} \frac{\partial}{\partial r} \Big(r \frac{\partial}{\partial r} \Big) + \frac{1}{r^2} \frac{\partial^2}{\partial \theta^2} \Big)^2 \psi = 0. \tag{1.9.4}$$

Separating the radial from the angular dependencies, we write

$$\psi(r, \theta) = q(r)\, f(\theta), \tag{1.9.5}$$

where $q(r)$ and $f(\theta)$ are two unknown functions to be computed subject to appropriate boundary conditions.

1.9.2 Power-law radial dependence

Now working in hindsight, we stipulate the power-law functional dependence

$$q(r) = A\, r^{\lambda}, \tag{1.9.6}$$

where the complex constant A is a measure of the intensity of the outer flow and the complex exponent λ determines the structure of the inner Stokes flow. It is understood that either the real or the imaginary part of the complex stream function ψ provides us with an acceptable solution.

Substituting (1.9.6) into (1.9.4), we obtain a fourth-order homogeneous ordinary differential equation for f,

$$f'''' + 2\,(\lambda^2 - 2\lambda + 2)\, f'' + \lambda^2 (\lambda - 2)^2 f = 0, \tag{1.9.7}$$

where a prime denotes a derivative with respect to θ. Substituting into this equation an eigensolution of the form

$$f(\theta) = \exp(\kappa \theta), \tag{1.9.8}$$

we obtain a bi-quadratic equation for the generally complex constant κ,

$$\kappa^4 + 2\,(\lambda^2 - 2\lambda + 2)\, \kappa^2 + \lambda^2 (\lambda - 2)^2 = 0. \tag{1.9.9}$$

When κ has an imaginary component, the eigen-solutions can be expressed in terms of trigonometric functions. Solving for κ and substituting the result into (1.9.8), we obtain the general solution

$$f(\theta) = \begin{cases} B\sin(\lambda\theta - \beta) + C\sin[(\lambda-2)\theta - \gamma] & \text{if } \lambda \neq 0, 1, 2, \\ B\sin(2\theta - \beta) + C\theta + D & \text{if } \lambda = 0, 2, \\ B\sin(\theta - \beta) + C\theta\sin(\theta - \gamma) & \text{if } \lambda = 1, \end{cases} \tag{1.9.10}$$

where B, C, and D are complex constants and β and γ are real constants.

1.9.3 *Flow near and around a corner*

As an application, we consider flow near or around a corner that is confined by two stationary intersecting walls located at $\theta = \pm\alpha$. The local flow near the corner is driven by the motion of the fluid far from the walls.

Because all boundary conditions for the stream function are homogeneous, the solution must be found by solving an eigenvalue problem for the exponent λ introduced in (1.9.6). Stated differently, the exponent λ cannot be assigned *a priori* but must be found as part of the solution.

We will consider separately the cases of flow with antisymmetric or symmetric streamline pattern, as shown in Figure 1.9.1. Symmetric flow occurs when the outer flow exhibits a corresponding symmetry, or else when the mid-plane of symmetry represents a free surface with vanishing shear stress.

1.9.4 *Antisymmetric flow*

Concentrating first on the case of antisymmetric flow, we require that the velocity is zero at the walls and derive the boundary conditions

$$\psi = 0, \qquad \frac{\partial\psi}{\partial\theta} = 0 \qquad \text{at} \qquad \theta = \pm\alpha. \tag{1.9.11}$$

Assuming that $\lambda \neq 0, 1, 2$, using (1.9.10), and stipulating that the velocity is antisymmetric with respect to the mid-plane, $\partial\psi/\partial\theta = 0$ at

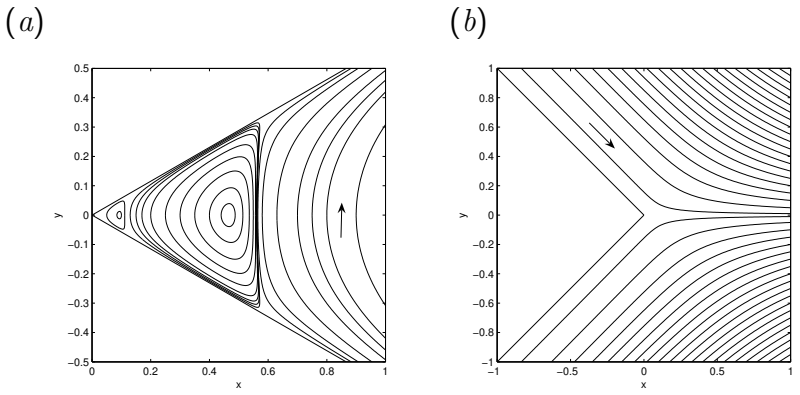


FIGURE 1.9.1 (a) Antisymmetric and (b) symmetric Stokes flow near two intersecting stationary plane walls

$\theta = 0$, we find that

$$f(\theta) = A\cos\lambda\theta + B\cos[(\lambda-2)\theta], \tag{1.9.12}$$

where A and B are two complex constants. When λ is complex, the complex conjugate of the right-hand side of (1.9.12) is also added to extract the real part.

Enforcing the boundary conditions at the walls, we obtain two homogeneous equations,

$$\begin{bmatrix} \cos\lambda\alpha & \cos[(\lambda-2)\alpha] \\ \lambda\sin\lambda\alpha & (\lambda-2)\sin[(\lambda-2)\alpha] \end{bmatrix} \cdot \begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \tag{1.9.13}$$

A nontrivial solution for A and B is possible only when the determinant of the coefficient matrix is zero. Setting the determinant to zero provides us with the secular equation

$$(\lambda-2)\cos\lambda\alpha\sin[(\lambda-2)\alpha] - \lambda\sin\lambda\alpha\cos[(\lambda-2)\alpha] = 0, \tag{1.9.14}$$

which can be recast into the compact form of a nonlinear algebraic equation for λ,

$$\sin[2\alpha(\lambda-1)] = (1-\lambda)\,\sin 2\alpha. \tag{1.9.15}$$

An obvious solution is $\lambda = 1$. However, since for this value the third rather than the first expression in (1.9.10) should have been chosen, this choice is disqualified. Combining (1.9.13) with (1.9.12) and (1.9.5), we find that the stream function is given by

$$\psi = Cr^{\lambda}\big(\cos\lambda\theta\cos[(\lambda-2)\alpha] - \cos\lambda\alpha\cos[(\lambda-2)\theta]\big), \tag{1.9.16}$$

where C is a real constant.

To account for the possibility of complex solutions, we write $\lambda = \lambda_{\mathrm{R}} + \mathrm{i}\lambda_{\mathrm{I}}$ and decompose (1.9.15) into its real and imaginary parts, indicated by the subscripts R and I, obtaining

$$\begin{aligned} &\sin[2\alpha(\lambda_{\mathrm{R}}-1)]\cosh(2\alpha\lambda_{\mathrm{I}}) = (1-\lambda_{\mathrm{R}})\sin 2\alpha, \\ &\cos[2\alpha(\lambda_{\mathrm{R}}-1)]\sinh(2\alpha\lambda_{\mathrm{I}}) = -\lambda_{\mathrm{I}}\sin 2\alpha. \end{aligned} \tag{1.9.17}$$

To simplify the notation, we introduce the auxiliary variables

$$\xi = 2\alpha(\lambda_{\mathrm{R}}-1), \qquad \eta = 2\alpha\lambda_{\mathrm{I}}, \qquad \kappa = \frac{\sin 2\alpha}{2\alpha}, \tag{1.9.18}$$

note that

$$\lambda = 1 + \frac{\xi + \mathrm{i}\eta}{2\alpha}, \tag{1.9.19}$$

and recast (1.9.17) into the form

$$\sin\xi\cosh\eta = -\kappa\xi, \qquad \cos\xi\sinh\eta = -\kappa\eta. \tag{1.9.20}$$

Since both the sine and cosine of ξ must be negative, ξ must lie in the range $(2n-1)\pi < \xi < (2n-\frac{1}{2})\pi$, where n is an integer. The solution of the two nonlinear equations (1.9.20) can be computed using Newton's method for two equations with suitable initial guesses for ξ and η.

Real and positive solutions ($\eta = 0$) other than the trivial solution exist in the range $73^\circ < \alpha < 180^\circ$ or $0.41\pi < \alpha < \pi$, where the lower limit is a numerical approximation. For lower values of α we obtain complex solutions.

1.9.5 Computer code

The following Matlab code, located in directory CORNER of SEMLIB, solves the nonlinear equations for ξ and η for an array of values of α using the Newton–Raphson method.

To ensure a good initial guess for the Newton–Raphson method, the solution space with respect to α is computed in two passes, one for complex and the second for real solutions. The code calls the function *newton2* located in directory NEWTON2 of SEMLIB:

```
global alpha

%------------------
% menu = 1: antisymmetric flow in a corner
%      = 2:     symmetric flow in a corner
%------------------

%---
% prepare
%---

   eps   = 0.001;    % epsilon for Newton iterations
   Niter = 100;      % maximum number of iterations
   Nint  = 128;      % number of intervals

   menu = 1;

   figure(1)
   hold on
   set(gca,'fontsize',15)

   alphai = 73.0;  % alpha (initial) in degrees
   alphai = alphai/180.0*pi;

%---
% scan alpha
%---

   for iscan=1:2
```

```
 x(1)   = 3.3;   % initial guess for xi and eta
 x(2)   = 1.75

 if(iscan==1)
   alphaf = 0.01;  % alpha(final) in degrees
 elseif(iscan==2)
   alphaf = 179.0;  % alpha(final) in degrees
 end

alphaf = alphaf/180.0*pi;
dalpha = (alphaf-alphai)/Nint;

for i=1:Nint+1

   alpha = alphai+(i-1.0)*dalpha;

   [x,f,Iflag] = newton2 ...
    ...
   (menu ...
   ,Niter ...
   ,eps ...
   ,x ...
   );

   lamR = x(1)/(2*alpha)+1.0;
   lamI = x(2)/(2*alpha);

   alphaplot(i) = alpha/pi;

   xiplot(i) = x(1); etplot(i) = x(2);
   lamRplot(i) = lamR; lamIplot(i) = lamI;

  end

  plot(alphaplot,lamRplot,'k')
  plot(alphaplot,lamIplot,'r')
  plot(alphaplot,xiplot,'k--')
  plot(alphaplot,etplot,'r--')
```

```
  end
```

The following function is called by *newton2*:

```
function f = newton2_fun(menu,x)

global alpha

%---
% function evaluation for Newton's method
%---

     q = 2.0*alpha;
     k = sin(q)/q;
     coshy = 0.5*(exp(x(2))+exp(-x(2)));
     sinhy = 0.5*(exp(x(2))-exp(-x(2)));

%-----------------------
      if(menu == 1)   % anti-symmetric Stokes flow
%-----------------------

      f(1) = sin(x(1))*coshy + k*x(1);
      f(2) = cos(x(1))*sinhy + k*x(2);

%-----------------------
      elseif(menu == 2) % symmetric Stokes flow
%-----------------------

      f(1) = sin(x(1))*coshy - k*x(1);
      f(2) = cos(x(1))*sinhy - k*x(2);

%-----------------------
      end
%-----------------------

     return
```

The graph generated by the code is shown in Figure 1.9.2. The results for a particular value of α can be used to compute streamline patterns.

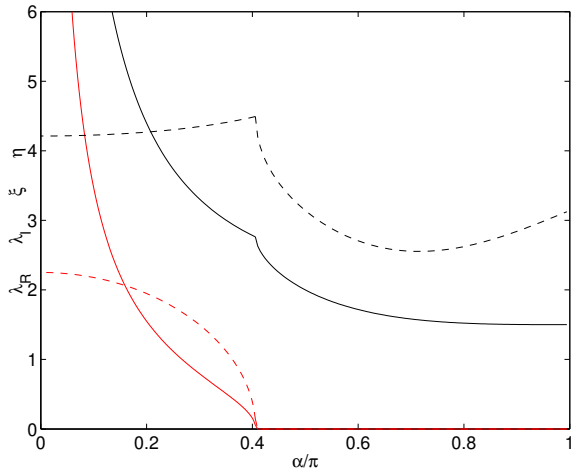


FIGURE 1.9.2 Viscous antisymmetric flow in a corner. Solution for ξ (solid red line), η (broken red line), Solution for $\lambda_{\rm R}$ (solid black line), and $\lambda_{\rm I}$ (broken black line) against the aperture semi-angle, α.

1.9.6 Symmetric flow

Next, we address the case of symmetric flow. Requiring that the velocity vanishes at the walls located at $\theta = \pm\alpha$ and the shear stress vanishes at the mid-plane $\theta = 0$, we derive the boundary conditions

$$\psi = 0, \qquad \frac{\partial \psi}{\partial \theta} = 0 \tag{1.9.21}$$

at $\theta = \pm\alpha$, and

$$\psi = 0, \qquad \frac{\partial^2 \psi}{\partial \theta^2} = 0 \tag{1.9.22}$$

at $\theta = 0$. Working as in the case of the antisymmetric flow, we set

$$f(\theta) = A \sin \lambda\theta + B \sin[(\lambda - 2)\theta], \tag{1.9.23}$$

where A and B are two complex constants. Enforcing the boundary conditions, we derive an eigenvalue problem expressed by the equation

$$\sin[2\alpha(\lambda - 1)] = (\lambda - 1)\, \sin 2\alpha. \tag{1.9.24}$$

The obvious solutions, $\lambda = 1, 2$, are disqualified for the reason that the second and third rather than the first expression in (1.9.10) should have been chosen.

Real and positive solutions ($\eta = 0$) other than the trivial solution exist in the range $78^\circ < \alpha < 180^\circ$ or $43\pi < \alpha < 180^\circ$, where the lower limit is a numerical approximation. For lower values of α we obtain complex solutions.

1.9.7 Computer code

The following Matlab code, located in directory CORNER of SEMLIB, solves the nonlinear equations for ξ and η for an array of values of α.

To ensure good initial guesses for Newton's method, the solution space with respect to α is computed in three passes, one for complex and the second and third for real solutions. The general structure of the code is similar to that discussed previously for antisymmetric flow:

```
global alpha

%------------------
% menu = 1: antisymmetric flow in a corner
%      = 2:     symmetric flow in a corner
%------------------

%---
% prepare
%---

   eps   = 0.001;  % epsilon for Newton iterations
   Niter = 100;      % maximum number of iterations
   Nint  = 128;      % number of intervals

   menu = 2;

   figure(1)
   hold on
   set(gca,'fontsize',15)
```

```
%---
% scan alpha
%---

   for iscan=1:3

   if(iscan==1)
      x(1)   = 7.1;  % initial guess for xi and eta
      x(2)   = 0.1;
      alphai = 78.0;  % alpha (initial)
      alphaf = 0.01;  % alpha (final)
   elseif(iscan==2)
      x(1)   = 7.1;
      x(2)   = 0.1;
      alphai = 78.0;
      alphaf = 124.0;
   elseif(iscan==3)
      x(1)   = 3.2;
      x(2)   = 0.0;
      alphai = 179.0;
      alphaf = 124.0;
   end

   alphaf = alphaf/180.0*pi;
   alphai = alphai/180.0*pi;

   dalpha = (alphaf-alphai)/Nint;

     for i=1:Nint+1

      alpha = alphai+(i-1.0)*dalpha;

      [x,f,Iflag] = newton2 ...
       ...
      (menu ...
      ,Niter ...
      ,eps ...
      ,x ...
      );
```

```
      lamR = x(1)/(2*alpha)+1.0;
      lamI = x(2)/(2*alpha);

      alphaplot(i) = alpha/pi;

      xiplot(i) = x(1);
      etplot(i) = x(2);
      lamRplot(i) = lamR;
      lamIplot(i) = lamI;

     end

     plot(alphaplot,lamRplot,'k')
     plot(alphaplot,lamIplot,'r')
     plot(alphaplot,xiplot,'k--')
     plot(alphaplot,etplot,'r--')

end
```

The graph generated by the code is shown in Figure 1.9.3. When $\alpha = 90^\circ$, we find that $\lambda = 2$, corresponding to simple shear flow along the x axis. When $\alpha = 180^\circ$, we find that $\lambda = 1.5$, corresponding to flow along a semi-infinite flat plate.

Exercise

1.9.1 Derive equation (1.9.9).

1.10 Similarity solutions

Similarity solutions can be regarded as the antithesis of solutions by separation of variables in that the independent variables, typically spatial coordinates and time, combine into dimensionless similarity groups. To illustrate this distinction, we consider the unsteady heat conduction

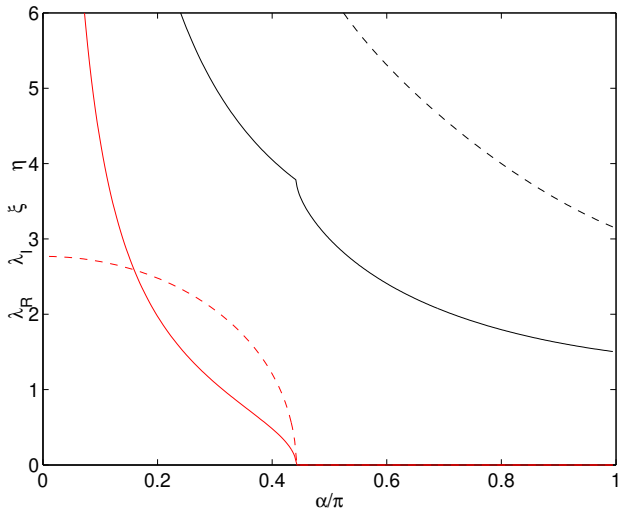


FIGURE 1.9.3 Same as Figure 1.9.2 but for symmetric flow.

equation for the temperature, T,

$$\frac{\partial T}{\partial t} = \kappa \frac{\partial^2 T}{\partial x^2}, \tag{1.10.1}$$

where t stands for times, x is a spatial coordinate, and κ is the thermal diffusivity with dimensions of length square divided by time.

1.10.1 *Separation of variables*

To develop a solution by separation of variables, we set

$$T(x,t) = \sum_{n=1}^{\infty} f_n(t)\, q_n(x), \tag{1.10.2}$$

where $f_n(t)$ and $q_n(x)$ are requisite functions. Substituting this expansion into the governing equation (1.10.1) and working as in Section 1.1, we obtain the double equality

$$\frac{1}{\kappa}\frac{1}{f_n}\frac{\mathrm{d}f_n(t)}{\mathrm{d}t} = \frac{1}{q_n}\frac{\mathrm{d}^2 q_n}{\mathrm{d}x^2} = -\frac{\alpha_n^2}{a^2}, \tag{1.10.3}$$

where a is a specified length and α_n is a dimensionless constant to be found as part of the solution in order to satisfy boundary conditions. The minus sign in the last term will be shortly explained.

The double equality provides us with two ordinary differential equations,

$$\frac{\mathrm{d}f_n}{\mathrm{d}t} + \kappa \frac{\alpha_n^2}{a^2} f_n = 0, \qquad \frac{\mathrm{d}^2 q_n}{\mathrm{d}x^2} + \frac{\alpha_n^2}{a^2} q_n = 0. \tag{1.10.4}$$

The general solutions are

$$f_n(t) = A_n \, \exp(-\alpha_n^2 \frac{\kappa t}{a^2}) \tag{1.10.5}$$

and

$$q_n(z) = C_n \cos(\alpha_n \frac{x}{a}) + D_n \sin(\alpha_n \frac{x}{a}), \tag{1.10.6}$$

where A_n, C_n, and D_n, are constant coefficients computed to satisfy boundary and initial conditions. The minus sign in the last term of (1.10.3) guarantee solutions that decay in time.

The solution by separation of variables can be applied so long as a length scale, a, can be identified. The method cannot be applied otherwise. For example, the method cannot be applied for a semi-infinite or infinite solution domain with respect to x.

1.10.2 Heat transport in a semi-infinite domain

Consider heat transport in a semi-infinite medium $x \geq 0$, due to the sudden elevation of the boundary temperature. The absence of a length scale, a, prevents us from generating a solution by separation of variables, as discussed earlier in this section. To make progress, we note that the set of independent parameters and physical constants involved in the governing equation include t, x, and κ.

Suppose that a solution involves an exponential function for time, t,

$$T(x,t) \sim \exp(\beta t), \tag{1.10.7}$$

where β is a constant with dimensions of inverse time. Our only choice for β is the combination

$$\beta = \gamma \frac{\kappa}{x^2}, \tag{1.10.8}$$

where γ is a dimensionless constant. Consequently,

$$T(x,t) \sim \exp(\gamma \frac{\kappa t}{x^2}). \tag{1.10.9}$$

We have found that the absence of a length scale requires that the time and space dependencies must combine into a unified dimensionless similarity variable, which is nondimensionalized by the thermal diffusivity, κ.

1.10.3 Similarity variable

Motivated by these observations, we introduce a dimensionless similarity variable,

$$\eta \equiv \frac{x}{\sqrt{\kappa t}}, \tag{1.10.10}$$

which ranges from zero to infinity, and consider a solution in the form

$$T(x,t) = f(\eta), \tag{1.10.11}$$

where $f(\eta)$ is a function of the similarity variable. This solution is sensible only if the boundary conditions and far-field conditions allow for this functional form.

The functional form shown in (1.10.11) implies that the temperature as seen by a traveling observer located at $x = \sqrt{\kappa\, t}$, and thus moving along the x axis with velocity $\mathrm{d}x/\mathrm{d}t = \sqrt{\kappa/(4t)}$, remains constant in time.

Next, we note that

$$\frac{\partial T}{\partial t} = \frac{\mathrm{d}f}{\mathrm{d}\eta}\frac{\partial \eta}{\partial t}, \qquad \frac{\partial T}{\partial x} = \frac{\mathrm{d}f}{\mathrm{d}\eta}\frac{\partial \eta}{\partial x}, \tag{1.10.12}$$

where

$$\frac{\partial \eta}{\partial t} = -\tfrac{1}{2}\frac{1}{\kappa^{1/2}}\frac{x}{t^{3/2}}, \qquad \frac{\partial \eta}{\partial x} = \frac{1}{\sqrt{\kappa t}}. \tag{1.10.13}$$

Consequently,

$$\frac{\partial^2 T}{\partial x^2} = \frac{\partial}{\partial x}\left(\frac{\mathrm{d}f}{\mathrm{d}\eta}\frac{\partial \eta}{\partial x}\right) = \frac{\mathrm{d}^2 f}{\mathrm{d}\eta^2}\frac{1}{\kappa t}. \tag{1.10.14}$$

Substituting these expressions into the governing equation (1.10.1), we obtain

$$-\tfrac{1}{2}\,\frac{\mathrm{d}f}{\mathrm{d}\eta}\frac{1}{\kappa^{1/2}}\,\frac{x}{t^{3/2}} = \frac{\mathrm{d}^2 f}{\mathrm{d}\eta^2}\frac{1}{t}. \tag{1.10.15}$$

Rearranging, we derive a second-order *nonlinear* ordinary differential equation for $f(\eta)$,

$$\boxed{\frac{\mathrm{d}^2 f}{\mathrm{d}\eta^2} = -\tfrac{1}{2}\,\eta\,\frac{\mathrm{d}f}{\mathrm{d}\eta}.} \tag{1.10.16}$$

To compute the solution, we recast (1.10.16) into the form

$$\frac{\mathrm{d}}{\mathrm{d}\eta}\left(\ln\frac{\mathrm{d}f}{d\eta}\right) = -\tfrac{1}{2}\,\eta. \tag{1.10.17}$$

Integrating, we obtain

$$\frac{\mathrm{d}f}{\mathrm{d}\eta} = -A\,\exp(-\tfrac{1}{4}\,\eta^2), \tag{1.10.18}$$

where A is an integration constant. Carrying out a second integration, we find that

$$f(\eta) = B - A\int_0^\eta \exp(-\tfrac{1}{4}\,q^2)\,\mathrm{d}q, \tag{1.10.19}$$

where B is a new integration constant and q is an auxiliary integration variable. Setting $v \equiv \frac{1}{2}q$, we obtain

$$\boxed{f(\eta) = B - 2A\int_0^{\eta/2} \exp(-v^2)\,\mathrm{d}v.} \tag{1.10.20}$$

The integral on the right-hand side is not available in analytical form.

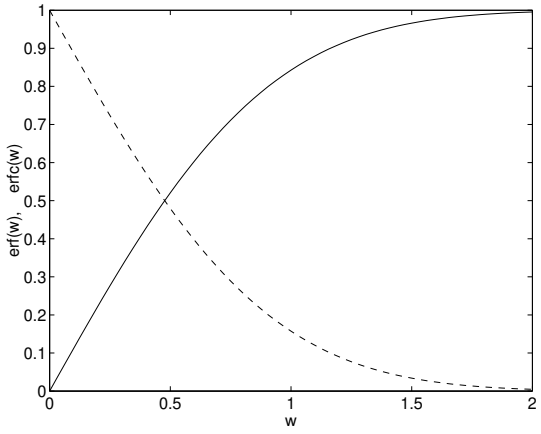


FIGURE 1.10.1 Graphs of the error function (solid line) and complementary error function (broken line) defined in equations (1.10.20) and (1.10.21).

1.10.4 Error function

To formalize the solution, we introduce the error function, defined as

$$\operatorname{erf}(w) \equiv \frac{2}{\sqrt{\pi}} \int_0^w \exp(-v^2)\, \mathrm{d}v, \tag{1.10.21}$$

and the complementary error function, defined as

$$\operatorname{erfc}(w) = 1 - \operatorname{erf}(w). \tag{1.10.22}$$

Graphs of these functions are shown in Figure 1.10.1. As w tends to infinity, the error function tends to the asymptotic value of unity; correspondingly, the complementary error function tends to zero.

In terms of the error function, the solution (1.10.20) is given by

$$f(\eta) = B - A\sqrt{\pi}\operatorname{erf}(\tfrac{1}{2}\eta), \tag{1.10.23}$$

where the constants A and B will be determined by initial and boundary conditions.

1.10.5 Step change in the boundary temperature

Now assume that the temperature has a uniform value $T_{\rm A}$ at $t = 0$ for any $x \geq 0$, and is changed to a persistent constant value $T_{\rm B}$ at the boundary, $x = 0$. Requiring that

$$T(x, t = 0) = f(\infty) = T_{\rm A} \tag{1.10.24}$$

and

$$T(x = 0, t > 0) = f(0) = T_{\rm B}, \tag{1.10.25}$$

we obtain

$$A\sqrt{\pi} = T_{\rm B} - T_{\rm A}, \qquad B = T_{\rm B}. \tag{1.10.26}$$

The temperature profile is described by

$$f(\eta) = T_{\rm B} - (T_{\rm B} - T_{\rm A})\,\mathrm{erf}(\tfrac{1}{2}\eta), \tag{1.10.27}$$

which can be restated as

$$f(\eta) = T_{\rm A} + (T_{\rm B} - T_{\rm A})\,\mathrm{erfc}(\tfrac{1}{2}\eta). \tag{1.10.28}$$

The dependencies on x and t are encapsulated in the similarity variable η.

Exercise

1.10.1 Derive the solution (1.10.18) by integrating equation (1.10.17).

1.11 Blasius boundary layer

Consider uniform viscous flow with velocity U past a semi-infinite flat plate, as shown in Figure 1.11.1. A boundary layer defined as a region where the slope of the streamwise velocity component, u_x, is substantial develops next to the plate.

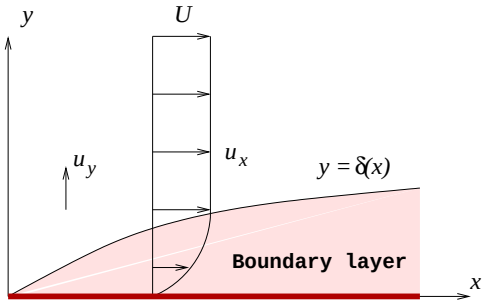


FIGURE 1.11.1 Illustration of a boundary layer developing along a semi-infinite flat plate that is held parallel to a uniform incident stream.

It can be shown that the flow inside the boundary layer is governed by a nonlinear homogeneous convection–diffusion equation for the x and y velocity components, u_x and u_y,

$$u_x \frac{\partial u_x}{\partial x} + u_y \frac{\partial u_x}{\partial y} = \nu \frac{\partial^2 u_x}{\partial y^2}, \tag{1.11.1}$$

known as the Blasius equation, where ν is a physical constant called the kinematic viscosity with units of length squared divided by time. The Blasius equation is accompanied by the continuity equation for an incompressible fluid,

$$\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} = 0. \tag{1.11.2}$$

The left-hand side of this equation is the divergence of the velocity.

To satisfy the continuity equation, we express the velocity components in terms of the stream function, ψ, defined by the equations

$$u_x = \frac{\partial \psi}{\partial y}, \qquad u_y = -\frac{\partial \psi}{\partial x}. \tag{1.11.3}$$

Substituting these expressions into the Blasius equation (1.11.1), we

obtain a third-order partial differential equation,

$$\frac{\partial \psi}{\partial y}\frac{\partial^2 \psi}{\partial x \partial y} - \frac{\partial \psi}{\partial x}\frac{\partial^2 \psi}{\partial y^2} = \nu \frac{\partial^3 \psi}{\partial y^3}. \tag{1.11.4}$$

Appropriate boundary conditions must be satisfied.

1.11.1 Similarity solution

It is evident from its definition that the stream function has units of velocity multiplied by length. The kinematic viscosity also has units of velocity multiplied by length. Based on this observation and guided by intuition, we speculate a similarity solution of equation (1.11.4) in the form

$$\psi(x, y) = \nu\, \xi\, f(\eta), \tag{1.11.5}$$

where

$$\xi \equiv \sqrt{\frac{Ux}{\nu}}, \qquad \eta \equiv \sqrt{\frac{U}{\nu x}}\, y \tag{1.11.6}$$

are two dimensionless variables. Note that ξ is a pure dimensionless variable defined in terms of x, whereas η is a similarity dimensionless variable defined in terms of x and y.

As a preliminary, we compute the derivatives

$$\frac{\partial \eta}{\partial x} = -\tfrac{1}{2}\frac{y}{x}\sqrt{\frac{U}{\nu x}} = -\tfrac{1}{2}\frac{1}{x}\eta, \qquad \frac{\partial \eta}{\partial y} = \sqrt{\frac{U}{\nu x}} = \frac{1}{y}\eta. \tag{1.11.7}$$

Moreover,

$$\frac{\partial f}{\partial x} = \frac{\mathrm{d}f}{\mathrm{d}\eta}\frac{\partial \eta}{\partial x}, \qquad \frac{\partial f}{\partial y} = \frac{\mathrm{d}f}{\mathrm{d}\eta}\frac{\partial \eta}{\partial y}, \tag{1.11.8}$$

Differentiating (1.11.5) with respect to y or x and using these relations we derive the x velocity component,

$$u_x(x, y) = \frac{\partial \psi}{\partial y} = U\sqrt{\nu\, Ux}\ \frac{\mathrm{d}f}{\mathrm{d}\eta}\frac{\mathrm{d}\eta}{\mathrm{d}y} \tag{1.11.9}$$

or

$$u_x(x,y) = U\sqrt{\nu U x}\ \frac{\mathrm{d}f}{\mathrm{d}\eta}\sqrt{\frac{U}{\nu x}}, \tag{1.11.10}$$

yielding

$$u_x(x,y) = U\,\frac{\mathrm{d}f}{\mathrm{d}\eta}, \tag{1.11.11}$$

We see that only the similarity variable η is involved in the expression for the x velocity component. This result provides justification for classifying the method as a similarity solution.

The y velocity component is given by

$$u_y(x,y) = -\frac{\partial\psi}{\partial x} = -(U\nu)^{1/2}\,\frac{\partial(\sqrt{x}\,f(\eta)\,)}{\partial x}, \tag{1.11.12}$$

yielding

$$u_y(x,y) = -\tfrac{1}{2}\sqrt{\frac{U\nu}{x}}\,f - \sqrt{\nu U x}\,\frac{\mathrm{d}f}{\mathrm{d}\eta}\,\frac{\partial\eta}{\partial x} \tag{1.11.13}$$

and then

$$u_y(x,y) = -\tfrac{1}{2}\sqrt{\frac{U\nu}{x}}\,(\,f - \eta\,\frac{\mathrm{d}f}{\mathrm{d}\eta}\,). \tag{1.11.14}$$

Further differentiation yields

$$\frac{\partial u_x}{\partial x} = U\,\frac{\mathrm{d}^2 f}{\mathrm{d}\eta^2}\,\frac{\partial\eta}{\partial x}, \qquad \frac{\partial u_x}{\partial y} = U\,\frac{\mathrm{d}^2 f}{\mathrm{d}\eta^2}\,\frac{\partial\eta}{\partial y}, \tag{1.11.15}$$

and

$$\frac{\partial^2 u_x}{\partial y^2} = U\,\frac{\mathrm{d}^3 f}{\mathrm{d}\eta^3}\left(\frac{\partial\eta}{\partial y}\right)^2. \tag{1.11.16}$$

1.11.2 Blasius equation

Substituting the preceding expressions into the boundary-layer equation (1.11.1), and simplifying, we obtain a third-order nonlinear ordinary differential equation for the function $f(\eta)$,

$$\frac{\mathrm{d}^3 f}{\mathrm{d}\eta^3} + \tfrac{1}{2}\, f\, \frac{\mathrm{d}^2 f}{\mathrm{d}\eta^2} = 0. \tag{1.11.17}$$

Enforcing the no-slip and no-penetration conditions and requiring that the flow in the boundary layer reduces to the outer uniform flow far from the plate, we obtain

$$f = 0 \quad \text{and} \quad \frac{\mathrm{d}f}{\mathrm{d}\eta} = 0 \quad \text{at} \quad \eta = 0, \tag{1.11.18}$$

and

$$\frac{\mathrm{d}f}{\mathrm{d}\eta} \to 1 \quad \text{as} \quad \eta \to \infty. \tag{1.11.19}$$

These equations provide us with boundary and far-field conditions to be used in solving the Blasius equation (1.11.17).

1.11.3 Formulation of a boundary-value problem

To solve the Blasius equation (1.11.17), it is convenient to rename $x_1 = f$, denote the first and second derivative of the function f as

$$x_2 \equiv \frac{\mathrm{d}f}{\mathrm{d}\eta}, \qquad x_3 \equiv \frac{\mathrm{d}x_2}{\mathrm{d}\eta} = \frac{\mathrm{d}^2 f}{\mathrm{d}\eta^2}, \tag{1.11.20}$$

and resolve the third-order equation into a system of three first-order nonlinear equations,

$$\frac{\mathrm{d}x_1}{\mathrm{d}\eta} = x_2, \qquad \frac{\mathrm{d}x_2}{\mathrm{d}\eta} = x_3, \qquad \frac{\mathrm{d}x_3}{\mathrm{d}\eta} = -\tfrac{1}{2}\, x_1\, x_3. \tag{1.11.21}$$

This system is accompanied by three boundary conditions,

$$x_1(\eta = 0) = 0, \quad x_2(\eta = 0) = 0, \quad x_2(\eta = \infty) = 1, \tag{1.11.22}$$

originating from (1.11.18) and (1.11.19). Since boundary conditions are provided at both ends of the solution domain with respect to η, extending from 0 to ∞, we are presented with a two-point boundary-value problem involving three first-order differential equations.

The solution can be computed using a shooting method according to the following steps:

1. Guess the value of $x_3(0) \equiv f''(\eta = 0)$.
2. Integrate equations (1.11.21) from $\eta = 0$ to $\eta = \infty$, subject to the initial conditions (1.11.18), using the value of $x_3(\eta = 0)$.
3. Check whether the far-field condition $x_3(\eta = \infty) = 1$ is satisfied. If not, improve the guess for $x_3(\eta = 0)$ and return to Step 2.

In practice, integrating up to $\eta = 10$ in Step 2 yields satisfactory accuracy. The improvement in Step 3 can be made using several a standard method for solving nonlinear algebraic equations, in including the Newton–Raphson method.

1.11.4 Newton–Raphson method

To implement the Newton–Raphson method, we observe that the initial guess for $x_3(0)$ affects the entire solution obtained in Step 2. The sensitivity of x_1, x_2, and x_3 on the initial guess is expressed by the derivatives

$$x_4(\eta) \equiv \frac{\mathrm{d}x_1(\eta)}{\mathrm{d}x_3(0)}, \quad x_5(\eta) \equiv \frac{\mathrm{d}x_2(\eta)}{\mathrm{d}x_3(0)}, \quad x_6(\eta) \equiv \frac{\mathrm{d}x_3(\eta)}{\mathrm{d}x_3(0)} \tag{1.11.23}$$

for $\eta \geq 0$. Differentiating the three differential equations given in (1.11.21) with respect to $x_3(0)$, and using the definitions of x_4, x_5, and x_6 given in (1.11.23), we obtain three additional differential equations,

$$\frac{\mathrm{d}x_4}{\mathrm{d}\eta} = x_5, \qquad \frac{\mathrm{d}x_5}{\mathrm{d}\eta} = x_6, \qquad \frac{\mathrm{d}x_6}{\mathrm{d}\eta} = -\tfrac{1}{2}\,(x_3\,x_4 + x_1 x_6). \tag{1.11.24}$$

These three equations comprise a suspended system of ordinary differential equations satisfying the initial conditions

$$x_4(\eta = 0) = 0, \qquad x_5(\eta = 0) = 0, \qquad x_6(\eta = 0) = 1. \tag{1.11.25}$$

The primary system of equations (1.11.21) is uncoupled from the that shown in (1.11.25) and integrated *a posteriori*. However, it is convenient to integrate all six equations at once.

The shooting method combined with the Newton–Raphson method involves the following steps:

1. Guess a value for $x_3(\eta = 0)$.

2. Integrate with respect to η a system of six ordinary differential equations comprised of (1.11.21) and (1.11.25) with initial conditions $x_1(\eta = 0) = 0$, $x_2(\eta = 0) = 0$, $x_4(\eta = 0) = 0$, $x_5(\eta = 0) = 0$, and $x_6(\eta = 0) = 1$.

3. Replace the current initial value, $x_3(t = 0)$, with the updated value

$$x_3(\eta = 0) \leftarrow x_3(\eta = 0) - \frac{x_2(\eta = \infty) - 1}{x_5(\eta = \infty)}, \tag{1.11.26}$$

   and repeat until convergence.

In practice, infinity is replaced by a sufficiently large value. The following Matlab function evaluates the right-hand sides of the six differential equations:

```
function dxdt = blas_fnc(x)

 dxdt = [x(2);
         x(3);
        -0.5*x(1)*x(3);
         x(5);
         x(6);
        -0.5*(x(4)*x(3)+x(6)*x(1)) ];

return
```

The method is implemented in the following Matlab code located in directory BLASIUS6 of SEMLIB. The integration is the Step 2 is carried out using the fourth-order Runge–Kutta method.

```
Nstep = 4*64;
etamax = 10.0;

deta = etamax/Nstep;
deth = 0.5*deta;

%---
% initial guess
%---

x3init = 0.277;

%---
% iterate
%---

 for iter=1:10

   x(1) = 0.0;
   x(2) = 0.0;
   x(3) = x3init;
   x(4) = 0.0;
   x(5) = 0.0;
   x(6) = 1.0;

%---
% integrate the six ODEs
%---

  eta = 0.0;

  etaa(1) = eta;

  for j=1:6
    xall(j,1) = x(j);
  end
```

```
%---
  for i=1:Nstep
%---

   for j=1:6
    xsave(j) = x(j);
   end

   xp0 = blasius6_fnc(x);

   for j=1:6
    x(j) = xsave(j) + deth*xp0(j);
   end

   xp1 = blasius6_fnc(x);

   for j=1:6
    x(j) = xsave(j) + deth*xp1(j);
   end

   xp2 = blasius6_fnc(x);

   for j=1:6
    x(j) = xsave(j) + deta*xp2(j);
   end

   xp3 = blasius6_fnc(x);

   for j=1:6
     vel = (xp0(j)+2.0*xp1(j)+2.0*xp2(j)+xp3(j))/6.0;
     x(j) = xsave(j) + deta*vel;
   end

   eta = eta + deta;
   etaa(i+1) = eta;

   for j=1:6
    xall(j,i+1) = x(j);
```

```
   end

%---
  end
%---

  residual = x(2)-1.0;
  x3init = x3init-residual/x(5);
  if(abs(residual)<0.00000001) break; end

%---
% end of iterations
%---

 end

%---
% converged value for x3(0)
%---

format long
[residual x3init]
format short

%---
% plot the solution
%---

figure(1)
hold on
plot(xall(1,:),etaa,'k--')
plot(xall(2,:),etaa,'r-')
plot(xall(3,:),etaa,'k-.')
ylabel('\eta','fontsize',15)
```

Running the code generates the following output for the residual and

```
  -0.00000000000203   0.33205733741221
```

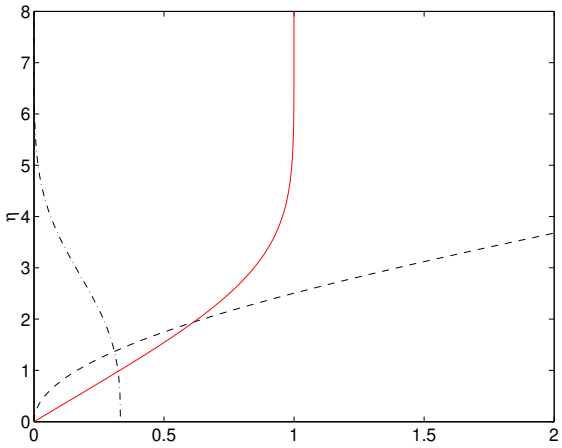


FIGURE 1.11.2 Graphs of the Blasius self-similar streamwise velocity profile $u/U = f'$ (dashed line), its integral f (solid line), and derivative f'' (dot-dashed line).

The first value is the residual and the second value is $f''(0)$ at the end of the iterations, accurate to the eighth decimal place. The streamwise velocity profile, $u/U = f' \equiv \mathrm{d}f/\mathrm{d}\eta$, is shown in in Figure 1.11.2 along with the profiles of f and $f'' \equiv \mathrm{d}^2 f/\mathrm{d}\eta^2$.

Exercise

1.11.1 Discuss the choice of boundary conditions in the boundary-layer solution.

Chapter 2

Mathematical modeling

Mathematical modeling involves writing equations that describe the state or govern the evolution of physical or engineering systems with varying complexity. A complete set of equations can be solved by analytical or numerical methods subject to boundary, far-field, and initial conditions.

Solving the vast majority of problems in science and engineering requires the use of numerical methods for all but the simplest configurations. Even if an analytical solution is possible, numerical evaluation is invariably necessary in the end.

2.1 Drainage of a tank

Consider the drainage of a fluid with low viscosity from a tank through a hole at the bottom, as shown in Figure 2.1.1. The exit velocity can be computed from Bernoulli's equation for irrotational flow at quasi-steady state, resulting in Torricelli's law,

$$U = \left(2 \frac{\Delta p}{\rho} + 2gh \right)^{1/2}, \tag{2.1.1}$$

where h is the instantaneous liquid height inside the tank, $\Delta p = p_1 - p_0$ is a pressure difference, p_0 is the ambient pressure at the point of discharge, and p_1 is the gas pressure above the liquid in the container, as shown in Figure 2.1.1.

2.1.1 Mass balance

Torricelli's law can be used to compute the time it takes for a fluid to drain from a tank with a given geometry. During a small period of

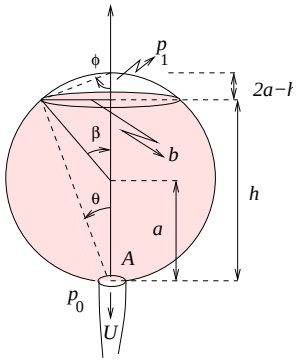


FIGURE 2.1.1 Illustration of the gravitational drainage of a fluid from a spherical tank.

time, $\mathrm{d}t$, the volume of liquid in the tank decreases by

$$\mathrm{d}V = S(h)\,\mathrm{d}h, \tag{2.1.2}$$

where $S(h)$ is the tank cross-sectional area. Setting this change in volume equal to $-UA\,\mathrm{d}t$, where A is the cross-sectional area of the drainage hole, we obtain

$$S(h)\,\mathrm{d}h = -UA\,\mathrm{d}t. \tag{2.1.3}$$

Both sides of this equation are negative. Substituting into the right-hand side Torricelli's law, we obtain

$$S(h)\,\mathrm{d}h = -\Big(2\,\frac{\Delta p}{\rho} + 2gh\Big)^{1/2} A\,\mathrm{d}t, \tag{2.1.4}$$

where Δp may be a function of h.

For a spherical tank,

$$S(h) = \pi b^2, \tag{2.1.5}$$

where the square of the cross-sectional radius, b, defined in Figure 2.1.1, is given by

$$b^2 = h\,(2a - h). \tag{2.1.6}$$

To derive this equation, we note that

$$\tan\theta = \frac{b}{h}, \qquad \tan\phi = \frac{b}{2a-h}, \tag{2.1.7}$$

where the angles θ and ϕ are defined in Figure 2.1.1. Having a common side and two perpendicular sides, drawn with dashed lines in Figure 2.1.1, these two angles have to add to $\frac{1}{2}\pi$. Because $\theta + \phi = \frac{1}{2}\pi$, we have that $\tan\theta\tan\phi = 1$, yielding (2.1.6).

Substituting (2.1.5) into (2.1.4) and rearranging, we obtain

$$\frac{\mathrm{d}t}{\mathrm{d}h} = -\frac{\pi}{\sqrt{2}}\frac{\sqrt{\rho}}{A}\frac{h\,(2a-h)}{\sqrt{\Delta p + \rho g h}}. \tag{2.1.8}$$

Further progress requires a specification of the pressure difference, Δp.

2.1.2 *Driving pressure*

The pressure difference, Δp, can either be constant or change in time independently or according to the liquid height, h. In the case of a closed tank occupied by a gas above the liquid, the pressure above the liquid, p_1, changes in time according to its volume, as determined by an equation of state,

$$p_1 V_g^{\gamma} = q, \tag{2.1.9}$$

where γ is an appropriate exponent and q is a constant determined by the initial condition. The volume of the gas above the tank is given by

$$V_g = \pi\,\tfrac{1}{3}\,(2 - 3\cos\beta + \cos^3\beta)\,a^3, \tag{2.1.10}$$

where $\cos\beta = (h-a)/a$ and the angle β is shown in Figure 2.1.1. In the case of a closed tank occupied by the liquid vapor above the liquid, the pressure p_1 is constant.

2.1.3 *Separation of variables*

When Δp depends on time only implicitly through $h(t)$, the absence of explicit time dependence on the left-hand side of (2.1.8) allows us to

solve this ordinary differential equation by separation of variables based on the following rearrangement:

$$\mathrm{d}t = -\frac{\pi}{\sqrt{2}}\frac{\sqrt{\rho}}{A}\frac{h\,(2a-h)}{\sqrt{\Delta p(h)+\rho g h}}\,\mathrm{d}h. \tag{2.1.11}$$

Integrating, we obtain an expression for the drainage time from an initial height, h_0, to a specified height, h,

$$t_{\rm drain}(h) = \frac{\pi}{\sqrt{2}}\frac{\sqrt{\rho}}{A}\int_h^{h_0}\frac{w\,(2a-w)}{\sqrt{\Delta p(w)+\rho g w}}\,\mathrm{d}w, \tag{2.1.12}$$

where w is an integration variable. Rearranging the integral, we find

$$t_{\rm drain}(h) = \frac{\pi}{\sqrt{2}}\frac{a^{5/2}}{A\sqrt{g}}\int_{\hat h}^{\hat h_0}\frac{v\,(2-v)}{\sqrt{c(v)+v}}\,\mathrm{d}v, \tag{2.1.13}$$

where $v = w/a$,

$$c \equiv \frac{\Delta p}{\rho g a} \tag{2.1.14}$$

is a dimensionless pressure difference, $\hat h_0 = h_0/a$, and $\hat h = h/a$. Once $c(v)$ has been specified, the integral in (2.1.13) can be computed by numerical methods.

When c is a constant, the integration on the right-hand side can be performed readily using the formulas

$$\int \frac{v}{\sqrt{c+v}}\,\mathrm{d}v = \Phi_1(v) \equiv \frac{2}{3}\,(v-2\,c)\sqrt{c+v} \tag{2.1.15}$$

and

$$\int \frac{v^2}{\sqrt{c+v}}\,\mathrm{d}v = \Phi_2(v) = \frac{2}{15}\,(8\,c^2-4\,cv+3\,v^2)\sqrt{c+v}. \tag{2.1.16}$$

We find that

$$t_{\rm drain}(\hat h) = \pi\,\frac{a^{5/2}}{A\sqrt{2g}}\,\big(\,2(\Phi_1(\hat h)-\Phi_1(\hat h_0)) - 2(\Phi_2(\hat h)-\Phi_2(\hat h_0)). \tag{2.1.17}$$

In the case of completely filled tank, $\hat{h}_0 = 2$.

When $\Delta p = 0$, corresponding to $c = 0$, we derive the simple formula

$$t_{\rm drain}(\hat{h}) = \frac{a^2}{A}\sqrt{\frac{a}{g}}\,\frac{\sqrt{2}\pi}{15}\,\mathcal{F}(\hat{h},\hat{h}_0), \tag{2.1.18}$$

where

$$\mathcal{F}(\hat{h},\hat{h}_0) = 10\,(\hat{h}_0^{3/2} - \hat{h}^{3/2}) - 3\,(\hat{h}_0^{5/2} - \hat{h}^{5/2}). \tag{2.1.19}$$

The time for complete drainage arises by setting $h = 0$.

Exercises

2.1.1 Derive a formula for the drainage time of a vertical cylindrical tank.

2.2 Collapse of a fluid carrying tube

The flow rate along the x axis through a tube with elliptical cross-section is given by

$$Q = -\frac{\pi}{4\mu}\,\frac{{\rm d}p}{{\rm d}x}\,\frac{a^3b^3}{a^2+b^2}, \tag{2.2.1}$$

where $-{\rm d}p/{\rm d}x$ is the negative of the pressure gradient along the tube length, μ is the fluid viscosity, and a and b are the ellipse semi-axes.

2.2.1 Poiseuille's law

When $a = b$ we obtain Poiseuille's law for for a circular tube,

$$Q = -\frac{\pi a^4}{8\mu}\,\frac{{\rm d}p}{{\rm d}x}. \tag{2.2.2}$$

We observe that the flow rate is proportional to the fourth power of the tube radius.

2.2.2 Transmural pressure

As expected on physical grounds, the pressure decreases along the tube length due to the fluid viscosity, while the flow rate, Q, remains constant at steady state. The transmural pressure is defined as

$$\Delta p \equiv p_{\text{outer}} - p, \tag{2.2.3}$$

where p_{outer} is the ambient pressure. If the tube wall is deformable, a negative transmural pressure causes the tube to collapse initially into an ellipse and eventually into a narrow slit. By contrast, a positive transmural pressure causes dilation as the tube wall pushes against the surroundings.

From https://en.wikipedia.org/wiki/Collapsible_flow: "*Venous collapse is important during exercise, when muscular compression of leg veins is used to pump blood against gravity up to the heart, and in therapeutic compression of leg veins for the treatment of deep-vein thrombosis.*"

2.2.3 Aspect ratio v. transmural pressure

We may assume that the tube retains an elliptical shape whose aspect ratio, b/a, is a function of the transmural pressure, Δp, as determined by the wall material,

$$\gamma(x) \equiv \frac{b}{a} = \mathcal{F}(\Delta p), \tag{2.2.4}$$

where $\mathcal{F}$ is a collapse function of the transmural pressure. If the tube wall is made of an inextensible material, the total arc length of the tube perimeter, $\mathcal{L}(a, b)$, remains constant along a collapsing tube.

Collapse occurs when Δp becomes higher than a critical value for collapse, denoted by Δp_c. For example, according to the theory of thin shells, $\Delta p_c = 3\,\kappa_{\text{B}}/a^3$, where κ_{B} is the modulus of bending of a circular shell of radius a, as discussed in Section 2.5.

A reasonable collapse function for a tube with a circular uncollapsed cross-section is

$$\mathcal{F} = \begin{cases} 1 & \text{if} \quad \Delta p < \Delta p_c, \\ 1 - \alpha\, \tanh\left(\beta\,(\frac{\Delta p}{\Delta p_c} - 1)\right) & \text{if} \quad \Delta p > \Delta p_c, \end{cases} \tag{2.2.5}$$

where α and β are dimensionless coefficient. As Δp increases, the aspect ratio tends to $1-\alpha$.

2.2.4 *Pressure distribution*

Rearranging (2.2.1), we obtain

$$\frac{\mathrm{d}p}{\mathrm{d}x} = -\tfrac{4}{\pi}\,\mu\,Q\,\frac{1}{a^4}\,\frac{1+\gamma^2}{\gamma^3}, \tag{2.2.6}$$

where a and γ are functions of x. Integrating, we obtain

$$p(x) = -\tfrac{4}{\pi}\,\mu\,Q\int_0^x \frac{1}{a^4}\,\frac{1+\gamma^2}{\gamma^3}\,\mathrm{d}x' + p_0, \tag{2.2.7}$$

where p_0 is the initial pressure.

2.2.5 *Perimeter of an ellipse*

Parametrizing the elliptical shape as $x = a\cos\eta$ and $y = b\sin\eta$, we find that the differential arc length along the contour of an ellipse is given by

$$\mathrm{d}\ell \equiv \sqrt{\mathrm{d}x^2+\mathrm{d}y^2} = \sqrt{a^2\sin^2\eta + b^2\,\cos^2\eta}\,\mathrm{d}\eta \tag{2.2.8}$$

for $0 \le \eta \le 2\pi$, which can be restated as

$$\mathrm{d}\ell = \sqrt{a^2-(a^2-b^2)\,\cos^2\eta}\,\mathrm{d}\eta = a\,\sqrt{1-k^2\cos^2\eta}\,\mathrm{d}\eta, \tag{2.2.9}$$

where

$$k^2 \equiv 1-\frac{b^2}{a^2} = 1-\gamma^2. \tag{2.2.10}$$

For convenience and without loss of generality, we have assumed that $b \le a$.

The arc length of the perimeter of the ellipse is given by

$$\mathcal{L} = \oint \mathrm{d}\ell = 4\,a\int_0^{\pi/2}\sqrt{1-k^2\cos^2\eta}\,\mathrm{d}\eta \tag{2.2.11}$$

or

$$\mathcal{L} = 4a \int_0^{\pi/2} \sqrt{1 - k^2 \sin^2\eta}\, \mathrm{d}\eta = 4a\, \mathrm{E}(k), \tag{2.2.12}$$

where

$$\mathrm{E}(k) \equiv \int_0^{\pi/2} \sqrt{1 - k^2 \sin^2\eta}\, \mathrm{d}\eta \tag{2.2.13}$$

is the *complete elliptic integral of the second kind*. When $a = b$, we find that $k^2 = 0$ and recover the well-known result for the perimeter of a circle, $L = 2\pi a$.

We have found that

$$a = \frac{\mathcal{L}}{4\,\mathrm{E}(k)}, \tag{2.2.14}$$

where $\mathcal{L}$ is the fixed perimeter of the ellipse.

Substituting (2.2.14) into (2.2.6), we obtain

$$\frac{\mathrm{d}p}{\mathrm{d}x} = -\frac{1024}{\pi} \frac{\mu\, Q}{\mathcal{L}^4} \mathcal{F}(\gamma), \tag{2.2.15}$$

where

$$\mathcal{F}(\gamma) \equiv \mathrm{E}^4(\sqrt{1-\gamma^2})\, \frac{1+\gamma^2}{\gamma^3} \tag{2.2.16}$$

and γ is a function of Δp according to (2.2.3). The function $\mathcal{F}(\gamma)$ takes a minimum value at $\gamma = 1$, which shows that minimum pressure drop for a certain flow rate is achieved for a circular tube.

2.2.6 Computation of complete elliptic integrals

The complete elliptic integrals can be computed efficiently by numerical approximation or iteration. An efficient method is based on the following expressions involving rapidly converging infinite products:

$$\mathrm{F}(k) = \tfrac{1}{2}\pi\,(1 + K_1)(1 + K_2)\cdots, \tag{2.2.17}$$

and

$$\mathrm{E}(k) = \mathrm{F}(k)\,(1 - \tfrac{1}{2}\,k^2\,P), \tag{2.2.18}$$

where

$$P = 1 + \tfrac{1}{2}\,K_1\big[\,1 + \tfrac{1}{2}\,K_2\,(1 + \tfrac{1}{2}\,K_3 \ldots)\,\big]. \tag{2.2.19}$$

The sequence K_p is computed using the recursive formula

$$K_p = \frac{1 - \sqrt{1 - K_{p-1}^2}}{1 + \sqrt{1 - K_{p-1}^2}}, \tag{2.2.20}$$

starting with $K_0 = k$. Matlab implements the computation of complete elliptic integrals in the internal function *ellipke*.

2.2.7 *Code for the evaluation of complete elliptic integrals*

The iterative method is implemented in the following Matlab function named *ell_int*, located in directory COLLAPSE of SEMLIB:

```
function [F,E] = ell_int(K)

%-------------------------------------------
% Complete elliptic integrals of the first
% and second kind
%-------------------------------------------

tol = 0.0000000000001;       % tolerance

%--------
% prepare
%--------

F = 0.5*pi; P = 1.0; G = 1.0; B = K;

D = 10.0*tol;

%-----------
```

```
% iterations
%-----------

while (abs(D)>tol)
  C = sqrt(1.0-B*B);
  B = (1.0-C)/(1.0+C);
  D = F*B;
  F = F+D;
  G = 0.50*G*B;
  P = P+G;
end

E = F*(1.0-0.5*K*K*P);

%-----
% done
%-----

return
```

Graphs of the complete elliptic integrals in their domain of definition are shown in Figure 2.2.1. Note that, as k tends to unity, $\mathrm{E}(k)$ remains finite but $\mathrm{F}(k)$ diverges to infinity.

2.2.8 Numerical integration of an ode

The ordinary differential equation (2.2.15) can be integrated with respect to x subject to an initial condition, $p(x = 0) = p_0$, using a standard numerical method. Examples are Euler's method, the modified Euler method, or a Runge–Kutta method.

Euler's method involves computing $\mathrm{d}p/\mathrm{d}x$, and then advancing the pressure, p, from a station, x, to a station $x+\mathrm{d}x$, where $\mathrm{d}x$ is a specified step. The aspect ratio and other variables of interest are updated after advancement.

The method is implemented in the following Matlab code named *collapse*, located in directory COLLAPSE of SEMLIB:

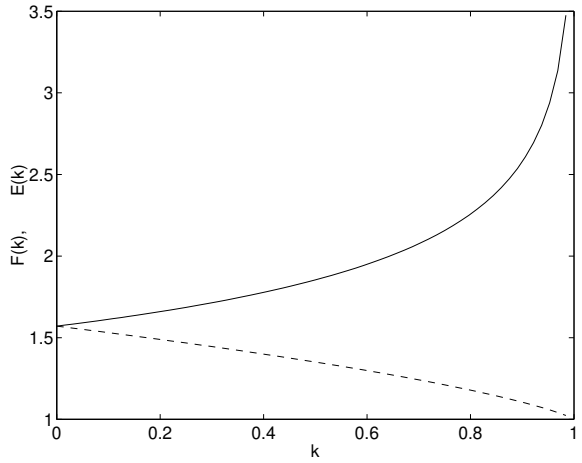


FIGURE 2.2.1 Graphs of the complete elliptic integral of the first kind, $\mathrm{F}(k)$ (solid line), and second kind, $\mathrm{E}(k)$ (dashed line).

```
%---
% data
%---

chi0 = 1.0;
mu = 1.0;
a = 1.0;   % initial radius
alpha = 0.8;
beta = 0.01;
Pouter = 1.0;
pc = 0.0;
Dx = 0.2*a;
gamma = 1.0;
nsteps = 4*32;

%---
% prepare
%---

L = 2*pi*a;
b = a;
```

```
Q = pi/4*chi0/mu*a^4;

p0 = pc;

Dpc = Pouter-pc;
Dp0 = Pouter-p0;

p = p0;
Dp = Pouter-p;

%---
% prepare for drawing cross-sections
%---

ndiv = 16;
Deta = 2*pi/ndiv;

for i=1:ndiv+1
 eta(i) = (i-1)*Deta;
 cseta(i) = cos(eta(i));
 sneta(i) = sin(eta(i));
end

%---
% integrate in x
%---

x = 0.0;
fc = -4/pi * mu*Q;

for ns=1:nsteps

 dpdx = fc/a^4 * (1+gamma^2)/gamma^3;

 x = x+Dx;
 p = p + dpdx*Dx;
 Dp = Pouter-p;
 E = ell_int(sqrt(1-gamma^2));
 gamma = 1.0 - alpha*tanh(beta*(Dp/Dpc-1));
```

```
 a = L/(4*E);
 b = a*gamma;

 if(gamma<0.01) break; end;

 plotx(ns)=x;
 plotgamma(ns)=gamma;

 for i=1:ndiv+1
  xprof(ns,i) = x;
  yprof(ns,i) = a*cseta(i);
  zprof(ns,i) = b*sneta(i);
 end

%---
end % of integration
%---

nsteps = ns-1;

%---
% plot profiles
%---

 figure(1)
 surf(xprof,yprof,zprof)

%---
% another plot
%---

 figure(2)
 plot(plotx, plotgamma,'-k')
```

Running the code generates the graphs and depiction shown in Figure 2.2.2.

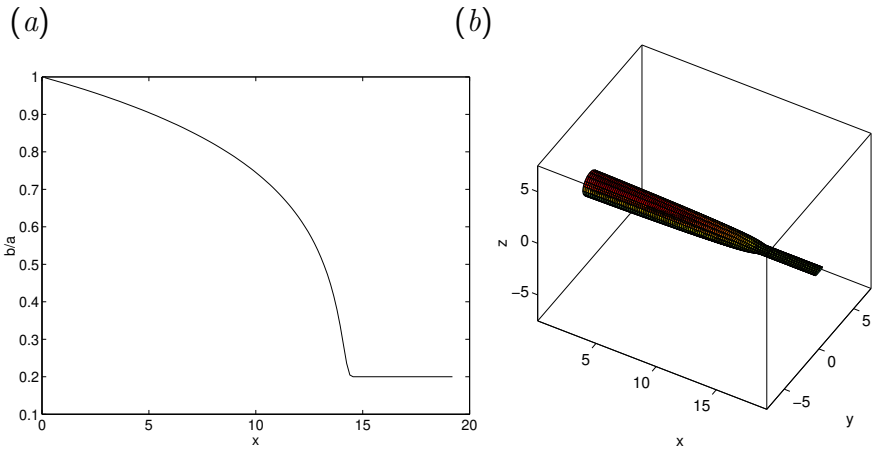


FIGURE 2.2.2 Flow in a collapsing tube with elliptical cross-section due to a positive transmural pressure. (a) Evolution of the aspect ratio with downstream position and (b) tube shape.

Exercises

2.2.1 Discuss the physical relevance of the parameter β involved in the collapse function $\mathcal{F}$.

2.2.2 Run the code given in the text to assess the effect of the parameter β.

2.3 Buckled shape of a cylindrical shell

We will show that the buckled shape of a cylindrical shell with a circular undeformed shape of radius a is governed by the following second-order differential equation for the curvature, κ,

$$\frac{\mathrm{d}^2\kappa}{\mathrm{d}\ell^2} = -\tfrac{1}{2}\,\kappa\,\frac{a^2\kappa^2 + c}{a^2} + \frac{\Delta p}{\kappa_{\mathrm{B}}}, \tag{2.3.1}$$

where ℓ is the arc length around the shell perimeter, c is a dimensionless constant,

$$\Delta p \equiv p_{\mathrm{outer}} - p_{\mathrm{inner}} \tag{2.3.2}$$

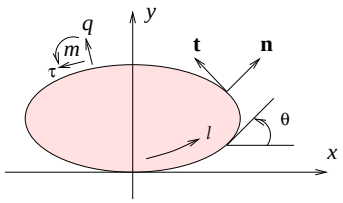


FIGURE 2.3.1 Illustration of a cylindrical shell deforming under the action of a pressure difference between the exterior and interior.

is the transmural pressure, and $\kappa_{\rm B}$ is a modulus of bending, as shown in Figure 2.3.1.

2.3.1 Curvature

The curvature is defined by the Frenet–Serret relations

$$\frac{\mathrm{d}\mathbf{t}}{\mathrm{d}\ell} = -\kappa\,\mathbf{n}, \qquad \frac{\mathrm{d}\mathbf{n}}{\mathrm{d}\ell} = \kappa\,\mathbf{t}, \tag{2.3.3}$$

where $\mathbf{t}$ and $\mathbf{n}$ are tangential and normal unit vectors. Projecting the first of these equations onto $\mathbf{n}$ and the second equation into $\mathbf{t}$, we find that

$$\kappa = -\frac{\mathrm{d}\mathbf{t}}{\mathrm{d}\ell}\cdot\mathbf{n} = \frac{\mathrm{d}\mathbf{n}}{\mathrm{d}\ell}\cdot\mathbf{t}. \tag{2.3.4}$$

Regarding that x and y coordinates around the shell as functions of arc length, ℓ, we find that

$$\kappa = -x''y' + x'y'' = -\frac{x''}{y'} = \frac{y''}{x'}, \tag{2.3.5}$$

where a prime denotes a derivative with respect to ℓ. To derive the second and fourth expression, we have noted that $x'^2 + y'^2 = 1$, and therefore $x'x'' = -y'y''$.

The dimensionless constant c and the curvature at the designated origin of arc length, κ_0, are to be found as part of the solution. For a circular shell of radius a, $\kappa_0 = 1/a$ and

$$c = 2\,\frac{a^3}{\kappa_{\rm B}}\,\Delta p - 1, \tag{2.3.6}$$

which arises by setting the right-hand side of (2.3.1) to zero.

2.3.2 Derivation of the equilibrium equation

To derive the equilibrium equation (2.3.1), we consider the differential force balance

$$\frac{\mathrm{d}}{\mathrm{d}\ell}(\tau\,\mathbf{t}+q\,\mathbf{n})-\Delta p\,\mathbf{n}=0, \tag{2.3.7}$$

where τ is the in-plane tension and q is the transverse shear tension given by

$$q=\frac{\mathrm{d}m}{\mathrm{d}\ell}, \tag{2.3.8}$$

and m is a bending moment. We will adopt the linear constitutive equation

$$m=\kappa_{\mathrm{B}}\,(\kappa-\frac{1}{a}) \tag{2.3.9}$$

where $1/a$ is the curvature in the unstressed configuration.

Resolving (2.3.7) into tangential and normal components and using the Frenet-Serret relations, we obtain

$$\frac{\mathrm{d}\tau}{\mathrm{d}\ell}+q\,\kappa=0, \qquad -\tau\,\kappa+\frac{\mathrm{d}q}{\mathrm{d}\ell}-\Delta p=0. \tag{2.3.10}$$

Expressing q in terms of the curvature, we obtain

$$\frac{\mathrm{d}\tau}{\mathrm{d}\ell}+\tfrac{1}{2}\,\kappa_{\mathrm{B}}\,\frac{\mathrm{d}\kappa^2}{\mathrm{d}\ell}=0 \tag{2.3.11}$$

and

$$-\tau\,\kappa+\kappa_{\mathrm{B}}\frac{\mathrm{d}^2\kappa}{\mathrm{d}\ell^2}-\Delta p=0. \tag{2.3.12}$$

Integrating equation (2.3.11), we obtain

$$\tau=-\tfrac{1}{2}\,\kappa_{\mathrm{B}}\,\big(\kappa^2+\frac{c}{a^2}\big). \tag{2.3.13}$$

Substituting this expression into (2.3.12) yields the governing equation (2.3.1).

2.3.3 Symmetry constraints

The shell profile slope angle θ shown in Figure 2.3.1 is defined in terms of the curvature, κ, and the arc length, ℓ, by the relation

$$\frac{\mathrm{d}\theta}{\mathrm{d}\ell} = \kappa. \tag{2.3.14}$$

For a shell with m-fold symmetry, defined such that $m = 2$ describes an ellipse or a biconcave disk, we require that

$$f_1(c, \kappa_0) \equiv \theta(\ell = \frac{2\pi a}{m}) - \frac{2\pi}{m} = 0 \tag{2.3.15}$$

and

$$f_2(c, \kappa_0) \equiv \kappa(\ell = \frac{2\pi a}{m}) - \kappa_0 = 0. \tag{2.3.16}$$

These constraints provide us with a system of two nonlinear algebraic equations for c and κ_0 whose roots can be found using Newton's method discussed in Section 2.3.3.

2.3.4 Buckling

The governing equation for the curvature, equation (2.3.1), is a nonlinear second-order ordinary differential equation. Linearizing about the circular shape, we set

$$\kappa = \frac{1}{a} + \hat{\kappa}, \qquad c = 2\,a^3\,\frac{\Delta p}{\kappa_B} - 1 + \hat{c}, \tag{2.3.17}$$

where a hat denotes a perturbation, and obtain

$$\frac{\mathrm{d}^2\hat{\kappa}}{\mathrm{d}\ell^2} = -(\frac{1}{a} + \hat{\kappa})\,\frac{\hat{\kappa}}{a} - \tfrac{1}{2}\,\frac{1}{a^2}\,(\frac{1}{a} + \hat{\kappa})\,(2\,a^3\,\frac{\Delta p}{\kappa_B} + \hat{c}) + \frac{\Delta p}{\kappa_B}. \tag{2.3.18}$$

Linearizing by neglecting products of hatted variables, we obtain

$$\frac{\mathrm{d}^2\hat{\kappa}}{\mathrm{d}\ell^2} = -\frac{1}{a^2}\,\big(\,1 + a^3\,\frac{\Delta p}{\kappa_B}\,\big)\hat{\kappa} - \tfrac{1}{2}\,\frac{1}{a^3}\,\hat{c}. \tag{2.3.19}$$

The solution is $\hat{c} = 0$ and $\hat{\kappa} = A\cos(\lambda\ell/a)$, where A is an arbitrary constant and

$$\lambda^2 \equiv 1 + a^3 \frac{\Delta p}{\kappa_B}. \tag{2.3.20}$$

The inherent periodicity condition with respect to ℓ requires that $\lambda = m$, yielding

$$a^3 \frac{\Delta p}{\kappa_B} = m^2 - 1, \tag{2.3.21}$$

where m is an integer. The first buckling mode corresponds to $m = 2$.

2.3.5 *Extended system of governing equations*

To solve the governing equations using a numerical method, we make guesses for c and κ_0, integrate the ordinary differential equation (2.3.1) from $\ell = 0$ to $\ell = \pi a/m$, and update the guesses according to the Newton–Raphson method.

The shell shape in the xy plane and the slope angle θ defined in Figure 2.3.1 can be generated simultaneously by replacing equation (2.3.1) with a system of seven differential equations. For an inextensible shell such as a garden hose, we define

$$\pi_1 = \kappa(\ell), \qquad \pi_3 = x(\ell), \qquad \pi_4 = y(\ell), \qquad \pi_7 = \theta(\ell), \tag{2.3.22}$$

and obtain the equations

$$\frac{\mathrm{d}\pi_1}{\mathrm{d}\ell} = \pi_2, \qquad \frac{\mathrm{d}\pi_2}{\mathrm{d}\ell} = -\tfrac{1}{2}\,\kappa\,\frac{a^2\pi_1^2 + c}{a^2} + \frac{\Delta p}{\kappa_B}, \qquad \frac{\mathrm{d}\pi_3}{\mathrm{d}\ell} = \pi_5,$$

$$\frac{\mathrm{d}\pi_4}{\mathrm{d}\ell} = \pi_6, \qquad \frac{\mathrm{d}\pi_5}{\mathrm{d}\ell} = -\pi_1\pi_6, \qquad \frac{\mathrm{d}\pi_6}{\mathrm{d}\ell} = \pi_1\pi_5, \qquad \frac{\mathrm{d}\pi_7}{\mathrm{d}\ell} = \pi_1. \tag{2.3.23}$$

The first two of these equations implement the governing equation (2.3.1) for the curvature. The third through the sixth equations implement the relations

$$x'' = -\kappa\, y', \qquad y'' = \kappa\, x', \tag{2.3.24}$$

originating from (2.3.5), where a prime denotes a derivative with respect to ℓ. The last equation is the definition of the slope angle θ.

The right-hand sides of equations (2.3.23) comprise the components of the *phase space velocity*. The following Matlab code, located in directory SHELL of SEMLIB, returns these velocities:

```
function f = shell_vel (a,c,Dp,kappaB,x)
   f(1) = x(2);
   f(2) = -0.5*x(1)*(a^2*x(1)^2 + c)/a^2 + Dp/kappaB;
   f(3) = x(5);
   f(4) = x(6);
   f(5) = -x(1)*x(6);
   f(6) =  x(1)*x(5);
   f(7) =  x(1);
return
```

For a horizontal shell, as shown in Figure 2.3.1, the initial condition corresponding to $\ell = 0$ is

$$\begin{aligned} &\pi_1 = \kappa_0, \quad \pi_2 = 0, \quad \pi_3 = 0, \quad \pi_4 = 0, \\ &\pi_5 = 1, \quad \pi_6 = 0, \quad \pi_7 = 0, \end{aligned} \tag{2.3.25}$$

where κ_0 is the *a priori* unknown curvature at $\ell = 0$.

2.3.6 *Runge–Kutta integration*

The following Matlab function, located in directory SHELL of SEMLIB, integrates equations (2.3.23) using the fourth-order Runge–Kutta method:

```
function sarray = shell_ode(a,c,kappa0,Dp,x ...
                            ,kappaB,Nstp,Dl)
%---
% prepare
%---

 Dlh = 0.5*Dl;

%---
```

```
% initial condition
%---

  x(1) = kappa0;  x(2) = 0.0;  x(3) = 0.0;  x(4) = 0.0;
  x(5) = 1.0;  x(6) = 0.0;  x(7) = 0.0;

  for j=1:7
    sarray(j,1) = x(j);
  end

%---
% integrate
%---

 for i=1:Nstp

      al = (i-1.0)*Dl;

      for j=1:7
       xsave(j) = x(j);
      end

      g0 = shell_vel(a,c,Dp,kappaB,x);

      for j=1:7
        x(j) = xsave(j) + Dlh*g0(j);
      end

      g1 = shell_vel(a,c,Dp,kappaB,x);

      for j=1:7
        x(j) = xsave(j) + Dlh*g1(j);
      end

      g2 = shell_vel(a,c,Dp,kappaB,x);

      for j=1:7
        x(j) = xsave(j) + Dl*g2(j);
      end
```

```
      g3 = shell_vel(a,c,Dp,kappaB,x);

      for j=1:7
        vel = (g0(j) + 2.0*g1(j) + 2.0*g2(j) + g3(j))/6.0;
        x(j) = xsave(j) + Dl*vel;
      end

      for j=1:7
       sarray(j,i+1) = x(j);
      end
 end

return
```

2.3.7 *Newton's method*

The initial curvature, κ_0, and the constant c, must be found simultaneously as part of the solution. The following Matlab code, located in directory SHELL of SEMLIB, performs the Newton iterations, as discussed in Section 2.3.2, computes an effective shape aspect ratio, and displays buckled shell profiles:

```
%---
% parameters
%---

a = 1;
kappaB = 1.0;
Nstp = 2*64;   % number of integration steps
Niter = 50;  % number of Newton iterations
mode = 2;    % buckling mode
eps = 0.001;  % for the Jacobian
tol = 0.000001; % iteration tolerance

%---
% prepare
%---
```

```
Dl = 2*pi*a/(mode*Nstp);

figure(1)
box on

p = 3.2;
fcc =  0.8195;
fck = 0.1879;

c_circle = 2*Dp*a^3/kappaB - 1.0;
kappa0_circle = 1.0/a;

c = fcc*c_circle;
kappa0 = fck*kappa0_circle;

%----------------------------
% start the Newton iterations
%----------------------------

   for Iter=1:Niter

    sarray = shell_ode(a,c,kappa0,Dp ...
                          ,kappaB,Nstp,Dl);
    f(1) = sarray(7,Nstp+1)-2*pi/mode;
    f(2) = sarray(1,Nstp+1)-kappa0;

%---------------------
% compute the Jacobian
% by numerical differentiation
%---------------------

    c = c+eps;      % perturb
    sarray = shell_ode(a,c,kappa0,Dp ...
                      ,kappaB,Nstp,Dl);
    c = c-eps;      % reset
    f1(1) = sarray(7,Nstp+1)-2*pi/mode;
    f1(2) = sarray(1,Nstp+1)-kappa0;
    Jac11 = (f1(1)-f(1))/eps;
    Jac21 = (f1(2)-f(2))/eps;
```

```
    kappa0 = kappa0+eps;      % perturb
    sarray = shell_ode(a,c,kappa0,Dp ...
                      ,kappaB,Nstp,Dl);
    kappa0 = kappa0-eps;      % reset
    f1(1) = sarray(7,Nstp+1)-2*pi/mode;
    f1(2) = sarray(1,Nstp+1)-kappa0;
    Jac12 = (f1(1)-f(1))/eps;
    Jac22 = (f1(2)-f(2))/eps;

%---
% solve the equation: Jac . Dx = - f
% for the correction vector Dx
% by Cramer's rule
%---

      b1  = -f(1);
      b2  = -f(2);
      Det = Jac11*Jac22-Jac12*Jac21;
      dc = (b1*Jac22-Jac12*b2)/Det;
      dkappa0 = (b2*Jac11-Jac21*b1)/Det;

%--------
% correct
%--------

     c = c + dc;
     kappa0 = kappa0 + dkappa0;

%-------
% escape
%-------

      iescape = 1;
      if(abs(dc) > tol) iescape = 0; end
      if(abs(dkappa0) > tol) iescape = 0; end

      if(iescape==1)
       Iflag = 0;
```

```
       sarray = shell_ode(a,c,kappa0,Dp ...
                          ,kappaB,Nstp,Dl);
       f(1) = sarray(7,Nstp+1)-2*pi/mode;
       f(2) = sarray(1,Nstp+1)-kappa0;
       break
      end

%----
 end  % of iterations
%----

% shape aspect ratio

 xs = 0.0;
 ys = 0.0;

 xshift = 0.0;
 yshift = 0.5*(sarray(4,Nstp+1)-sarray(4,1));

 for i=1:Nstp
  xs = xs + (sarray(3,i)-xshift)^2;
  ys = ys + (sarray(4,i)-yshift)^2;
 end

 aspect = sqrt(ys/xs);
 Dpplot = Dp*a^3/kappaB;

 plot( sarray(3,:),sarray(4,:),'k')
 plot(-sarray(3,:),sarray(4,:),'k')
```

A family of computed shell shapes are shown in Figure 2.3.2(a). As the scaled transmural pressure $\widehat{\Delta p} \equiv \Delta p\, a^3/\kappa_B$ increases, the profiles become elliptical and then biconcave. Touching of the two sides of the shell occurs at the critical value $\widehat{\Delta p} = 5.247$.

The dependence of the shape aspect ratio, ϱ, on the transmural pressure is illustrated in Figure 2.3.2(b). The dashed line represents the phenomenological correlation

$$\varrho = 1 - \tfrac{1}{2}\sqrt{\widehat{\Delta p} - 3}. \tag{2.3.26}$$

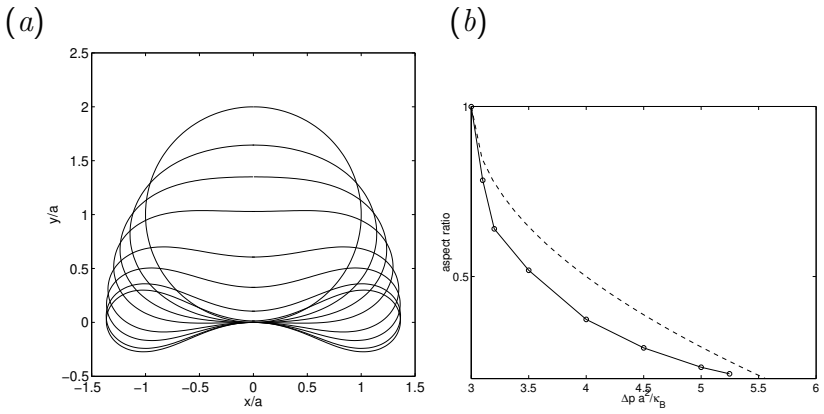


FIGURE 2.3.2 (a) Family of computed cell shapes for scaled transmural pressure $\Delta p a^3/\kappa_B = 3, 3.1, 3.2, 3.5, 4.0, 4.5, 5.0$, and 5.247. (b) Dependence of shell aspect ration on the transmural pressure.

Improved correlations can be developed by using data-fitting algorithms.

Exercise

2.3.1 Prepare a plot of the scaled curvature, $a\kappa_0$, against transmural pressure, and identify the transmural pressure when κ_0 becomes zero.

2.4 Heat transport across a slab

Consider unsteady heat transport across a conductive slab of thickness h, as illustrated in Figure 2.4.1. The temperature is held at a value T_1 at the lower surface of the slab and at a value T_2 and the upper surface of the slab. These boundary temperatures are allowed to vary in time, t.

2.4.1 Unsteady heat conduction equation

The evolution of the temperature profile across the slab is governed by

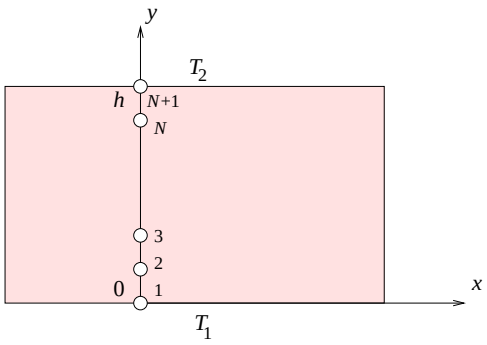


FIGURE 2.4.1 A one-dimensional finite-difference grid is used to compute the velocity profile in unidirectional channel flow.

the unsteady heat conduction equation,

$$\frac{\partial T}{\partial t} = \kappa \frac{\partial^2 T}{\partial y^2}, \tag{2.4.1}$$

where κ is the thermal diffusivity of the slab material with dimensions of length squared divided by time. An initial condition for the temperature profile must be provided.

2.4.2 *Explicit finite-difference method*

A finite-difference method for computing the evolution of the temperature profile can be developed by dividing the cross-section of the slab extending from $y = 0$ to h into N intervals defined by $N + 1$ grid points, as shown in Figure 2.4.1. For convenience, the temperature at the i grid point is denoted as

$$T_i(t) \equiv T(y_i, t) \tag{2.4.2}$$

for $i = 1, \ldots, N + 1$, where T_1 and T_{N+1} are the boundary values.

Next, we evaluate both sides of (2.4.1) at the ith interior grid point at time t for $i = 2, \ldots, N$, and approximate the time derivative on the left-hand side with a first-order forward finite difference (Forward Time, FT) and the second derivative on the right-hand side with a second-order centered finite difference (Centered Space, CS). The result is the

following FTCS finite-difference equation (FDE)

$$\frac{T_i(t+\Delta t) - T_i(t)}{\Delta t} = \kappa \, \frac{T_{i-1}(t) - 2\,T_i(t) + T_{i+1}(t)}{\Delta y^2}. \tag{2.4.3}$$

Solving for $T_i(t+\Delta t)$ on the left-hand side, we obtain

$$T_i(t+\Delta t) = \alpha\, T_{i-1}(t) + (1-2\,\alpha)\, T_i(t) + \alpha\, T_{i+1}(t) \tag{2.4.4}$$

for $i = 2, \ldots, N$, where

$$\alpha \equiv \frac{\kappa \Delta t}{\Delta y^2} \tag{2.4.5}$$

is a dimensionless parameter called the *numerical diffusion number.*

Equation (2.4.4) allows us to update the velocity at the interior grid points explicitly, starting from a specified initial condition, subject to the prescribed boundary conditions

$$T_1(t) = T_1(t), \qquad T_{N+1}(t) = T_2(t). \tag{2.4.6}$$

The algorithm involves merely evaluating the right-hand side of (2.4.4) at each grid point at a succession of time levels.

2.4.3 FTCS code

The following Matlab code named *slab_ftcs*, located in directory SLAB of SEMLIB, performs the simulation of the evolving temperature profile

```
h = 1.0;
kappa = 1.0;

N = 2*32;

T1 = 0.0;
T2 = 1.0;

al  = 0.505;  % alpha

nstep = 2*128;  % number of steps
```

```
mstep = 32;  % number of steps for plotting

%---
% prepare
%---

Dy = h/N;
Dt = al*Dy*Dy/kappa;

%---
% plotting
%---

figure(2)

%---
% grid and initial condition
%---

for i=1:N+1
 y(i) = (i-1)*Dy;
 T(i) = 0.0;
end

T(1) = T1;
T(N+1) = T2;

t = 0.0;
istep = mstep;

%---
% time stepping
%---

for step=1:nstep

 Tnew(1) = T1;

 for i=2:N
```

```
  Tnew(i) = al*T(i-1) + (1.0-2.0*al)*T(i) + al*T(i+1);
 end

 Tnew(N+1) = T2;
 T = Tnew;

 t = t+Dt;

 if(step==1)
  figure(1)
  Handle1 = plot(T,y,'.k-');
  set(Handle1, 'erasemode', 'xor');
  set(gca,'fontsize',15)
  axis([0 1.0 0 h])
  xlabel('T','fontsize',15)
  ylabel('y','fontsize',15)
 else
  figure(1)
  set(Handle1,'XData',T,'YData',y);
  hold on
  pause(0.02)
  drawnow
 end

  if(istep==mstep)
   figure(2)
   plot(T,y,'.k-');
   istep = 0;
  end

  istep = istep+1;

end
%---
```

Evolving temperature profiles are shown in Figure 2.4.2 for two values of the dimensionless numerical parameter α, equal to 0.495 and 0.505.

The numerical results suggest that the explicit method of updating the temperature based on equation (2.4.4) is free of artificial oscillations

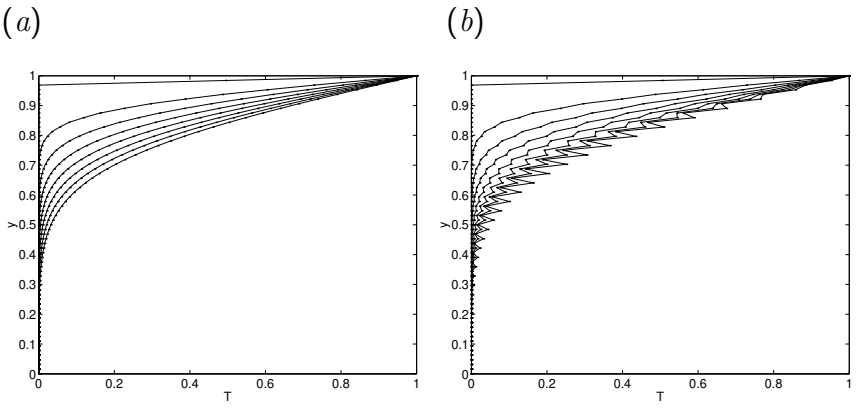


FIGURE 2.4.2 Evolving temperature profiles across a slab computed by an explicit finite-difference method for numerical diffusion number (a) $\alpha = 0.495$ and (b) 0.505.

only when the time step, Δt, is small enough so that the dimensionless numerical diffusion number α defined in (2.4.5) is less than $\frac{1}{2}$.

For larger time steps, the velocity profile develops unphysical growing numerical oscillations unrelated to the physics of the motion, as illustrated in Figure 2.4.2(b) for $\alpha = 0.505$. We say that the explicit finite-difference method is *conditionally stable*.

2.4.4 *Numerical stability analysis*

To understand the origin of the instability, we examine the behavior of the numerical solution subject to a sinusoidal initial condition with a specified wave length, L. Motivated by the linearity of the governing equation, we separate the temporal from the spatial dependence, writing

$$T_i^n = A^n \exp(-\mathrm{i}\, i\theta) = A^n \left(\cos(i\theta) - \mathrm{i}\, \sin(i\theta) \right), \tag{2.4.7}$$

where $T_i^n = T_i(n\Delta t)$, i is the imaginary unit, the coefficient A^n depends on the time level n alone, and

$$\theta \equiv \frac{2\pi \Delta x}{L} \tag{2.4.8}$$

is the phase angle. The superscript n of A^n is an index, not an exponent.

The minimum wave length supported by the grid is $L = 2\Delta x$, corresponding to the maximum phase angle $\theta_{\max} = \pi$. Thus, the phase angle varies from zero for long waves, to π for short waves.

Substituting (2.4.7) into the difference equation (2.4.4), we find that

$$A^{n+1} = \alpha A^n \exp(\mathrm{i}\,\theta) + (1 - 2\alpha)A_n + \alpha A^n \exp(-\mathrm{i}\,\theta). \quad (2.4.9)$$

Rearranging and simplifying, we obtain

$$G \equiv \frac{A^{n+1}}{A^n} = 1 - 4\alpha \sin^2 \frac{\theta}{2}, \quad (2.4.10)$$

where G is the gain or amplification factor.

When $\alpha > \frac{1}{2}$, the magnitude of the amplification factor, $|G|$, is greater than unity for a certain range of values of θ near π, and the numerical method is unstable. When $\alpha < \frac{1}{2}$, $|G|$ is less than unity for any value of θ, and the numerical method is stable.

2.4.5 *Implicit time integration*

To avoid the restriction on the time step Δt for numerical stability, we implement an *implicit* finite-difference method.

The procedure involves enforcing equation (2.4.1) at the ith interior grid point at time $t + \Delta t$ for $i = 2, \ldots, N$, and then approximating the time derivative on the left-hand side with a first-order backward finite difference in time (BT) and the second derivative on the right-hand side with a second-order centered finite difference (CS) to obtain the BTCS difference equation

$$\frac{T_i(t + \Delta t) - T_i(t)}{\Delta t} = \kappa \frac{T_{i+1}(t + \Delta t) - 2\,T_i(t + \Delta t) + T_{i-1}(t + \Delta t)}{\Delta y^2}. \quad (2.4.11)$$

Rearranging, we obtain

$$-\alpha\,T_{i-1}(t + \Delta t) + (1 + 2\,\alpha)\,T_i(t + \Delta t) - \alpha\,T_{i+1}(t + \Delta t) = T_i(t), \quad (2.4.12)$$

where α is a dimensionless numerical diffusion number defined in (2.4.5) as $\alpha \equiv \kappa \Delta t / \Delta y^2$.

Equation (2.4.12) allows us to compute the temperature at the interior grid points at the time level $t + \Delta t$ in an implicit fashion that requires simultaneously solving for all unknown grid values, subject to the prescribed boundary conditions at the lower and upper slab surface.

2.4.6 Formulation as a linear system

To formalize the solution algorithm, we write equation (2.4.12) for the interior nodes, $i = 2, \ldots, N$, and enforce the boundary conditions to obtain a system of $N - 1$ linear equations for the nodal temperatures at time $t + \Delta t$ encapsulated in a vector $\mathbf{T}$,

$$\mathbf{A} \cdot \mathbf{T}(t + \Delta t) = \mathbf{T}(t). \tag{2.4.13}$$

We have introduced the tridiagonal coefficient matrix

$$\mathbf{A} = \begin{bmatrix} 1+2\alpha & -\alpha & 0 & \cdots & 0 & 0 & 0 \\ -\alpha & 1+2\alpha & -\alpha & \cdots & 0 & 0 & 0 \\ 0 & -\alpha & 1+2\alpha & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1+2\alpha & -\alpha & 0 \\ 0 & 0 & 0 & \cdots & -\alpha & 1+2\alpha & -\alpha \\ 0 & 0 & 0 & \cdots & 0 & -\alpha & 1+2\alpha \end{bmatrix}, \tag{2.4.14}$$

the vector of unknown and known temperatures

$$\mathbf{T}(t + \Delta t) = \begin{bmatrix} T_2(t + \Delta t) \\ T_3(t + \Delta t) \\ \vdots \\ T_{N-1}(t + \Delta t) \\ T_N(t + \Delta t) \end{bmatrix}, \qquad \mathbf{T}(t) = \begin{bmatrix} T_2(t) \\ T_3(t) \\ \vdots \\ T_{N-1}(t) \\ T_N(t) \end{bmatrix}, \tag{2.4.15}$$

and a sparse vector incorporating the boundary conditions,

$$\mathbf{b} = \alpha \begin{bmatrix} T_1(t+\Delta t) \\ 0 \\ \vdots \\ 0 \\ T_2(t+\Delta t) \end{bmatrix}. \tag{2.4.16}$$

The numerical method involves solving the linear system (2.4.13) at the current time instant, t, to obtain the temperature profile at the next time instant, $t + \Delta t$, beginning from a specified initial state. The process is then repeated for a specified sequence of time steps.

The tridiagonal structure of the matrix $\mathbf{A}$ displayed in (2.4.14) allows us to compute the solution efficiently using the efficient Thomas algorithm discussed in Section 6.3.

2.4.7 BTCS code

The following Matlab code named *slab_btcs*, located in directory SLAB of SEMLIB, performs the simulation of the evolving temperature profile:

```
N = 2*32;

h = 1.0;
kappa = 1.0;

T1 = 0.0;
T2  = 1.0;

al = 1.00;  % alpha

nstep = 8*128;  % number of steps

mstep = 32;  % number of steps for plotting

%---
% prepare
%---
```

```
Dy = h/N;
Dt = al*Dy*Dy/kappa;

%---
% grid and initial condition
%---

for i=1:N+1
 y(i) = (i-1)*Dy;
 T(i) = 0.0;
end

T(1) = T1;
T(N+1) = T2;

%---
% tridiagonal projection matrix
%---

for i=1:N-1
 atr(i) = 1.0+2*al;
 btr(i) = -al;
 ctr(i) = -al;
end

%---
% plotting
%---

figure(2)
axis([0 1.0 0 h])
xlabel('T','fontsize',15)
ylabel('y','fontsize',15)
set(gca,'fontsize',15)
hold on
box on

%---
% time stepping
```

```
%---

t = 0.0;
istep = mstep;

%---

for step=1:nstep

 for i=1:N-1    % right-hand side
  s(i) = T(i+1);
 end

 T(1)  = T1;
 T(N+1) = T2;
 s(1)   =s(1)  + al*T(1);
 s(N-1) =s(N-1)+ al*T(N+1);

 sol = thomas (N-1,atr,btr,ctr,s);

 for i=2:N
  T(i) = sol(i-1);
 end

 t=t+Dt;

 if(step==1)
   figure(1)
   Handle1 = plot(T,y,'ko-');
   set(Handle1, 'erasemode', 'xor');
   set(gca,'fontsize',15)
   axis([min(T1,T2) max(T1,T2) 0 h])
   xlabel('T','fontsize',15)
   ylabel('y','fontsize',15)
 else
   figure(1)
   set(Handle1,'XData',T,'YData',y);
   pause(0.2)
   drawnow
```

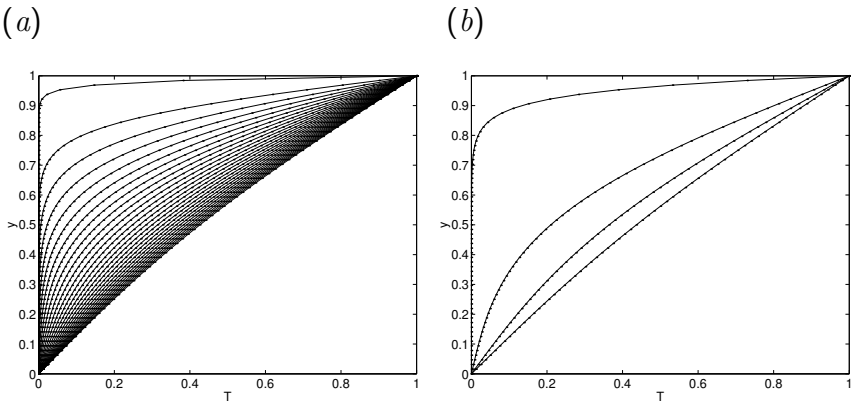


FIGURE 2.4.3 Evolving temperature profiles in a layer computed with an implicit finite-difference method for numerical diffusion number (a) $\alpha = 0.495$ and (b) 0.505.

```
 end

  if(istep==mstep)
   figure(2)
   plot(T,y,'.k-');
   istep = 0;
  end

  istep = istep+1;

end
%---
```

Evolving profiles are shown in Figure 2.4.3 for two values of the dimensionless numerical parameter α. The numerical simulation is stable even for arbitrarily large time steps, Δt.

2.4.8 Numerical stability

To confirm the numerical stability, we substitute the expression (2.4.7)

into the difference equation (2.4.12), and find that

$$G \equiv \frac{A^{n+1}}{A^n} \frac{1}{1 + 4\alpha \sin^2(\frac{1}{2}\theta)}. \tag{2.4.17}$$

Since $|G| < 1$ for any value of α and θ, the method is confirmed to be numerically stable.

Exercise

2.4.1 Derive formula (2.4.17).

2.5 *Heat transport across two adjacent slabs*

Generalizing the discussion of Section 2.4, now we consider unsteady heat transport across two superimposed infinite slabs, as illustrated in Figure 2.5.1. The lower slab is labeled 1 and the upper slab is labeled 2. The slabs are separated by a flat interface located at $y = h_1$, where $h_1 < h$ is the lower slab thickness and h is the slab assembly thickness. The upper-slab thickness is $h_2 = h - h_1$.

The temperature is held at a value T_1 at the lower surface of the lower slab and at a value T_2 at the upper surface of the upper slab. These boundary values are allowed to vary in time, t.

2.5.1 *Unsteady heat conduction equation*

The evolution of the temperature in the lower or upper slab is governed by the unsteady heat conduction equation,

$$\frac{\partial T^{(1)}}{\partial t} = \kappa_1 \frac{\partial^2 T^{(1)}}{\partial y^2}, \qquad \frac{\partial T^{(2)}}{\partial t} = \kappa_2 \frac{\partial^2 T^{(2)}}{\partial y^2}, \tag{2.5.1}$$

where κ_1, κ_2, are the thermal diffusivities of the two materials. An initial condition for the temperature profile in each slab must be provided.

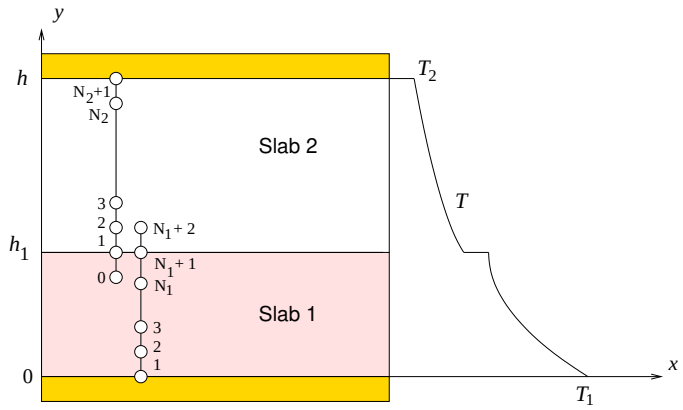


FIGURE 2.5.1 Illustration of a composite finite-difference grid with phantom nodes on either side an interface between two conductive slabs used to compute the evolving temperature profile. The interface is located at $y = h_1$.

2.5.2 *Interfacial conditions*

At the interface, we require continuity of thermal flux expressed by the balance equation

$$q \equiv -k_1 \left(\frac{\partial T^{(1)}}{\partial y}\right)_{y=h_1} = -k_2 \left(\frac{\partial T^{(2)}}{\partial y}\right)_{y=h_1}, \tag{2.5.2}$$

where k_1 and k_2, are the thermal conductivities of the two materials.

In addition, we require an equation for the interfacial jump in temperature related to the thermal flux by

$$q = k_c \, (T^{(1)} - T^{(2)})_{y=h_1}, \tag{2.5.3}$$

where k_c is the *contact conductivity* ranging from zero to infinity.

As k_c becomes larger, the interfacial jump in temperature tends to zero, that is, the composite temperature profile shown in Figure 2.5.1 tends to become continuous. In the diametrically opposite limit where k_c tends to zero, the interfacial flux q decreases and the interfacial jump in temperature rises.

2.5.3 Steady state

The temperature distribution at steady state is governed by the ordinary differential equations

$$\frac{\mathrm{d}^2 T^{(1)}}{\mathrm{d}y^2} = 0, \qquad \frac{\mathrm{d}^2 T^{(2)}}{\mathrm{d}y^2} = 0, \tag{2.5.4}$$

subject to the aforementioned boundary and interface conditions. Integrating these equations, we find the linear profiles

$$T^{(1)} = T_1 + \lambda\,\xi\,y, \qquad T^{(2)} = T_2 + \xi\,(y-h), \tag{2.5.5}$$

where $\lambda \equiv k_2/k_1$ is the conductivity ratio and

$$\xi = \frac{T_2 - T_1}{h}\,\frac{1}{k_2/(hk_c) + (\lambda - 1)\,h_1/h + 1} \tag{2.5.6}$$

is the slope of the temperature profile in the upper slab. The fraction on the right-hand side can be regarded as a correction factor with respect to that arising in the case of a single homogeneous slab, $(k)_c \to \infty$ and $\lambda = 1$. The flux at steady state is given by

$$q = -k_2 \xi. \tag{2.5.7}$$

where ξ is given in (2.5.6).

2.5.4 Finite-difference method

To develop a finite-difference method for computing the evolution of the temperature profile, we divide the lower slab into N_1 evenly spaced intervals defined by $N_1 + 1$ nodes,

$$y_1^{(1)}, y_2^{(1)}, \ldots, y_{N_1+1}^{(1)}, \tag{2.5.8}$$

and the upper slab into N_2 evenly spaced intervals defined by $N_2 + 1$ nodes,

$$y_1^{(2)}, y_2^{(2)}, \ldots, y_{N_1+1}^{(2)}, \tag{2.5.9}$$

as shown in Figure 2.5.1. To simplify the notation, we denote

$$T_i^{(1)} \equiv T^{(1)}(y_i^{(1)}), \qquad T_i^{(2)} \equiv T^{(2)}(y_i^{(2)}), \tag{2.5.10}$$

where i varies over the lower or upper grid.

2.5.5 Phantom nodes

For reasons that will become apparent in hindsight, we also extend the domain of definition of each slab into the adjacent slab by one artificial grid point labeled $N_1 + 2$ for the lower slab or 0 for the upper slab, as shown in Figure 2.5.1.

Approximating the derivatives in (2.5.2) and (2.5.3) with centered finite differences, we derive two equations relating the values of the temperature at the extended nodes, $T^{(1)}_{N_1+2}$ and $T^{(2)}_0$,

$$q \simeq -k_1 \frac{T^{(1)}_{N_1+2} - T^{(1)}_{N_1}}{2\Delta y_1} = -k_2 \frac{T^{(2)}_2 - T^{(2)}_0}{2\Delta y_2} \tag{2.5.11}$$

and

$$k_c \, (T^{(1)}_{N_1+1} - T^{(2)}_1) = -k_1 \frac{T^{(1)}_{N_1+2} - T^{(1)}_{N_1}}{2\Delta y_1}, \tag{2.5.12}$$

where $\Delta y_1 \equiv h_1/N_1$ and $\Delta y_2 \equiv h_2/N_2$ are the grid spacings. Equations (2.5.11) and (2.5.12) can be rearranged as

$$T^{(2)}_0 = \frac{\beta}{\lambda} T^{(1)}_{N_1} + T^{(2)}_2 - \frac{\beta}{\lambda} T^{(1)}_{N_1+2} \tag{2.5.13}$$

and

$$\lambda_c \frac{2}{N_1} (T^{(1)}_{N_1+1} - T^{(2)}_1) = -T^{(1)}_{N_1+2} + T^{(1)}_{N_1}, \tag{2.5.14}$$

where

$$\lambda \equiv \frac{k_2}{k_1}, \qquad \lambda_c \equiv \frac{k_c h_1}{k_1}, \qquad \beta \equiv \frac{\Delta y_2}{\Delta y_1}. \tag{2.5.15}$$

2.5.6 Objective functions

From equation (2.5.14), we find that

$$\Phi_1 \equiv T^{(1)}_{N_1+2} - T^{(1)}_{N_1} + \lambda_c \frac{2}{N_1} (T^{(1)}_{N_1+1} - T^{(2)}_1) = 0, \tag{2.5.16}$$

involving the lower-grid phantom value $T^{(1)}_{N_1+2}$. Combining this expression with (2.5.13) to eliminate $T^{(1)}_{N_1+2}$, we obtain

$$\Phi_2 \equiv T^{(2)}_0 - T^{(2)}_2 - \beta \frac{\lambda_c}{\lambda} \frac{2}{N_1} (T^{(1)}_{N_1+1} - T^{(2)}_1) = 0, \tag{2.5.17}$$

involving the upper-grid phantom value, $T^{(2)}_0$. We recall that β is the ratio of the upper to lower grid spacings.

2.5.7 *Explicit time integration*

The finite-difference equation arising from the explicit discretization of the unsteady heat conduction equation for the lower slab reads

$$\frac{T^{(1)}_i(t+\Delta t) - T^{(1)}_i(t)}{\Delta t} = \kappa_1 \frac{T^{(1)}_{i+1}(t) - 2T^{(1)}_i(t) + T^{(1)}_{i-1}(t)}{\Delta y_1^2}, \tag{2.5.18}$$

where Δt is a chosen time step. The finite-difference equation arising from the explicit discretization of the unsteady heat condition equation for the upper slab reads

$$\frac{T^{(2)}_i(t+\Delta t) - T^{(2)}_i(t)}{\Delta t} = \kappa_1 \frac{T^{(2)}_{i+1}(t) - 2T^{(2)}_i(t) + T^{(2)}_{i-1}(t)}{\Delta y_2^2}. \tag{2.5.19}$$

Rearranging, we obtain

$$T^{(1)}_i(t+\Delta t) = \alpha_1 T^{(1)}_{i+1}(t) + (1 - 2\alpha_1) T^{(1)}_i(t) + \alpha_1 T^{(1)}_{i-1}(t) \tag{2.5.20}$$

for the lower slab, and a corresponding equation for the upper slab,

$$T^{(2)}_i(t+\Delta t) = \alpha_2 T^{(2)}_{i+1}(t) + (1 - 2\alpha_2) T^{(2)}_i(t) + \alpha_2 T^{(2)}_{i-1}(t), \tag{2.5.21}$$

where

$$\alpha_1 \equiv \frac{\kappa_1 \Delta t}{\Delta y_1^2}, \qquad \alpha_2 \equiv \frac{\kappa_2 \Delta t}{\Delta y_2^2} \tag{2.5.22}$$

are the numerical diffusion numbers for the lower and upper slab.

The numerical procedure involves the following steps:

1. Initialize the nodal temperatures.
2. Compute the temperature at the lower extended node, $T^{(1)}_{N_1+2}$, from equation (2.5.16).
3. Compute the temperature at the upper extended node, $T^{(2)}_0$, from equation (2.5.17).
4. Use equation (2.5.20) to update the temperature at the grid points in the lower layer for $i = 2, \ldots, N_1 + 1$.
5. Use equation (2.5.21) to update the temperature at the internal grid nodes in the upper layer for $i = 1, \ldots, N_2$.
6. Use the boundary conditions to update the temperature at the lower and upper walls.
7. Return to Step 2 and repeat the computation for another step.

Unfortunately, the explicit method requires a sufficiently small time step to prevent the occurrence of numerical instability. To overcome this obstacle, we develop an implicit finite-difference method.

2.5.8 Implicit time integration

The finite-difference equation arising from the implicit discretization of the unsteady heat conduction equation for the lower slab reads

$$\frac{T^{(1)}_i(t+\Delta t) - T^{(1)}_i(t)}{\Delta t} = \kappa_1 \frac{T^{(1)}_{i+1}(t+\Delta t) - 2T^{(1)}_i(t+\Delta t) + T^{(1)}_{i-1}(t+\Delta t)}{\Delta y_1^2} \tag{2.5.23}$$

for $i = 2, \ldots, N_1 + 1$, where Δt is a chosen time step. The finite-difference equation arising from the explicit discretization of the unsteady heat condition equation for the upper slab reads

$$\frac{T^{(2)}_i(t+\Delta t) - T^{(2)}_i(t)}{\Delta t} = \kappa_1 \frac{T^{(2)}_{i+1}(t+\Delta t) - 2T^{(2)}_i(t+\Delta t) + T^{(2)}_{i-1}(t+\Delta t)}{\Delta y_2^2} \tag{2.5.24}$$

for $i = 1, \ldots, N_2$. Rearranging, we obtain a difference equation for the lower slab,

$$-\alpha_1 T_{i+1}^{(1)}(t+\Delta t) + (1+2\,\alpha_1)\, T_i^{(1)}(t+\Delta t) - \alpha_1 T_{i-1}^{(1)}(t+\Delta t) = T_i^{(1)}(t), \tag{2.5.25}$$

and a corresponding equation for the upper slab,

$$-\alpha_2 T_{i+1}^{(2)}(t+\Delta t) + (1+2\,\alpha_2)\, T_i^{(1)}(t+\Delta t) - \alpha_2 T_{i-1}^{(2)}(t+\Delta t) = T_i^{(2)}(t). \tag{2.5.26}$$

Equations (2.5.25) and(2.5.26) provide us with two tridiagonal systems of linear equations for the nodal temperatures at time $t + \Delta t$ in terms of nodal values at time t.

2.5.9 *Thomas algorithm*

The following Matlab function named *slabs_solve*, located in directory SLABS of SEMLIB, compiles and solves two tridiagonal systems of equations for given values at the phantom nodes, $T_{N_1+2}^{(1)}$ and $T_0^{(2)}$. The function also returns the values of the objective functions Φ_1 and Φ_2 defined in (2.5.16) and (2.5.17).

The tridiagonal systems are solved by the Thomas algorithm discussed in Section 6.3:

```
function [T1,T2,Phi1,Phi2] = slabs_solve ...
 ...
         (T1,alpha1,T1w,T1_phan,N1 ...
         ,T2,alpha2,T2w,T2_phan,N2 ...
         ,beta,lambda,lambdac)

%---
% Thomas coefficients
%---

for i=1:N1
 atr1(i) = 1.0+2.0*alpha1;
 btr1(i) = -alpha1;
 ctr1(i) = -alpha1;
```

```
end

for i=1:N2
 atr2(i) = 1.0+2.0*alpha2;
 btr2(i) = -alpha2;
 ctr2(i) = -alpha2;
end

%---
% right-hand sides
%---

for i=1:N1
 s1(i) = T1(i+1);
end

s1(1)  = s1(1)  + alpha1*T1w;
s1(N1) = s1(N1) + alpha1*T1_phan;

for i=1:N2
 s2(i) = T2(i);
end

s2(1)  = s2(1)  + alpha2*T2_phan;
s2(N2) = s2(N2) + alpha2*T2w;

%---
% solutions by the Thomas algorithm
%---

sol1 = thomas (N1,atr1,btr1,ctr1,s1);

for i=1:N1
 T1(i+1) = sol1(i);
end

sol2 = thomas (N2,atr2,btr2,ctr2,s2);

for i=1:N2
```

```
 T2(i) = sol2(i);
end

%---
% compute Phi1 and Phi2
%---

DTI = T1(N1+1)-T2(1);
fc = 2.0*DTI/N1;
Phi1 = T1_phan - T1(N1) + lambdac*fc;
Phi2 = T2_phan - T2(2)  - beta*lambdac/lambda*fc;

%---
% done
%---

return
```

2.5.10 *Phantom node values*

The phantom values passed to the function *slabs_solve*, $T^{(1)}_{N_1+2}$ and $T^{(2)}_0$, must be found as part of the solution such that $\Phi_1 = 0$ and $\Phi_2 = 0$ according to equations (2.5.16) and (2.5.17). A key observation is that, because Φ_1 and Φ_2 are linear functions of the two phantom values, we may write

$$\begin{aligned} \Phi_1 &= A_{1,1}\, T^{(1)}_{N_1+2} + A_{1,2}\, T^{(2)}_0 + b_1, \\ \Phi_2 &= A_{2,1}\, T^{(1)}_{N_1+2} + A_{2,2}\, T^{(2)}_0 + b_2, \end{aligned} \tag{2.5.27}$$

where $\mathbf{A}$ is a certain matrix and $\mathbf{b}$ is a certain vector evaluated by computing Φ_1 and Φ_2 for $T^{(1)}_{N_1+2} = 0$ or 1 and $T^{(2)}_0 = 0$ or 1, in three combinations. For example, $b_1 = \Phi_1$ for $T^{(1)}_{N_1+2} = 0$ and $T^{(2)}_0 = 0$.

Accordingly, $T^{(1)}_{N_1+2}$ and $T^{(2)}_0$ can be evaluated by solving the system of two equations in two unknowns originating from (2.5.27),

$$\mathbf{A} \cdot \boldsymbol{\phi} = -\mathbf{b}, \tag{2.5.28}$$

where $\boldsymbol{\phi} = [T^{(1)}_{N_1+2}, T^{(2)}_0]$ is the doublet of *a priori* unknown phantom values.

2.5.11 Simulation code

The numerical method incorporating time stepping toward the steady state is implemented in the following code named *slabs_btcs* located in directory SLABS of SEMLIB:

```
%---
% parameters
%---

h1 = 1.0;
h2 = 2.0*h1;

N1 = 16;
N2 = 48;

T1w = 0.0;
T2w = 1.0;

kappa1 = 1.0;
k1 = 1.0;

kappa2 = 3.0*kappa1;
k2 = 0.4*k1;
kc = 1.1*k1/h1;

Dt = 0.1;   % time step

%---
% prepare
%---

Dy1 = h1/N1;
Dy2 = h2/N2;

alpha1 = kappa1*Dt/Dy1^2;
alpha2 = kappa2*Dt/Dy2^2;

beta = Dy2/Dy1;
lambda = k2/k1;
```

```
lambdac = kc*h1/k1;

h = h1+h2;

% for steady state:

xi = (T2w-T1w)/h/(k2/(h*kc)+(lambda-1)*h1/h+1.0);

%---
% grid, initial conditions, steady state
%---

for i=1:N1+1
 y1(i) = (i-1)*Dy1;
 T1(i) = 0;
 T1steady(i) = T1w + lambda*xi*y1(i);
end

T1(1) = T1w;

for i=1:N2+1
 y2(i) = h1+(i-1)*Dy2;
 T2(i) = 0;
 T2steady(i) = T2w + xi*(y2(i)-h);
end

T2(N2+1) = T2w;

%---
% time stepping
%---

t = 0.0;

nstep = 32;

%---
for step=1:nstep
%---
```

```
T1save = T1; T2save = T2;

T1_phan = 0.0; T2_phan = 0.0;  % phantom nodes

[T1,T2,Phi1,Phi2] = slabs_solve ...
  ...
                   (T1,alpha1,T1w,T1_phan,N1 ...
                   ,T2,alpha2,T2w,T2_phan,N2 ...
                   ,beta,lambda,lambdac);
RHS(1) = -Phi1;
RHS(2) = -Phi2;

T1_phan = 1.0; T2_phan = 0.0;
T1 = T1save; T2 = T2save;

[T1,T2,Phi1,Phi2] = slabs_solve ...
 ...
                   (T1,alpha1,T1w,T1_phan,N1 ...
                   ,T2,alpha2,T2w,T2_phan,N2 ...
                   ,beta,lambda,lambdac);
AMAT(1,1) = Phi1+RHS(1);
AMAT(2,1) = Phi2+RHS(2);

T1_phan = 0.0; T2_phan = 1.0;
T1 = T1save; T2 = T2save;

[T1,T2,Phi1,Phi2] = slabs_solve ...
...
                  (T1,alpha1,T1w,T1_phan,N1 ...
                  ,T2,alpha2,T2w,T2_phan,N2 ...
                  ,beta,lambda,lambdac);
AMAT(1,2) = Phi1+RHS(1);
AMAT(2,2) = Phi2+RHS(2);

PHAN = RHS/AMAT';
T1_phan = PHAN(1);
T2_phan = PHAN(2);
```

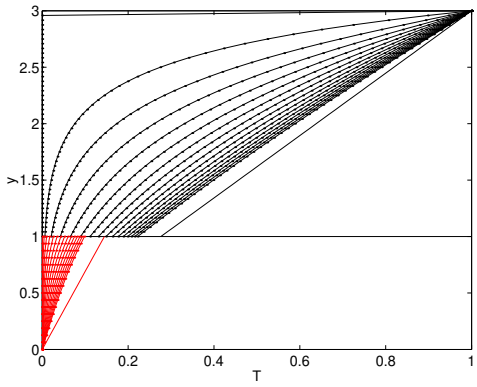


FIGURE 2.5.2 A sequence of profiles evolving toward the steady state profile across two slabs in the presence of contact resistance.

```
T1 = T1save; T2 = T2save;

[T1,T2,Phi1,Phi2] = slabs_solve ...
   ...
                     (T1,alpha1,T1w,T1_phan,N1 ...
                     ,T2,alpha2,T2w,T2_phan,N2 ...
                     ,beta,lambda,lambdac);
t = t+Dt;

%---
end
%---
```

A sequence of profiles advancing toward the steady state represented by the linear profiles are shown in Figure 2.5.2. Thanks to the implicit time integration, which requires solving system of tridiagonal equation at a minimal computation cost, an arbitrarily long time step can be afforded.

Exercise

2.5.1 Sketch the steady-state temperature profile in the limit as k_c tends to zero.

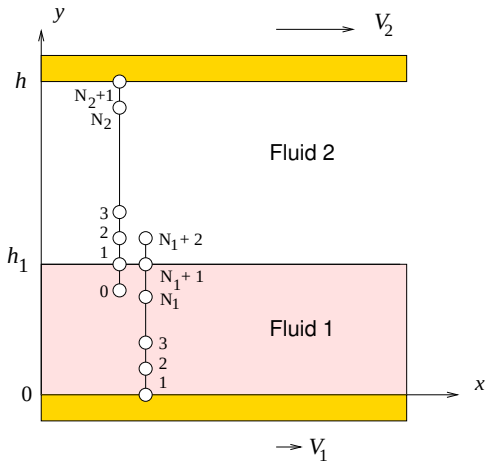


FIGURE 2.6.1 Illustration of a composite finite-difference grid with phantom nodes on either side an interface used to compute the velocity profile of unidirectional two-fluid channel flow. The interface is located at $y = h_1$.

2.6 Two-layer flow in a channel

In an application related to that discussed in Section 25. for heat conduction across two adjacent slabs, now we consider the unidirectional flow of two superimposed layers in a channel, as illustrated in Figure 2.6.1. The lower layer is labeled 1 and the upper layer is labeled 2. The fluids are separated by a flat interface located at $y = h_1$, where h_1 is the lower-layer thickness. The upper-layer thickness is $h_2 = h - h_1$, where h is the channel width.

2.6.1 Equations of viscous unidirectional flow

The evolution of the streamwise velocity component, u_x, is governed by the unsteady diffusion equation

$$\frac{\partial u_x^{(1)}}{\partial t} = -\frac{1}{\rho_1}\frac{\partial p^{(1)}}{\partial x} + \nu_1 \frac{\partial^2 u_x^{(1)}}{\partial y^2} + g_x \tag{2.6.1}$$

in the lower layer, and by a corresponding equation in the upper layer,

$$\frac{\partial u_x^{(2)}}{\partial t} = -\frac{1}{\rho_2}\frac{\partial p^{(1)}}{\partial x} + \nu_2 \frac{\partial^2 u_x^{(2)}}{\partial y^2} + g_x, \tag{2.6.2}$$

where $u_x^{(1)}$ and $p^{(1)}$ are the lower-layer velocity and pressure, $u_x^{(2)}$ are the upper-layer velocity and pressure, ρ_1 and ρ_2, are the fluid densities, $\nu_1 = \mu_1/\rho_1$ and $\nu_2 = \mu_2/\rho_2$ are the kinematic viscosities, μ_1 and μ_2 are the viscosities, and g_x is the streamwise component of the gravitational acceleration.

2.6.2 *Wall conditions*

The lower and upper walls translate parallel to themselves with velocities V_1 and V_2, as shown in Figure 2.6.1. The no-slip boundary condition requires that

$$u_x^{(1)}(y=0) = V_1, \qquad u_x^{(2)}(y=h) = V_2. \tag{2.6.3}$$

A slip boundary condition could have been used in a generalized formulation.

2.6.3 *Interfacial conditions*

At the interface, we require three conditions: continuity of velocity, continuity of shear stress, and continuity of normal stress. To ensure continuity of shear stress, we require that the streamwise pressure gradient is shared by the two layers,

$$\chi \equiv -\frac{\partial p^{(1)}}{\partial x} = -\frac{\partial p^{(2)}}{\partial x}, \tag{2.6.4}$$

where χ is a constant. Continuity of velocity at the interface requires that

$$u_x^{(1)}(y=h_1) = u_x^{(2)}(y=h_1). \tag{2.6.5}$$

Using the equations of motion (2.6.1) and (2.6.2), we find that, if condition (2.6.5) is true at the initial instant, it will remain true, provided that

$$-\frac{1}{\rho_1}\frac{\partial p}{\partial x} + \nu_1 \left(\frac{\partial^2 u_x^{(1)}}{\partial y^2}\right)_{y=h_1} = -\frac{1}{\rho_2}\frac{\partial p}{\partial x} + \nu_2 \left(\frac{\partial^2 u_x^{(2)}}{\partial y^2}\right)_{y=h_1}, \tag{2.6.6}$$

where the second partial derivatives are evaluated at the interface.

2.6.4 Pressure distribution

The pressure profile is given by

$$p^{(1)}(x,y) = -\chi x + \rho_1 g_y\,(y-h_1) + \pi_0 \tag{2.6.7}$$

in the lower layer and

$$p^{(2)}(x,y) = -\chi x + \rho_2 g_y\,(y-h_1) + \pi_0 \tag{2.6.8}$$

in the upper layer, where π_0 is an unspecified reference pressure.

2.6.5 Dimensionless ratios

For future reference, we introduce the viscosity ratio, the density ratio, the kinematic viscosity ratio, and the layer thickness ratio,

$$\lambda \equiv \frac{\mu_2}{\mu_1}, \qquad \varrho \equiv \frac{\rho_2}{\rho_1}, \qquad \gamma \equiv \frac{\nu_2}{\nu_1} = \frac{\lambda}{\varrho}, \qquad r \equiv \frac{h_2}{h_1}. \tag{2.6.9}$$

2.6.6 Steady state

At steady state, the velocity profiles across the two layers are governed by the equations

$$\frac{\mathrm{d}^2 u_x^{(1)}}{\mathrm{d}y^2} = -\frac{\chi + \rho_1 g_x}{\mu_1}, \qquad \frac{\mathrm{d}^2 u_x^{(2)}}{\mathrm{d}y^2} = -\frac{\chi + \rho_2 g_x}{\mu_2}. \tag{2.6.10}$$

To facilitate forthcoming algebraic manipulations, we integrate equations (2.6.10) twice with respect to y and express the solutions in the convenient forms

$$u_x^{(1)}(y) = -\tfrac{1}{2}\,\frac{\chi + \rho_1 g_x}{\mu_1}\,(y-h_1)^2 + \xi_1\,(y-h_1) + u_{\mathrm{I}} \tag{2.6.11}$$

and

$$u_x^{(2)}(y) = -\tfrac{1}{2}\,\frac{\chi + \rho_2 g_x}{\mu_2}\,(y-h_1)^2 + \xi_2\,(y-h_1) + u_{\mathrm{I}}, \tag{2.6.12}$$

where $u_{\rm I}$ is the common interfacial velocity and

$$\xi_1 \equiv \left(\frac{\mathrm{d}u_x^{(1)}}{\mathrm{d}y}\right)_{y=h_1}, \qquad \xi_2 \equiv \left(\frac{\mathrm{d}u_x^{(2)}}{\mathrm{d}y}\right)_{y=h_1} \tag{2.6.13}$$

are the interfacial shear rates expressing the slope of the velocity profile on either side of the interface.

To compute the three unknowns,$u_{\rm I}$, ξ_1, and ξ_2, we enforce the no-slip boundary condition at the lower and upper walls, and also require that the shear stress is continuous across the interface,

$$\mu_1 \xi_1 = \mu_2 \, \xi_2. \tag{2.6.14}$$

After a fair amount of algebra, we derive the expression.

$$u_{\rm I} = \frac{1}{2\mu_1} \frac{h_1 h_2}{\lambda + r} \left[(1+r)\,\chi + \rho_1 g_x \,(1 + \varrho\, r) \right] + \frac{r\, V_1 + \lambda\, V_2}{r + \lambda}. \tag{2.6.15}$$

The shear rates are given by

$$\xi_1 = -\tfrac{1}{2} \frac{\chi + \rho_1 \, g_x}{\mu_1} \, h_1 + \frac{u_{\rm I} - V_1}{h_1} \tag{2.6.16}$$

and

$$\xi_2 = \tfrac{1}{2} \frac{\chi + \rho_2 \, g_x}{\mu_2} \, h_2 - \frac{u_{\rm I} - V_2}{h_2}, \tag{2.6.17}$$

respectively, at the lower or upper side of the interface.

2.6.7 Finite-difference grid

To implement a finite-difference method, we divide the lower layer into N_1 evenly spaced intervals defined by $N_1 + 1$ grid points, $y_i^{(1)}$ for $i = 1, \ldots, N_1 + 1$, and the upper layer into N_2 evenly spaced intervals defined by $N_2 + 1$ grid points, $y_i^{(2)}$ for $i = 1, \ldots, N_2 + 1$, as shown in Figure 2.6.1. The grid spacings are $\Delta y_1 \equiv h_1/N_1$ and $\Delta y_2 \equiv h_2/N_2$.

To simplify the notation, we denote

$$u_i^{(1)} \equiv u_x^{(1)}(y_i^{(1)}), \qquad u_i^{(2)} \equiv u_x^{(2)}(y_i^{(2)}) \tag{2.6.18}$$

for the ith node.

For reasons that will become apparent in hindsight, we also extend the domain of definition of each layer into the adjacent layer by one artificial grid point labeled N_1+2 for the lower layer or 0 for the upper layer.

2.6.8 Discretization of interfacial conditions

Approximating the derivatives in (2.5.3) and (2.6.6) with centered finite differences, we derive two equations relating the values of the velocity at the extended nodes, $u^{(1)}_{N_1+2}$ and $u^{(2)}_0$,

$$\mu_1 \frac{u^{(1)}_{N_1+2} - u^{(1)}_{N_1}}{2\Delta y_1} = \mu_2 \frac{u^{(2)}_2 - u^{(2)}_0}{2\Delta y_2} \tag{2.6.19}$$

and

$$\frac{\chi}{\rho_1} + \nu_1 \frac{u^{(1)}_{N_1+2} - 2\,u^{(1)}_{N_1+1} + u^{(1)}_{N_1}}{\Delta y_1^2} = \frac{\chi}{\rho_2} + \nu_2 \frac{u^{(2)}_2 - 2\,u^{(2)}_1 + u^{(2)}_0}{\Delta y_2^2}. \tag{2.6.20}$$

Setting $u^{(1)}_{N_1+1} = u^{(2)}_1$ to ensure that the velocity is continuous across the interface, we recast equations (2.6.19) and (2.6.20) into a system of two linear equations for the velocity at the extended nodes,

$$\beta\, u^{(1)}_{N_1+2} + \lambda\, u^{(2)}_0 = \beta\, u^{(1)}_{N_1} + \lambda\, u^{(2)}_2 \tag{2.6.21}$$

and

$$\begin{aligned} \beta^2\, u^{(1)}_{N_1+2} - \gamma\, u^{(2)}_0 = 2\,(\beta^2 - \gamma)\, u^{(1)}_{N_1+1} - \beta^2 u^{(1)}_{N_1} + \gamma\, u^{(2)}_2 \\ -\frac{\Delta y_2^2}{\mu_1}\,(1 - \frac{1}{\varrho})\,\chi, \end{aligned} \tag{2.6.22}$$

where $\beta \equiv \Delta y_2/\Delta y_1$ is the grid-size ratio.

In matrix notation,

$$\begin{bmatrix} \beta & \lambda \\ \beta^2 & -\gamma \end{bmatrix} \cdot \begin{bmatrix} u^{(1)}_{N_1+2} \\ u^{(2)}_0 \end{bmatrix} = \mathbf{c}, \tag{2.6.23}$$

where

$$\mathbf{c} \equiv \begin{bmatrix} \beta\, u_{N_1}^{(1)} + \lambda\, u_2^{(2)}, \\ \\ 2\,(\beta^2 - \gamma)\, u_{N_1+1}^{(1)} - \beta^2 u_{N_1}^{(1)} + \gamma u_2^{(2)} - \dfrac{\Delta y_2^2}{\mu_1}\,(1 - \dfrac{1}{\varrho})\,\chi \end{bmatrix} \tag{2.6.24}$$

Solving for the velocity at the lower and upper extended nodes, we obtain the expressions

$$u_{N_1+2}^{(1)} = a_1\, u_{N_1}^{(1)} + a_2\, u_{N_1+1}^{(1)} + a_3\, u_2^{(2)} - a_4\, \chi \tag{2.6.25}$$

and

$$u_0^{(2)} = -\frac{\beta}{\lambda}\, u_{N_1+2}^{(1)} + \frac{\beta}{\lambda}\, u_{N_1}^{(1)} + u_2^{(2)}, \tag{2.6.26}$$

where

$$\begin{aligned} a_1 &= \frac{\gamma - \beta\lambda}{\gamma + \beta\lambda}, & a_2 &= 2\lambda \frac{\beta^2 - \gamma}{\beta(\gamma + \beta\lambda)}, \\ a_3 &= \frac{2\,\gamma\lambda}{\beta(\gamma + \beta\lambda)}, & a_4 &= \frac{\lambda}{\beta(\gamma + \beta\lambda)}\, \frac{\Delta y_2^2}{\mu_1} \left(1 - \frac{1}{\varrho}\right) \end{aligned} \tag{2.6.27}$$

are four constants.

2.6.9 *Explicit time integration*

To develop an explicit time integration method, we evaluate both sides of (2.6.1) and (2.6.2) at the ith node and approximate the time derivative on the left-hand side with a first-order forward finite difference and the second derivative on the right-hand side with a second-order centered finite difference. The result is the finite-difference equation

$$\frac{u_i^{(1)}(t + \Delta t) - u_i^{(1)}(t)}{\Delta t} = \frac{\chi}{\rho_1} + \nu_1\, \frac{u_{i-1}(t) - 2\, u_i(t) + u_{i+1}(t)}{\Delta y_1^2} + g_x, \tag{2.6.28}$$

for the lower layer, and a corresponding equation

$$\frac{u_i^{(1)}(t + \Delta t) - u_i^{(1)}(t)}{\Delta t} = \frac{\chi}{\rho_2} + \nu_2\, \frac{u_{i-1}(t) - 2\, u_i(t) + u_{i+1}(t)}{\Delta y_2^2} + g_x \tag{2.6.29}$$

for the upper layer. Rearranging, we derive the explicit finite-difference equations

$$u_i^{(1)}(t+\Delta t) \tag{2.6.30}$$
$$= \alpha_1\, u_{i+1}^{(1)}(t) + (1-2\,\alpha_1)\, u_i^{(1)}(t) + \alpha_1\, u_{i-1}^{(1)}(t) - \Delta t\,\big(\frac{\chi}{\rho_1} + g_x\big)$$

for the lower layer, and

$$u_i^{(2)}(t+\Delta t) \tag{2.6.31}$$
$$= \alpha_2\, u_{i+1}^{(2)}(t) + (1-2\,\alpha_2)\, u_i^{(2)}(t) + \alpha_2\, u_{i-1}^{(2)}(t) - \Delta t\,\big(\frac{\chi}{\rho_2} + g_x\big),$$

for the upper layer, where

$$\alpha_1 \equiv \frac{\nu_1 \Delta t}{\Delta y_1^2}, \qquad \alpha_2 \equiv \frac{\nu_2 \Delta t}{\Delta y_2^2} \tag{2.6.32}$$

are the numerical diffusion numbers for the lower and upper layer.

The numerical procedure involves the following steps:

1. Initialize the nodal velocities.
2. Compute the velocity at the lower extended node, $u_{N_1+2}^{(1)}$, from equation (2.6.25).
3. Use equation (2.6.31) to update the velocity at the grid points in the lower layer for $i = 2, \ldots, N_1 + 1$.
4. Set $u_1^{(2)} = u_{N_1+1}^{(1)}$.
5. Use equation (2.6.32) to update the velocity at the internal grid nodes in the upper layer for $i = 2, \ldots, N_2$.
6. Use the boundary conditions to update the velocity at the lower and upper walls.
7. Return to Step 2 and repeat the computation for another step.

The method is implemented in a Matlab code named *layers_ftcs*, located in directory LAYERS of SEMLIB.

2.6.10 *Implicit time integration*

As in the heat transport problem discuss in Section 2.5, the explicit method requires a sufficiently small time step to prevent the occurrence of numerical instability. To overcome this practical difficulty, we develop an implicit finite-difference method based on the backward-time discretization.

An implicit time integration method can be developed working as in Section 2.8 for the heat transport problem. The finite-difference equation arising from the implicit discretization of equation (2.6.1) reads

$$\frac{u_i^{(1)}(t+\Delta t)-u_i^{(1)}(t)}{\Delta t}=\frac{\chi(t+\Delta t)}{\rho_1} \tag{2.6.33}$$
$$+\nu_1\,\frac{u_{i+1}^{(1)}(t+\Delta t)-2u_i^{(1)}(t+\Delta t)+u_{i-1}^{(1)}(t+\Delta t)}{\Delta y_1^2}+g_x(t+\Delta t)$$

for $i=2,\ldots,N_1+1$. The finite-difference equation arising from the implicit discretization of equation (2.6.2) reads

$$\frac{u_i^{(2)}(t+\Delta t)-u_i^{(2)}(t)}{\Delta t}=\frac{\chi(t+\Delta t)}{\rho_1} \tag{2.6.34}$$
$$\kappa_1\,\frac{u_{i+1}^{(2)}(t+\Delta t)-2u_i^{(2)}(t+\Delta t)+u_{i-1}^{(2)}(t+\Delta t)}{\Delta y_2^2}+g_x(t+\Delta t)$$

for $i=1,\ldots,N_2$. Rearranging, we obtain the difference equation

$$-\alpha_1\,u_{i+1}^{(1)}(t+\Delta t)+(1+2\,\alpha_1)\,u_i^{(1)}(t+\Delta t)-\alpha_1\,u_{i-1}^{(1)}(t+\Delta t)$$
$$=u_i^{(1)}(t)+\Delta t\left(\frac{\chi+\rho_1 g_x}{\rho_1}\right)_{t+\Delta t} \tag{2.6.35}$$

for the lower layer, and

$$-\alpha_2\,u_{i+1}^{(2)}(t+\Delta t)+(1+2\,\alpha_2)\,u_i^{(2)}(t+\Delta t)-\alpha_2\,u_{i-1}^{(2)}(t+\Delta t)$$
$$=u_i^{(2)}(t)+\Delta t\left(\frac{\chi+\rho_2 g_x}{\rho_2}\right)_{t+\Delta t} \tag{2.6.36}$$

for the upper layer. Equations (2.6.35) and(2.6.36) provide us with two tridiagonal systems of linear equations for the nodal temperature values at time $t+\Delta t$ in terms of nodal values at time t.

The implicit numerical method is similar to that discussed in Section 2.5 for heat transport across two adjacent slabs. The objective functions Φ_1 and Φ_2 are now defined with reference to (2.6.25) and (2.6.26) as

$$\Phi_1 \equiv u^{(1)}_{N_1+2} - a_1\, u^{(1)}_{N_1} - a_2\, u^{(1)}_{N_1+1} - a_3\, u^{(2)}_2 + a_4\, \chi = 0 \tag{2.6.37}$$

and

$$\Phi_2 \equiv u^{(2)}_0 + \frac{\beta}{\lambda}\, u^{(1)}_{N_1+2} - \frac{\beta}{\lambda}\, u^{(1)}_{N_1} - u^{(2)}_2. \tag{2.6.38}$$

Note that all nodal values are implicit functions of the values at the phantom nodes, $u^{(1)}_{N_1+2}$ and $u^{(2)}_0$.

2.6.11 Thomas algorithm

The following Matlab function named *layers_btcs_solve*, located in directory LAYERS of SEMLIB, compiles and solves two tridiagonal systems of equations for given values at the phantom nodes, $u^{(1)}_{N_1+2}$ and $u^{(2)}_0$, and also returns the values of the functions Φ_1 and Φ_2 defined in (2.6.37) and (2.6.38). The tridiagonal systems are solved by the Thomas algorithm discussed in Section 99.9:

```
function [u1,u2,Phi1,Phi2] = layers_btcs_solve ...
 ...
        (u1,alpha1,u1w,u1_phan,N1 ...
        ,u2,alpha2,u2w,u2_phan,N2 ...
        ,Dt,chi,rho1,rho2,gx ...
        ,beta,lambda ...
        ,a1,a2,a3,a4)

%---
% Thomas coefficients
%---

for i=1:N1
 atr1(i) = 1.0+2.0*alpha1;
 btr1(i) = -alpha1;
 ctr1(i) = -alpha1;
end
```

```
for i=1:N2
 atr2(i) = 1.0+2.0*alpha2;
 btr2(i) = -alpha2;
 ctr2(i) = -alpha2;
end

%---
% right-hand sides
%---

tmp1 = Dt*(chi+rho1*gx)/rho1;

for i=1:N1
 s1(i) = u1(i+1) + tmp1;
end

s1(1)  = s1(1)  + alpha1*u1w;
s1(N1) = s1(N1) + alpha1*u1_phan;

tmp2 = Dt*(chi+rho2*gx)/rho2;

for i=1:N2
 s2(i) = u2(i) + tmp2;
end

s2(1)  = s2(1)  + alpha2*u2_phan;
s2(N2) = s2(N2) + alpha2*u2w;

%---
% solutions by the Thomas algorithm
%---

sol1 = thomas (N1,atr1,btr1,ctr1,s1);

for i=1:N1
 u1(i+1) = sol1(i);
end
```

```
sol2 = thomas (N2,atr2,btr2,ctr2,s2);

for i=1:N2
 u2(i) = sol2(i);
end

%---
% compute Phi(1) and Phi(2)
%---

Phi1 = u1_phan - a1*u1(N1) - a2*u1(N1+1) - a3*u2(2) + a4*chi;
Phi2 = u2_phan + beta/lambda*(u1_phan - u1(N1)) - u2(2);

%---
% done
%---

return
```

2.6.12 Simulation code

The numerical method incorporating time stepping toward the steady state is implemented in the following Matlab code named *layers_btcs*, located in directory LAYERS of SEMLIB:

```
h1 = 1.0;
h2 = 2.0*h1;

N1 = 16;
N2 = 48;

u1w = 0.0;
u2w = 0.8;

mu1 = 1.0;
rho1 = 1.0;

mu2 = 0.7*mu1;
rho2 = 0.4*rho1;
```

```
chi =-0.4;
gx = 0.0;

Dt = 0.5;   % time step

%---
% prepare
%---

Dy1 = h1/N1;
Dy2 = h2/N2;

nu1 = mu1/rho1;
nu2 = mu2/rho1;

alpha1 = nu1*Dt/Dy1^2;
alpha2 = nu2*Dt/Dy2^2;

h = h1+h2;
beta = Dy2/Dy1;
lambda = mu2/mu1;
vr = rho2/rho1;
hr = h2/h1;

gamma = lambda/vr;
tmp = gamma+beta*lambda;
a1 = (gamma-beta*lambda)/tmp;
a2 = 2.0*lambda*(beta*beta-gamma)/(beta*tmp);
a3 = 2.0*gamma*lambda/(beta*tmp);
a4 = lambda*Dy2*Dy2*(1.0-1.0/vr)/(beta*mu1*tmp);

%---
% grid, initial conditions
%---

for i=1:N1+1
 y1(i) = (i-1)*Dy1;
 u1(i) = 0;
```

```
end

u1(1) = u1w;

for i=1:N2+1
 y2(i) = h1+(i-1)*Dy2;
 u2(i) = 0;
end

u2(N2+1) = u2w;

%---
% time stepping
%---

t = 0.0;

nstep = 128;

%---
for step=1:nstep
%---

u1save = u1;
u2save = u2;

u1_phan = 0.0;
u2_phan = 0.0;

[u1,u2,Phi1,Phi2] = solve_layers ...
 ...
        (u1,alpha1,u1w,u1_phan,N1 ...
        ,u2,alpha2,u2w,u2_phan,N2 ...
        ,Dt,chi,rho1,rho2,gx ...
        ,beta,lambda ...
        ,a1,a2,a3,a4);

RHS(1) = -Phi1;
RHS(2) = -Phi2;
```

```
u1_phan = 1.0;
u2_phan = 0.0;

u1 = u1save;
u2 = u2save;

[u1,u2,Phi1,Phi2] = solve_layers ...
 ...
        (u1,alpha1,u1w,u1_phan,N1 ...
        ,u2,alpha2,u2w,u2_phan,N2 ...
        ,Dt,chi,rho1,rho2,gx ...
        ,beta,lambda ...
        ,a1,a2,a3,a4);

AMAT(1,1) = Phi1+RHS(1);
AMAT(2,1) = Phi2+RHS(2);

u1_phan = 0.0;
u2_phan = 1.0;

u1 = u1save;
u2 = u2save;

[u1,u2,Phi1,Phi2] = solve_layers ...
 ...
        (u1,alpha1,u1w,u1_phan,N1 ...
        ,u2,alpha2,u2w,u2_phan,N2 ...
        ,Dt,chi,rho1,rho2,gx ...
        ,beta,lambda ...
        ,a1,a2,a3,a4);

AMAT(1,2) = Phi1+RHS(1);
AMAT(2,2) = Phi2+RHS(2);

PHAN = RHS/AMAT';
u1_phan = PHAN(1);
u2_phan = PHAN(2);
```

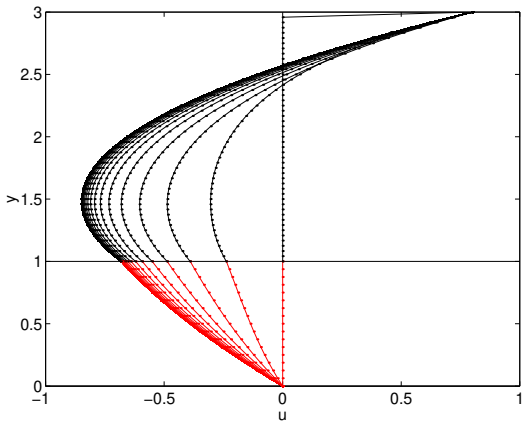


FIGURE 2.6.2 A sequence of velocity profiles evolving toward the steady state profile across two viscous layers in unidirectional flow.

```
u1 = u1save;
u2 = u2save;

[u1,u2,Phi1,Phi2] = solve_layers ...
 ...
        (u1,alpha1,u1w,u1_phan,N1 ...
        ,u2,alpha2,u2w,u2_phan,N2 ...
        ,Dt,chi,rho1,rho2,gx ...
        ,beta,lambda ...
        ,a1,a2,a3,a4);

t = t+Dt;

%---
end
%---
```

A sequence of profiles advancing toward the steady state are shown in Figure 2.6.2. Thanks to the implicit time integration, an arbitrarily large time step can be afforded.

Exercise

2.6.1 Discuss the implementation of the implicit method to three-layer flow.

2.7 Moving particles

Consider an array of N particles, each moving along the x axis with its own velocity, v, from a specified initial position. The ith particle position, X_i, is governed by the equation

$$\frac{\mathrm{d}X_i}{\mathrm{d}t} = v_i(t) \tag{2.7.1}$$

for $i = 1, \ldots, N$. We can prepare a plot of v_i against X_i at any time instant, t. Connecting the dots provides us with a generally evolving function, $v(x, t)$. If all particle velocities are the same, $v_i = V(t)$, then the function $v(x, t) = V(t)$ is a rising or plunging flat distribution.

2.7.1 Burgers equation

If the velocity of each particle remains constant in time, $v_i = V_i$, then the interpolated function, $v(x, t)$, satisfies the Burgers equation,

$$\frac{\partial v}{\partial t} + v\,\frac{\partial v}{\partial x} = 0. \tag{2.7.2}$$

The reason is that the left-hand side of this equation is the material derivative of v expressing the rate of change of v following each point particle, which is zero by assumption. Physically, when particles move with a constant velocity, they preserve the value of the velocity.

The following Matlab code, located in directory PARTICLES of SEMLIB, computes and visualizes the particle motion:

```
%-------------------------
% particle solution of
% the convection equation
%-------------------------
```

```
N  = 2*2*2*64; % number of particles

a  =-10;
b  = 10;

Dt = 0.00010; % time step

%---
% prepare
%---

Dx = (b-a)/N;

%---
% initial position and velocity
%---

for i=1:N
 x(i) = a+(i-0.5)*Dx;
 vinit(i) = exp(-x(i)^2);
 v(i) = vinit(i); % Burgers
end

time = 0.0;

%-----
for istep=1:30000
%-----

 for i=1:N
  x = x + Dt*v;
 end

 time = time + Dt;

 if(istep==1)
  Handle1 = plot(x,vinit,'ko-');
  set(Handle1, 'erasemode','xor');
```

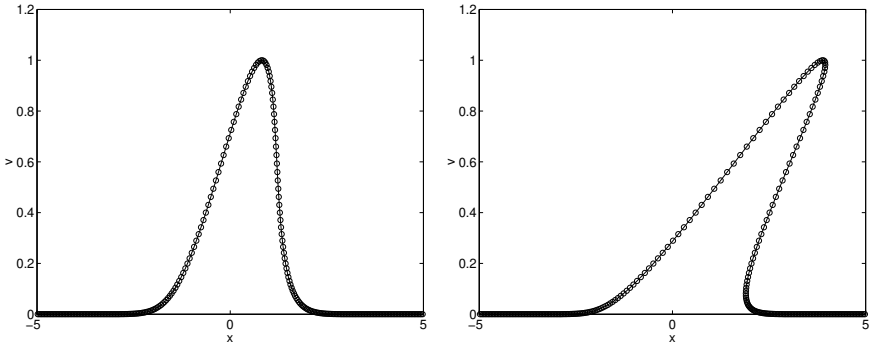


FIGURE 2.7.1 Snapshots of an evolving particle distribution representing the solution of the Burgers equation.

```
  axis([-5 5 -0.0 1.2])
  xlabel('x','fontsize',15)
  ylabel('v','fontsize',15)
  set(gca,'fontsize',15)
 else
  set(Handle1,'XData',x,'YData',vinit);
  pause(0.2)
  drawnow
 end

%-----
end
%-----
```

Two snapshots of an evolving distribution are shown in Figure 2.7.1. Because particles with higher velocity move faster to the right, the initial Gaussian distribution evolves into a discontinuous and then a multi-valued front.

Exercise

2.7.1 Run the code for a particle velocity distribution of your choice and discuss the results.

Chapter 3

Kronecker tensor product

The Kronecker tensor product of two matrices is a new matrix that encapsulates all possible products of the elements of two matrices in a a particular and carefully chosen order. In practice, the Kronecker tensor product allows us to implement elegant constructions that simplify numerical implementations. Applications can be found in matrix calculus, numerical solutions of differential equations, system theory and identification, and other related disciplines.

In this chapter, we review the basic properties of the Kronecker tensor product and discuss selected applications with reference to the numerical solution of the Poisson equation and the graph Laplacian of a square or toroidal lattice, including the Möbius strip.

3.1 Kronecker tensor product and sum

Consider an $M \times N$ matrix, $\mathbf{A}$, and another $L \times K$ matrix, $\mathbf{B}$. The Kronecker tensor product of these matrices is an $(ML) \times (NK)$ matrix,

$$\mathbf{C} = \mathbf{A} \otimes \mathbf{B}. \tag{3.1.1}$$

The matrix $\mathbf{C}$ arises by multiplying the matrix $\mathbf{B}$ with each element of the matrix $\mathbf{A}$, and then making space to accommodate the $L \times K$ sub-matrices, as follows:

$$\mathbf{C} = \begin{bmatrix} A_{11}\mathbf{B} & \cdots & A_{1N}\mathbf{B} \\ \vdots & \vdots & \vdots \\ A_{M1}\mathbf{B} & \cdots & A_{MM}\mathbf{B} \end{bmatrix}. \tag{3.1.2}$$

We see that $\mathbf{C}$ is a block matrix where each block is a multiple of $\mathbf{B}$ multiplied by each element of $\mathbf{A}$. By construction, the matrix $\mathbf{C}$ encapsulates all possible combinations of products of the elements of $\mathbf{A}$ and $\mathbf{B}$.

3.1.1 Computer code

The following Matlab function named *kron_cp*, located in directory MATENS of SEMLIB, generates the matrix $\mathbf{C}$ from given matrices $\mathbf{A}$ and $\mathbf{B}$:

```
function C =  kron_cp (M,N,A ...
                      ,L,K,B)

%---
% Kronecker product of two matrices
%---

i = 0;

for iA=1:M
 for iB=1:L

  i = i+1;
  j = 0;

  for jA=1:N
   for jB=1:K
     j = j+1;
     C(i,j) = A(iA,jA)*B(iB,jB);
   end
  end

 end
end

%---
% done
%---
```

```
return
```

In this implementation, the matrix dimensions, M, N, K, L, are supplied in the input. As an alternative, these dimensions can be deduced inside the function using native Matlab functions.

3.1.2 kron of Matlab

An internal Matlab function named *kron* generates the Kronecker tensor product $\mathbf{C} = \mathbf{A} \otimes \mathbf{B}$ from two given matrices, $\mathbf{A}$ and $\mathbf{B}$. The following Matlab script named *matens*, located in directory MATENS of SEMLIB, illustrates the construction:

```
%---
% tensor product of two matrices
%---

A = [-1 0 1;
     -2 1 2];

B = [1;
     2;
     3];

kron(A,B)
```

Running the code generates the following output:

```
    -1     0     1
    -2     0     2
    -3     0     3
    -2     1     2
    -4     2     4
    -6     3     6
```

In this case, $M = 2$, $N = 3$, $L = 3$, $K = 1$, and the tensor product is a 6×3 matrix.

As a second example, we run the following script:

```
A = [ 1 9;
     -2 8]

B = [1 3;
     2 4;
     3 5]

C = kron(A,B)
```

and obtain the following output:

```
C =
     1     3     9    27
     2     4    18    36
     3     5    27    45
    -2    -6     8    24
    -4    -8    16    32
    -6   -10    24    40
```

In this case, $M = 2$, $N = 2$, $L = 3$, $K = 2$, and the tensor product is a 6×4 matrix.

3.1.3 Properties of the Kronecker tensor product

The Kronecker tensor product satisfies familiar associative, left- and right-distributive properties with respect to addition and multiplication, as shown in Table 3.1.1.

3.1.4 Square matrices

Let $\mathbf{A}$ be a $M \times M$ square matrix, and $\mathbf{B}$ be another $L \times L$ square matrix. We find that the inverses are related by

$$(\mathbf{A} \otimes \mathbf{B})^{-1} = \mathbf{A}^{-1} \otimes \mathbf{B}^{-1}, \tag{3.1.3}$$

where the superscript -1 denotes the matrix inverse. The traces of the matrices are related by

$$\text{trace}(\mathbf{A} \otimes \mathbf{B}) = \text{trace}(\mathbf{A}) \times \text{trace}(\mathbf{B}), \tag{3.1.4}$$

where $\times$ denotes scalar multiplication. The determinants are related by

$$\det(\mathbf{A} \otimes \mathbf{B}) = \det{}^{L}(\mathbf{A}) \times \det{}^{M}(\mathbf{B}). \tag{3.1.5}$$

$$(\mathbf{A} \otimes \mathbf{B}) \otimes \mathbf{C} = \mathbf{A} \otimes (\mathbf{B} \otimes \mathbf{C})$$

$$(\mathbf{A} + \mathbf{B}) \otimes \mathbf{C} = \mathbf{A} \otimes \mathbf{C} + \mathbf{B} \otimes \mathbf{C}$$

$$\mathbf{A} \otimes (\mathbf{B} + \mathbf{C}) = \mathbf{A} \otimes \mathbf{B} + \mathbf{A} \otimes \mathbf{C}$$

$$(\alpha\mathbf{A}) \otimes \mathbf{B} = \mathbf{A} \otimes (\alpha\mathbf{B}) = \alpha(\mathbf{A} \otimes \mathbf{B})$$

$$(\mathbf{A} \otimes \mathbf{B})^{\mathrm{T}} = \mathbf{A}^{\mathrm{T}} \otimes \mathbf{B}^{\mathrm{T}}$$

$$(\mathbf{A} \otimes \mathbf{B}) \cdot (\mathbf{P} \otimes \mathbf{Q}) = (\mathbf{A} \cdot \mathbf{P}) \otimes (\mathbf{B} \cdot \mathbf{Q})$$

TABLE 3.1.1 Properties of the Kronecker product, where α is a scalar, the superscript T denotes the matrix transpose, where $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{P}$, $\mathbf{Q}$ are pairs of matrices with appropriate dimensions.

This property can be understood when $\mathbf{B}$ is the unit matrix.

3.1.5 *Eigenvalues and eigenvectors*

Let $\mathbf{a}$ be an eigenvector of a square matrix $\mathbf{A}$ with eigenvalue λ_A, and $\mathbf{b}$ be an eigenvector of a square matrix $\mathbf{B}$ with eigenvalue λ_B, so that

$$\mathbf{A} \cdot \mathbf{a} = \lambda_A \, \mathbf{a}, \qquad \mathbf{B} \cdot \mathbf{b} = \lambda_B \, \mathbf{b}. \tag{3.1.6}$$

It can be shown by direct substitution that

$$\mathbf{c} = \mathbf{a} \otimes \mathbf{b} = \begin{bmatrix} a_1 \, \mathbf{b} \\ a_2 \, \mathbf{b} \\ \vdots \\ a_M \, \mathbf{b} \end{bmatrix} \tag{3.1.7}$$

is an eigenvector of $\mathbf{A} \otimes \mathbf{B}$ with corresponding eigenvalue $\lambda_C = \lambda_A \lambda_B$, where $\mathbf{a}$ and $\mathbf{b}$ are regarded as slender vertical arrays.

3.1.6 Kronecker sum

Let $\mathbf{E}$ be a $M \times M$ square matrix, and $\mathbf{F}$ be another $L \times L$ square matrix. The Kronecker sum of these two matrices is defined as

$$\mathbf{E} \oplus \mathbf{F} \equiv \mathbf{I}_L \otimes \mathbf{E} + \mathbf{F} \otimes \mathbf{I}_M, \tag{3.1.8}$$

where $\mathbf{I}_L$ is the $L \times L$ identity matrix and $\mathbf{I}_M$ is the $M \times M$ identity matrix. Using the definition of the Kronecker product, we find that

$$\mathbf{E} \oplus \mathbf{F} = \begin{bmatrix} \mathbf{E} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{E} & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{E} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{E} \end{bmatrix} + \begin{bmatrix} F_{11}\mathbf{I}_M & \cdots & F_{1L}\mathbf{I}_M \\ \vdots & \vdots & \vdots \\ F_{L1}\mathbf{I}_M & \cdots & F_{LL}\mathbf{I}_M \end{bmatrix}. \tag{3.1.9}$$

The first matrix on the right-hand side contains L diagonal blocks.

The Kronecker sum of two matrices is useful for constructing finite-difference coefficients matrices, Laplacian matrices of Cartesian networks or grids, as discussed in the remainder of this chapter.

Let $\mathbf{e}$ be an eigenvector of $\mathbf{E}$ with eigenvalue λ_E, and $\mathbf{f}$ be an eigenvector of $\mathbf{F}$ with eigenvalue λ_F, so that

$$\mathbf{E} \cdot \mathbf{e} = \lambda_E \, \mathbf{e}, \qquad \mathbf{F} \cdot \mathbf{f} = \lambda_F \, \mathbf{f}. \tag{3.1.10}$$

Now consider the vector

$$\mathbf{g} \equiv \mathbf{f} \otimes \mathbf{e} = \begin{bmatrix} f_1\mathbf{e} \\ f_2\mathbf{e} \\ \vdots \\ f_{M-1}\mathbf{e} \\ f_M\mathbf{e} \end{bmatrix}, \tag{3.1.11}$$

where $\mathbf{e}$ and $\mathbf{f}$ are regarded as slender vertical arrays. We find that

$$(\mathbf{I}_L \otimes \mathbf{U}) \cdot \mathbf{g} = \lambda_E \, \mathbf{g}, \qquad (\mathbf{F} \otimes \mathbf{I}_M \cdot \mathbf{g} = \lambda_F \, \mathbf{g}, \tag{3.1.12}$$

which shows that $\mathbf{g}$ is an eigenvector of $\mathbf{E} \oplus \mathbf{F}$ with corresponding eigenvalue $\lambda_E + \lambda_F$.

Exercises

3.1.1 Show that (a) the Kronecker product of two diagonal matrices is a diagonal matrix, (b) the Kronecker product of two upper triangular matrices is an upper triangular matrix, and (c) the Kronecker product of two lower triangular matrices is an upper triangular matrix.

3.1.2 Confirm the properties shown in Table 3.1.1 for matrices of your choice.

3.2 Poisson equation on a Cartesian grid

Consider the Poisson equation in the xy plane for a function, $f(x, y)$,

$$\nabla^2 f + s = 0, \tag{3.2.1}$$

where $s(x, y)$ is a specified source term. The solution domain is a rectangle, as shown in Figure 3.2.1.

3.2.1 Boundary conditions

The Dirichlet boundary condition is specified along the left, bottom, and top edges,

$$f(x = a_x) = w(y), \quad f(y = a_y) = z(x), \quad f(y = b_y) = v(x), \tag{3.2.2}$$

and the Neumann boundary condition is specified along the left edge,

$$\frac{\partial f}{\partial x} = q(y) \quad \text{at} \quad x = b_x, \tag{3.2.3}$$

as shown in Figure 3.2.1, where w, z, v, and q are given functions.

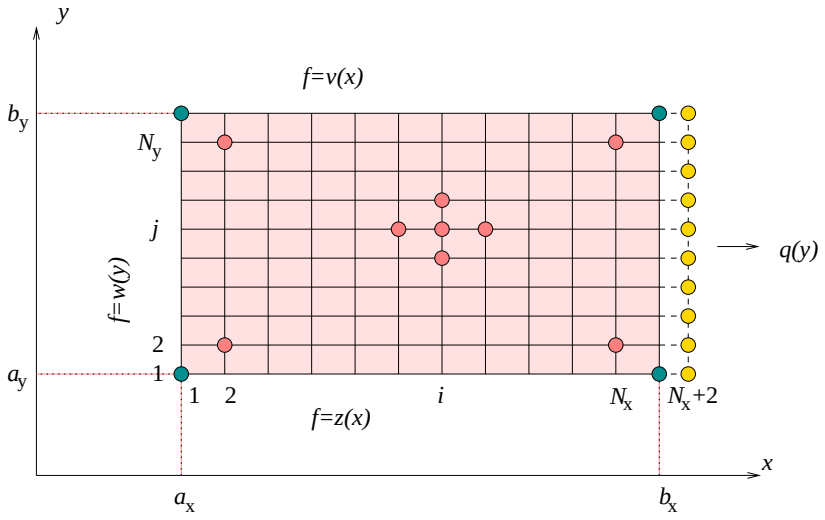


FIGURE 3.2.1 Illustration of a uniform finite-difference grid for solving the Poisson equation in a rectangular domain. Dirichlet or Neumann boundary conditions are specified along the four edges.

3.2.2 *Finite-difference grid*

To implement a finite-difference method, we divide the x interval, (a_x, b_x), into N_x evenly spaced subintervals separated by the spacing $\Delta x = (b_x - a_x)/N_x$, and draw the vertical grid lines $x = x_i$, where

$$x_i = a_x + (i-1)\,\Delta x \tag{3.2.4}$$

for $i = 1, \ldots, N_x + 1$, as shown in Figure 3.2.1. Correspondingly, we divide the y interval (a_y, b_y) into N_y evenly spaced subintervals separated by the spacing $\Delta y = (b_y - b_y)/N_y$, and draw the horizontal grid lines $y = y_j$, where

$$y_j = a_y + (j-1)\,\Delta y \tag{3.2.5}$$

for $j = 1, \ldots, N_y + 1$, as shown in Figure 3.2.1.

The intersections of vertical and horizontal grid lines define grid nodes. For simplicity, we denote the value of the function f at the

(i, j) node as

$$f_{i,j} \equiv f(x_i, y_j). \tag{3.2.6}$$

The Dirichlet boundary conditions (3.2.2) on the left, bottom, and top require that

$$f_{1,j} = w(y_j), \qquad f_{i,1} = z(x_i), \qquad f_{i,N_y+1} = v(x_i). \tag{3.2.7}$$

Our objective is to compute the unknown nodal values, $f_{i,j}$, for $i = 2, \ldots, N_x + 1$, and $j = 2, \ldots, N_y$, comprising a set of

$$K = N_x(N_y - 1) \tag{3.2.8}$$

unknowns.

3.2.3 *Discretization*

To build a system of linear equations for the unknowns, we apply the governing differential equation at the (i, j) node and approximate the second partial derivatives with central differences, setting

$$\left(\frac{\partial^2 f}{\partial x^2}\right)_{i,j} \simeq \frac{f_{i-1,j} - 2f_{i,j} + f_{i+1,j}}{\Delta x^2} \tag{3.2.9}$$

and

$$\left(\frac{\partial^2 f}{\partial y^2}\right)_{i,j} \simeq \frac{f_{i,j-1} - 2f_{i,j} + f_{i,j+1}}{\Delta y^2}, \tag{3.2.10}$$

with an error on the order of Δx^2 and Δy^2.

Now we apply the differential equations at the (i, j) node and derive the difference equation

$$\frac{f_{i-1,j} - 2\,f_{i,j} + f_{i+1,j}}{\Delta x^2} + \frac{f_{i,j-1} - 2\,f_{i,j} + f_{i,j+1}}{\Delta y^2} + s_{i,j} = 0, \tag{3.2.11}$$

which can be rearranged into the equation

$$f_{i+1,j} - 2\,(1 + \beta)\,f_{i,j} + f_{i-1,j} + \beta\,f_{i,j+1} + \beta\,f_{i,j-1} = -\Delta x^2\,s_{i,j}, \tag{3.2.12}$$

where

$$\beta \equiv \left(\frac{\Delta x}{\Delta y}\right)^2. \tag{3.2.13}$$

The left-hand side of (3.2.12) is a weighted average of function values at a group of five neighboring nodes, forming a cross-shaped stencil, as shown in Figure 3.2.1

3.2.4 Algebraic balance

The difference equation (3.2.12) can be applied at the interior grid nodes for $i = 2, \ldots, N_x$ and $j = 2, \ldots, N_y$, to yield

$$K_{\text{int}} = (N_x - 1)(N_y - 1) \tag{3.2.14}$$

linear equations. However, the equation cannot be applied at the boundary nodes since some of the grid values will lie outside the solution domain. To achieve closure, we must supply a number of

$$K - K_{\text{int}} = N_x(N_y - 1) - (N_x - 1)(N_y - 1) = N_y - 1 \tag{3.2.15}$$

additional equations.

3.2.5 Neumann boundary condition

The missing equations implement the Neumann boundary condition on the right side of the plate, $x = b_x$. We want to implement this boundary condition with an error that is comparable to the size of Δx^2, which matches the error due to the discretization of the differential equation.

With this goal in mind, we extend the domain of solution beyond the physical boundary $x = b_x$, and introduce the phantom grid line, $x_{N+2} = b_x + \Delta x$, and associated values, $f_{N+2,j}$, as shown in Figure 3.2.1. The central difference approximation

$$\left(\frac{\partial f}{\partial x}\right)_{N_x+1,j} \simeq \frac{f_{N_x+2,j} - f_{N_x,j}}{2\,\Delta x} \tag{3.2.16}$$

carries a numerical error on the order of Δx^2. Using this approximation, we replace the boundary condition (3.2.3) with the difference equation

$$\frac{f_{N_x+2,j} - f_{N_x,j}}{2\,\Delta x} = q_j, \tag{3.2.17}$$

which can be rearranged to give

$$f_{N+2,j} = f_{N,j} + 2\,\Delta x\, q_j \tag{3.2.18}$$

for $j = 2, \ldots, N_y$.

Having introduced phantom nodes, we may apply the difference equation (3.2.12) at the right-edge boundary nodes corresponding to $i = N_x + 1$. Using (3.2.18) to express the exterior value, $f_{N_x+2,j}$, in terms of the interior value $f_{N_x,j}$, we derive the difference equation

$$\begin{gathered} -2(1+\beta) f_{N_x+1,j} + 2 f_{N_x,j} + \beta f_{N_x+1,j+1} + \beta f_{N_x+1,j-1} \\ = -\Delta x^2\, s_{N_x+1,j} - 2\Delta x\, q_j \end{gathered} \tag{3.2.19}$$

for $j = 2, \ldots, N_y$.

3.2.6 *Compiling a linear system*

Equations (3.2.12) and (3.2.19) provide us with the desired system of $N_x(N_y - 1)$ linear algebraic equations for the unknown nodal values. To obtain the standard form of a linear system, we compile the values $f_{i,j}$ for $i = 2, \ldots, N_x + 1$ and $j = 2, \ldots, N_y$ into a long $N_x(N_y - 1)$-dimensional vector $\mathbf{s}$, row by row starting from the bottom, obtaining

$$\mathbf{f} = \begin{bmatrix} \mathbf{f}^{(2)} \\ \mathbf{f}^{(3)} \\ \vdots \\ \mathbf{f}^{(N_y-1)} \\ \mathbf{f}^{(N_y)} \end{bmatrix}. \tag{3.2.20}$$

The sub-arrays,

$$\mathbf{f}^{(2)} = \begin{bmatrix} f_{2,2} \\ f_{3,2} \\ \vdots \\ f_{N_x,2} \\ f_{N_x+1,2} \end{bmatrix}, \quad \ldots, \quad \mathbf{f}^{(N_y)} = \begin{bmatrix} f_{2,N_y} \\ f_{3,N_y} \\ \vdots \\ f_{N_x,N_y} \\ f_{N_x+1,N_y} \end{bmatrix}, \tag{3.2.21}$$

encapsulate horizontal profiles. There is nothing special about the row by row compilation, and a column by column compilation will work just as well.

Compiling the derived finite-difference equations also row by row starting from the bottom, we formulate the linear system

$$\mathbf{A}\cdot\mathbf{f}=\mathbf{b}, \tag{3.2.22}$$

where $\mathbf{A}$ is a block-tridiagonal matrix consisting of diagonal and tridiagonal blocks,

$$\mathbf{A}=\begin{bmatrix} \mathbf{T} & \mathbf{D} & \mathbf{0} & \cdots & \cdots & \cdots & \cdots & \cdots \\ \mathbf{D} & \mathbf{T} & \mathbf{D} & \mathbf{0} & \cdots & \cdots & \cdots & \cdots \\ \mathbf{0} & \mathbf{D} & \mathbf{T} & \mathbf{D} & \mathbf{0} & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \mathbf{0} & \mathbf{D} & \mathbf{T} & \mathbf{D} & \mathbf{0} \\ \cdots & \cdots & \cdots & \cdots & \mathbf{0} & \mathbf{D} & \mathbf{T} & \mathbf{D} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \mathbf{0} & \mathbf{D} & \mathbf{T} \end{bmatrix}. \tag{3.2.23}$$

We have introduced the $N_x\times N_x$ tridiagonal matrix

$$\mathbf{T}=\begin{bmatrix} -2\,(1+\beta) & 1 & \cdots & \cdots & \cdots \\ 1 & -2\,(1+\beta) & 1 & \cdots & \cdots \\ 0 & 1 & 1 & 0 & \cdots \\ \cdots & \cdots & \ddots & \cdots & \cdots \\ \cdots & 1 & -2\,(1+\beta) & 1 & 0 \\ \cdots & 0 & 1 & -2\,(1+\beta) & 1 \\ \cdots & \cdots & 0 & 2 & -2\,(1+\beta) \end{bmatrix} \tag{3.2.24}$$

and the $N_x\times N_x$ diagonal matrix

$$\mathbf{D}=\begin{bmatrix} \beta & 0 & \cdots & \cdots & \cdots & \cdots \\ 0 & \beta & 0 & \cdots & \cdots & \cdots \\ \cdots & 0 & \beta & 0 & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & 0 & \beta & 0 \\ \cdots & \cdots & \cdots & \cdots & 0 & \beta \end{bmatrix}. \tag{3.2.25}$$

The entry of *two* in the penultimate element of the last row of $\mathbf{T}$ implements the Neumann boundary condition at the right edge of the solution domain.

3.2.7 *Right-hand side*

The right-hand side of the linear system is the partitioned vector

$$\mathbf{b} = \begin{bmatrix} \mathbf{b}^{(2)} \\ \mathbf{b}^{(3)} \\ \vdots \\ \mathbf{b}^{(N_y-1)} \\ \mathbf{b}^{(N_y)} \end{bmatrix}, \tag{3.2.26}$$

where

$$\mathbf{b}^{(2)} = -\begin{bmatrix} \beta\, z(x_2) + \Delta x^2\, s_{2,2} + w(y_2) \\ \beta\, z(x_3) + \Delta x^2\, s_{3,2} \\ \vdots \\ \beta\, z(x_{N_x}) + \Delta x^2\, s_{N_x,2} \\ \beta\, z(x_{N_x+1}) + \Delta x^2\, s_{N_x+1,2} + 2\,\Delta x\, q(y_2) \end{bmatrix} \tag{3.2.27}$$

for the first block corresponding to $j = 2$,

$$\mathbf{b}^{(j)} = -\begin{bmatrix} \Delta x^2\, s_{2,j} + w(y_j) \\ \Delta x^2\, s_{3,j} \\ \vdots \\ \Delta x^2\, s_{N_x,j} \\ \Delta x^2\, s_{N_x+1,j} + 2\,\Delta x\, q(y_j) \end{bmatrix} \tag{3.2.28}$$

for the intermediate blocks corresponding to $j = 3, \ldots, N_y - 1$, and

$$\mathbf{b}^{(N_y)} = -\begin{bmatrix} \beta\, v(x_2) + \Delta x^2\, s_{2,N_y} + w(y_{N_y}) \\ \beta\, v(x_3) + \Delta x^2\, s_{3,N_y} \\ \vdots \\ \beta\, v(x_{N_x}) + \Delta x^2\, s_{N_x,N_y} \\ \beta\, v(x_{N_x+1}) + \Delta x^2\, s_{N_x+1,N_y} + 2\,\Delta x\, q(y_{N_y}) \end{bmatrix} \tag{3.2.29}$$

for the last block corresponding to $j = N_y$. To interpret these structures, we make the following observations:

- The first term in the first and last blocks, $\mathbf{b}^{(2)}$ and $\mathbf{b}^{(N_y)}$, implement the Dirichlet boundary condition at the bottom and top sides of the rectangular solution domain.

- The last term in the first entry of each block, $\mathbf{b}^{(j)}$ for $j = 2, \ldots, N_y$, implement the Dirichlet boundary condition at the left side of the rectangular solution domain.

- The last term in the last entry of each block, $\mathbf{b}^{(j)}$ for $j = 2, \ldots, N_y$, implements the Dirichlet boundary condition at the right edge of the solution domain.

The linear system (3.2.22) can be solved by a standard direct numerical method or by special-purpose iterative numerical methods.

3.2.8 Robin boundary condition

Suppose that the Dirichlet boundary condition specified in (3.2.3) on the left side is replaced by the Robin boundary condition

$$\frac{\partial f}{\partial x} = q(y) = -\xi\,(f - f_0) \quad \text{at} \quad x = b_x, \tag{3.2.30}$$

where ξ and f_0 are specified constants. To implement this boundary condition, we perform two modifications:

- Add to the last diagonal element of the matrix $\mathbf{T}$ shown in (3.2.24) the term $-2\Delta x\,\xi$.

- Replace the last term involving $q(y_j)$ in the last entry of each block, $\mathbf{b}^{(j)}$ for $j = 2, \ldots, N_y$, with $2\Delta x\,\xi f_0$.

The homogeneous Neumann boundary condition discussed earlier in this section arises for $\xi = 0$.

Exercise

3.2.1 When the Neumann boundary condition is imposed along the four edges. we obtain a linear system of $(N_x + 1)(N_y + 1)$ unknowns. Display the corresponding structure of the coefficient matrix $\mathbf{A}$ and right-hand side, $\mathbf{b}$.

3.3 Finite-difference matrix as a tensor sum

In fact, the finite-difference matrix $\mathbf{A}$ shown in (3.2.23) can be constructed as the Kronecker tensor sum of two matrices, $\mathbf{U}^{\mathrm{DN}}$ and $\beta\mathbf{V}^{\mathrm{DD}}$,

$$\mathbf{A} = \mathbf{U}^{\mathrm{DN}} \oplus (\beta\mathbf{V}^{\mathrm{DD}}). \tag{3.3.1}$$

where we recall that $\beta \equiv (\Delta x/\Delta y)^2$. We have introduced an $N_x \times N_x$ tridiagonal matrix associated with the Dirichlet–Neumann boundary conditions at the left and right sides of the solution domain,

$$\mathbf{U}^{\mathrm{DN}} = \begin{bmatrix} -2 & 1 & \cdots & \cdots & \cdots & \cdots \\ 1 & -2 & 1 & 0 & \cdots & \cdots \\ 0 & 1 & -2 & 1 & 0 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & 0 & 1 & -2 & 1 & 0 \\ \cdots & \cdots & 0 & 1 & -2 & 1 \\ \cdots & \cdots & \cdots & 0 & 2 & -2 \end{bmatrix}, \tag{3.3.2}$$

and a $(N_y - 1) \times (N_y - 1)$ tridiagonal Toeplitz matrix associated with the Dirichlet–Dirichlet boundary condition at the bottom and top of the solution domain,

$$\mathbf{V}^{\mathrm{DD}} = \begin{bmatrix} -2 & 1 & \cdots & \cdots & \cdots & \cdots \\ 1 & -2 & 1 & 0 & \cdots & \cdots \\ 0 & 1 & -2 & 1 & 0 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & 0 & 1 & -2 & 1 & 0 \\ \cdots & \cdots & 0 & 1 & -2 & 1 \\ \cdots & \cdots & \cdots & 0 & 1 & -2 \end{bmatrix}. \tag{3.3.3}$$

By definition, a Toeplitz matrix has constant elements along each diagonal. Note the *two* in the subdiagonal element in the last row of $\mathbf{U}^{\mathrm{DN}}$ implementing the Neumann boundary condition at the right edge.

Resolving the Kronecker tensor sum into Kronecker products, we obtain

$$\mathbf{A} = \mathbf{I}_{N_y-1} \otimes \mathbf{U}^{\mathrm{DN}} + \beta\, \mathbf{V}^{\mathrm{DD}} \otimes \mathbf{I}_{N_x}, \tag{3.3.4}$$

where $\mathbf{I}_{N_y-1}$ is the $(N_y-1)\times(N_y-1)$ identity matrix and $\mathbf{I}_{N_x}$ is the $N_x \times N_x$ identity matrix. Explicitly, the two constituents of the finite-difference matrix $\mathbf{A}$ involved in the Kronecker sum are given by

$$\mathbf{I}_{N_y-1} \otimes \mathbf{U}^{\mathrm{DN}} = \begin{bmatrix} \mathbf{U}^{\mathrm{DN}} & \cdots & \mathbf{0} \\ \vdots & \mathbf{U}^{\mathrm{DN}} & \vdots \\ \mathbf{0} & \cdots & \mathbf{U}^{\mathrm{DN}} \end{bmatrix} \tag{3.3.5}$$

and

$$\mathbf{V}^{\mathrm{DD}} \otimes \mathbf{I}_{N_x} = \begin{bmatrix} -2\mathbf{I}_{N_x} & \mathbf{I}_{N_x} & \cdots & \cdots & \cdots & \cdots \\ \mathbf{I}_{N_x} & -2\mathbf{I}_{N_x} & \mathbf{I}_{N_x} & \mathbf{0} & \cdots & \cdots \\ \mathbf{0} & \mathbf{I}_{N_x} & -2\mathbf{I}_{N_x} & \mathbf{I}_{N_x} & \mathbf{0} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \mathbf{0} & \mathbf{I}_{N_x} & -2\mathbf{I}_{N_x} & \mathbf{I}_{N_x} & \mathbf{0} \\ \cdots & \cdots & \mathbf{0} & \mathbf{I}_{N_x} & -2\mathbf{I}_{N_x} & \mathbf{I}_{N_x} \\ \cdots & \cdots & \cdots & \mathbf{0} & \mathbf{I}_{N_x} & -2\mathbf{I}_{N_x} \end{bmatrix}. \tag{3.3.6}$$

The first constituent shown in (3.3.5) involves N_x diagonal blocks.

3.3.1 DNDD coefficient matrix as a tensor product

The following Matlab code named *matens_DNDD*, located in directory MATENS of SEMLIB, generates the coefficient matrix $\mathbf{A}$ using the Kronecker tensor product:

```
Nx = 6;
Ny = 12;

%---
% tridiagonal in x
%---

U(1,1) = -2; U(1,2) = 1;
U(Nx,Nx) = -2; U(Nx,Nx-1) = 2;

for i=2:Nx-1
```

```
 U(i,i) = -2;
 U(i,i-1) = 1;
 U(i,i+1) = 1;
end

%---
% tridiagonal in y
%---

V(1,1) = -2; V(1,2) = 1;
V(Ny-1,Ny-1) = -2; V(Ny-1,Ny-2) = 1;

for i=2:Ny-2
 V(i,i) = -2;
 V(i,i-1) = 1;
 V(i,i+1) = 1;
end

beta = 3.0;   % arbitrary

A = kron(eye(Ny-1),U) + beta*kron(V,eye(Nx));
A
```

Running the code generates the following output, as instructed by the last line of the code:

```
  A =

    -8   1   0   3   0   0   0   0   0
     1  -8   1   0   3   0   0   0   0
     0   2  -8   0   0   3   0   0   0
     3   0   0  -8   1   0   3   0   0
     0   3   0   1  -8   1   0   3   0
     0   0   3   0   2  -8   0   0   3
     0   0   0   3   0   0  -8   1   0
     0   0   0   0   3   0   1  -8   1
     0   0   0   0   0   3   0   2  -8
```

Cursory inspection reveals that this display is consistent with the structure of the matrix shown in (3.2.23).

3.3.2 Solution of the DNDD Poisson equation

The following Matlab code named *poisson_DNDD*, located in directory MATENS of SEMLIB, generates and solves linear system discussed in Section 3.2:

```
%----
% Solution of Poisson equation
% in a rectangle confined between
% -a < x < a and -b < y < b
% with DNDD boundary conditions
%----

Nx = 16;
Ny = 14;

a = 1.0;
b = 0.5;

%---
% prepare
%---

Dx = 2.0*a/Nx; Dy = 2.0*b/Ny;

beta = Dx^2/Dy^2;

Dx2 = 2.0*Dx;

%---
% grid
%---

for i=1:Nx+1
 x(i) = -a + (i-1)*Dx;
end

for i=1:Ny+1
 y(i) = -b + (i-1)*Dy;
end
```

```
%---
% boundary conditions (typical)
%---

for i=1:Nx+1
 z(i) = 0.2; % bottom (Dirichlet)
 v(i) = 0.1; % top    (Dirichlet)
end

for j=1:Ny+1
 w(j) = 0.0; % left  (Dirichlet)
 q(j) = 2.0; % right (Neumann)
end

%---
% specify the source (typical)
%---

for j=2:Ny
 for i=2:Nx+1
  source(i,j) = 10.0*x(i)*y(j);
 end
end

%---
% tridiagonal in x
%---

U(1,1) = -2; U(1,2) = 1;
U(Nx,Nx) = -2; U(Nx,Nx-1) = 2;

for i=2:Nx-1
 U(i,i) = -2; U(i,i-1) = 1; U(i,i+1) = 1;
end

%---
% tridiagonal in y
%---
```

```
V(1,1) = -2; V(1,2) = 1;
V(Ny-1,Ny-1) = -2; V(Ny-1,Ny-2) = 1;

for i=2:Ny-2
 V(i,i) = -2; V(i,i-1) = 1; V(i,i+1) = 1;
end

%---
% coefficient matrix
%---

A = kron(eye(Ny-1),U) + beta*kron(V,eye(Nx));

%---
% right-hand side
%---

Ic = 0; % counter

for j=2:Ny
 for i=2:Nx+1

  Ic = Ic+1;

  rhs(Ic) = source(i,j)*Dx^2;

  if(j==2)
    rhs(Ic) = rhs(Ic) - beta*z(i);
  end

  if(j==Ny)
    rhs(Ic) = rhs(Ic) - beta*v(i);
  end

  if(i==2)
    rhs(Ic) = rhs(Ic) - w(j);
  end
```

```
  if(i==Nx+1)
    rhs(Ic) = rhs(Ic) - Dx2*q(j);
  end

 end
end

%---
% solve
%---

SOL = rhs/A';

%---
% distribute
%---

Ic = 0;

for j=2:Ny
 for i=2:Nx+1
  Ic = Ic+1;
  f(i,j) = SOL(Ic);
 end
end

%---
% boundary conditions
%---

for i=1:Nx+1
 f(i,1) = z(i);    % bottom
 f(i,Ny+1) = v(i); % top
end

for j=1:Ny+1
 f(1,j) = w(j);     %  left
end
```

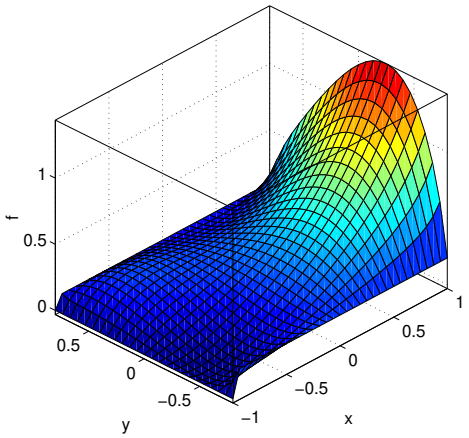


FIGURE 3.3.1 Solution of the Poisson equation with random forcing computed by a finite-difference method in a rectangular domain subject to the homogeneous Dirichlet condition along three edges and the homogeneous Neumann condition along the fourth edge. The coefficient matrix is constructed conveniently as the Kronecker tensor product of two matrices.

```
%---
% plot
%---

surf(x,y,f')
```

The graphics generated by the code is shown in Figure 3.3.1. Note that the homogeneous Dirichlet boundary condition (zero value) along three edges and the homogeneous Neumann boundary condition (zero normal derivative) along fourth edge are satisfied.

3.3.3 *Eigenvalues and eigenvectors of the coefficient matrix*

For convenience, we denote $\mathbf{U}^{\mathrm{DN}}$ by $\mathbf{U}$ and $\mathbf{V}^{\mathrm{DN}}$ by $\mathbf{V}$. Let the N_x-dimensional vertical array $\boldsymbol{\phi}$ contain an eigenvector of the constituent matrix $\mathbf{U}$, and the $(N_y - 1)$-dimensional vertical array $\boldsymbol{\psi}$ contain an

eigenvector of the constituent matrix $\mathbf{V}$, such that

$$\mathbf{U} \cdot \boldsymbol{\phi} = \lambda_\phi \, \boldsymbol{\phi}, \qquad \mathbf{V} \cdot \boldsymbol{\psi} = \lambda_\psi \, \boldsymbol{\psi}, \tag{3.3.7}$$

where λ_ψ and λ_ψ are eigenvalues. The array $\boldsymbol{\phi}$ is regarded as a slender $N_x \times 1$ matrix, and the array ψ is regarded as another slender $(N_y - 1) \times 1$ matrix. The Kronecker matrix product of the eigenvector arrays is a slender $N_x(N_y - 1) \times 1$ array given by

$$\boldsymbol{\chi} \equiv \boldsymbol{\psi} \otimes \boldsymbol{\phi} = \begin{bmatrix} \psi_1 \boldsymbol{\phi} \\ \vdots \\ \psi_{N_y - 1} \boldsymbol{\phi} \end{bmatrix}, \tag{3.3.8}$$

where $\otimes$ is the Kronecker matrix product. Referring to (3.3.5) and (3.3.6), we find that

$$(\mathbf{I}_{N_y - 1} \otimes \mathbf{U}) \cdot \boldsymbol{\phi} = \lambda_\phi \, \boldsymbol{\chi}, \qquad (\mathbf{V} \otimes \mathbf{I}_{N_x}) \cdot \boldsymbol{\psi} = \lambda_\psi \, \boldsymbol{\chi}. \tag{3.3.9}$$

By combining these results and recalling that

$$\mathbf{A} = \mathbf{I}_{N_y - 1} \otimes \mathbf{U} + \beta \, \mathbf{V} \otimes \mathbf{I}_{N_x}, \tag{3.3.10}$$

we deduce that $\boldsymbol{\chi}$ is an eigenvector of the coefficient matrix $\mathbf{A}$ with corresponding eigenvalue

$$\lambda = \lambda_\phi + \beta \, \lambda_\psi, \tag{3.3.11}$$

as discussed in Section 3.1. Similar results are obtained for other constituent matrices implementing different boundary conditions.

Exercise

3.3.1 Reproduce the results shown in Figure 3.3.1 with an inhomogeneous (non-zero) Neumann boundary condition of your choice.

3.4 Finite-difference matrices as tensor products

In Sections 3.2 and 3.3 we discussed the Poisson equation in a rectangular domain with DNDD boundary conditions at the left, right, bottom, and top boundaries, where D stands for Dirichlet and N stands for Neumann. Other Dirichlet–Neumann combinations can be implemented by straightforward modifications. The Robin boundary condition (R) and the periodic boundary condition (P) can also be implemented by straightforward modifications, as discussed in the following few sections.

3.4.1 DDXX

The $N_x \times N_x$ matrix $\mathbf{U}^{\mathrm{DN}}$ shown in (3.3.2) corresponds to the Dirichlet–Neumann (DN) boundary condition along the x axis. For the DD boundary condition, we use the alternative $(N_x-1)\times(N_x-1)$ Toeplitz matrix

$$\mathbf{U}^{\mathrm{DD}} = \begin{bmatrix} -2 & 1 & \cdots & \cdots & \cdots & \cdots \\ 1 & -2 & 1 & 0 & \cdots & \cdots \\ 0 & 1 & -2 & 1 & 0 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & 0 & 1 & -2 & 1 & 0 \\ \cdots & \cdots & 0 & 1 & -2 & 1 \\ \cdots & \cdots & \cdots & 0 & 1 & -2 \end{bmatrix}, \tag{3.4.1}$$

which is a symmetric tridiagonal matrix with *ones* along the subdiagonal and superdiagonal lines.

3.4.2 DDDD

Assume that the Dirichlet boundary condition is specified around all four edges or a rectangular solution domain, (DDDD),

$$\begin{aligned} &f = w(y) \quad \text{at} \quad x = a_x, \qquad && f = q(y) \quad \text{at} \quad x = b_x, \\ &f = z(x) \quad \text{at} \quad y = a_y, \qquad && f = v(x) \quad \text{at} \quad y = b_y, \end{aligned} \tag{3.4.2}$$

where $w(y)$, $q(y)$, $z(x)$, and $v(x)$ are given functions along the left, right, bottom, and top edges. A solution vector, $\mathbf{f}$, can be compiled as

a sequence of horizontal layers from the second level, $j = 2$, up to the second-to-top level, $j = N_y$, into the partitioned solution vector

$$\mathbf{f} = \begin{bmatrix} \mathbf{f}^{(2)} \\ \mathbf{f}^{(3)} \\ \vdots \\ \mathbf{f}^{(N_y-1)} \\ \mathbf{f}^{(N_y)} \end{bmatrix}. \tag{3.4.3}$$

The sub-arrays,

$$\mathbf{f}^{(2)} \equiv \begin{bmatrix} f_{2,2} \\ f_{3,2} \\ \vdots \\ f_{N_x-1,2} \\ f_{N_x,2} \end{bmatrix}, \quad \ldots, \quad \mathbf{f}^{(N_2)} \equiv \begin{bmatrix} f_{2,N_2} \\ f_{3,N_2} \\ \vdots \\ f_{N_x-1,N_2-1} \\ f_{N_x,N_2} \end{bmatrix}, \tag{3.4.4}$$

encapsulate horizontal profiles. The solution vector, $\mathbf{f}$, consists of $(N_x - 1) \times (N_y - 1)$ scalar unknowns and requires an equal number of equations.

Applying the finite-difference equation at all interior nodes, we derive a system of linear equations

$$\mathbf{A} \cdot \mathbf{f} = \mathbf{b}. \tag{3.4.5}$$

The coefficient matrix is given by

$$\mathbf{A} = \mathbf{I}_{N_y+1} \otimes \mathbf{U}^{\mathrm{DD}} + \beta\, \mathbf{V}^{\mathrm{DD}} \otimes \mathbf{I}_{N_x+1}. \tag{3.4.6}$$

The right-hand side of the linear system is the partitioned array

$$\mathbf{b} = \begin{bmatrix} \mathbf{b}^{(2)} \\ \mathbf{b}^{(3)} \\ \vdots \\ \mathbf{b}^{(N_y-1)} \\ \mathbf{b}^{(N_y)} \end{bmatrix}, \tag{3.4.7}$$

where

$$\mathbf{b}^{(2)} = - \begin{bmatrix} \beta\, z(x_2) + \Delta x^2\, s_{2,2} + w(y_2) \\ \beta\, z(x_3) + \Delta x^2\, s_{3,2} \\ \vdots \\ \beta\, z(x_{N_x-1}) + \Delta x^2\, s_{N_x-1,2} \\ \beta\, z(x_{N_x}) + \Delta x^2\, s_{N_x,2} + q(y_2) \end{bmatrix} \tag{3.4.8}$$

for the first block corresponding to $j = 2$,

$$\mathbf{b}^{(j)} = - \begin{bmatrix} \Delta x^2\, s_{2,j} + w(y_j) \\ \Delta x^2\, s_{3,j} \\ \vdots \\ \Delta x^2\, s_{N_x-1,j} \\ \Delta x^2\, s_{N_x,j} + q(y_j) \end{bmatrix} \tag{3.4.9}$$

for the intermediate blocks corresponding to $j = 3, \ldots, N_y - 1$, and

$$\mathbf{b}^{(N_y)} = - \begin{bmatrix} \beta\, v(x_2) + \Delta x^2\, s_{2,N_y} + w(y_{N_y}) \\ \beta\, v(x_3) + \Delta x^2\, s_{3,N_y} \\ \vdots \\ \beta\, v(x_{N_x-1}) + \Delta x^2\, s_{N_x-1,N_y} \\ \beta\, v(x_{N_x}) + \Delta x^2\, s_{N_x,N_y} + q(y_{N_y}) \end{bmatrix} \tag{3.4.10}$$

for the last block corresponding to $j = N_y$. Note that $z(x)$ appears in the first block, $\mathbf{b}^{(2)}$, $v(x)$ appears in the last block, $\mathbf{b}^{(N_y)}$, $w(y)$ appears in the first entry of each block, and $q(y)$ appears in the last entry of each block.

3.4.3 DDDD code

The following Matlab code named *poisson_DDDD*, located in directory MATENS of SEMLIB, generates and solves a linear system for homogeneous (zero value) Dirichlet boundary conditions:

```
%============================
% Solution of Poisson equation
% in a rectangle confined between
% -a < x < a and -b < y < b
```

```
% with DDDD boundary conditions
%
% f = w at the left
% f = q at the right
% f = z at the bottom
% f = v at the top
%============================

Nx = 16;
Ny = 14;

a = 1.0;
b = 0.5;

%---
% prepare
%---

Dx = 2.0*a/Nx;
Dy = 2.0*b/Ny;

beta = Dx^2/Dy^2;

%---
% grid
%---

for i=1:Nx+1
 x(i) = -a + (i-1)*Dx;
end

for j=1:Ny+1
 y(j) = -b + (j-1)*Dy;
end

%---
% boundary conditions
%---
```

```
for i=1:Nx+1
 z(i) = 1-1.2*i*(Nx+1-i)/Nx^2;  % bottom
 v(i) = 1.0;                    % top
end

for j=1:Ny+1
 w(j) = 1.0;   % left
 q(j) = 1.0;   % right
end

%---
% specify the source (typical)
%---

for j=2:Ny
 for i=2:Nx
  source(i,j) = 200.0*x(i)*y(j);
 end
end

%---
% tridiagonal in x
%---

U(1,1) =-2; U(1,2) = 1;
U(Nx-1,Nx-1) =-2; U(Nx-1,Nx-2) = 1;

for i=2:Nx-2
 U(i,i) = -2;
 U(i,i-1) = 1;
 U(i,i+1) = 1;
end

%---
% tridiagonal in y
%---

V(1,1) =-2; V(1,2) = 1;
V(Ny-1,Ny-1) =-2; V(Ny-1,Ny-2) = 1;
```

```
for i=2:Ny-2
 V(i,i) = -2;
 V(i,i-1) = 1;
 V(i,i+1) = 1;
end

%---
% coefficient matrix
%---

A = kron(eye(Ny-1),U) + beta*kron(V,eye(Nx-1));

%---
% right-hand side
% involving the source and the Dirichlet boundary conditions
%---

Ic = 0;

for j=2:Ny
 for i=2:Nx

  Ic = Ic +1;

  rhs(Ic) = -source(i,j)*Dx^2;

  if(j==2)
    rhs(Ic) = rhs(Ic) - beta*z(i);
  end

  if(j==Ny)
    rhs(Ic) = rhs(Ic) - beta*v(i);
  end

  if(i==2)
    rhs(Ic) = rhs(Ic) - w(j);
  end
```

```
  if(i==Nx)
    rhs(Ic) = rhs(Ic) - q(j);
  end

 end
end

%---
% solve
%---

SOL = rhs/A';

%---
% distribute
%---

Ic = 0;

for j=2:Ny
 for i=2:Nx
  Ic = Ic+1;
  f(i,j) = SOL(Ic);
 end
end

%---
% fill in the boundary conditions
%---

for i=1:Nx+1
 f(i,1) = z(i);    % bottom
 f(i,Ny+1) = v(i); % top
end

for j=1:Ny+1
 f(1,j) = w(j);     %  left
 f(Nx+1,j) = q(j);  % right
end
```

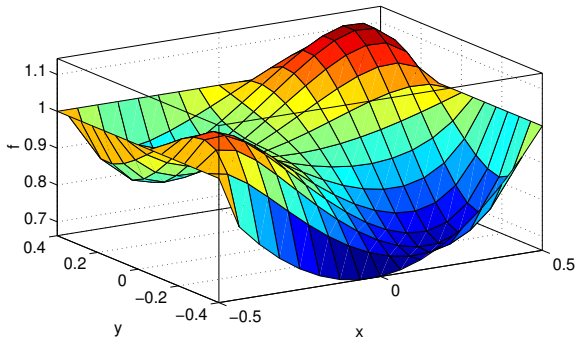


FIGURE 3.4.1 Solution of the Poisson equation with uniform forcing subject to the Dirichlet condition along the four edges of a rectangular solution domain. The coefficient matrix is constructed as the tensor product of two matrices.

```
%---
% plot
%---

mesh(x,y,f')
```

The graphics generated by the code is shown in Figure 3.4.1. The Dirichlet boundary conditions implemented in the code are satisfied.

3.4.4 NNXX

When the Neumann boundary condition is specified at the left and right edges of the solution domain along the x axis, we use the following $(N_x + 1) \times (N_x + 1)$ tridiagonal constituent matrix:

$$\mathbf{U}^{\mathrm{NN}} = \begin{bmatrix} -2 & 2 & \cdots & \cdots & \cdots & \cdots \\ 1 & -2 & 1 & 0 & \cdots & \cdots \\ 0 & 1 & -2 & 1 & 0 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & 0 & 1 & -2 & 1 & 0 \\ \cdots & \cdots & 0 & 1 & -2 & 1 \\ \cdots & \cdots & \cdots & 0 & 2 & -2 \end{bmatrix}. \tag{3.4.11}$$

Note the two *two* entries in the first and last rows implementing the Neumann boundary condition at the corresponding nodes.

Exercises

3.4.1 Write and run a code that solves the Poisson equation with $NNDD$ boundary conditions.

3.4.2 Write and run a code that solves the Poisson equation with $NNND$ boundary conditions.

3.5 The Moore–Penrose inverse

When the Neumann boundary condition is applied around the four edges of the solution domain, (NNNN), the solution vector is compiled as a sequence of horizontal layers from the first level, $j = 1$, to the top level, $j = N_y + 1$, into a partitioned solution vector involving $(N_x + 1) \times (N_y + 1)$ unknowns,

$$\mathbf{f} = \begin{bmatrix} \mathbf{f}^{(1)} \\ \mathbf{f}^{(2)} \\ \vdots \\ \mathbf{f}^{(N_y)} \\ \mathbf{f}^{(N_y+1)} \end{bmatrix}. \tag{3.5.1}$$

The sub-arrays,

$$\mathbf{f}^{(1)} \equiv \begin{bmatrix} f_{1,1} \\ f_{2,1} \\ \vdots \\ f_{N_x,1} \\ f_{N_x+1,1} \end{bmatrix}, \quad \ldots, \quad \mathbf{f}^{(N_y+1)} \equiv \begin{bmatrix} f_{1,N_y+1} \\ f_{2,N_y+1} \\ \vdots \\ f_{N_x,N_y+1} \\ f_{N_x+1,N_y+2} \end{bmatrix}, \tag{3.5.2}$$

encapsulate horizontal profiles.

The Neumann boundary condition specifies the outward derivatives of the function $f(x, y)$,

$$-\frac{\partial f}{\partial x} = w(y) \quad \text{at} \quad x = a_x, \qquad \frac{\partial f}{\partial x} = q(y) \quad \text{at} \quad x = b_x \tag{3.5.3}$$

and

$$-\frac{\partial f}{\partial y} = z(x) \quad \text{at} \quad y = a_y, \qquad \frac{\partial f}{\partial y} = v(x) \quad \text{at} \quad y = b_y, \tag{3.5.4}$$

where $w(y)$, $q(y)$, $z(x)$, and $v(x)$ are given functions along the left, right, bottom, and top edges of the rectangular solution domain.

3.5.1 Finite-difference equations

Collecting the finite-difference equations at all interior and boundary nodes, we derive a system of linear equations for the nodal values of the solution,

$$\mathbf{A} \cdot \mathbf{f} = \mathbf{b}, \tag{3.5.5}$$

where the coefficient matrix is given by a Kronecker sum,

$$\mathbf{A} = \mathbf{U}^{\mathrm{NN}} \oplus \mathbf{V}^{\mathrm{NN}} = \mathbf{I}_{N_y+1} \otimes \mathbf{U}^{\mathrm{NN}} + \beta\, \mathbf{V}^{\mathrm{NN}} \otimes \mathbf{I}_{N_x+1}, \tag{3.5.6}$$

and the constituent matrix $\mathbf{U}^{\mathrm{NN}}$ is given in (3.4.11). The $(N_y + 1) \times (N_y + 1)$ constituent matrix $\mathbf{V}^{\mathrm{NN}}$ is given by a similar expression.

The right-hand side of the linear system is the partitioned array

$$\mathbf{b} = \begin{bmatrix} \mathbf{b}^{(1)} \\ \mathbf{b}^{(2)} \\ \vdots \\ \mathbf{b}^{(N_y)} \\ \mathbf{b}^{(N_y+1)} \end{bmatrix}, \tag{3.5.7}$$

where

$$\mathbf{b}^{(1)} = - \begin{bmatrix} \beta\, 2\Delta y\, z(x_1) + \Delta x^2\, s_{1,1} + 2\Delta x\, w(y_1) \\ \beta\, 2\Delta y\, z(x_2) + \Delta x^2\, s_{2,1} \\ \vdots \\ \beta\, 2\Delta y\, z(x_{N_x}) + \Delta x^2\, s_{N_x,1} \\ \beta\, 2\Delta y\, z(x_{N_x+1}) + \Delta x^2\, s_{N_x+1,1} + 2\Delta x\, q(y_2) \end{bmatrix} \tag{3.5.8}$$

for the first block corresponding to $j = 1$,

$$\mathbf{b}^{(j)} = - \begin{bmatrix} \Delta x^2\, s_{1,j} + 2\Delta x\, w(y_j) \\ \Delta x^2\, s_{2,j} \\ \vdots \\ \Delta x^2\, s_{N_x,j} \\ \Delta x^2\, s_{N_x+1,j} + 2\Delta x\, q(y_j) \end{bmatrix} \tag{3.5.9}$$

for the intermediate blocks corresponding to $j = 2, \ldots, N_y$, and

$$\mathbf{b}^{(N_y+1)} = - \begin{bmatrix} \beta\, 2\Delta y\, v(x_1) + \Delta x^2\, s_{1,N_y+1} + 2\Delta x\, w(y_{N_y+1}) \\ \beta\, 2\Delta y\, v(x_2) + \Delta x^2\, s_{2,N_y+1} \\ \vdots \\ \beta\, 2\Delta y\, v(x_{N_x}) + \Delta x^2\, s_{N_x-1,N_y} \\ \beta\, 2\Delta y\, v(x_{N_x+1}) + \Delta x^2\, s_{N_x,N_y} + 2\Delta x\, q(y_{N_y+1}) \end{bmatrix} \tag{3.5.10}$$

for the last block corresponding to $j = N_y + 1$. Note that $z(x)$ appears in the first block, $\mathbf{b}^{(1)}$, $v(x)$ appears in the last block, $\mathbf{b}^{(N_y+1)}$, $w(y)$ appears in the first entry of each block, and $q(y)$ appears in the last entry of each block.

3.5.2 *Singular coefficient matrix*

The coefficient matrix $\mathbf{A}$ generated by (3.5.6) is singular, having one zero eigenvalue with a uniform corresponding eigenvector, $\mathbf{u}$, comprised of the same arbitrary scalar element. The associated eigenvector of the transpose of $\mathbf{A}$, called the left eigenvector or the adjoint eigenvector, denoted by $\mathbf{v}$, satisfies

$$\mathbf{v} \cdot \mathbf{A} = \mathbf{A}^{\mathrm{T}} \cdot \mathbf{v} = \mathbf{0}, \tag{3.5.11}$$

where the superscript T denotes the matrix transpose. The components of the left eigenvector, $\mathbf{v}$, can be apportioned to grid nodes following the compilation of the solution vector, $\mathbf{f}$, thereby generating a corresponding areal nodal field denoted by $\mathbf{w}$, parametrized by two indices.

For example, when $N_x = 2$, $N_y = 2$, and $\beta = 1$, we obtain the following singular 9×9 matrix:

```
A =

    -4     2     0     2     0     0     0     0     0
     1    -4     1     0     2     0     0     0     0
     0     2    -4     0     0     2     0     0     0
     1     0     0    -4     2     0     1     0     0
     0     1     0     1    -4     1     0     1     0
     0     0     1     0     2    -4     0     0     1
     0     0     0     2     0     0    -4     2     0
     0     0     0     0     2     0     1    -4     1
     0     0     0     0     0     2     0     2    -4
```

The sum of the element in each row of this matrix is zero. The associated adjoint eigenvector corresponding to the null eigenvalue is

$$\mathbf{v} = \tfrac{1}{4}\begin{bmatrix} 1 \\ 2 \\ 1 \\ 2 \\ 4 \\ 2 \\ 1 \\ 2 \\ 1 \end{bmatrix} \tag{3.5.12}$$

The solution domain can be divided into $2 \times 2 = 4$ squares defined by the $3 \times 3 = 9$ nodes. The components of the vertical array on the right-hand side of (3.5.12) is the number of squares hosting a particular nodes, which is 4 for the central node, 1 for the four corner nodes, and 2 for all other nodes. In fact, the eigenvector shown in (3.5.12) applies for any value of $\beta \equiv (\Delta x/\Delta y)^2$.

3.5.3 Adjoint eigenvector as a tensor product

The adjoint eigenvector, $\mathbf{v}$, can be constructed as the Kronecker tensor product of an (N_x+1)-dimensional vertical array, $\mathbf{v}^x$, and an (N_y+1)-dimensional vertical array, $\mathbf{v}^y$, as

$$\mathbf{v} \equiv \mathbf{v}^y \otimes \mathbf{v}^x, \tag{3.5.13}$$

where $\otimes$ is the Kronecker matrix product, not to be confused with the vector tensor product,

$$\mathbf{v}^x = \frac{1}{2}\begin{bmatrix} 1 \\ 2 \\ \vdots \\ 2 \\ 1 \end{bmatrix}, \qquad \mathbf{v}^y = \frac{1}{2}\begin{bmatrix} 1 \\ 2 \\ \vdots \\ 2 \\ 1 \end{bmatrix}. \tag{3.5.14}$$

The tensor product of these arrays, is a slender $(N_x+1)(N_y+1)\times 1$ array, given by

$$\mathbf{v} = \begin{bmatrix} v_1^y \mathbf{v}^x \\ v_2^y \mathbf{v}^x \\ \vdots \\ v_{N_y}^y \mathbf{v}^x \\ v_{N_y+1}^y \mathbf{v}^x \end{bmatrix}. \tag{3.5.15}$$

The associated nodal field is represented by the $(N_x+1)\times(N_y+1)$ matrix

$$\mathbf{w} = \begin{bmatrix} 1/4 & 1/2 & 1/2 & \cdots & 1/2 & 1/2 & 1/4 \\ 1/2 & 1 & 1 & \cdots & 1 & 1 & 1/2 \\ 1/2 & 1 & 1 & \cdots & 1 & 1 & 1/2 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 1/2 & 1 & 1 & \cdots & 1 & 1 & 1/2 \\ 1/2 & 1 & 1 & \cdots & 1 & 1 & 1/2 \\ 1/4 & 1/2 & 1/2 & \cdots & 1/2 & 1/2 & 1/4 \end{bmatrix}. \tag{3.5.16}$$

The sum of all elements in this matrix is equal to $N_x N_y$.

In fact, the matrix $\mathbf{w}$ is the tensor product of the vertical array $\mathbf{v}^x$ and the horizontal array $\mathbf{v}^{y^{\mathrm{T}}}$,

$$\mathbf{w} = \mathbf{v}^x \otimes \mathbf{v}^{y^{\mathrm{T}}}, \tag{3.5.17}$$

where the superscript T is denotes the transpose.

3.5.4 Trapezoidal rule integration weights

The nodal field encapsulated in the array $\mathbf{w}$ shown in (3.5.16), generated by $\mathbf{v}^x$ and $\mathbf{v}^y$ according to (3.5.13), contains integration weights of the double trapezoidal rule over the rectangular solution domain,

$$\iint \phi(x,y)\,\mathrm{d}x\,\mathrm{d}y \simeq \Delta x\,\Delta y \sum_{i=1}^{N_x+1}\sum_{j=1}^{N_y+1} w_{i,j}\,\phi(x_i,y_j), \tag{3.5.18}$$

where $\phi(x,y)$ is an arbitrary function.

3.5.5 Compatibility condition

For a linear system, $\mathbf{A}\cdot\mathbf{f}=\mathbf{b}$, to have a solution, the right-hand side must satisfy the compatibility condition

$$\mathbf{v}\cdot\mathbf{b}=0, \tag{3.5.19}$$

which arises by multiplying the vector equation $\mathbf{A}\cdot\mathbf{x}=\mathbf{b}$ by $\mathbf{v}$. and recalling that $\mathbf{v}\cdot\mathbf{A}=\mathbf{0}$. This discrete compatibility condition is the counterpart of the continuous compatibility condition

$$\oint \mathbf{n}\cdot\boldsymbol{\nabla} f\,\mathrm{d}\ell + \iint s(x,y)\,\mathrm{d}x\,\mathrm{d}y = 0, \tag{3.5.20}$$

where $\mathbf{n}$ is the outward unit vector normal to the boundary of the solution domain, and ℓ is the associated arc length. The continuous compatibility condition arises by integrating the Poisson equation, $\nabla^2 f + s = 0$ over the solution domain, and using the divergence theorem to convert the areal integral of the Laplacian into a boundary integral.

3.5.6 Extracting a solution

To satisfy the discrete compatibility condition, we introduce the normalized adjoint eigenvector,

$$\widehat{\mathbf{v}} = \frac{1}{|\mathbf{v}|}\,\mathbf{v}, \tag{3.5.21}$$

where

$$|\mathbf{v}| = \sqrt{v_1^2+\cdots+v_N^2}, \tag{3.5.22}$$

and replace the right-hand side of the linear system, $\mathbf{b}$, with an adjusted right-hand side,

$$\mathbf{b}^{\text{adjusted}} = \mathbf{b} - (\mathbf{b}\cdot\widehat{\mathbf{v}})\,\widehat{\mathbf{v}} = (\mathbf{I} - \widehat{\mathbf{v}}\otimes\widehat{\mathbf{v}})\cdot\mathbf{b}, \tag{3.5.23}$$

which lacks a component in the direction of $\widehat{\mathbf{v}}$,

$$\mathbf{b}^{\text{adjusted}}\cdot\widehat{\mathbf{v}} = 0. \tag{3.5.24}$$

We then solve the altered system

$$\mathbf{A}\cdot\mathbf{f} = (\mathbf{I} - \widehat{\mathbf{v}}\otimes\widehat{\mathbf{v}})\cdot\mathbf{b}, \tag{3.5.25}$$

which admits an infinite number of solutions differing by a constant. Note that the matrix $\mathbf{I} - \widehat{\mathbf{v}}\otimes\widehat{\mathbf{v}}$ on the right-hand side is singular, as is the coefficient matrix, $\mathbf{A}$.

With the discrete compatibility condition met, any solution can be offset by an arbitrary value. In practice to solve (3.5.25), we may overlook one equation, typically the last equation, and set the value of the solution at the last grid node involved in all other equations to zero. The solution is guaranteed to satisfy the overlooked equation.

3.5.7 NNNN code

The following Matlab code *poisson_NNNN*, located in directory MATENS of SEMLIB, generates the coefficient matrix $\mathbf{A}$ and solves the Poisson equation for the homogeneous (zero value) Neumann boundary condition around the four edges using this method:

```
%================================
% Solution of the Poisson equation
% in a rectangle confined between
% -a < x < a and -b < y < b
% with the Neumann
% boundary condition all around
%
% -df/dx = w at the left
%  df/dx = q at the right
% -df/dy = z at the bottom
```

```
%  df/dy = v at the top
%================================

Nx = 24;
Ny = 16;

a = 1.0;
b = 0.5;

%---
% prepare
%---

Dx = 2.0*a/Nx;
Dy = 2.0*b/Ny;

beta = Dx^2/Dy^2;

Dx2 = 2.0*Dx;
Dy2 = 2.0*Dy;

%---
% grid
%---

for i=1:Nx+1
 x(i) = -a + (i-1)*Dx;
end

for i=1:Ny+1
 y(i) = -b + (i-1)*Dy;
end

%---
% Neumann boundary conditions
%---

for i=1:Nx+1
 z(i) =  0.1; % bottom
```

```
 v(i) = -0.1; % top
end

for j=1:Ny+1
 w(j) =  0.2; % left
 q(j) = -0.2; % right
end

%---
% specify the source (typical)
%---

for j=1:Ny+1
 for i=1:Nx+1
  source(i,j) = 20.0*x(i)*y(j);
 end
end

%---
% tridiagonal in x
%---

U(1,1) =-2; U(1,2) = 2;
U(Nx+1,Nx+1) =-2; U(Nx+1,Nx) = 2;

for i=2:Nx
 U(i,i)   =-2;
 U(i,i-1) = 1;
 U(i,i+1) = 1;
end

%---
% tridiagonal in y
%---

V(1,1) =-2; V(1,2)  = 2;
V(Ny+1,Ny+1) =-2; V(Ny+1,Ny) = 2;

for i=2:Ny
```

```
 V(i,i)  = -2;
 V(i,i-1) = 1;
 V(i,i+1) = 1;
end

%---
% coefficient matrix
%---

A = kron(eye(Ny+1),U) + beta*kron(V,eye(Nx+1));

%---
% system size
%---

Nsys = (Nx+1)*(Ny+1);

%---
% compose the adjoint eigenvector
% of the null eigenvalue
% as a Kronecker tensor product
%---

eigvx(1) = 0.5;

for i=2:Nx
 eigvx(i) = 1.0;
end

eigvx(Nx+1) = 0.5;

eigvy(1) = 0.5;

for j=2:Ny
 eigvy(j) = 1.0;
end

eigvy(Ny+1) = 0.5;
```

```
eigv = kron(eigvy',eigvx');

%---
% normalize eigv
%---

vnorm = 0;

for i=1:Nsys
 vnorm = vnorm + eigv(i)^2;
end

vnorm = sqrt(vnorm);

for i=1:Nsys
 eigvn(i) = eigv(i)/vnorm;
end

%---
% define the normalized eigenvector
% of the null eigenvalue
%---

for i=1:Nsys
 eigun(i) = 1.0/sqrt(Nsys);
end

%---
% right-hand side
%---

Ic = 0; % counter

for j=1:Ny+1
 for i=1:Nx+1

  Ic = Ic +1;

  rhs(Ic) = -source*Dx^2;
```

```
  if(j==1)
    rhs(Ic) = rhs(Ic) - beta*Dy2*z(i);
  end

  if(j==Ny+1)
    rhs(Ic) = rhs(Ic) - beta*Dy2*v(i);
  end

  if(i==1)
    rhs(Ic) = rhs(Ic) - Dx2*w(j);
  end

  if(i==Nx+1)
    rhs(Ic) = rhs(Ic) - Dx2*q(j);
  end

 end
end

%---
% regularize the right-hand side
%---

rhsdotvn = 0.0;

for i=1:Nsys
 rhsdotvn = rhsdotvn + rhs(i)*eigvn(i);
end

for i=1:Nsys
 rhsn(i) = rhs(i) - rhsdotvn*eigvn(i);
end

%---
% solve
%---

Ared   = A(1:Nsys-1,1:Nsys-1);
```

```
rhsred = rhsn(1:Nsys-1);
sol    = rhsred/Ared';
sol(Nsys) = 0.0;

%---
% solution by the Moore--Penrose inverse
% (optional)
%---

MPinv = pinv(A);
MPsol = MPinv*rhs';
shift = MPsol(Nsys);

for i=1:Nsys
 MPsol(i) = MPsol(i)-shift;
end

%---
% distribute the solution
%---

Ic = 0;

for j=1:Ny+1
 for i=1:Nx+1
  Ic = Ic+1;
  f(i,j) = sol(Ic);
 end
end

%---
% plot
%---

figure(1)
mesh(x,y,f')
```

The graphics generated by the code for the parameters and conditions implemented in the code is shown in Figure 3.5.1.

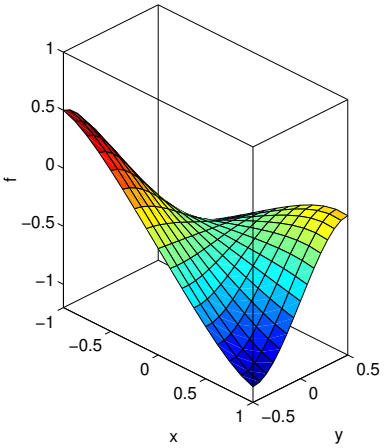


FIGURE 3.5.1 Solution of the Poisson equation by a finite-difference method with the Neumann boundary condition around the four edges of a rectangular solution domain and a random source term.

3.5.8 *Solution by the Moore–Penrose inverse*

Precisely the same solution shifted by an arbitrary constant is obtained by multiplying the uncorrected right-hand side, $\mathbf{b}$, by the Moore–Penrose inverse of the matrix $\mathbf{A}$, denoted by $\mathbf{A}^+$,

$$\mathbf{f} = \mathbf{A}^+ \cdot \mathbf{b} = \mathbf{A}^+ \cdot \mathbf{b}^{\text{adjusted}}, \tag{3.5.26}$$

and then shifting the solution to zero the last component of $\mathbf{f}$ zero. In the code, $\mathbf{A}^+$ is generated using the internal Matlab function `pinv`. As an alternative, now we derive a simple formula for $\mathbf{A}^+$ in terms of the eigenvector and adjoint eigenvector of $\mathbf{A}$ corresponding to the zero eigenvalue discussed earlier in this section.

3.5.9 *Properties of the Moore–Penrose inverse*

Referring to the definition of $\mathbf{b}^{\text{adjusted}} = (\mathbf{I} - \widehat{\mathbf{v}} \otimes \widehat{\mathbf{v}}) \cdot \mathbf{b}$, we find that

$$\mathbf{A}^+ \cdot \widehat{\mathbf{v}} = \mathbf{0}, \tag{3.5.27}$$

which shows that the adjoint eigenvector, $\widehat{\mathbf{v}}$, is an eigenvector of $\mathbf{A}^+$

with zero corresponding eigenvalue, thereby indicating that $\mathbf{A}^+$ is singular.

The $N \times N$ matrices $\mathbf{A}$ and $\mathbf{A}^+$ satisfy

$$\mathbf{A} \cdot \mathbf{A}^+ = \mathbf{I} - \widehat{\mathbf{v}} \otimes \widehat{\mathbf{v}}, \qquad \mathbf{A}^+ \cdot \mathbf{A} = \mathbf{I} - \widehat{\mathbf{u}} \otimes \widehat{\mathbf{u}}, \tag{3.5.28}$$

where $\widehat{\mathbf{u}}$ is the normalized eigenvector of $\mathbf{A}$ corresponding to the null eigenvalue with elements $\widehat{u}_i = 1/\sqrt{N}$ for $i = 1, \ldots, N$. Based on these relations, we find that

$$\mathbf{A}^+ \cdot \mathbf{A} \cdot \mathbf{A}^+ = \mathbf{I}, \qquad \mathbf{A} \cdot \mathbf{A}^+ \cdot \mathbf{A} = \mathbf{I}. \tag{3.5.29}$$

To confirm that (3.5.26) provides us with the regularized solution, we substitute $\mathbf{f} = \mathbf{A}^+ \cdot \mathbf{b}$ into (3.5.25) and obtain

$$\mathbf{A} \cdot \mathbf{A}^+ \cdot \mathbf{b} = (\mathbf{I} - \widehat{\mathbf{v}} \otimes \widehat{\mathbf{v}}) \cdot \mathbf{b}, \tag{3.5.30}$$

which is true in light of the first relation in (3.5.28).

3.5.10 *Computation of the Moore–Penrose inverse*

To compute $\mathbf{A}^+$, we stipulate that

$$\widehat{\mathbf{u}} \cdot \mathbf{A}^+ = 0, \tag{3.5.31}$$

note that

$$(\mathbf{A} + \widehat{\mathbf{v}} \otimes \widehat{\mathbf{u}}) \cdot (\mathbf{A}^+ + \widehat{\mathbf{u}} \otimes \widehat{\mathbf{v}}) = \mathbf{I}, \tag{3.5.32}$$

and set

$$\mathbf{A}^+ = (\mathbf{A} + \widehat{\mathbf{v}} \otimes \widehat{\mathbf{u}})^{-1} - \widehat{\mathbf{u}} \otimes \widehat{\mathbf{v}}. \tag{3.5.33}$$

The inverse of the deflated matrix on the right-hand side, $(\mathbf{A} + \widehat{\mathbf{v}} \otimes \widehat{\mathbf{u}})^{-1}$, is well defined. Other advanced methods that circumvent the computation of the matrix inverse are available.

Exercise

3.5.1 Modify the code discussed in the text to implement an arbitrary Neumann boundary condition around the four edges of the solution domain.

3.6 Periodic boundary conditions

Next, we consider the periodic condition along the x axis and introduce the $N_x \times N_x$ constituent matrix

$$\mathbf{U}^{\mathrm{PP}} = \begin{bmatrix} -2 & 1 & \cdots & \cdots & \cdots & 1 \\ 1 & -2 & 1 & 0 & \cdots & \cdots \\ 0 & 1 & -2 & 1 & 0 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & 0 & 1 & -2 & 1 & 0 \\ \cdots & \cdots & 0 & 1 & -2 & 1 \\ 1 & \cdots & \cdots & 0 & 2 & -2 \end{bmatrix}. \tag{3.6.1}$$

Note the two *ones* at the corner elements along the back-diagonal implementing the periodicity condition.

3.6.1 PPPP

When the periodicity condition is applied in both the x and y directions, (PPPP), the solution vector is compiled as a sequence of horizontal layers from the first level, $j = 1$, to the second-to-top level, $j = N_y$, into a partitioned solution vector

$$\mathbf{f} = \begin{bmatrix} \mathbf{f}^{(1)} \\ \mathbf{f}^{(2)} \\ \vdots \\ \mathbf{f}^{(N_y-1)} \\ \mathbf{f}^{(N_y)} \end{bmatrix}, \tag{3.6.2}$$

involving $N_x \times N_y$ scalar unknowns. The sub-arrays

$$\mathbf{f}^{(1)} \equiv \begin{bmatrix} f_{1,1} \\ f_{2,1} \\ \vdots \\ f_{N_x-1,1} \\ f_{N_x,1} \end{bmatrix}, \quad \ldots, \quad \mathbf{f}^{(N_y)} \equiv \begin{bmatrix} f_{1,N_y} \\ f_{2,N_y} \\ \vdots \\ f_{N_x-1,N_y} \\ f_{N_x,N_y} \end{bmatrix} \tag{3.6.3}$$

encapsulate horizontal profiles

3.6.2 Finite-difference equations

Collecting the finite-difference equations at nodes with indices $i = 1, \cdots, N_x$ and $j = 1, \cdots, N_y$, we derive a system of linear equations for the nodal values of the solution,

$$\mathbf{A} \cdot \mathbf{f} = \mathbf{b}, \tag{3.6.4}$$

where the coefficient matrix is given by a Kronecker sum,

$$\mathbf{A} = \mathbf{U}^{\rm PP} \oplus \mathbf{V}^{\rm PP} = \mathbf{I}_{N_y} \otimes \mathbf{U}^{\rm PP} + \beta\, \mathbf{V}^{\rm PP} \otimes \mathbf{I}_{N_x}, \tag{3.6.5}$$

and the constituent matrix $\mathbf{U}^{\rm PP}$ is given in (3.6.1). A similar $N_y \times N_y$ matrix is used for the constituent matrix $\mathbf{V}^{\rm PP}$.

The right-hand side of the linear system is a partitioned array,

$$\mathbf{b} = \begin{bmatrix} \mathbf{b}^{(1)} \\ \mathbf{b}^{(2)} \\ \vdots \\ \mathbf{b}^{(N_y-1)} \\ \mathbf{b}^{(N_y)} \end{bmatrix}, \qquad \mathbf{b}^{(j)} = -\Delta x^2 \begin{bmatrix} s_{1,j} \\ s_{2,j} \\ \vdots \\ s_{N_x-1,j} \\ s_{N_x,j} \end{bmatrix} \tag{3.6.6}$$

for $j = 1, \ldots, N_y$.

3.6.3 Singular coefficient matrix

The coefficient matrix $\mathbf{A}$ is singular, having one zero eigenvalue with a uniform corresponding eigenvector, $\mathbf{u}$, comprised of the same arbitrary

scalar element. Because the coefficient matrix $\mathbf{A}$ is symmetric, the associated adjoint eigenvector, $\mathbf{v}$, is also $\mathbf{u}$.

For example, when $N_x = 3$, $N_y = 3$, and $\beta = 4$, we obtain the following singular 9×9 matrix:

```
A =
   -10     1     1     4     0     0     4     0     0
     1   -10     1     0     4     0     0     4     0
     1     1   -10     0     0     4     0     0     4
     4     0     0   -10     1     1     4     0     0
     0     4     0     1   -10     1     0     4     0
     0     0     4     1     1   -10     0     0     4
     4     0     0     4     0     0   -10     1     1
     0     4     0     0     4     0     1   -10     1
     0     0     4     0     0     4     1     1   -10
```

The sum of the element in each row or column is zero.

Because the coefficient matrix $\mathbf{A}$ is singular, a solution can be found only if the sum of all elements of the right-hand side of the linear system, $\mathbf{b}$, is zero. To satisfy this condition, we work as in Section 3.5 and find that, to satisfy the compatibility condition, we can merely shift all nodal values of the source in the Poisson equation by an appropriate amount.

3.6.4 PPPP code

The following Matlab code named *poisson_PPPP*, located in directory MATENS of SEMLIB, generates the coefficient matrix $\mathbf{A}$ and solves the Poisson equation subject to the doubly periodic condition:

```
%================================
% Solution of the Poisson equation
% in a rectangle confined between
% -a < x < a and -b < y < b
%
% with the doubly periodic condition
%================================

Nx = 16;
Ny = 14;
```

```
a = 1.0;
b = 0.5;

%---
% prepare
%---

Dx = 2.0*a/Nx;
Dy = 2.0*b/Ny;

beta = Dx^2/Dy^2;

%---
% grid
%---

for i=1:Nx+1
 x(i) = -a + (i-1)*Dx;
end

for i=1:Ny+1
 y(i) = -b + (i-1)*Dy;
end

%---
% specify the source (typical)
%---

for j=1:Ny
 for i=1:Nx
  source(i,j) = 50.0*x(i)*y(j);
 end
end

%---
% circulant tridiagonal in x
%---
```

```
U(1,1) =-2; U(1,2) = 1; U(1,Nx) = 1;
U(Nx,1) = 1; U(Nx,Nx) =-2; U(Nx,Nx-1) = 1;

for i=2:Nx-1
 U(i,i)   =-2;
 U(i,i-1) = 1;
 U(i,i+1) = 1;
end

%---
% circulant tridiagonal in y
%---

V(1,1) =-2; V(1,2) = 1; V(1,Ny) = 1;
V(Ny,1) = 1; V(Ny,Ny) =-2; V(Ny,Ny-1) = 1;

for i=2:Ny-1
 V(i,i)  = -2;
 V(i,i-1) = 1;
 V(i,i+1) = 1;
end

A = kron(eye(Ny),U) + beta*kron(V,eye(Nx));

%---
% system size
%---

Nsys = Nx*Ny;

%---
% right-hand side
%---

Ic = 0;
sumrhs = 0;

for j=1:Ny
 for i=1:Nx
```

```
  Ic = Ic+1;
  rhs(Ic) = -source(i,j)*Dx^2;
  sumrhs = sumrhs + rhs(Ic);
 end
end

shift = sumrhs/Nsys;

for i=1:Nsys
 rhsn(i) = rhs(i)-shift;
end

%---
% solve
%---

Ared   = A(1:Nsys-1,1:Nsys-1);
rhsred = rhsn(1:Nsys-1);
sol    = rhsred/Ared';
sol(Nsys) = 0.0;
residual = A*sol'-rhsn';

%---
% distribute the solution
%---

Ic = 0;

for j=1:Ny
 for i=1:Nx
  Ic = Ic+1;
  f(i,j) = sol(Ic);
 end
end

for j=1:Ny
 f(Nx+1,j) = f(1,j);
end
```

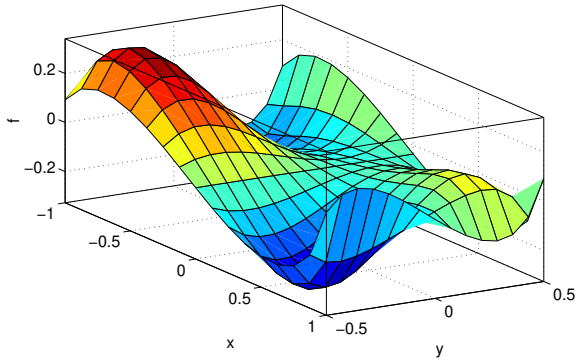


FIGURE 3.6.1 Solution of the Poisson equation by a finite-difference method with the doubly periodic condition and a random distributed source term.

```
for i=1:Nx
 f(i,Ny+1) = f(i,1);
end

f(Nx+1,Ny+1) = f(1);

%---
% plot
%---

figure(1)
mesh(x,y,f')
```

The graphics generated by the code is shown in Figure 3.6.1. The solution displayed can be shifted upward or downward by an arbitrary constant.

Exercise

3.6.1 Discuss the eigenvalues of the coefficient matrix with the doubly periodic condition.

3.7 Flow through a rectangular channel

Continuing the discussion of the Poisson equation in a rectangular solution domain, now we consider a physical application concerning unidirectional fluid flow along the x axis through a tube with a rectangular cross-section in the yz plane is confined inside $-a \leq y \leq a$ and $-b \leq z \leq b$.

The flow is governed by a Poisson equation for the axial velocity component, u_x, with a constant right-hand side,

$$\frac{\partial^2 u_x}{\partial y^2} + \frac{\partial^2 u_x}{\partial z^2} = \frac{\chi + \rho g_x}{\mu}, \tag{3.7.1}$$

where ρ is the fluid density, μ is the fluid viscosity, χ is a constant expressing the negative of the axial pressure gradient, g_x is the x component of the gravitational acceleration.

3.7.1 Slip boundary condition

The slip boundary condition expressed by (1.2.1) and (1.2.2),

$$u_x(y = \pm a, z) = \overline{+}\ell \left(\frac{\partial u_x}{\partial y} \right)_{y=\pm a} \tag{3.7.2}$$

and

$$u_x(y, z = \pm b) = \overline{+}\ell \left(\frac{\partial u_x}{\partial z} \right)_{z=\pm b} \tag{3.7.3}$$

are imposed around the edges. In the present context, the slip boundary condition is classified as a Robin boundary condition. A solution by separation of variables was developed in Section 7.2.

3.7.2 Code

The following Matlab code, located in directory TUBE_REC_SLIP of SEMLIB, solves the Poisson equation in the yz plane to generate the velocity distribution over the tube cross-section. The finite-difference coefficient matrix is constructed as a tensor sum of two tridiagonal matrices involving the slip coefficients:

```
%============================
% Flow in a rectangular channel
% inside -a<y<a and -b<z<b
% with the slip boundary condition
%============================

a = 1.0;
b = 0.5;

Ny = 2*16;
Nz = 2*16;


slipl = 0.1;

pg = 1.0; % effective pressure gradient

%---
% prepare
%---

Dy = 2.0*a/Ny;
Dz = 2.0*b/Nz;

Dys = Dy*Dy;

beta = Dy^2/Dz^2;

%---
% grid
%---

for i=1:Ny+1
 x(i) = -a + (i-1)*Dy;
end

for i=1:Nz+1
 y(i) = -b + (i-1)*Dz;
end
```

```
%---
% tridiagonal in y involving slip
%---

slp = 2.0*Dy/slipl;

U(1,1) = -2-slp;
U(1,2) = 2;
U(Ny+1,Nz+1) =-2-slp;
U(Ny+1,Nz) = 2;

for i=2:Ny
 U(i,i)   =-2;
 U(i,i-1) = 1;
 U(i,i+1) = 1;
end

%---
% tridiagonal in z involving slip
%---

slp = 2.0*Dz/slipl;

V(1,1) = -2-slp;
V(1,2) =  2;
V(Nz+1,Nz+1) =-2-slp;
V(Nz+1,Nz) = 2;

for i=2:Nz
 V(i,i) = -2;
 V(i,i-1) = 1;
 V(i,i+1) = 1;
end

%---
% coefficient matrix
%---
```

```
A = kron(eye(Nz+1),U) + beta*kron(V,eye(Ny+1));

%---
% system size
%---

Nsys = (Ny+1)*(Nz+1);

%---
% right-hand side
%---

for i=1:Nsys
 rhs(i) = -pg*Dys;
end

%---
% solve
%---

SOL = rhs/A';

%---
% distribute
%---

Ic = 0;

for j=1:Nz+1
 for i=1:Ny+1
  Ic = Ic+1;
  f(i,j) = SOL(Ic);
 end
end

%---
% plot
%---
```

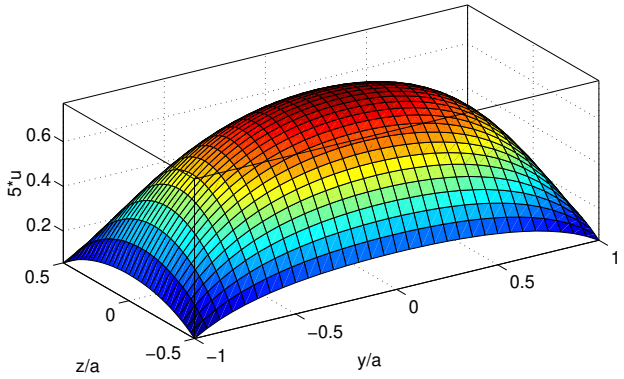


FIGURE 3.7.1 Velocity distribution over the cross-section of a rectangular tube with aspect ratio $a/b = 2$ for scaled slip coefficient $\lambda \equiv \ell/b = 0.2$.

```
figure(1)
surf(x,y,5.0*f')
```

The graphics generated by the code is shown in Figure 3.7.1. The velocity profile is in excellent agreement with that displayed in Figure 4.2.4 based on an analytical yet approximate solution derived by a Fourier series expansion.

3.7.3 Flow rate

The flow rate through the rectangular tube, Q, can be computed by integrating the velocity profile over the rectangular cross-section using the double trapezoidal rule according to the following computational module:

```
flow_rate = 0.0;

for j=1:Nz
 for i=1:Ny
  flow_rate = flow_rate + 0.25*(f(i,j)+f(i+1,j) ...
              +f(i+1,j+1)+f(i,j+1) );
```

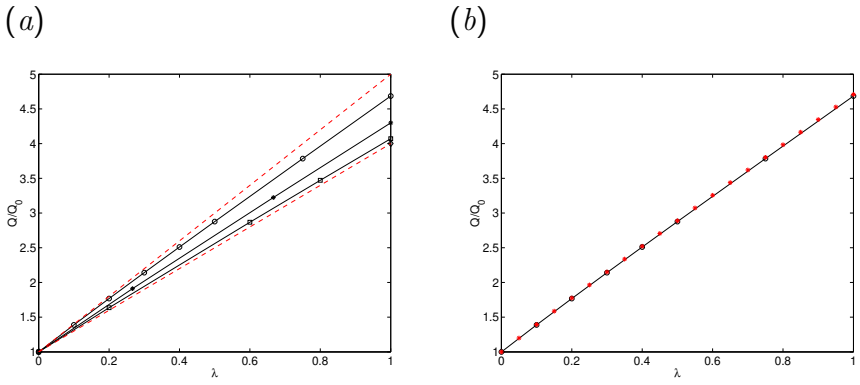


FIGURE 3.7.2 (a) Ratio of the flow rate, Q, to that corresponding to the no-slip condition, Q_0, for $b/a = 1$ (circular symbols), 0.75, 0.50, and 0.20, plotted against the scaled slip length. The blue dashed line corresponds to a circular tube of radius a and the red dashed line corresponds to a two-dimensional channel of width $2b$. (b) Ratio Q/Q_0 for a square tube, $b/a = 1$. The circular symbols correspond to the finite-difference solution and the asterisks correspond to the analytical one-Fourier-term solution.

```
 end
end

flow_rate = flow_rate*Dy*Dz;
```

A graph of the flow rate through the tube, Q, scaled by that corresponding to the no-slip boundary condition applies along all four walls, Q_0, is shown in Figure 3.7.2(a). The results are nearly identical to those shown in Figure 6.1.3, obtained by an analytical method.

The agreement is confirmed further by the graphs presented in Figure 3.7.2(b) for a tube with square cross-section, $a = b$. The asterisks correspond to the approximate analytical one-Fourier-term solution derived in Section 4.2.

3.7.4 Flow through a rectangular duct

A slight modification of the channel code allows us to compute unidirectional flow along the x axis through a rectangular duct. The flow is driven by the translation of the upper wall in the absence of gravity or a pressure gradient. As a generalization, we allow for two slip lengths, one for the moving top and the second for the three stationary side and bottom walls.

3.7.5 Duct code

The following Matlab code, located in directory DUCT_REC_SLIP of SEMLIB, solves the Laplace equation in the yz plane to generate the velocity distribution over the duct cross-section:

```
a = 1.0;
b = 0.5;

Ny = 2*16;
Nz = 2*16;

slipltop  = 0.1;
sliplwall = 0.001;

Vtop = 1.0;

%---
% prepare
%---

Dy = 2.0*a/Ny;
Dz = 2.0*b/Nz;

beta = Dy^2/Dz^2;

%---
% grid
%---

for i=1:Ny+1
```

```
 x(i) = -a + (i-1)*Dy;
end

for i=1:Nz+1
 y(i) = -b + (i-1)*Dz;
end

%---
% tridiagonal in y
%---

slpwall = 2.0*Dy/sliplwall;

U(1,1) = -2-slpwall;
U(1,2) = 2;
U(Ny+1,Ny+1) =-2-slpwall;
U(Ny+1,Ny)   = 2;

for i=2:Ny
 U(i,i)   =-2;
 U(i,i-1) = 1;
 U(i,i+1) = 1;
end

%---
% tridiagonal in z
%---

slptop = 2.0*Dz/slipltop;

V(1,1) = -2-slpwall;
V(1,2) =  2;
V(Nz+1,Nz+1) =-2-slptop;
V(Nz+1,Nz)   = 2;

for i=2:Nz
 V(i,i) = -2;
 V(i,i-1) = 1;
 V(i,i+1) = 1;
```

```
end

%---
% coefficient matrix
%---

A = kron(eye(Nz+1),U) + beta*kron(V,eye(Ny+1));

%---
% system size
%---

Nsys = (Ny+1)*(Nz+1);

%---
% right-hand side
%---

for i=1:Nsys
 rhs(i) = 0.0;
end

for i=1:Ny+1
 rhs(Nsys+1-i) = -2.0*beta*Dz*Vtop/slipltop;
end

%---
% solve
%---

SOL = rhs/A';

%---
% distribute
%---

Ic = 0;

for j=1:Nz+1
```

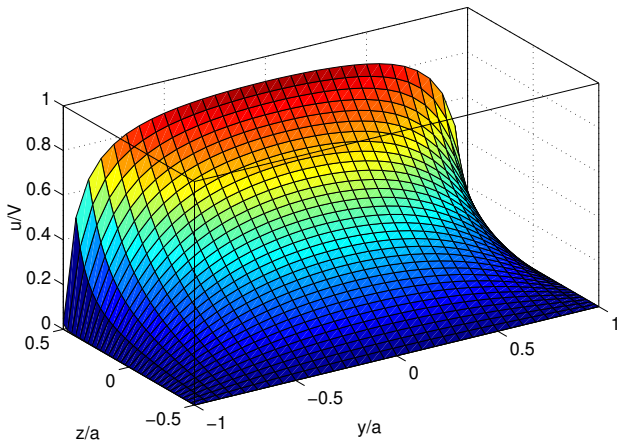


FIGURE 3.7.3 Velocity distribution over the cross-section of a rectangular duct with aspect ratio $a/b = 2$. The slip boundary condition is imposed along the three walls.

```
 for i=1:Ny+1
  Ic = Ic+1;
  f(i,j) = SOL(Ic);
 end
end

%---
% plot
%---

figure(1)
surf(x,y,f')
```

The graphics generated by the code is shown in Figure 3.7.3.

Exercises

3.7.1 Modify the code *tube_rec_slip* so that each wall has its own slip length.

3.7.2 Prepare a graph of the flow rate against the slip length of the moving top and no-slip side and bottom walls for a square duct.

3.8 Graph Laplacian

A network is represented by a mathematical graph consisting of a set of N nodes, also called vertices, connected by L links, also called edges. The degree of the ith node, denoted by d_i, is defined the number of links attached to the node, connecting the node to its nearest neighbors. A secluded node is devoid of nearest or any neighbors. In a complete network, each node is connected to every other node with a uniform node degree distribution, $d_i = N - 1$ for any node index, i.

3.8.1 Adjacency matrix

It is useful to introduce an $N \times N$ adjacency matrix, $\mathbf{A}$, defined such that $A_{ij} = 1$ if nodes i and j are connected by a link, and $A_{ij} = 0$ otherwise, where $i, j = 1, \ldots, N$. By convention, the diagonal elements of the adjacency matrix are zero. By construction, the adjacency matrix is symmetric. The degree of the ith node, d_i, is equal to the number of *ones* in the ith row or column of the adjacency matrix. The total number of links in a network, L, is equal to the number of *ones* in the upper or lower triangular part of the adjacency matrix,

3.8.2 Graph Laplacian

Let $\mathbf{D}$ be a diagonal matrix whose ith diagonal element is equal to the degree of the ith node, d_i. The $N \times N$ graph Laplacian matrix, $\mathbf{L}$, is defined in terms of $\mathbf{D}$, and the adjacency matrix, $\mathbf{A}$, as

$$\mathbf{L} = \mathbf{D} - \mathbf{A}. \tag{3.8.1}$$

For example, for a network of two connected nodes,

$$\mathbf{L} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}. \tag{3.8.2}$$

For a complete triangular network of three connected, where each node

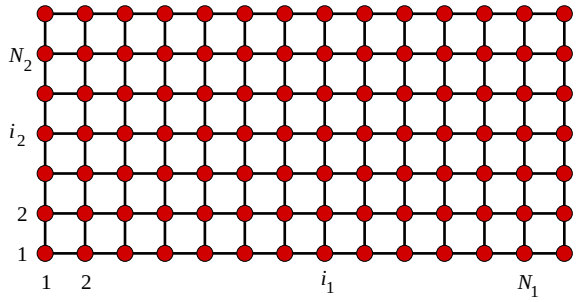


FIGURE 3.8.1 Illustration of a rectangular patch of a square lattice containing N_1 links in the first direction and N_2 links in the second direction.

is connected to each other node,

$$\mathbf{L} = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}. \tag{3.8.3}$$

Because the sum of elements in each row is zero, the Laplacian matrix of any network is singular.

3.8.3 Square lattice

A network consisting of a rectangular patch of a square lattice containing N_1 links in the first direction, parametrized by an index i_1, and N_2 links in the second direction, parametrized by another index i_2, is shown in Figure 3.8.1. Isolated, periodic, doubly periodic, and other configurations of a distributed nodal field can be envisioned.

A theorem due to Fielder states that the eigenvectors of the Laplacian matrix for certain types of boundary conditions are tensor products of those of the constituent one-dimensional graphs. The associated eigenvalues are the sums of the eigenvalues of the Laplacian of the constituent one-dimensional graphs. This property reflects the separability of the discrete Laplace operator in Cartesian coordinates.

3.8.4 Isolated square lattice

The total number of nodes in the isolated network shown in Figure 3.8.1 is

$$N = (N_1 + 1)(N_2 + 1) \tag{3.8.4}$$

and the total number of links is

$$L = N_1(N_2 + 1) + (N_1 + 1)N_2. \tag{3.8.5}$$

Note that the number of links is significantly higher than the number of nodes, which is typical of a well-connected network.

The nodal values of a nodal scalar field, f, can be compiled in a sequence of horizontal layers from the bottom where $i_2 = 1$ to the top where $i_2 = N_2 + 1$, into an N-dimensional vector

$$\mathbf{f} = \begin{bmatrix} \mathbf{f}^{(1)} \\ \mathbf{f}^{(2)} \\ \vdots \\ \mathbf{f}^{(N_2)} \\ \mathbf{f}^{(N_2+1)} \end{bmatrix}, \tag{3.8.6}$$

where the subvectors

$$\mathbf{f}^{(1)} \equiv \begin{bmatrix} f_{1,1} \\ f_{2,1} \\ \vdots \\ f_{N_1+1,1} \end{bmatrix}, \quad \ldots, \quad \mathbf{f}^{(N_2+1)} \equiv \begin{bmatrix} f_{1,N_2+1} \\ f_{2,N_2+1} \\ \vdots \\ f_{N_1+1,N_2+1} \end{bmatrix} \tag{3.8.7}$$

encapsulate horizontal profiles. Note that the vector $\mathbf{f}$ is identical to that employed in the solution of the Poisson equation using a finite-difference methods with $\Delta x = \Delta y$, as discussed in Section 3.9.

The Laplacian matrix consists of N_2+1 rows of $(N_1+1)\times(N_1+1)$ square tridiagonal blocks, $\mathbf{F}$, $\mathbf{E}$, and $\mathbf{I}$, in the following configuration:

$$\mathbf{L} = \begin{bmatrix} \mathbf{F} & -\mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ -\mathbf{I} & \mathbf{E} & -\mathbf{I} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\mathbf{I} & \mathbf{E} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{E} & -\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & -\mathbf{I} & \mathbf{E} & -\mathbf{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & -\mathbf{I} & \mathbf{F} \end{bmatrix}, \tag{3.8.8}$$

where $\mathbf{I}$ is the $(N_1+1)\times(N_1+1)$ identity matrix. When $N_1 = 3$,

$$\mathbf{F} = \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 3 & -1 & 0 \\ 0 & -1 & 3 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix}, \quad \mathbf{E} = \begin{bmatrix} 3 & -1 & 0 & 0 \\ -1 & 4 & -1 & 0 \\ 0 & -1 & 4 & -1 \\ 0 & 0 & -1 & 3 \end{bmatrix}. \tag{3.8.9}$$

The *two* entries of $\mathbf{F}$ correspond to corner nodes, the *three* entries of $\mathbf{F}$ and $\mathbf{E}$ correspond to boundary nodes, and the *four* entries of $\mathbf{E}$ correspond to interior nodes.

3.8.5 Laplacian as a tensor product

In fact, the graph Laplacian shown in (3.8.8) is given by a Kronecker sum,

$$\mathbf{A} = \mathbf{U} \oplus \mathbf{V} = \mathbf{I}_{N_2+1} \otimes \mathbf{U} + \mathbf{V} \otimes \mathbf{I}_{N_1+1}, \tag{3.8.10}$$

where $\mathbf{I}_{N_2+1}$ is the $(N_2+1)\times(N_2+1)$ identity matrix, $\mathbf{I}_{N_1+1}$ is the $(N_1+1)\times(N_1+1)$ identity matrix,

$$\mathbf{U} = \begin{bmatrix} 1 & -1 & \cdots & \cdots & \cdots & \cdots \\ -1 & 2 & -1 & 0 & \cdots & \cdots \\ 0 & -1 & 2 & -1 & 0 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & 0 & -1 & 2 & -1 & 0 \\ \cdots & \cdots & 0 & -1 & 2 & -1 \\ \cdots & \cdots & \cdots & 0 & -1 & 1 \end{bmatrix} \tag{3.8.11}$$

is an $(N_1+1)\times(N_1+1)$ tridiagonal matrix, and

$$\mathbf{V}=\begin{bmatrix} 1 & -1 & \cdots & \cdots & \cdots & \cdots \\ -1 & 2 & -1 & 0 & \cdots & \cdots \\ 0 & -1 & 2 & -1 & 0 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & 0 & -1 & 2 & -1 & 0 \\ \cdots & \cdots & 0 & -1 & 2 & -1 \\ \cdots & \cdots & \cdots & 0 & -1 & 1 \end{bmatrix} \tag{3.8.12}$$

is a $(N_2+1)\times(N_2+1)$ tridiagonal matrix. It is not surprising that the matrices $\mathbf{U}$ and $\mathbf{V}$ may differ only in size and have identical forms.

The eigenvalues of the singular matrices $\mathbf{U}$ and $\mathbf{V}$ are given by

$$\lambda_{n_1}=4\,\sin^2(\tfrac{1}{2}\,\alpha_{n_1}),\qquad \lambda_{n_2}=4\,\sin^2(\tfrac{1}{2}\,\beta_{n_2}), \tag{3.8.13}$$

where

$$\alpha_{n_1}=\frac{n_1-1}{N_1+1}\,\pi,\qquad \beta_{n_2}=\frac{n_2-1}{N_2+1}\,\pi \tag{3.8.14}$$

for $n_1=1,\ldots,N_1+1$ and $n_2=1,\ldots,N_2+1$. for $n_1=1,\ldots,N_1+1$ and $n_2=1,\ldots,N_2+1$. The eigenvalues of the Laplacian matrix are the sums of these eigenvalues, given by

$$\lambda_{n_1,n_2}=4\,\sin^2(\tfrac{1}{2}\,\alpha_{n_1})+4\,\sin^2(\tfrac{1}{2}\,\beta_{n_2}) \tag{3.8.15}$$

or

$$\lambda_{n_1,n_2}=4-2\cos\alpha_{n_1}-2\cos\beta_{n_2}, \tag{3.8.16}$$

or

$$\lambda_{n_1,n_2}=4-2\cos\alpha_{n_1}-2\cos\beta_{n_2}. \tag{3.8.17}$$

The corresponding eigenvectors, $\mathbf{u}^{(n_1,n_2)}$, normalized so that their lengths are equal to unity, $\mathbf{u}^{(n_1,n_2)}\cdot\mathbf{u}^{(n_1,n_2)}=1$, are

$$u_{i_1,i_2}^{n_1,n_2}=A_{n_1}B_{n_2}\frac{2}{\sqrt{(N_1+1)(N_2+1)}} \times\cos\big((i_1-\tfrac{1}{2})\,\alpha_{n_1}\big)\cos\big((i_2-\tfrac{1}{2})\,\beta_{n_2}\big) \tag{3.8.18}$$

for $n_1, i_1 = 1, \ldots, N_1 + 1$ and $n_2, i_2 = 1, \ldots, N_2 + 1$, where $A_{n_1} = 1$, except that $A_1 = 1/\sqrt{2}$ and $B_1 = 1/\sqrt{2}$.

3.8.6 Code for the graph Laplacian

The construction of the Laplacian is implemented in the following Matlab function located in directory GRAPHS of SEMLIB:

```
N1 = 3;
N2 = 2;

%---
% tridiagonal in x
%---

U(1,1) = 1; U(1,2) = -1;
U(N1+1,N1) = -1; U(N1+1,N1+1) = 1;

for i=2:N1
 U(i,i) = 2;
 U(i,i-1) = -1;
 U(i,i+1) = -1;
end

%---
% tridiagonal in y
%---

V(1,1) = 1; V(1,2) = -1;
V(N2+1,N2) = -1; V(N2+1,N2+1) = 1;

for i=2:N2
 V(i,i) = 2;
 V(i,i-1) = -1;
 V(i,i+1) = -1;
end

Lapl = kron(eye(N2+1),U) + kron(V,eye(N1+1));
Lapl
```

Running the code generates the following output, as instructed by the last line of the code:

```
Lapl =

     2    -1     0     0    -1     0     0     0     0     0     0     0
    -1     3    -1     0     0    -1     0     0     0     0     0     0
     0    -1     3    -1     0     0    -1     0     0     0     0     0
     0     0    -1     2     0     0     0    -1     0     0     0     0
    -1     0     0     0     3    -1     0     0    -1     0     0     0
     0    -1     0     0    -1     4    -1     0     0    -1     0     0
     0     0    -1     0     0    -1     4    -1     0     0    -1     0
     0     0     0    -1     0     0    -1     3     0     0     0    -1
     0     0     0     0    -1     0     0     0     2    -1     0     0
     0     0     0     0     0    -1     0     0    -1     3    -1     0
     0     0     0     0     0     0    -1     0     0    -1     3    -1
     0     0     0     0     0     0     0    -1     0     0    -1     2
```

We recall that the diagonal *threes* correspond to the boundary nodes and the diagonal *fours* correspond to the interior nodes.

3.8.7 Periodic lattice

When the lattice supports a field that is periodic in the first direction, we use the $N_1 \times N_1$ constituent matrix

$$\mathbf{U} = \begin{bmatrix} 2 & -1 & \cdots & \cdots & \cdots & -1 \\ -1 & 2 & -1 & 0 & \cdots & \cdots \\ 0 & -1 & 2 & -1 & 0 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & 0 & -1 & 2 & -1 & 0 \\ \cdots & \cdots & 0 & -1 & 2 & -1 \\ -1 & \cdots & \cdots & 0 & -1 & 2 \end{bmatrix}. \tag{3.8.19}$$

The eigenvalues of this constituent matrix are bea

$$\lambda_{n_1} = 4 \sin^2(\tfrac{1}{2}\gamma_{n_1}), \tag{3.8.20}$$

where

$$\gamma_{n_1} = \frac{n_1 - 1}{N_1} 2\pi \tag{3.8.21}$$

for $n_1 = 1, \ldots, N_1$. In fact, the matrix $\mathbf{U}$ shown in (3.8.19) is the negative of that shown in (3.6.1).

For $N_1 = 3$ and $N_2 = 3$, we obtain the following 12×12 graph Laplacian:

```
Lapl =

     3 -1 -1 -1  0  0  0  0  0  0  0  0
    -1  3 -1  0 -1  0  0  0  0  0  0  0
    -1 -1  3  0  0 -1  0  0  0  0  0  0
    -1  0  0  4 -1 -1 -1  0  0  0  0  0
     0 -1  0 -1  4 -1  0 -1  0  0  0  0
     0  0 -1 -1 -1  4  0  0 -1  0  0  0
     0  0  0 -1  0  0  4 -1 -1 -1  0  0
     0  0  0  0 -1  0 -1  4 -1  0 -1  0
     0  0  0  0  0 -1 -1 -1  4  0  0 -1
     0  0  0  0  0  0 -1  0  0  3 -1 -1
     0  0  0  0  0  0  0 -1  0 -1  3 -1
     0  0  0  0  0  0  0  0 -1 -1 -1  3
```

(3.8.22)

The eigenvalues of the Laplacian can be constructed readily as sums of all possible combinations of the eigenvalues of $\mathbf{U}$ and $\mathbf{V}$.

Exercise

3.8.1 Compute the eigenvalues of a rectangular patch that is periodic in the first direction.

3.9 Toroidal lattice

Continuing the discussion of Section 3.8, now we consider a nodal scalar field that is periodic in two directions, so that

$$f_{1,i_2} = f_{N_1+1,i_2}, \qquad f_{i_1,1} = f_{i_1,N_2+1}. \tag{3.9.1}$$

The vector of nodal values inside each period consists of N_2 blocks in the following configuration:

$$\mathbf{f} = \begin{bmatrix} \mathbf{f}^{(1)} \\ \mathbf{f}^{(2)} \\ \vdots \\ \mathbf{f}^{(N_2-1)} \\ \mathbf{f}^{(N_2)} \end{bmatrix}, \tag{3.9.2}$$

where the subvectors

$$\mathbf{f}^{(1)} = \begin{bmatrix} f_{1,1} \\ f_{2,1} \\ \vdots \\ f_{N_1,1} \end{bmatrix}, \quad \ldots, \quad \mathbf{f}^{(N_2)} = \begin{bmatrix} f_{1,N_2} \\ f_{2,N_2} \\ \vdots \\ f_{N_1,N_2} \end{bmatrix} \tag{3.9.3}$$

encapsulate horizontal profiles. The doubly periodic configuration describes a square network on the surface of an untwisted torus resembling a doughnut.

3.9.1 Doubly periodic Laplacian

The doubly periodic Laplacian consists of N_2 rows of $N_1 \times N_1$ nearly tridiagonal blocks, $\mathbf{E}$, upper and lower diagonal negative unit blocks, $-\mathbf{I}$, and two negative unit corner blocks implementing the periodicity condition in the second direction in the following configuration:

$$\mathbf{L} = \begin{bmatrix} \mathbf{E} & -\mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & -\mathbf{I} \\ -\mathbf{I} & \mathbf{E} & -\mathbf{I} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\mathbf{I} & \mathbf{E} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{E} & -\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & -\mathbf{I} & \mathbf{E} & -\mathbf{I} \\ -\mathbf{I} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & -\mathbf{I} & \mathbf{E} \end{bmatrix}, \tag{3.9.4}$$

where $\mathbf{I}$ is the $N_1 \times N_1$ identity matrix. When $N_1 = 5$, the matrix $\mathbf{E}$

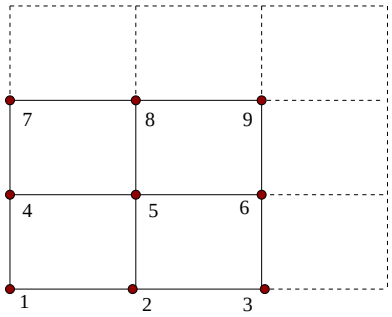


FIGURE 3.9.1 Illustration of a doubly periodic network with $N_1 = 3$ and $N_2 = 3$ showing the node numbering scheme inside one period.

takes the form

$$\mathbf{E} = \begin{bmatrix} 4 & -1 & 0 & 0 & -1 \\ -1 & 4 & -1 & 0 & 0 \\ 0 & -1 & 4 & -1 & 0 \\ 0 & 0 & -1 & 4 & -1 \\ -1 & 0 & 0 & -1 & 4 \end{bmatrix}. \tag{3.9.5}$$

The northeastern and southwestern corner elements of $\mathbf{E}$ implement the periodicity condition in the first direction. The northeastern and southwestern corner blocks of $\mathbf{L}$ implement the periodicity condition in the second direction.

The doubly periodic network for $N_1 = 3$ and $N_2 = 3$ is shown in Figure 3.9.1. The node labels shown in this figure mark the node location in the nodal vector field, $\mathbf{f}$. The associated 9×9 doubly periodic Laplacian is shown in Table 3.9.1(a). The four near neighbors of node 1 are node 2, node 3 due to the periodicity condition in the first direction, node 4, and node 7 due to the periodicity condition in the second direction.

3.9.2 Laplacian in terms of tensor products

The doubly periodic graph Laplacian can be expressed as the sum of a

(a)

```
 4   -1   -1   -1    0    0   -1    0    0
-1    4   -1    0   -1    0    0   -1    0
-1   -1    4    0    0   -1    0    0   -1
-1    0    0    4   -1   -1   -1    0    0
 0   -1    0   -1    4   -1    0   -1    0
 0    0   -1   -1   -1    4    0    0   -1
-1    0    0   -1    0    0    4   -1   -1
 0   -1    0    0   -1    0   -1    4   -1
 0    0   -1    0    0   -1   -1   -1    4
```

(b)

```
 4   -1    0   -1    0   -1   -1    0    0
-1    4   -1    0   -1    0    0   -1    0
 0   -1    4    0    0   -1   -1    0   -1
-1    0    0    4   -1    0   -1    0   -1
 0   -1    0   -1    4   -1    0   -1    0
-1    0   -1    0   -1    4    0    0   -1
-1    0   -1   -1    0    0    4   -1    0
 0   -1    0    0   -1    0   -1    4   -1
 0    0   -1   -1    0   -1    0   -1    4
```

(c)

```
 4   -1    0   -1    0    0   -1    0   -1
-1    4   -1    0   -1    0    0   -1    0
 0   -1    4   -1    0   -1    0    0   -1
-1    0   -1    4   -1    0   -1    0    0
 0   -1    0   -1    4   -1    0   -1    0
 0    0   -1    0   -1    4   -1    0   -1
-1    0    0   -1    0   -1    4   -1    0
 0   -1    0    0   -1    0   -1    4   -1
-1    0   -1    0    0   -1    0   -1    4
```

TABLE 3.9.1 The 9×9 Laplacian of the (a) untwisted torus, (b) twisted torus with $m = 1$, and (c) twisted torus with $m = 2$.

Kronecker sum,

$$\mathbf{A} = \mathbf{U} \oplus \mathbf{V} = \mathbf{I}_{N_2} \otimes \mathbf{U} + \mathbf{V} \otimes \mathbf{I}_{N_1}, \tag{3.9.6}$$

where the constituent $N_1 \times N_1$ matrix $\mathbf{U}$ shown in (3.8.19) is used for $\mathbf{U}$ and the same matrix with dimensions $N_2 \times N_2$ is used for $\mathbf{V}$.

3.9.3 Eigenvalues

The eigenvalues of the doubly periodic Laplacian can be deduced by summation from those of the constituent matrices $\mathbf{U}$ and $\mathbf{V}$, and are given by

$$\lambda_{n_1,n_2} = 4 - 2\,\cos\alpha_{n_1} - 2\,\cos\beta_{n_2}, \tag{3.9.7}$$

where

$$\alpha_{n_1} = \frac{n_1 - 1}{N_1}\,2\pi, \qquad \beta_{n_2} = \frac{n_2 - 1}{N_2}\,2\pi \tag{3.9.8}$$

for $n_1 = 1, \ldots, N_1$ and $n_2 = 1, \ldots, N_2$.

3.9.4 Twisted torus

The doubly periodic Laplacian describes a square network on the surface of an untwisted torus. The toroidal network can be cut at a point across the centerline, and one end can be twisted around the centerline by an angle

$$\phi_m = m\,\frac{2\pi}{N_2} \tag{3.9.9}$$

for $m = 0, \ldots, N_2 - 1$ while the other end is held firm, as shown in Figure 3.9.2(a). Although higher values of m are permissible, they can be mapped to lower values $m - N_2$. Finally, the two ends of the can be rejoined to yield a twisted toroidal network, as shown in Figure 3.9.1(b, c) for $m = 1, 2$.

The nodal profile of the twisted torus satisfies the regular periodicity condition in the second direction,

$$f_{i_1,N_2+1} = f_{i_1,1}, \tag{3.9.10}$$

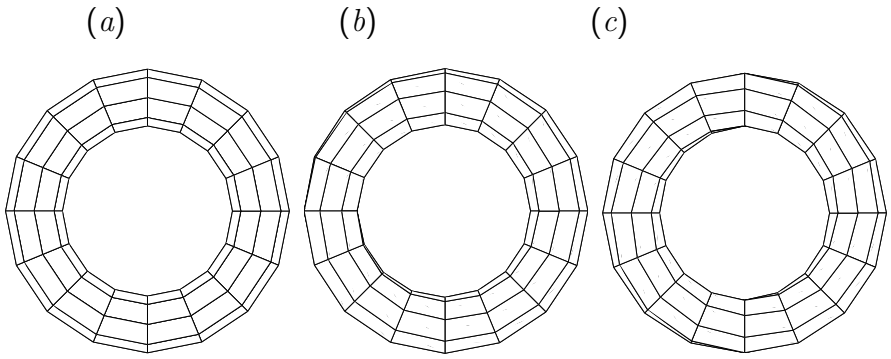


FIGURE 3.9.2 Illustration of (a) an untwisted square network ($m = 0$) and (b,c) a twisted square network on the surface of a torus with $m = 1, 2$.

and a modified periodicity condition in the first direction,

$$f_{N_1+1,i_2} = f_{1,i_2-m} \tag{3.9.11}$$

for $i_2 = 1, \ldots, N_2$. When $m = 0$, we obtain the usual periodicity condition discussed previously in this section.

3.9.5 *Direct construction of the Laplacian*

The following Matlab code entitled *torus_lapl*, located in directory TORUS of SEMLIB, generates the doubly periodic Laplacian by direct construction:

```
N1 = 3;
N2 = 3;

m = 1;

%-----------
% initialize
%-----------

for j=1:N2
 for i=1:N1
```

```
  Lap(i,j) = 0;
 end
end

%---
% r(i,j):
% location of the i,j node
% in the solution vector
%---

Ic = 0;

for j=1:N2
 for i=1:N1
  Ic = Ic+1;
  r(i,j) = Ic;
  Lap(Ic,Ic) = 4;
 end
end

%---
% fill in the Laplacian matrix
%---

for j=1:N2
 for i=1:N1

   rnod = r(i,j);

   if(i==1)           % left link
    jlef = j+m;
    if(jlef>N2) jlef = jlef-N2; end
    if(jlef<1)  jlef = jlef+N2; end
    rlef = r(N1,jlef);
   else
    rlef = r(i-1,j);
   end

   if(i==N1)            %right
```

```
    jrit = j-m;
    if(jrit<1)  jrit = jrit+N2; end
    if(jrit>N2) jrit = jrit-N2; end
    rrit = r(1,jrit);
   else
    rrit = r(i+1,j);
   end

   if(j==1)            % bottom
    rbot = r(i,N2);
   else
    rbot = r(i,j-1);
   end

   if(j==N2)           % top
    rtop = r(i,1);
   else
    rtop = r(i,j+1);
   end

   Lap(rnod,rlef) = -1;
   Lap(rnod,rrit) = -1;
   Lap(rnod,rbot) = -1;
   Lap(rnod,rtop) = -1;
 end
end
```

Consider a small network with $N_1 = 3$ and $N_2 = 3$, as shown in Figure 3.9.1. For $m = 1$, we obtain the 9×9 Laplacian shown in Table 3.9.1(b). For $m = 2$, we obtain the 9×9 Laplacian shown in Table 3.9.1(c). These two Laplacians can be compared with that listed in Table 3.9.1(a) for the untwisted torus, $m = 0$.

The corresponding eigenvalues are shown in Table 3.9.3. We observe that the eigenvalues for $m = 1$ and 2 are the same and different than those for $m = 0$. The reason is that, if a traveler starts moving along a certain line i_2, he/she will remain on that line after one complete revolution. By contrast, for $m = 1$, a traveler who starts on the line $i_2 = 1$ will transition to $i_2 = 2$, then to $i_2 = 3$, and back to $i_2 = 1$.

$m = 0$	$m = 1, 2$
0.0000	0.0000
3.0000	3.0000
3.0000	3.0000
3.0000	3.4679
3.0000	3.4679
6.0000	4.6527
6.0000	4.6527
6.0000	6.8794
6.0000	6.8794

TABLE 3.9.3 Eigenvalues of the untwisted ($m = 0$) and twisted torus for $N_1 = 3$ and $N_2 = 3$.

For $m = 2$, a travelers who starts on the line $i_2 = 1$ will transition to $i_2 = 3$, then to $i_2 = 2$, and then back to $i_2 = 1$. Thus, the paths $m = 1, 2$ are topologically equivalent.

For $N_2 = 4$, we obtain the following orbital sequences:

$$\begin{aligned} m = 0 \quad &\rightarrow \quad i_2 = 1 \\ m = 1 \quad &\rightarrow \quad i_2 = 1, 2, 3, 4, 1 \\ m = 2 \quad &\rightarrow \quad i_2 = 1, 3, 1 \\ m = 3 \quad &\rightarrow \quad i_2 = 1, 4, 3, 2, 1 \end{aligned} \tag{3.9.12}$$

These sequences are independent of N_1. Because the sizes of the orbital set for $m = 1, 3$ are the same, the corresponding eigenvalues of the twisted Laplacian are identical and different than those for $m = 0$ and $m = 2$, as shown in Table 3.9.2.

For $N_2 = 5$, we obtain the following orbital sequences:

$$\begin{aligned} m = 0 \quad &\rightarrow \quad i_2 = 1 \\ m = 1 \quad &\rightarrow \quad i_2 = 1, 2, 3, 4, 5, 1 \\ m = 2 \quad &\rightarrow \quad i_2 = 1, 3, 5, 2, 4, 1 \\ m = 3 \quad &\rightarrow \quad i_2 = 1, 4, 2, 5, 3, 1 \\ m = 4 \quad &\rightarrow \quad i_2 = 1, 5, 4, 3, 1 \end{aligned} \tag{3.9.13}$$

$m = 0$	$m = 1, 3$	$m = 2$
0.0000	0.0000	0.0000
2.0000	2.0000	2.0000
2.0000	2.0000	2.0000
2.0000	2.1522	2.5858
2.0000	2.1522	2.5858
4.0000	3.2346	2.5858
4.0000	3.2346	2.5858
4.0000	4.0000	4.0000
4.0000	4.5858	4.0000
4.0000	4.5858	5.4142
4.0000	4.7654	5.4142
6.0000	4.7654	5.4142
6.0000	5.8478	5.4142
6.0000	5.8478	6.0000
6.0000	7.4142	6.0000
8.0000	7.4142	8.0000

TABLE 3.9.2 Eigenvalues of the untwisted ($m = 0$) and twisted torus for $N_1 = 4$ and $N_2 = 4$.

Three sets of eigenvalues are expected for this configuration.

Exercise

3.9.1 Compute the eigenvalues of the twisted torus for $N_1 = 3$, $N_2 = 3$, and $m = 1, 2$.

3.10 Møbius strip

The rectangular $N_1 \times N_2$ patch of the square lattice shown in Figure 3.8.1 can be twisted by 180° around its length, and the twisted strip can be bent and its narrow edges can be glued to yield the Möbius strip

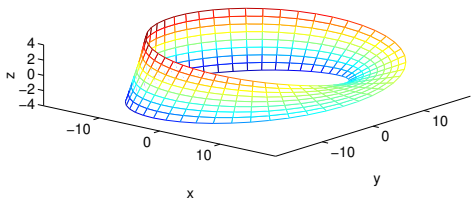


FIGURE 3.10.1 A Cartesian strip can be twisted by 180° along its length to give a helical strip, and the helical strip can be bent and the narrow edges and then attached to yield the Möbius strip.

shown in Figure 3.9.1. In principle, a higher number of twists can be performed before the edges are glued.

3.10.1 Construction

The Möbius strip shown in Figure 3.10.1 was generated by the following Matlab function entitled *mobius_strip1*, located in directory MOBIUS of SEMLIB, implementing a standard Möbius transformation:

```
N1 = 12*4; % number of links along the strip
N2 = 2*4; % number of links across the strip

a1 = 2.0; % link length along the strip
a2 = 1.0; % link length  across the strip

%---
% prepare
%---

L = N1*a1;  % nominal strip length
W = N2*a2;  % nominal strip width

ushift = 0.18771; % arbitrary
vshift = 0.5; % fixed
```

```
%---
% transform
%---

for j=1:N2+1

 v = (j-1)/N2-vshift;

 for i=1:N1+1
  u = 2*pi*(i-1-ushift)/N1-pi;
  tmp = 1.0+0.5*v*cos(0.5*u);
  px(i,j) = tmp*cos(u);
  py(i,j) = tmp*sin(u);
  pz(i,j) = 0.5*v*sin(0.5*u);
  pc(i,j) = pz(i,j);   % color
  px(i,j) = L/(2*pi)*px(i,j); % scale
  py(i,j) = L/(2*pi)*py(i,j); % scale
  pz(i,j) = 2*W*pz(i,j);      % scale
 end

end

%---
% plot
%---

 mesh(px,py,pz,pc)
```

In the code, v is the twist angle about the strip centerline, and the u is the rotation angle around the centerline.

3.10.2 Nodal field

The nodal profile of a Möbius strip in the direction of the first index, i_1, satisfies the reverse periodicity condition

$$f_{1,i_2} = f_{N_1+1,N_2+2-i_2} \tag{3.10.1}$$

for $i_2 = 1, \ldots, N_2 + 1$. For example, the southwestern nodal value is

equal to the northeastern nodal value,

$$f_{1,1} = f_{N_1+1,N_2+1}. \tag{3.10.2}$$

The vector of nodal values encapsulating $N = N_1\,(N_2+1)$ unknowns is given by

$$\mathbf{f} = \begin{bmatrix} \mathbf{f}^{(1)} \\ \mathbf{f}^{(2)} \\ \vdots \\ \mathbf{f}^{(N_2)} \\ \mathbf{f}^{(N_2+1)} \end{bmatrix}, \tag{3.10.3}$$

where

$$\mathbf{f}^{(1)} = \begin{bmatrix} f_{1,1} \\ f_{2,1} \\ \vdots \\ f_{N_1-1,1} \\ f_{N_1,1} \end{bmatrix}, \quad \ldots, \quad \mathbf{f}^{(N_2+1)} = \begin{bmatrix} f_{1,N_2+1} \\ f_{2,N_2+1} \\ \vdots \\ f_{N_1-1,N_2+1} \\ f_{N_1,N_2+1} \end{bmatrix}. \tag{3.10.4}$$

The associated doubly periodic Laplacian matrix consists of N_2+1 rows of $N_1 \times N_1$ square tridiagonal blocks, $\mathbf{F}$ and $\mathbf{E}$, and a chain of sparse back-diagonal blocks,

$$\mathbf{L} = \begin{bmatrix} \mathbf{F} & -\mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & -\mathbf{J} \\ -\mathbf{I} & \mathbf{E} & -\mathbf{I} & \cdots & \mathbf{0} & -\mathbf{J} & \mathbf{0} \\ \mathbf{0} & -\mathbf{I} & \mathbf{E} & \cdots & -\mathbf{J} & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & -\mathbf{J} & \cdots & \mathbf{E} & -\mathbf{I} & \mathbf{0} \\ \mathbf{0} & -\mathbf{J} & \mathbf{0} & \cdots & -\mathbf{I} & \mathbf{E} & -\mathbf{I} \\ -\mathbf{J} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & -\mathbf{I} & \mathbf{F} \end{bmatrix}, \tag{3.10.5}$$

where $\mathbf{I}$ is the $N_1 \times N_1$ unit matrix. The $N_1 \times N_1$ matrix $\mathbf{J}$ is null, expect that the northeastern and southwestern corner elements are equal to unity. When N_2 is even, the central block of $\mathbf{L}$ is $\mathbf{E} - \mathbf{J}$.

For $N_1 = 4$, we obtain the matrices

$$\mathbf{F} = \begin{bmatrix} 3 & -1 & 0 & 0 \\ -1 & 3 & -1 & 0 \\ 0 & -1 & 3 & -1 \\ 0 & 0 & -1 & 3 \end{bmatrix}, \qquad \mathbf{E} = \begin{bmatrix} 4 & -1 & 0 & 0 \\ -1 & 4 & -1 & 0 \\ 0 & -1 & 4 & -1 \\ 0 & 0 & -1 & 4 \end{bmatrix} \tag{3.10.6}$$

and

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad \mathbf{J} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}. \tag{3.10.7}$$

The *three* entries of $\mathbf{F}$ correspond to the bottom and top edge nodes.

3.10.3 Graph Laplacian as a sum of tensor products

In fact, the graph Laplacian shown in (3.10.5) is given by the sum of three tensor products,

$$\mathbf{A} = \mathbf{I}_{N_2+1} \otimes \mathbf{U} - \mathbf{K}_{N_2+1} \otimes \mathbf{J} + \mathbf{V} \otimes \mathbf{I}_{N_1}, \tag{3.10.8}$$

where

- $\mathbf{I}_{N_2+1}$ is the $(N_2+1) \times (N_2+1)$ identity matrix
- $\mathbf{K}_{N_2+1}$ is the $(N_2+1) \times (N_2+1)$ back-diagonal unit matrix
- $\mathbf{I}_{N_1}$ is the $N_1 \times N_1$ identity matrix

Moreover,

$$\mathbf{U} = \begin{bmatrix} 2 & -1 & \cdots & \cdots & \cdots & \cdots \\ -1 & 2 & -1 & 0 & \cdots & \cdots \\ 0 & -1 & 2 & -1 & 0 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & 0 & -1 & 2 & -1 & 0 \\ \cdots & \cdots & 0 & -1 & 2 & -1 \\ \cdots & \cdots & \cdots & 0 & -1 & 2 \end{bmatrix} \tag{3.10.9}$$

is an $N_1 \times N_1$ tridiagonal matrix, and

$$\mathbf{V} = \begin{bmatrix} 1 & -1 & \cdots & \cdots & \cdots & \cdots \\ -1 & 2 & -1 & 0 & \cdots & \cdots \\ 0 & -1 & 2 & -1 & 0 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & 0 & -1 & 2 & -1 & 0 \\ \cdots & \cdots & 0 & -1 & 2 & -1 \\ \cdots & \cdots & \cdots & 0 & -1 & 1 \end{bmatrix} \tag{3.10.10}$$

is a $(N_2 + 1) \times (N_2 + 1)$ tridiagonal matrix.

Replacing the back-diagonal unit matrix $\mathbf{K}_{N_2+1}$ with identity matrix $\mathbf{I}_{N_2+1}$ with identity matrix yields the periodic Laplacian shown in (3.8.22).

3.10.4 Code for the Laplacian

The construction of the Laplacian is implemented in the following Matlab code entitled *mobius_lapl* located in directory MOBIUS of SEMLIB:

```
%==================
% Graph Laplacian matrix
% of the Mobius network
%==================

N1 = 3;
N2 = 3;

%---
% unit back-diagonal
%---

for i=1:N2+1
  K(i,N2+2-i) = 1.0;
end

%---
% tridiagonal
% in the first direction
```

```
%---

U(1,1) = 2; U(1,2) = -1;
U(N1,N1) = 2; U(N1,N1-1) = -1;

for i=2:N1-1
 U(i,i) = 2;
 U(i,i-1) = -1;
 U(i,i+1) = -1;
end

%---
% tridiagonal
% in the second direction
%---

V(1,1) = 1; V(1,2) = -1;
V(N2+1,N2+1) = 1; V(N2+1,N2) = -1;

for i=2:N2
 V(i,i) = 2;
 V(i,i-1) = -1;
 V(i,i+1) = -1;
end

%---
% matrix J
%---

J(1,N1) = 1.0;
J(N1,1) = 1.0;

%---
% Laplacian
%---

Lapl = kron(eye(N2+1),U) - kron(K,J) + kron(V,eye(N1));
Lapl
```

Running the code generates the following output, as instructed by the

last line of the code:

```
Lapl =

   3  -1   0  -1   0   0   0   0   0   0   0  -1
  -1   3  -1   0  -1   0   0   0   0   0   0   0
   0  -1   3   0   0  -1   0   0   0  -1   0   0
  -1   0   0   4  -1   0  -1   0  -1   0   0   0
   0  -1   0  -1   4  -1   0  -1   0   0   0   0
   0   0  -1   0  -1   4  -1   0  -1   0   0   0
   0   0   0  -1   0  -1   4  -1   0  -1   0   0
   0   0   0   0  -1   0  -1   4  -1   0  -1   0
   0   0   0  -1   0  -1   0  -1   4   0   0  -1
   0   0  -1   0   0   0  -1   0   0   3  -1   0
   0   0   0   0   0   0   0  -1   0  -1   3  -1
  -1   0   0   0   0   0   0   0  -1   0  -1   3
```

This display is consistent with the structure shown in (3.10.5) involving the back-diagonal block of the matrix $-\mathbf{J}$.

3.10.5 Eigenvalues

The eigenvalues of the Laplacian are given by

$$\lambda_{n_1,n_2} = 4\,\sin^2(\tfrac{1}{2}\,\alpha_{n_1,n_2}) + 4\,\sin^2(\tfrac{1}{2}\,\beta_{n_2}) \tag{3.10.11}$$

or

$$\lambda_{n_1,n_2} = 4 - 2\cos\alpha_{n_1,n_2} - 2\cos\beta_{n_2}, \tag{3.10.12}$$

where

$$\alpha_{n_1,n_2} = \frac{n_1 - 1 + \gamma_{n_2}}{N_1}\,2\pi, \qquad \beta_{n_2} = \frac{n_2 - 1}{N_2 + 1}\,\pi \tag{3.10.13}$$

for $n_1 = 1, \ldots, N_1$ and $n_2 = 1, \ldots, N_2 + 1$, where $\gamma_{n_2} = 0$ if n_2 is odd and $\gamma_{n_2} = 1/2$ if n_2 is even. Formally, we write

$$\gamma_{n_2} = \frac{1 + (-1)^{n_2}}{4}. \tag{3.10.14}$$

Note that these expressions differ from those given (3.9.7) for the periodic rectangular network only by the presence of γ_{n_2} in the first fraction.

3.10.6 Confirmation of eigenvalues

The following Matlab code, continuing the code listed previously in this section, confirms the eigenvalues:

```
egvl = eig(Lapl);

%---
% eigenvalues analytically
%---

Ic = 0;

for n1=1:N1
 for n2=1:N2+1
  gamma = ( 1+(-1)^n2 )/4.0;
  alpha = ( n1-1+gamma )*2.0*pi/N1;
  beta  = (n2-1)*pi/(N2+1);
  Ic = Ic+1;
  lmd(Ic) = 4*sin(0.5*alpha)^2 + 4*sin(0.5*beta)^2;
  lambda(n1,n2) = lmd(Ic);
 end
end

lmd = sort(lmd);

[egvl lmd']
```

Running the code generates the following output:

```
   0.00000   0.00000
   1.58579   1.58579
   1.58579   1.58579
   2.00000   2.00000
   3.00000   3.00000
   3.00000   3.00000
   4.41421   4.41421
   4.41421   4.41421
   4.58579   4.58579
   5.00000   5.00000
```

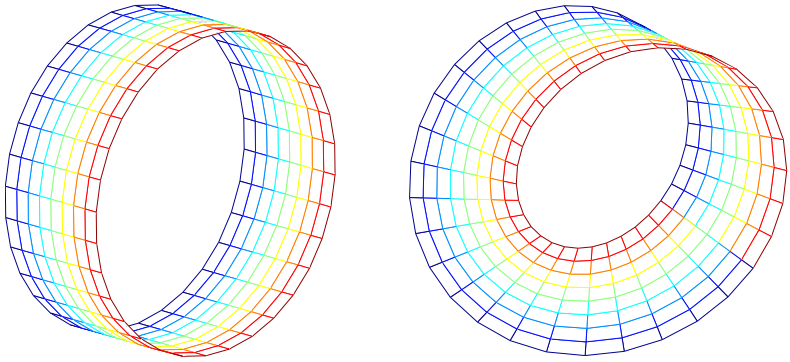


FIGURE 3.11.1 To generate the Möbius strip, we may bend a rectangular patch of the square lattice into a cylindrical strip, as shown on the left, and then twist the cross-section of the strip along the centerline.

```
5.00000    5.00000
7.41421    7.41421
```

The results reconcile the numerical with the analytical evaluation.

Exercises

3.10.1 Explain why the Möbius strip has a single twisted boundary consisting of the upper and lower edges of the generating rectangular patch.

3.11 Direct construction of the Mobius strip

The Möbius strip shown in Figure 3.10.1 was constructed by a parametric coordinate transformation. In an alternative direct construction, we roll a rectangular patch of the square lattice into a cylindrical strip, cut the strip across its width, and then twist the cross-section around the centerline to obtain the Möbius strip shown in Figure 3.11.1.

3.11.1 Rotation about a Cartesian axis

To implement the construction, we recall that the new position of a material point that has rotated about the x axis by an angle φ_x, around the y axis by an angle φ_y, or around the z axis by an angle φ_z, is given by

$$\mathbf{x}^{\text{new}} = \boldsymbol{\mathcal{R}}^{(x)}(\varphi_x)\cdot\mathbf{x}, \quad \mathbf{x}^{\text{new}} = \boldsymbol{\mathcal{R}}^{(y)}(\varphi_y)\cdot\mathbf{x}, \quad \mathbf{x}^{\text{new}} = \boldsymbol{\mathcal{R}}^{(z)}(\varphi_z)\cdot\mathbf{x}, \tag{3.11.1}$$

where $\mathbf{x}$ is the old position and

$$\boldsymbol{\mathcal{R}}^{(x)}(\varphi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \sin\varphi & \cos\varphi \end{bmatrix},$$

$$\boldsymbol{\mathcal{R}}^{(y)}(\varphi) = \begin{bmatrix} \cos\varphi & 0 & \sin\varphi \\ 0 & 1 & 0 \\ -\sin\varphi & 0 & \cos\varphi \end{bmatrix}, \tag{3.11.2}$$

$$\boldsymbol{\mathcal{R}}^{(z)}(\varphi) = \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

are orthogonal rotation matrices. The transpose of each one of these matrices is equal to the corresponding inverse.

3.11.2 Rotation around an arbitrary axis

To rotate a point by a specified angle φ about an axis x_1' that passes through the origin, as shown in Figure 3.11.2 it is convenient to introduce a rotated Cartesian coordinate system comprised of primed axes. We then apply a forward transformation, $\mathbf{x}\to\mathbf{x}'$, followed by a rotation and then a backward transformation, $\mathbf{x}'\to\mathbf{x}$, to obtain the overall transformation

$$\mathbf{x}^{\text{rotated}} = \mathbf{P}\cdot\mathbf{x}, \tag{3.11.3}$$

where

$$\mathbf{P} = \mathbf{A}^{\mathrm{T}}\cdot\boldsymbol{\mathcal{R}}^{(x)}(\varphi)\cdot\mathbf{A} \tag{3.11.4}$$

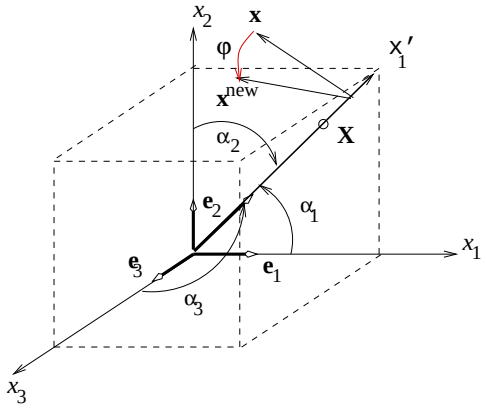


FIGURE 3.11.2 Rotation of a point $\mathbf{x}$ around the x_1' axis by angle φ. The new coordinates are found using the projection matrix (3.11.4).

is a projection matrix. The first row of the coordinate rotation matrix $\mathbf{A}$ contains the direction cosines of the x_1' axis,

$$a \equiv A_{11} = \cos\alpha_1, \qquad b \equiv A_{12} = \cos\alpha_2,$$
$$c \equiv A_{13} = \cos\alpha_3, \tag{3.11.5}$$

where $a^2 + b^2 + c^2 = 1$. The elements in the second and third row disappear after the multiplications are carried out in (3.11.4). In practice, we may specify a point $\mathbf{X}$ on the x_1' axis, and compute the direction cosines

$$a = \frac{X}{|\mathbf{X}|}, \qquad b = \frac{Y}{|\mathbf{X}|}, \qquad c = \frac{Z}{|\mathbf{X}|}, \tag{3.11.6}$$

as shown in Figure 3.11.1. Note that $\mathbf{P} \cdot \mathbf{X} = \mathbf{X}$, in agreement with physical intuition.

Carrying out the multiplications in (3.11.4) and simplifying, we derive the projection matrix

$$\mathbf{P} = \cos\varphi\,\mathbf{I} + (1-\cos\varphi)\begin{bmatrix} a^2 & ab & ac \\ ab & b^2 & bc \\ ac & bc & c^2 \end{bmatrix} + \sin\varphi\begin{bmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{bmatrix}, \tag{3.11.7}$$

where $\mathbf{I}$ is the identity matrix. The matrix in the second term on the right-hand side of (3.11.7) is symmetric, whereas the matrix in the third term is skew-symmetric.

Precisely the same rotated vector is obtained from the formula

$$\mathbf{x}^{\text{rotated}} = \mathbf{e}\,(\mathbf{x}\cdot\mathbf{e}) + \cos\varphi\,(\mathbf{I}-\mathbf{e}\otimes\mathbf{e})\cdot\mathbf{x} + \sin\varphi\,\mathbf{e}\times\mathbf{x}, \tag{3.11.8}$$

where $\mathbf{e} = (a, b, c)$ is a unit vector. We recall that

$$(\mathbf{I}-\mathbf{e}\otimes\mathbf{e})\cdot\mathbf{x} = \mathbf{e}\times\mathbf{x}\times\mathbf{e} \tag{3.11.9}$$

is the component of $\mathbf{x}$ parallel to $\mathbf{e}$.

3.11.3 Code

The construction is implemented in the following Matlab function located in directory MOBIUS of SEMLIB:

```
%=======================
% Mobius strip
%
% a1: link size in the first direction
% a2: link size in the second direction
%=======================

N1 = 8*4;
N2 = 2*4;

a1  = 2.0;
a2  = 1.0;
```

```
ntwists = 1;

angoff = 0.78615*pi;  % arbitrary
phioff = 0.27834*pi;  % arbitrary

%---
% prepare
%---

L = N1*a1;
W = N2*a2;

radius = L/(2.0*pi);

%---
% nodes in a planar rectangular strip
%
% -0.5*L < x < 0.5*L
% -0.5*W < y < 0.5*W
%---

for j=1:N2+1
 for i=1:N1+1
  p1(i,j) = -0.5*L+(i-1)*a1;
  p2(i,j) = -0.5*W+(j-1)*a2;
 end
end

%---
% run along the strip
%---

 for i=1:N1+1

%--------------------------------
% bend the strip around the second axis
% to form a ring of radius L/2pi
% around the second axis
```

```
%--------------------------------

  ang = 2*pi*(i-1)/N1-angoff;
  sang = sin(ang);
  cang = cos(ang);

  % centerline is a circle in the 31 plane

  pcl1(i) =  radius*cang; % centerline
  pcl2(i) =  0.0;         % centerline
  pcl3(i) =  radius*sang; % centerline

  % bend into a cylindrical strip

  for j=1:N2+1
   pp1(i,j) = radius*cang;
   pp2(i,j) = p2(i,j);
   pp3(i,j) = radius*sang;
   ppc(i,j) = p2(i,j); % color
  end

%---
% twist around the centerline
%---

  A = sang;
  B = 0.0;
  C =-cang;

  phi = ntwists*0.5*ang-phioff;
  sphi = sin(phi);
  cphi = cos(phi);

  for j=1:N2+1

   % distance of a node from the centerline
   % for a certain i

   dfc1 = pp1(i,j)-pcl1(i);
```

```
   dfc2 = pp2(i,j)-pcl2(i);
   dfc3 = pp3(i,j)-pcl3(i);

   R1 = [A*A A*B A*C;
         B*A B*B B*C;
         C*A C*B C*C];

   R2 = [0 -C  B;
         C  0 -A;
        -B  A  0];

   P = cphi*eye(3) + (1.0-cphi)*R1 + sphi*R2;

% distance from centerline

   dfc = P*[dfc1; dfc2; dfc3];

   pm1(i,j) = pcl1(i) + dfc(1);
   pm2(i,j) = pcl2(i) + dfc(2);
   pm3(i,j) = pcl3(i) + dfc(3);

  end

 end

%---
% plot
%---

 figure(1)
 mesh(pp1,pp2,pp3,ppc)

 figure(2)
 mesh(pm1,pm2,pm3,ppc)
```

Exercise

3.11.1 Derive the projection matrix (3.11.7).

Chapter 4

Fractional calculus

We are familiar with functions of one, two, or more independent variables and their ordinary or partial derivatives of integer order: first, second, and higher-order whole derivatives. These integral derivatives are defined as limits of differences between two neighboring function values divided by the infinitesimal increment in the associated independent variable.

Intermediate derivatives of fractional order can be introduced in several ways with the goal of describing physical, conceptual, or engineering processes that exhibit unusual behavior. Though the notion of fractional calculus sounds intuitive, fractional derivatives are not unique and their interpretation is not always clear. An exception pertains to the Riesz fractional derivative, also known as the fractional Laplacian. In general, fractional derivatives are best regarded as modeling tools.

In this chapter, we introduce the Riemann–Liouville left and right fractional integrals and derivatives, the backward and forward Grünwald–Letnikov fractional derivatives, the Caputo fractional derivative, and the fractional Laplacian. The Riemann–Liouville and the Grünwald–Letnikov fractional derivatives coincide for functions that obey sufficient regularity conditions.

The fractional Laplacian admits the most clear physical interpretation in the context of random processes involving random displacements or walks with a heavy-tailed power-law probability distribution.

4.1 Riemann–Liouville fractional integral

Consider a function, $f(x)$, and denote its definite integral from a specified fixed lower limit, $x_{\rm A}$, to an arbitrary upper limit, x,

$$[\,{\rm I}^{(1)} f\,]^{\rm RL}_{x_{\rm A}}(x) \equiv \int_{x_{\rm A}}^{x} f(\xi)\,{\rm d}\xi, \tag{4.1.1}$$

where the superscript ${\rm RL}$ stands for *Riemann–Liouville* and ξ is an integration variable. Next, we introduce the identity operation

$$[\,{\rm I}^{(0)} f\,]^{\rm RL}_{x_{\rm A}}(x) \equiv f(x). \tag{4.1.2}$$

Note the superscript (1) for the integral operator and the superscript (0) for the identity operator.

Having defined $[\,{\rm I}^{(0)} f\,]^{\rm RL}_{x_{\rm A}}(x)$, we introduce an integer, n, and define the integral

$$[\,{\rm I}^{(n)} f\,]^{\rm RL}_{x_{\rm A}}(x) \equiv \int_{x_{\rm A}}^{x} [\,{\rm I}^{(n-1)} f\,]^{\rm RL}_{x_{\rm A}}(\xi)\,{\rm d}\xi \tag{4.1.3}$$

recursively for $n = 1, 2, \ldots$. Cauchy showed that

$$[\,{\rm I}^{(n)} f\,]^{\rm RL}_{x_{\rm A}}(x) = \frac{1}{(n-1)\,!} \int_{x_{\rm A}}^{x} \frac{f(\xi)}{(x-\xi)^{1-n}}\,{\rm d}\xi \tag{4.1.4}$$

for $n = 1, 2, \ldots$ and $x \geq x_{\rm A}$, where an exclamation mark denotes the factorial, $(n-1)! = 1 \cdot 2 \cdots (n-1)$.

4.1.1 Fractional order

Motivated by (4.1.4), we note that $(n-1)\,! = \Gamma(n)$, where $\Gamma(n)$ is the Gamma function, and introduce the (left) Riemann–Liouville fractional integral of positive order, β, defined as

$$[\,{\rm I}^{(\beta)} f\,]^{\rm RL}_{x_{\rm A}}(x) \equiv \frac{1}{\Gamma(\beta)} \int_{x_{\rm A}}^{x} \frac{f(\xi)}{(x-\xi)^{1-\beta}}\,{\rm d}\xi \tag{4.1.5}$$

for $x \geq x_{\rm A}$ so that the difference $x - \xi$ in the denominator is positive. When $\beta = 1$, we obtain (4.1.1). As β tends to zero, we obtain (4.1.2). When β is an integer, we obtain (4.1.4).

It can be shown by direct substitution and using Fubini's theorem that

$$[\,\mathrm{I}^{(\beta)}\mathrm{I}^{(\gamma)}f\,]^{\mathrm{RL}}_{x_\mathrm{A}}(x) = [\,\mathrm{I}^{(\beta+\gamma)}f]^{\mathrm{RL}}_{x_\mathrm{A}}(x) \tag{4.1.6}$$

for any β and γ. This rule is reminiscent of that for the exponential function, $\exp(\beta)\exp(\gamma) = \exp(\beta+\gamma)$.

4.1.2 Integrable singularity

We are particularly interested in values of β in the range $0 < \beta < 1$, and in derivatives of the fractional integral with respect to the evaluation point, x, defining the fractional derivatives of $f(x)$, as discussed in Section 4.2.

When $0 < \beta < 1$, the kernel of the integrand on the right-hand side of (4.1.5),

$$\mathcal{K}(x,\xi) \equiv \frac{1}{(x-\xi)^{1-\beta}}, \tag{4.1.7}$$

is singular, this is, it takes an infinite value as $\xi \to x$. However, the singularity is integrable and the fractional integral is well defined. For example, when $\beta = \frac{1}{2}$, the kernel behaves as

$$\mathcal{K}(x,\xi) \equiv \frac{1}{\sqrt{x-\xi}} \tag{4.1.8}$$

whose indefinite integral with respect to ξ is $-2\sqrt{x-\xi}$. A singularity does not appear when $\beta \geq 1$,

4.1.3 Flat function

When the function $f(x)$ is flat, $f(x) = c$, where c is a constant, we perform the integration and obtain the fractional integral

$$[\,\mathrm{I}^{(\beta)}c\,]^{\mathrm{RL}}_{x_\mathrm{A}}(x) = \frac{c}{\beta\,\Gamma(\beta)}\,(x-x_\mathrm{A})^{\beta} \tag{4.1.9}$$

for $x > x_\mathrm{A}$. Using the properties of the Gamma function, we find that, as β tends to zero, the product $\beta\,\Gamma(\beta)$ tends to unity, and thereby obtain the expected result

$$[\,\mathrm{I}^{(0)}c\,]^{\mathrm{RL}}_{x_\mathrm{A}}(x) = c. \tag{4.1.10}$$

As β tends to unity, the product $\beta\,\Gamma(\beta)$ also tends to unity, providing us with the familiar expression for the whole integral,

$$[\,\mathrm{I}^{(1)}c]_{x_\mathrm{A}}^{\mathrm{RL}}(x) = c\,(x - x_\mathrm{A}). \tag{4.1.11}$$

Though the exponent on the right-hand side of (4.1.9) could have been deduced by intuition, the fraction in front of the gamma function is too specific to have been deduced by mere inspection.

4.1.4 Power function

In the case of a power function,

$$f(x) = (x - x_\mathrm{A})^p, \tag{4.1.12}$$

where p is a real positive exponent, we perform the integration and find the fractional integral

$$[\,\mathrm{I}^{(\beta)}f\,]_{x_\mathrm{A}}^{\mathrm{RL}}(x) = \frac{\Gamma(p+1)}{\Gamma(p+1+\beta)}\,(x - x_\mathrm{A})^{p+\beta} \tag{4.1.13}$$

for $x > x_\mathrm{A}$. This formula is consistent with (4.1.9) when $p = 0$, yielding the flat function. We see that the fractional integral of a power function is also a power function, which is satisfying.

Using the formula for the fractional derivative, we find that

$$[\,\mathrm{I}^{(\beta)}\mathrm{I}^{(\gamma)}f\,]_{x_\mathrm{A}}^{\mathrm{RL}}(x) = \frac{\Gamma(p+1)}{\Gamma(p+1+\beta+\gamma)}\,(x - x_\mathrm{A})^{p+\beta+\gamma} \tag{4.1.14}$$

for $x > x_\mathrm{A}$, consistent with (4.1.6).

4.1.5 Computation of the Riemann–Liouville fractional integral

To compute the integral in (4.1.5), for $0 < \beta \leq 1$, we subtract out the singularity by writing

$$\int_{x_\mathrm{A}}^{x} \frac{f(\xi)}{(x-\xi)^{1-\beta}}\,\mathrm{d}\xi = \int_{x_\mathrm{A}}^{x} \frac{f(\xi) - f(x)}{(x-\xi)^{1-\beta}}\,\mathrm{d}\xi + f(x)\int_{x_\mathrm{A}}^{x} \frac{\mathrm{d}x}{(x-\xi)^{1-\beta}}. \tag{4.1.15}$$

Now using a Taylor series expansion, we write

$$f(\xi) - f(x) \simeq f'(x)\,(\xi - x) + \cdots, \tag{4.1.16}$$

and find that integrand of the first integral on the right-hand side behaves like $(x-\xi)^\beta$, and is thus sufficiently regular for the integral to be computed by standard numerical methods.

The second integral can be computed analytically, and is given by

$$\int_{x_\mathrm{A}}^{x} \frac{\mathrm{d}\xi}{(x-\xi)^{1-\beta}} = \frac{1}{\beta}\,(x-x_\mathrm{A})^\beta \tag{4.1.17}$$

for $x > x_\mathrm{A}$, which is finite provided that $\beta > 0$.

4.1.6 Numerical evaluation

As a preliminary, we introduce a function named *fnc*, located in directory FRAC of SEMLIB, that receives x and returns the value $f(x)$ and its first few derivatives offered through an integer menu:

```
function  [f,fp,fpp,fppp] = fnc(x,menu)

  if(menu==1)
   dx = x - 2.3;f = 1.0/dx;fp = -1.0/dx^2;fpp = 2.0/dx^3;
   fppp = -6.0/dx^4;
  elseif(menu==2)
   a = 0.48;f = sin(a*x);fp = a*cos(a*x);fpp = -a*a*f;
   fppp = -a*a*a*fp;
  elseif(menu==3)
   f = x^3;fp = 3.0*x^2;fpp = 6.0*x;
   fppp = 6.0;
  elseif(menu==4)
   a = 1.0;f = exp(a*x);fp = a*exp(a*x);fpp = a*a*exp(a*x);
   fppp = a*a*a*exp(a*x);
  elseif(menu==5)
   t = exp(-x^2);f = t;fp = -2*x*t;fpp = 2*(2*x^2-1)*t;
   fppp = 4*x*(3-2*x^2)*t;
  elseif(menu==6)
   t = exp(-x^2);f = x*t;fp = (1-2.0*x^2)*t;
   fpp = 2*x*(2.0*x^2-3.0)*t;
   fppp = 2*(4.0*x^4-12.0*x^2+3.0)*t;
  end

%---
```

```
% done
%---

return
```

The following Matlab function named *rl_int*, located in directory FRAC of SEMLIB, evaluates the Riemann–Liouville fractional integral using this method. The regularized integral on the right-hand side of (4.1.15) is computed by the mid-point rule:

```
function rli = rl_fint (bet,xA,x,nint,menu)

%=======================================
% compute the Riemann--Liouville integral
%
% int_xA_x f(xi) /(x-xi)^bet d xi
%
% nint: integration intervals
%=======================================

%---
% prepare
%---

[f,fp,fpp,fppp] = fnc(x,menu);

Dx = (x-xA)/nint;

%---
% integrate by the mid-point rule
%---

intgrl = 0.0;

for i=1:nint

 xi(i) = xA + (i-0.5)*Dx;
 xr(i) = x-xi(i);
```

```
 [fxi(i),fxip(i),fxipp(i),fxippp(i)]  ...
        = fnc(xi(i),menu);

 intgrl = intgrl + (fxi(i)-f)/xr(i)^(1.0-bet);

end
%---

rli = intgrl*Dx;
rli = rli + f/bet * (x-xA)^bet;
rli = rli/gamma(bet);

%---
% done
%---

return
```

To prevent confusion with the beta function, the parameter β is named `bet` in the code.

4.1.7 Exponential function

The Riemann–Liouville fractional integral of the exponential function, $f(x) = \mathrm{e}^x$, at $x = 1$ is shown in Figure 4.1.1 for $x_\mathrm{A} = 0$.

As β tends to zero, the fractional integral tends to $f(1) = \mathrm{e}$. As β tends to unity, the fractional integral tends to

$$\int_0^1 e^\xi \,\mathrm{d}\xi = \mathrm{e} - 1. \tag{4.1.18}$$

As β tends to 2, the fractional integral tends to

$$\int_0^1 (1-\xi)\, e^\xi \,\mathrm{d}\xi = \mathrm{e} - 2. \tag{4.1.19}$$

Exercise

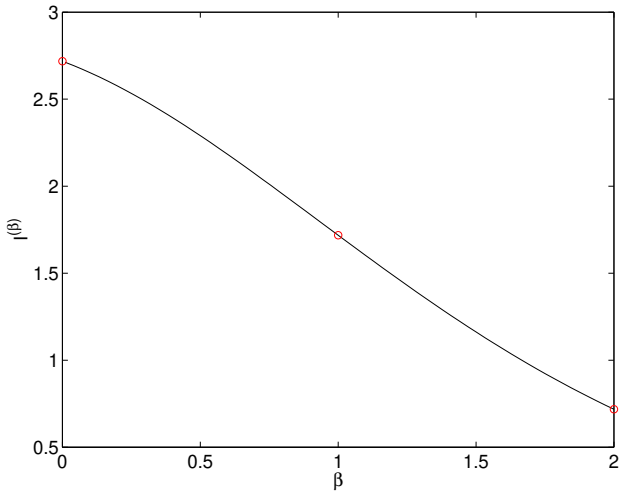


FIGURE 4.1.1 The Riemann–Liouville fractional integral of the exponential function, $f(x) = \mathrm{e}^x$, for $x_\mathrm{A} = 0$ and $x = 1$.

4.1.1 Generate the counterpart of Figure 4.1.1 for the function $f(x) = \exp(2x)$.

4.2 *Right Riemann–Liouville fractional integral*

The right Riemann–Liouville fractional integral of positive order, β, is defined by the formula

$$[\mathrm{I}^{(\beta)} f]^{x_\mathrm{B}}_{\mathrm{RL}}(x) \equiv \frac{1}{\Gamma(\beta)} \int_x^{x_\mathrm{B}} \frac{f(\xi)}{(\xi - x)^{1-\beta}} \, \mathrm{d}\xi, \tag{4.2.1}$$

where x_B is a specified upper limit and x is an evaluation point with $x \leq x_\mathrm{B}$. Since $\beta > 0$, the integrand exhibits either an integrable or a mild singularity, and the fractional integral is well defined. By way of this definition,

$$[\mathrm{I}^{(0)} f]^{x_\mathrm{B}}_{\mathrm{RL}}(x) = f(x), \tag{4.2.2}$$

and

$$[\mathrm{I}^{(1)} f]^{x_\mathrm{B}}_{\mathrm{RL}}(x) = \int_x^{x_\mathrm{B}} f(\xi)\,\mathrm{d}\xi. \tag{4.2.3}$$

as required.

4.2.1 *Power function*

In the case of a power function,

$$f(x) = (x_\mathrm{B} - x)^p, \tag{4.2.4}$$

where p is a real positive exponent, we find that

$$[\mathrm{I}^{(\beta)} f]^{x_\mathrm{B}}_{\mathrm{RL}}(x) = \frac{\Gamma(p+1)}{\Gamma(p+1+\beta)}\,(x_\mathrm{B} - x)^{p+\beta} \tag{4.2.5}$$

for $x \leq x_\mathrm{B}$. We see that the fractional integral of a power function is also a power function. The flat function arises by setting $p = 0$.

4.2.2 *Computation of the left Riemann–Liouville fractional integral*

The integral in (4.2.1) for $0 < \beta < 1$ can be computed by subtracting out the singularity, as discussed in Section 4.1 for the left Riemann–Liouville fractional integral, writing

$$\begin{aligned}\int_x^{x_\mathrm{B}} \frac{f(x)}{(\xi - x)^{1-\beta}}\,\mathrm{d}\xi = \int_x^{x_\mathrm{B}} \frac{f(\xi) - f(x)}{(\xi - x)^{1-\beta}}\,\mathrm{d}\xi \\ + f(x) \int_x^{x_\mathrm{B}} \frac{\mathrm{d}\xi}{(\xi - x)^{1-\beta}}.\end{aligned} \tag{4.2.6}$$

Using a Taylor series expansion, we write

$$f(\xi) - f(x) \simeq f'(x)\,(\xi - x) + \cdots, \tag{4.2.7}$$

and find that integrand of the first integral on the right-hand side behaves like $(\xi - x)^\beta$, and is thus sufficiently regular for the integral to be computed by standard numerical methods.

The second integral can be computed analytically, and is given by

$$\int_x^{x_\mathrm{B}} \frac{\mathrm{d}\xi}{(x_\mathrm{B} - x)^{1-\beta}} = \frac{1}{\beta}\,(x_\mathrm{B} - x)^\beta \tag{4.2.8}$$

for $x < x_\mathrm{B}$, which is finite provided that $\beta > 0$.

4.2.3 Numerical evaluation

The following Matlab function named *rl_intr*, located in directory FRAC of SEMLIB, evaluates the right Riemann–Liouville fractional integral using this method. The regularized integral on the right-hand side of (4.2.6) is computed by the mid-point rule:

```
function rli = rl_fint (bet,xA,x,nint,menu)

%=======================================
% compute the Riemann--Liouville integral
%
% int_xA_x f(xi) /(x-xi)^bet d xi
%
% nint: integration intervals
%=======================================

%---
% prepare
%---

[f,fp,fpp,fppp] = fnc(x,menu);

Dx = (x-xA)/nint;

%---
% integrate by the mid-point rule
%---

intgrl = 0.0;

for i=1:nint

 xi(i) = xA + (i-0.5)*Dx;
 xr(i) = x-xi(i);

 [fxi(i),fxip(i),fxipp(i),fxippp(i)]  ...
        = fnc(xi(i),menu);

 intgrl = intgrl + (fxi(i)-f)/xr(i)^(1.0-bet);
```

```
end
%---

rli = intgrl*Dx;
rli = rli + f/bet * (x-xA)^bet;
rli = rli/gamma(bet);

%---
% done
%---

return
```

4.2.4 Exponential function

As an example, the right Riemann–Liouville fractional integral of the exponential function, $f(x) = \mathrm{e}^x$, evaluated at $x = 1$ is shown in Figure 4.2.1 for $x_\mathrm{B} = 2$.

As β tends to zero, the fractional integral tends to $f(0) = \mathrm{e}$. As β tends to unity, the fractional integral tends to $\int_1^2 e^\xi \, \mathrm{d}\xi = \mathrm{e}\,(\mathrm{e} - 1)$. As β tends to 2, the fractional integral tends to $\int_1^2 (\xi - 1)\, e^\xi \, \mathrm{d}\xi = \mathrm{e}$.

4.2.5 Centered Riemann–Liouville fractional integral

The Riemann–Liouville fractional integral can be combined with the right Riemann–Liouville fractional integral into the centered fractional integral

$$ {}_{\mathrm{RL}}[\,\mathrm{I}^{(\beta)} f\,]_{x_\mathrm{A}}^{x_\mathrm{B}}(x) \equiv \frac{1}{\Gamma(\beta)} \int_{x_\mathrm{A}}^{x_\mathrm{B}} \frac{f(\xi)}{|\xi - x|^{1-\beta}} \, \mathrm{d}\xi \tag{4.2.9} $$

for $x_\mathrm{A} \le x \le x_\mathrm{B}$, where x_A and x_B are two specified limits. By way of this definition,

$$ {}_{\mathrm{RL}}[\,\mathrm{I}^{(0)} f\,]_{x_\mathrm{A}}^{x_\mathrm{B}}(x) = 2\, f(x) \tag{4.2.10} $$

and

$$ {}_{\mathrm{RL}}[\,\mathrm{I}^{(1)} f\,]_{x_\mathrm{A}}^{x_\mathrm{B}} = \int_{x_\mathrm{A}}^{x_\mathrm{B}} f(\xi) \, \mathrm{d}\xi. \tag{4.2.11} $$

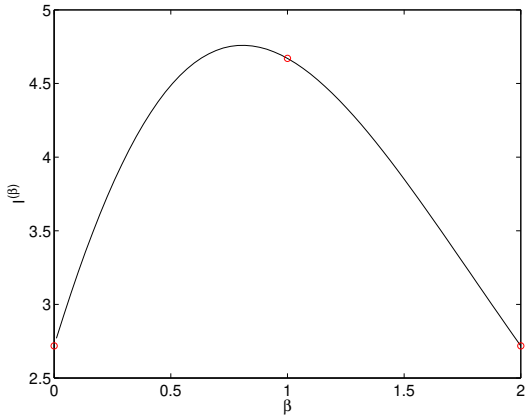


FIGURE 4.2.1 The right Riemann–Liouville fractional integral of the exponential function, $f(x) = \mathrm{e}^x$, for $x = 1$ and $x_\mathrm{B} = 2$.

Note that ${}_{\mathrm{RL}}[\,\mathrm{I}^{(1)} f\,]_{x_\mathrm{A}}^{x_\mathrm{B}}$ is independent of x.

The integral in (4.2.9) can be computed by subtracting out the singularity, as discussed for the right or left Riemann–Liouville fractional integral.

Exercise

4.2.1 Generate the counterpart of Figure 4.2.1 for the function $f(x) = \exp(2x)$.

4.3 Riemann–Liouville fractional derivative

A fractional derivative of positive order, α, can be defined in terms of whole derivatives of the Riemann–Liouville fractional integral of order $0 < \beta \leq 1$ introduced in (4.1.5), denoted as $[\,\mathrm{I}^{(\beta)} f\,]_{x_\mathrm{A}}^{\mathrm{RL}}(x)$, where x_A is a specified lower integration limit and x is a specified evaluation point.

4.3.1 *Case* $0 \leq \alpha < 1$

When the order of the fractional derivative lies in the range $0 \leq \alpha < 1$,

the Riemann–Liouville fractional derivative is defined as

$$[\,\mathrm{D}^{(\alpha)} f\,]^{\mathrm{RL}}_{x_\mathrm{A}}(x) \equiv \frac{\mathrm{d}}{\mathrm{d}x}\,[\,\mathrm{I}^{(1-\alpha)} f\,]^{\mathrm{RL}}_{x_\mathrm{A}}(x), \tag{4.3.1}$$

involving a fractional integral of order $\beta = 1-\alpha$. Making substitutions, we find that

$$[\,\mathrm{D}^{(\alpha)} f\,]^{\mathrm{RL}}_{x_\mathrm{A}}(x) = \frac{1}{\Gamma(1-\alpha)} \frac{\mathrm{d}}{\mathrm{d}x} \int_{x_\mathrm{A}}^{x} \frac{f(\xi)}{(x-\xi)^{\alpha}}\,\mathrm{d}\xi \tag{4.3.2}$$

for $x \geq x_\mathrm{A}$. Note that x appears both in the upper limit and in kernel of the integral on the right-hand side.

4.3.2 *Case* $1 \leq \alpha < 2$

When the order of the fractional derivative lies in the range $1 \leq \alpha < 2$, a fractional derivative can be defined as

$$[\,\mathrm{D}^{(\alpha)} f\,]^{\mathrm{RL}}_{x_\mathrm{A}}(x) \equiv \frac{\mathrm{d}^2}{\mathrm{d}x^2}\,[\,\mathrm{I}^{(2-\alpha)} f]^{\mathrm{RL}}_{x_\mathrm{A}}(x), \tag{4.3.3}$$

involving a fractional integral of order $\beta = 2-\alpha$. Making substitutions, we find that

$$[\,\mathrm{D}^{(\alpha)} f\,]^{\mathrm{RL}}_{x_\mathrm{A}} f(x) = \frac{1}{\Gamma(2-\alpha)} \frac{\mathrm{d}^2}{\mathrm{d}x^2} \int_{x_\mathrm{A}}^{x} \frac{f(\xi)}{(x-\xi)^{\alpha-1}}\,\mathrm{d}\xi \tag{4.3.4}$$

for $x \geq x_\mathrm{A}$. Note that x appears both in the upper limit and in kernel of the integral on the right-hand side.

4.3.3 *Case* $n-1 \leq \alpha < n$

More generally, when the order of the fractional derivative lies in the range $n-1 \leq \alpha < n$, where $n = 1, 2, \ldots$ is an integer,

$$[\,\mathrm{D}^{(\alpha)} f\,]^{\mathrm{RL}}_{x_\mathrm{A}}(x) \equiv \frac{\mathrm{d}^n}{\mathrm{d}x^n}\,[\,\mathrm{I}^{(n-\alpha)} f]^{\mathrm{RL}}_{x_\mathrm{A}}(x), \tag{4.3.5}$$

involving a fractional integral order $\beta = n-\alpha$. Making substitutions, we find that

$$[\,\mathrm{D}^{(\alpha)} f\,]^{\mathrm{RL}}_{x_\mathrm{A}}(x) = \frac{1}{\Gamma(n-\alpha)} \frac{\mathrm{d}^n}{\mathrm{d}x^n} \int_{x_\mathrm{A}}^{x} \frac{f(\xi)}{(x-\xi)^{\alpha+1-n}}\,\mathrm{d}\xi. \tag{4.3.6}$$

Note that the exponent in the denominator of the integrand satisfies the inequality

$$0 \leq \alpha + 1 - n < 1, \tag{4.3.7}$$

which ensures the integrability of the integral on the right-hand side.

4.3.4 Restatement

We may define $w = x - \xi$ and write $\xi = x - w$ to obtain

$$[\,\mathrm{D}^{(\alpha)} f\,]^{\mathrm{RL}}_{x_\mathrm{A}}(x) \equiv \frac{1}{\Gamma(n-\alpha)} \frac{\mathrm{d}^n}{\mathrm{d}x^n} \int_0^{x-x_\mathrm{A}} \frac{f(x-w)}{w^{\alpha+1-n}}\, \mathrm{d}w, \tag{4.3.8}$$

where the upper integration limit is zero or positive, $x \geq x_\mathrm{A}$.

We may define $v = \xi - x$, and write $\xi = x + v$ to obtain

$$[\,\mathrm{D}^{(\alpha)} f\,]^{\mathrm{RL}}_{x_\mathrm{A}}(x) \equiv \frac{1}{\Gamma(n-\alpha)} \frac{\mathrm{d}^n}{\mathrm{d}x^n} \int_{x_\mathrm{A}-x}^0 \frac{f(x+v)}{|v|^{\alpha+1-n}}\, \mathrm{d}v, \tag{4.3.9}$$

where the lower integration limit is zero or negative, $x_A \leq x$.

4.3.5 Whole differentiation

When the fractional order, α, is zero or a positive integer, m, the Riemann–Liouville fractional derivative reproduces the corresponding ordinary derivative, that is,

$$[\,\mathrm{D}^{(m)} f\,]^{\mathrm{RL}}_{x_\mathrm{A}}(x) = f^{(m)}(x) \tag{4.3.10}$$

for $m = 0, 1, \ldots$, subject to the convention that $f^{(0)}(x) = f(x)$.

4.3.6 Flat function

Using (4.1.9) for a flat function, $f(x) = c$, where c is a constant, we find that

$$[\,\mathrm{D}^{(\alpha)} c\,]^{\mathrm{RL}}_{x_\mathrm{A}}(x) = \frac{c}{(1-\alpha)\,\Gamma(1-\alpha)} \frac{\mathrm{d}}{\mathrm{d}x} (x - x_\mathrm{A})^{1-\alpha} \tag{4.3.11}$$

for $0 \leq \alpha < 1$. Performing the differentiation, we obtain

$$[\,\mathrm{D}^{(\alpha)} c\,]^{\mathrm{RL}}_{x_\mathrm{A}}(x) = \frac{c}{\Gamma(1-\alpha)} (x - x_\mathrm{A})^{-\alpha} \tag{4.3.12}$$

for $x \geq x_{\mathrm{A}}$. We observe that the fractional derivative of a flat function is not necessarily zero. However, since $\Gamma(0)$ is infinite, the first derivative corresponding to $\alpha = 1$ is reproduced to be zero.

Using (4.1.9) once more, we obtain

$$[\,\mathrm{D}^{(\alpha)} c\,]_{x_{\mathrm{A}}}^{\mathrm{RL}}(x) = \frac{c}{(2-\alpha)\,\Gamma(2-\alpha)}\,\frac{\mathrm{d}^2}{\mathrm{d}x^2}\,(x - x_{\mathrm{A}})^{2-\alpha} \tag{4.3.13}$$

for $1 \leq \alpha < 2$. Carrying out the differentiation, we find that

$$[\,\mathrm{D}^{(\alpha)} c\,]_{x_{\mathrm{A}}}^{\mathrm{RL}}(x) = c\,\frac{1-\alpha}{\Gamma(2-\alpha)}\,(x - x_{\mathrm{A}})^{-\alpha}. \tag{4.3.14}$$

Using the properties of the Gamma function, we find that this formula is identical to that shown in (4.3.12). Since $\Gamma(0)$ is infinite, the second derivative corresponding to $\alpha = 2$ is reproduced to be zero.

Working in a similar fashion, we find that expression (4.3.12) applies for any value of α.

4.3.7 *Power function*

In the case of a power function,

$$f(x) = (x - x_{\mathrm{A}})^p, \tag{4.3.15}$$

where p is a real positive exponent, we find that

$$[\,\mathrm{D}^{(\alpha)} f\,]_{x_{\mathrm{A}}}^{\mathrm{RL}}(x) = \frac{\Gamma(p+1)}{\Gamma(p+1-\alpha)}\,(x - x_{\mathrm{A}})^{p-\alpha} \tag{4.3.16}$$

for $\alpha \geq 0$ and $x > x_{\mathrm{A}}$. The fractional derivative of the flat function arises by setting $p = 0$.

4.3.8 *Computation of the Riemann–Liouville fractional derivative*

The following Matlab function named *rl_fder*, located in directory FRAC of SEMLIB, evaluates the Riemann–Liouville fractional derivative by numerical differentiation of the fractional integral using centered differences:

```
function rlip = rl_fder (alpha,xA,x,eps,nint,menu)

%====================================
% computation of the Riemann--Liouville
% fractional derivative
%
% eps: numerical differentiation step
%====================================

if(alpha<0)

 bet = -alpha;
 rlip = rl_int (bet,xA,x,nint,menu);

elseif(alpha >= 0 & alpha < 1)

 bet = 1.0-alpha;
 rli_2 = rl_int (bet,xA,x+eps,nint,menu);
 rli_1 = rl_int (bet,xA,x-eps,nint,menu);
 rlip = (rli_2-rli_1)/(2.0*eps);

elseif(alpha >= 1 & alpha < 2)

 bet = 2.0-alpha;
 rli_0 = rl_int (bet,xA,x,menu);
 rli_1 = rl_int (bet,xA,x-eps,nint,menu);
 rli_2 = rl_int (bet,xA,x+eps,nint,menu);
 rlip = (rli_2 -2*rli_0 +rli_1)/eps^2;

elseif(alpha >= 2 & alpha < 3)

 bet = 3.0-alpha;
 rli_A = rl_fint (bet,xA,x-2*eps,nint,menu);
 rli_1 = rl_fint (bet,xA,x-eps,nint,menu);
 rli_2 = rl_fint (bet,xA,x+eps,nint,menu);
 rli_B = rl_fint (bet,xA,x+2*eps,nint,menu);
 rlip = (-rli_A + 2*rli_1 - 2*rli_2 + rli_B)/(2*eps^3);

else
```

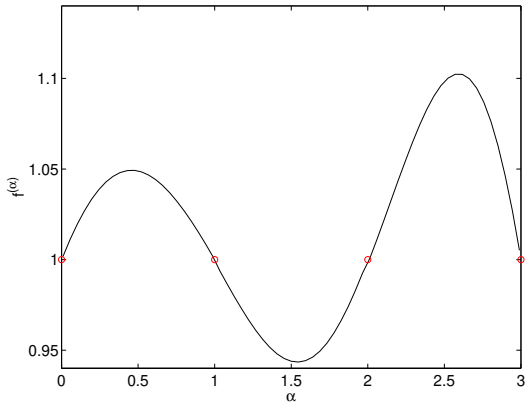


FIGURE 4.3.1 The Riemann–Liouville fractional derivative of the exponential function, $f(x) = \mathrm{e}^x$, for $x_\mathrm{A} = 0$, evaluated at $x = 1$.

```
 disp('rl_fder: this choice of alpha not yet implemented')

end

%---
% done
%---

return
```

The function makes a call to the function *rl_fint* listed in Section 4.1 to evaluate the fractional Riemann–Liouville integral.

4.3.9 Exponential function

As an example, the Riemann–Liouville fractional derivative of the exponential function, $f(x) = \mathrm{e}^x$, at $x = 1$ is shown in Figure 4.3.1 for $x_\mathrm{A} = 0$.

At $\alpha = 0, 1, 2$, the fractional derivative is equal to e. However, for intermediate values the fractional derivative is different than e.

Exercise

4.3.1 Generate the counterpart of Figure 4.3.1 for the Gaussian distribution, $f(x) = \exp(-x^2)$.

4.4 Right Riemann–Liouville fractional derivative

The right Riemann–Liouville fractional derivative of order α of a suitable function, $f(x)$, is defined with reference to the right Riemann–Liouville fractional integral defined in Section 4.2 as

$$\left[\mathrm{D}^{(\alpha)} f\right]_{\mathrm{RL}}^{x_{\mathrm{B}}}(x) \equiv (-1)^n \frac{1}{\Gamma(n-\alpha)} \frac{\mathrm{d}^n}{\mathrm{d}x^n} \int_x^{x_{\mathrm{B}}} \frac{f(x)}{(\xi - x)^{1-n+\alpha}} \, \mathrm{d}\xi \tag{4.4.1}$$

for $x \leq x_{\mathrm{B}}$, where x_{B} is a specified upper limit and n is an integer such that

$$n - 1 \leq \alpha < n. \tag{4.4.2}$$

Note the power $(-1)^n$ after the definition sign, $\equiv$.

We may define $v = \xi - x$, we obtain

$$\left[\mathrm{D}^{(\alpha)} f\right]_{\mathrm{RL}}^{x_{\mathrm{B}}}(x) = (-1)^n \frac{1}{\Gamma(n-\alpha)} \frac{\mathrm{d}^n}{\mathrm{d}x^n} \int_0^{x_{\mathrm{B}}-x_0} \frac{f(x+v)}{v^{1-n+\alpha}} \, \mathrm{d}v, \tag{4.4.3}$$

where v is the signed distance from the evaluation point.

4.4.1 Integer fractional orders

The right Riemann–Liouville fractional derivative reproduces the corresponding ordinary derivative when α is zero or an even integer, and the negative of the corresponding ordinary derivative when α is an odd integer,

$$\left[\,\mathrm{D}^{(m)} f\,\right]_{\mathrm{RL}}^{x_{\mathrm{B}}}(x) = (-1)^m \, f^{(m)}(x) \tag{4.4.4}$$

for $m = 0, 1, \ldots$, subject to the convention that $f^{(0)}(x) = f(x)$.

4.4.2 Power function

In the case of a power function,

$$f(x) = (x_{\mathrm{B}} - x)^p, \tag{4.4.5}$$

where p is a real positive exponent, we perform the integration and find that

$$[\,\mathrm{D}^{(\alpha)} f\,]_{\mathrm{RL}}^{x_{\mathrm{B}}}(x) = \frac{\Gamma(p+1)}{\Gamma(p+1-\alpha)}\,(x_{\mathrm{B}} - x)^{p-\alpha} \tag{4.4.6}$$

for $x < x_{\mathrm{B}}$. The fractional derivative of the flat function arises by setting $p = 0$.

4.4.3 Computation of the right Riemann–Liouville fractional derivative

The following Matlab function named *rl_fderr*, located in directory FRAC of SEMLIB, evaluates the right Riemann–Liouville fractional derivative by numerical differentiation of the fractional integral using centered differences with a sufficiently small numerical differentiation step:

```
function rlip = rl_fderr (alpha,xB,x,eps,nint,menu)

%====================================
% computation of the right Riemann--Liouville
% fractional derivative
%
% eps: numerical differentiation step
%====================================

if(alpha<0)

 bet = -alpha;
 rlip = rl_fintr (bet,xB,x,nint,menu);

elseif(alpha >= 0 & alpha < 1)

 bet = 1.0-alpha;
```

```
 rli_2 = rl_fintr (bet,xB,x+eps,nint,menu);
 rli_1 = rl_fintr (bet,xB,x-eps,nint,menu);
 rlip = (rli_2-rli_1)/(2.0*eps);
 rlip = - rlip;

elseif(alpha >= 1 & alpha < 2)

 bet = 2.0-alpha;
 rli_0 = rl_fintr (bet,xB,x,nint,menu);
 rli_1 = rl_fintr (bet,xB,x-eps,nint,menu);
 rli_2 = rl_fintr (bet,xB,x+eps,nint,menu);
 rlip = (rli_2 -2*rli_0 +rli_1)/eps^2;

elseif(alpha >= 2 & alpha < 3)

 bet = 3.0-alpha;
 rli_A = rl_fintr (bet,xB,x-2*eps,nint,menu);
 rli_1 = rl_fintr (bet,xB,x-eps,nint,menu);
 rli_2 = rl_fintr (bet,xB,x+eps,nint,menu);
 rli_B = rl_fintr (bet,xB,x+2*eps,nint,menu);
 rlip = (-rli_A + 2*rli_1 - 2*rli_2 + rli_B)/(2*eps^3);
 rlip = - rlip;

else

 disp('rl_fderr: this choice of alpha not yet implemented')

end

%---
% done
%---

return
```

The function makes a call to the function *rl_fintr* listed in Section 4.2 to evaluate the right fractional Riemann–Liouville integral.

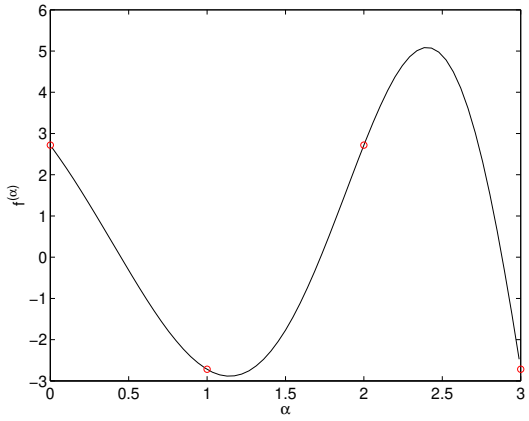


FIGURE 4.4.1 The right Riemann–Liouville fractional derivative of the exponential function, $f(x) = \mathrm{e}^x$, evaluated at $x_0 = 1$ for $x_\mathrm{B} = 2$.

4.4.4 Exponential function

The right Riemann–Liouville fractional derivative of the exponential function, $f(x) = \mathrm{e}^x$, evaluated at $x_0 = 1$ is shown in Figure 4.4.1 for $x_\mathrm{B} = 2$.

For $\alpha = 0, 2$, the fractional derivative is equal to e. For intermediate values of α, including the value $\alpha = 1$, the fractional derivative is different than e. For the exponential function, $f(x) = \mathrm{e}^x$, the fractional derivative at $\alpha = 1$ is equal to $-\mathrm{e}$.

Exercise

4.4.1 Generate the counterpart of Figure 4.4.1 for the first moment of the Gaussian distribution, $f(x) = x \exp(-x^2)$.

4.5 Centered Riemann–Liouville fractional derivative and the fractional Laplacian

Half the sum of the left and right Riemann–Liouville fractional derivatives discussed in Sections 4.2 and 4.4 is defined as the centered Riemann–Liouville fractional derivative,

$$ {}_{\rm RL}\big[\,{\rm D}^{(\alpha)} f\,\big]^{x_{\rm B}}_{x_{\rm A}}(x) \equiv \tfrac{1}{2}\Big(\,\big[\,{\rm D}^{(\alpha)} f\,\big]^{\rm RL}_{x_{\rm A}} + \big[\,{\rm D}^{(\alpha)} f\,\big]^{x_{\rm B}}_{\rm RL}\Big)(x) \tag{4.5.1} $$

for $x_A \leq x \leq x_{\rm B}$, where $x_{\rm A}$ is a specified lower limit and $x_{\rm B}$ is a specified upper limit. We are particularly interested in values of α in the range $[0, 2)$.

For $0 \leq \alpha < 1$, we obtain

$$ {}_{\rm RL}\big[\,{\rm D}^{(\alpha)} f\,\big]^{x_{\rm B}}_{x_{\rm A}}(x) = -\frac{1}{2\,\Gamma(1-\alpha)}\frac{\rm d}{{\rm d}x}\Big(\int_{x_{\rm A}-x}^{x_{\rm B}-x} \frac{v}{|v|}\,\frac{f(x+v)}{|v|^{\alpha}}\,{\rm d}v\Big) \tag{4.5.2} $$

The fraction $v/|v|$ inside the integral is equal to -1 when $v < 0$ or 1 when $v > 0$. Note that the lower integration limit is negative, whereas the upper integration limit is positive.

For $1 \leq \alpha < 2$, we obtain

$$ {}_{\rm RL}\big[\,{\rm D}^{(\alpha)} f\,\big]^{x_{\rm B}}_{x_{\rm A}}(x) = \frac{1}{2\,\Gamma(2-\alpha)}\frac{{\rm d}^2}{{\rm d}x^2}\Big(\int_{x_{\rm A}-x}^{x_{\rm B}-x} \frac{f(x+v)}{|v|^{\alpha-1}}\,{\rm d}v\Big). \tag{4.5.3} $$

Note that the exponent of the denominator inside the integral varies in the integrable range $0 \leq \alpha - 1 < 1$.

4.5.1 Integer fractional orders

As α tends to 0, the left and right Riemann–Liouville derivatives both tend to the function value, $f(x)$. The centered derivative also tends to $f(x)$ as α tends to 0,

$$ {}_{\rm RL}\big[\,{\rm D}^{(0)} f\,\big]^{x_{\rm B}}_{x_{\rm A}}(x) = f(x) \tag{4.5.4} $$

for $x_{\mathrm{A}} \leq x \leq x_{\mathrm{B}}$.

As α tends to 2, the left and right Riemann-Liouville derivatives both tend to the second ordinary derivative, $f''(x)$. The centered derivative also tends to $f''(x)$ as α tends to 2,

$$ {}_{\mathrm{RL}}\big[\,\mathrm{D}^{(2)} f\,\big]^{x_{\mathrm{B}}}_{x_{\mathrm{A}}}(x) = f''(x) \tag{4.5.5} $$

for $x_{\mathrm{A}} \leq x \leq x_{\mathrm{B}}$.

Since $\big[\,\mathrm{D}^{(\alpha)} f\,\big]^{\mathrm{RL}}_{x_{\mathrm{A}}}(x)$ tends to $f'(x)$ whereas $\big[\,\mathrm{D}^{(\alpha)} f\,\big]^{x_{\mathrm{B}}}_{\mathrm{RL}}(x)$ tends to $-f'(x)$ as α tends to 1, the centered derivative, ${}_{\mathrm{RL}}\big[\,\mathrm{D}^{(\alpha)} f\,\big]^{x_{\mathrm{B}}}_{x_{\mathrm{A}}}(x)$, tends to zero as α tends to 1,

$$ {}_{\mathrm{RL}}\big[\,\mathrm{D}^{(1)} f\,\big]^{x_{\mathrm{B}}}_{x_{\mathrm{A}}}(x) = 0 \tag{4.5.6} $$

for $x_{\mathrm{A}} \leq x \leq x_{\mathrm{B}}$. However, the ratio

$$ \frac{1}{\alpha - 1}\, {}_{\mathrm{RL}}\big[\,\mathrm{D}^{(\alpha)} f\,\big]^{x_{\mathrm{B}}}_{x_{\mathrm{A}}}(x) \tag{4.5.7} $$

tends to a finite limit as α tends to 1.

4.5.2 Fractional Laplacian

In the limit as $x_{\mathrm{A}} \to -\infty$ and $x_{\mathrm{B}} \to \infty$, the centered Riemann–Liouville fractional derivative reduces to the fractional Laplacian (FL), also known as the Riesz fractional derivative, according to the formula

$$ \big[\,\mathrm{L}^{(\alpha)} f\,\big](x) = -\frac{1}{\cos\big(\frac{\alpha\pi}{2}\big)}\, {}_{\mathrm{RL}}\big[\,\mathrm{D}^{(\alpha)} f\,\big]^{+\infty}_{-\infty}(x) \tag{4.5.8} $$

for $0 \leq \alpha < 2$. A necessary condition is that the function $f(x)$ decays rapidly enough for the pertinent integrals to be defined. Note the minus sign on the right-hand side of (4.5.8).

For $0 \leq \alpha < 1$, we obtain

$$ \big[\,\mathrm{L}^{(\alpha)} f\,\big](x) = -\frac{1}{\cos\big(\frac{\alpha\pi}{2}\big)}\, \frac{1}{\Gamma(1-\alpha)}\, \frac{\mathrm{d}\mathcal{J}(x)}{\mathrm{d}x}, \tag{4.5.9} $$

where

$$\mathcal{J}(x) \equiv \int_{-\infty}^{\infty} \frac{v}{|v|^{\alpha+1}}\, f(x+v)\, \mathrm{d}v. \tag{4.5.10}$$

The integral on the right-hand side converges in the entire range considered, $0 \leq \alpha < 1$.

For $1 \leq \alpha < 2$, we obtain

$$[\,\mathrm{L}^{(\alpha)} f\,](x) = -\frac{1}{\cos(\frac{\alpha\pi}{2})}\, \frac{1}{\Gamma(2-\alpha)} \frac{\mathrm{d}^2 \mathcal{J}(x)}{\mathrm{d}x^2}, \tag{4.5.11}$$

where

$$\mathcal{J}(x) \equiv \int_{-\infty}^{\infty} \frac{1}{|v|^{\alpha-1}}\, f(x+v)\, \mathrm{d}v. \tag{4.5.12}$$

Note that the exponent of the denominator inside the integral varies in the integrable range $0 \leq \alpha - 1 < 1$.

More will be said about the fractional Laplacian in Sections 4.10 and 4.11 where alternative definitions will be given and pertinent physical interpretations will be discussed.

4.5.3 Computation of the fractional Laplacian

The computation of the fractional Laplacian in terms of the Riemann–Liouville derivatives is implemented in the following Matlab code named *rl_flap*, located in directory FRAC of SEMLIB:

```
%==================================
% Evaluate the fractional laplacian
% in terms of the left and right
% Riemann--Liouville
% fractional derivatives
%==================================

xA = -2.0;  % arbitrary
xB =  2.0;  % arbitrary
```

```
alpha = 0.678; % arbitrary

eps = 0.01;   % differentiation step

nint = 2*32;

menu = 5;  % Gaussian distribution

%---
% evaluate
%---

x = 0.0;

rlder_left = rl_fder (alpha,xA,x,eps,nint,menu);
rlder_rite = rl_fderr(alpha,xB,x,eps,nint,menu);

rlflap =  rlder_left + rlder_rite;
rlflap = -rlflap/(2.0*cos(alpha*pi/2));

[x, rlflap]
```

4.5.4 Gaussian function

The fractional Laplacian of the Gaussian distribution, $f(x) = \mathrm{e}^{-x^2}$, is shown in Figure 4.5.1 for $\alpha = 1.5$. The solid line represents the whole distribution obtained by evaluating a principal-value integral, as discussed in Section 4.10. The circular symbols at $x = 0$ and 1 represent results obtained in terms of the centered Riemann–Liouville derivative.

Exercise

4.5.1 Generate the counterpart of Figure 4.5.1 for the first moment of the Gaussian distribution, $f(x) = x \exp(-x^2)$.

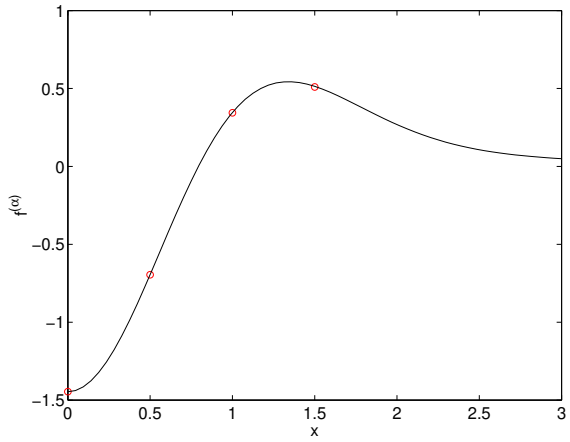


FIGURE 4.5.1 The fractional Laplacian of the Gaussian distribution, $f(x) = \mathrm{e}^{-x^2}$, for $\alpha = 1.5$. The circular symbols represents results obtained in terms of the centered Riemann–Liouville derivative.

4.6 Backward Grunwald–Letnikov derivative

The backward Grünwald–Letnikov fractional derivative discussed in this section, and the corresponding forward fractional derivative discussed in Section 4.7 are defined in terms of infinite sums associated with backward or forward finite-difference approximations. In the limit as the differentiation step tends to zero, these derivatives reduce to, and are mere restatements of the left and right Riemann–Liouville derivatives discussed in Sections 4.3 and 4.4.

4.6.1 Backward finite differences

The first few ordinary derivatives of a function, $f(x)$, at a point, x_i, can be approximated with the following backward finite-difference formulas in terms of function values at an array of evenly-spaced neighboring

points, $x_{i-p} = x - ph$,

$$\begin{bmatrix} f_i^{(0)} \\ h f_i^{(1)} \\ h^2 f_i^{(2)} \\ h^3 f_i^{(3)} \\ h^4 f_i^{(4)} \\ h^5 f_i^{(5)} \end{bmatrix} \simeq \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & -1 & 3 & -3 & 1 \\ 0 & 1 & -4 & 6 & -4 & 1 \\ -1 & 5 & -10 & 10 & -5 & 1 \end{bmatrix} \cdot \begin{bmatrix} f_{i-5} \\ f_{i-4} \\ f_{i-3} \\ f_{i-2} \\ f_{i-1} \\ f_i \end{bmatrix}, \quad (4.6.1)$$

where h is a specified interval and we have denoted $f_j = f(x_j)$. The associated numerical error is of order h for any derivative order.

Note the last column of *ones* in the matrix on the right-hand side of (4.6.1) multiplying f_i, the alternating signs along each row, and the constant signs down each column.

As an example,

$$f_i^{(4)} \simeq \frac{1}{h^4} \big(f_{i-4} - 4f_{i-3} + 6f_{i-2} - 4f_{i-1} + f_i \big), \quad (4.6.2)$$

with an error of order h.

4.6.2 General backward difference formula

The underlying finite-difference formula for the mth ordinary derivative is

$$f_i^{(m)} \simeq \frac{1}{h^m} \sum_{k=0}^{m} (-1)^k \binom{m}{k} f_{i-k}, \quad (4.6.3)$$

where the first set of tall parentheses on the right-hand side denote the binomial coefficient. Equation (4.6.3) for the mth whole derivative can be restated as

$$f_i^{(m)}(x) \simeq \frac{1}{h^m} \sum_{k=0}^{m} (-1)^k \binom{m}{k} f(x - kh), \quad (4.6.4)$$

where m is a positive integer and x is an arbitrary evaluation point.

4.6.3 Fractional derivative

To obtain a corresponding fractional derivative of order α, we replace the integer m in (4.6.3) with the fractional order, α, and modify the upper summation limit to obtain

$$[\mathrm{D}^{(\alpha)} f\,]^{\mathrm{GL}}_{x_\mathrm{A}}(x) \simeq \frac{1}{h^\alpha} \sum_{k=0}^{k_{\max}} (-1)^k \binom{\alpha}{k} f(x - kh), \tag{4.6.5}$$

where x is an evaluation point,

$$k_{\max} = \Big[\, \frac{x - x_\mathrm{A}}{h} \,\Big] \tag{4.6.6}$$

is a positive integer, the square brackets indicate the integral part, and x_A is a specified lower limit associated with the Grünwald-Letnikov derivative. We have introduced the fractional binomial coefficient

$$\binom{\alpha}{k} \equiv \frac{(\alpha - k + 1) \cdots (\alpha - 1)\, \alpha}{1 \cdot 2 \cdots k} = \prod_{p=1}^{k} \frac{\alpha - p + 1}{p} \tag{4.6.7}$$

for $k \geq 1$, subject to the convention that

$$\binom{\alpha}{0} = 1. \tag{4.6.8}$$

Using the properties of the Gamma function, we find that

$$\binom{\alpha}{k} = (-1)^k \frac{\Gamma(k - \alpha)}{\Gamma(-\alpha)\, \Gamma(k + 1)} \tag{4.6.9}$$

for any real and positive α and integer k. Given these definitions, negative values for α can be employed, providing us with fractional integrals.

The Grünwald–Letnikov fractional derivative arises from (4.6.5) in the limit as $h \to 0$, and correspondingly $k_{\max} \to \infty$, while x_A is held constant.

4.6.4 Differentiation coefficients

It is expedient to introduce the differentiation coefficients

$$\omega_{\alpha,k} \equiv (-1)^k \binom{\alpha}{k} = \frac{1}{\Gamma(-\alpha)} \frac{\Gamma(k-\alpha)}{\Gamma(k+1)}, \tag{4.6.10}$$

which can be expressed as

$$\omega_{\alpha,k} = (-1)^k \frac{\Gamma(\alpha+1)}{\Gamma(\alpha-k+1)\Gamma(k+1)}. \tag{4.6.11}$$

The definition (4.6.5) takes the form

$$[\mathrm{D}^{(\alpha)} f]^{\mathrm{GL}}_{x_{\mathrm{A}}}(x) \simeq \frac{1}{h^\alpha} \sum_{k=0}^{k_{\max}} \omega_{\alpha,k}\, f(x-kh), \tag{4.6.12}$$

where $\omega_{\alpha,0} = 1$.

Referring to the definition (4.6.7), we find that

$$\binom{\alpha}{k} = \frac{\alpha-k+1}{k} \binom{\alpha}{k-1}, \tag{4.6.13}$$

which shows that the differentiation coefficients satisfy the recursion formula

$$\omega_{\alpha,k} = \frac{k-\alpha-1}{k}\, \omega_{\alpha,k-1}, \tag{4.6.14}$$

starting with $\omega_{\alpha,0} = 1$. It can be shown that

$$\sum_{k=0}^{\infty} \omega_{\alpha,k} = 0. \tag{4.6.15}$$

In fact, these coefficient arise from the generating fractional power

$$(1-\xi)^\alpha = \sum_{k=0}^{\infty} \omega_{\alpha,k}\, \xi^k. \tag{4.6.16}$$

for $\xi < 1$.

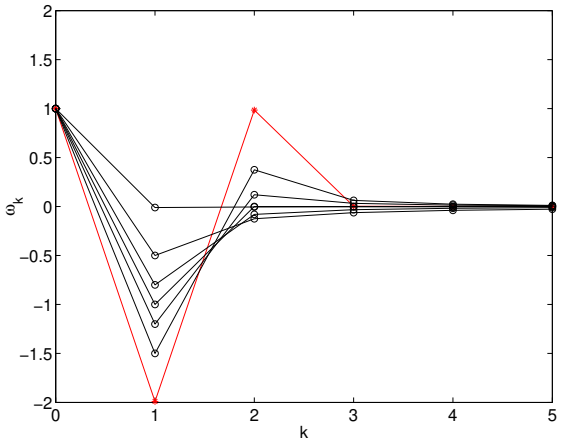


FIGURE 4.6.1 Differentiation coefficients of the Grünwald–Letnikov fractional derivative for $\alpha = 1.99$ (asterisks connected with a red line). 1.5, 1.2, 1.0, 0.8, 0.5, and 0.01.

A graph of the differentiation coefficients is shown in Figure 4.6.1 for several values of α.

4.6.5 *Numerical evaluation*

The computation of of the Grünwald–Letnikov fractional derivative is implemented in the following Matlab code named *gl_fder*, located in directory FRAC of SEMLIB:

```
function glp = gl_fder(alpha,xA,x,kmax,menu)

%==============================
% evaluate the Grunwald--Letnikov
% fractional derivative
%==============================

  h = (x-xA)/kmax;

  ishift = 1;

%---
```

```
% compute omegas
%---

  omg(ishift) = 1;

  for k=1:kmax
    omg(k+ishift) = omg(k-1+ishift)*(k-alpha-1)/k;
  end

%---
% sum
%---

  glp = 0.0;

  for k=0:kmax
   [f,fp,fpp,fppp] = fnc(x-k*h,menu);
   glp = glp + omg(k+ishift)*f;
  end

  glp = glp/h^alpha;

%---
% done
%---

return
```

An integer parameter `ishift` was introduced to ensure that the array `omg` is assigned positive and non-zero indices.

4.6.6 Equivalence to the Riemann–Liouville derivative

In fact, when the function $f(x)$ is sufficiently regular, the Grünwald–Letnikov fractional derivative reduces to the Riemann–Liouville fractional derivative, as shown in Figure 4.6.2 for the exponential function. Unfortunately, the rate of convergence with respect to the number of summed terms, $k_{\max}$, is slow, as shown in Figure 4.6.2 for up to 256 terms. By contrast, the results shown in Figure 4.3.1 are visually indistinguishable for numerical integration divisions 16, 32, and 64.

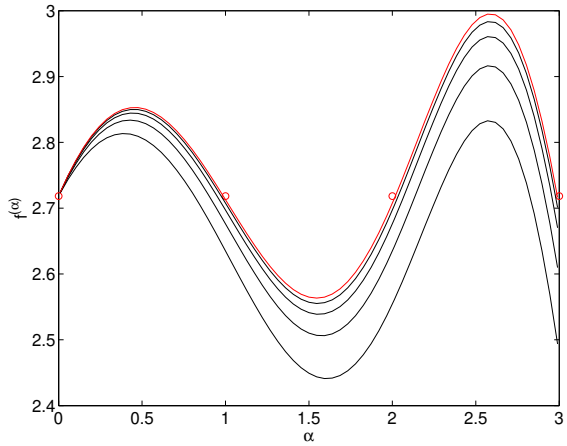


FIGURE 4.6.2 The Grünwald–Letnikov fractional derivative of the exponential function, $f(x) = \mathrm{e}^x$, evaluated at $x = 1$ for $x_{\mathrm{A}} = 0$ computed with $k_{\max} = 16$ (lowest line), 32, 64, 128, and 256 terms (red line). The circular symbols indicate exact values for whole derivatives.

However, the ease of computing the Grünwald–Letnikov fractional derivative in terms of differentiation coefficients is an important advantage.

Exercise

4.6.1 Generate the counterpart of Figure 4.6.1 for the Gaussian distribution, $f(x) = \exp(-x^2)$.

4.7 Forward Grunwald–Letnikov fractional derivative

The first five ordinary derivatives of a function, $f(x)$, at a point, x, can be approximated with the following forward finite-difference formulas in terms of values at a specified array of evenly-spaced neighboring points,

$x_{i+p} = x_i + ph,$

$$\begin{bmatrix} f_i^{(0)} \\ hf_i^{(1)} \\ h^2 f_i^{(2)} \\ h^3 f_i^{(3)} \\ h^4 f_i^{(4)} \\ h^5 f_i^{(5)} \end{bmatrix} \simeq \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 \\ -1 & 3 & -3 & 1 & 0 & 0 \\ 1 & -4 & 6 & -4 & 1 & 0 \\ -1 & 5 & -10 & 10 & -5 & 1 \end{bmatrix} \cdot \begin{bmatrix} f_i \\ f_{i+1} \\ f_{i+2} \\ f_{i+3} \\ f_{i+4} \\ f_{i+5} \end{bmatrix}, \quad (4.7.1)$$

where h is a specified interval and $f_j \equiv f(x_j)$. Note the diagonal line of *ones* and the first column of alternating *ones* and *negative ones* in the matrix on the right-hand side. The accuracy of these approximations is first-order with respect to h.

4.7.1 General forward difference formula

The underlying forward difference formula for the mth ordinary derivative is

$$f_i^{(m)} \simeq \frac{1}{h^m} \sum_{\ell=0}^{m} (-1)^\ell \binom{m}{\ell} f_{i+m-\ell} \quad (4.7.2)$$

for $m = 0, 1, \ldots$, where the tall parentheses denote the binomial coefficient. The first term in the sum corresponding to $\ell = 0$ implements the aforementioned diagonal line of *ones*. The last term in the sum corresponding to $\ell = m$ implements the aforementioned alternating *ones* and *negative ones*.

Since the binomial coefficient satisfies the relation

$$\binom{m}{\ell} = \binom{m}{m-\ell}, \quad (4.7.3)$$

we may define $k = m - \ell$, set $\ell = m - k$, and obtain the alternative forward-difference formula

$$f_i^{(m)} \simeq (-1)^m \frac{1}{h^m} \sum_{k=0}^{m} (-1)^k \binom{m}{k} f_{i+k}. \quad (4.7.4)$$

For example, in the case of the first derivative, $m = 1$, we obtain

$$f_i^{(1)} \simeq -\frac{1}{h}\left(\binom{1}{0} f_i - \binom{1}{1} f_{i+1}\right) = \frac{f_{i+1} - f_i}{h}, \qquad (4.7.5)$$

which is a familiar forward finite-difference approximation of the first derivative.

4.7.2 Fractional derivative

To obtain the standard forward Grünwald–Letnikov fractional derivative of order α, we replace m in the binomial coefficient of (4.7.4) with the fractional order, α, do away with the factor $(-1)^m$ after the approximation sign, and modify the upper summation limit to obtain

$$[\mathrm{D}^{(\alpha)} f]^{x_\mathrm{B}}_{\mathrm{GL}}(x) \simeq \frac{1}{h^\alpha} \sum_{k=0}^{k_{\max}} (-1)^k \binom{\alpha}{k} f(x + kh), \qquad (4.7.6)$$

where x is an evaluation point,

$$k_{\max} = \left[\frac{x_\mathrm{B} - x}{h}\right] \qquad (4.7.7)$$

is a positive integer, the square brackets indicate the integral part, x_B is a specified upper limit, and $x \leq x_\mathrm{B}$.

In fact, equation (4.7.6) is as a natural extension of the backward Grünwald-Letnikov derivative defined in (4.6.5), repeated below for convenience,

$$[\mathrm{D}^{(\alpha)} f]^{\mathrm{GL}}_{x_\mathrm{A}}(x) \simeq \frac{1}{h^\alpha} \sum_{k=0}^{k_{\max}} (-1)^k \binom{\alpha}{k} f(x - kh) \qquad (4.7.8)$$

where h is a positive increment. More will be said about doing away with the factor $(-1)^m$ later in this section.

In terms of the differentiation coefficients $\omega_{\alpha,k}$ defined in (4.6.11), we obtain

$$[\mathrm{D}^{(\alpha)} f]^{x_\mathrm{B}}_{\mathrm{GL}}(x) \simeq \frac{1}{h^\alpha} \sum_{k=0}^{k_{\max}} \omega_{\alpha,k} f(x + kh), \qquad (4.7.9)$$

which is the counterpart of formula (4.6.12) for the Grünwald–Letnikov fractional derivative.

The forward fractional derivative arises from (4.7.6) or (4.7.9) in the limit as h tends zero and correspondingly $k_{\max}$ tends to infinity, while x_{B} is held constant.

4.7.3 Numerical evaluation

The computation of the forward Grünwald–Letnikov fractional derivative is implemented in the following Matlab code named *gl_fderf*, located in directory FRAC of SEMLIB:

```
function glp = gl_fderf (alpha,xB,x,kmax,menu)

%=====================================
% evaluate the right Grunwald--Letnikov
% fractional derivative
%=====================================

  h = (xB-x)/kmax;

  ishift = 1;
  omg(ishift) = 1;
  for k=1:kmax
    omg(k+ishift) = omg(k-1+ishift)*(k-alpha-1)/k;
  end

  glp = 0.0;

  for k=0:kmax
   [f,fp,fpp,fppp] = fnc(x+kh,menu);
   glp = glp + omg(k+ishift)*f;
  end

  glp = glp/h^alpha;

%---
% done
%---
```

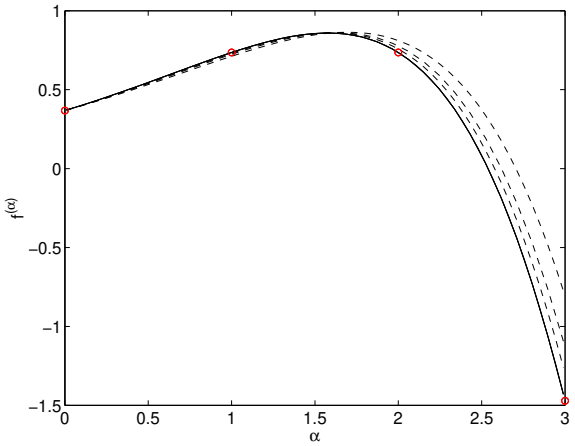


FIGURE 4.7.1 The forward Grünwald–Letnikov fractional derivative of the first moment of the Gaussian function, $f(x) = x\,\mathrm{e}^{-x^2}$, evaluated at $x = 1$, for $x_{\mathrm{B}} = 2$ computed with $k_{\max} = 16$ (lowest dashed line), 32, and 64 terms. The right Riemann–Liouville fractional derivative is drawn with the solid line.

```
return
```

An integer parameter `ishift` has been introduced to ensure that the array `omg` is assigned positive and non-zero index values.

4.7.4 Equivalence to the right Riemann–Liouville derivative

When the function $f(x)$ is sufficiently regular, the forward Grünwald–Letnikov fractional derivative reduces to the right Riemann–Liouville fractional derivative discussed in Section 4.4, as shown in Figure 4.7.1. Unfortunately, the rate of convergence with respect to the number of summed terms, $k_{\max}$, is slow.

4.7.5 Fractional Laplacian

Combining the backward with the forward fractional derivatives for $x_A = -\infty$ and $x_{\mathrm{B}} = \infty$, we obtain an expression for the fractional

Laplacian defined in (4.5.8),

$$[\,\mathrm{L}^{(\alpha)} f\,](x) = -\frac{1}{\cos(\dfrac{\alpha\pi}{2})}\,{}_{\mathrm{RL}}\big[\,\mathrm{D}^{(\alpha)} f\,\big]_{-\infty}^{+\infty}(x) \tag{4.7.10}$$

for $0 \le \alpha < 2$. Making substitutions, we find that

$$[\,\mathrm{L}^{(\alpha)} f\,](x) \simeq -\tfrac{1}{2}\,\frac{1}{h^{\alpha}}\,\frac{1}{\cos(\dfrac{\alpha\pi}{2})}\,\sum_{k=-\infty}^{\infty} \varphi_k f_{i+k}, \tag{4.7.11}$$

where

$$\varphi_0 = 2, \qquad \varphi_k = \omega_{\alpha,|k|} \tag{4.7.12}$$

for $k \ne 0$ are influence coefficients.

The method is implemented in the following Matlab code named *gl_flap*, located in directory FRAC of SEMLIB:

```
function glp = gl_flap (alpha,x,h,kmax,menu)

%==============================
% Evaluate the fractional Laplacian
% based on the Grunwald--Letnikov
% fractional derivatives
%==============================

  ishift = 1;

  omg(ishift) = 1;

  for k=1:kmax
    omg(k+ishift) = omg(k-1+ishift)*(k-alpha-1)/k;
  end

  glp = 0.0;

  for k=0:kmax
   [f,fp,fpp,fppp] = fnc(x-k*h,menu);
```

```
   glp = glp + omg(k+ishift)*f;
   [f,fp,fpp,fppp] = fnc(x+k*h,menu);
   glp = glp + omg(k+ishift)*f;
  end

  glp = glp/h^alpha;
  glp = - glp/(2.0*cos(alpha*pi/2));

%---
% done
%---

return
```

The fractional Laplacian of the first moment of the Gaussian distribution, $f(x) = x\,\mathrm{e}^{-x^2}$, is shown in Figure 4.7.2 for $\alpha = 0.678$. The solid line represents the whole distribution obtained by evaluating a principal-value integral, as discussed in Section 4.9.

The circular symbols at $x = 0$ and 1 represent results obtained using formula (4.7.11) with 64 terms. The results suggest that formula (4.7.11) provides us with an efficient method for approximating the fractional Laplacian in applications.

4.7.6 *Differentiation coefficients*

Equation (4.7.11) can be recast into the form

$$[\,\mathrm{L}^{(\alpha)} f\,](x) \simeq \frac{1}{h^\alpha} \sum_{k=-\infty}^{\infty} d_{|k|} f_{i+k}, \tag{4.7.13}$$

where d_k are differentiation coefficients. The distribution of these coefficients is shown in Figure 4.7.3 for several values of α. As α tends to zero, all coefficients but d_0 tend to zero while d_0 tends to -1, yielding the negative of the function. As α tends to unity, the coefficients diverge.

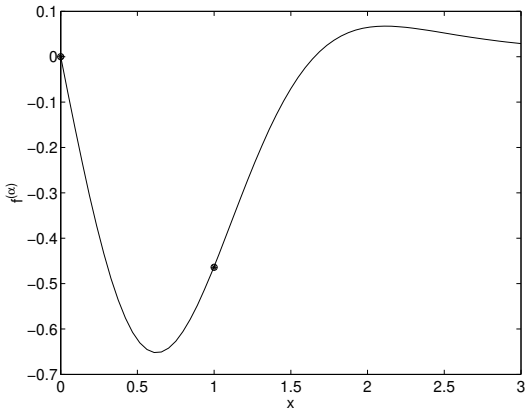


FIGURE 4.7.2 The fractional Laplacian of the first moment of the Gaussian distribution, $f(x) = x\,\mathrm{e}^{-x^2}$, for $\alpha = 0.678$. The symbols represents results obtained using formula (4.7.11).

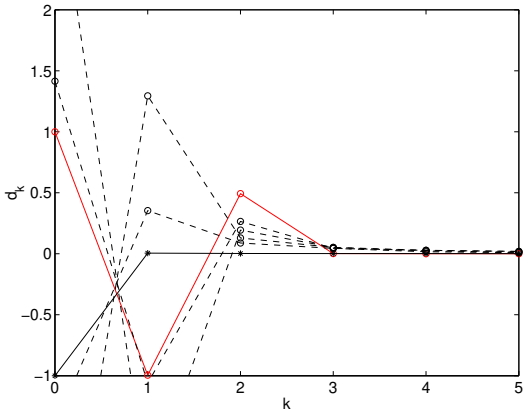


FIGURE 4.7.3 Differentiation coefficients for the fractional Laplacian for $\alpha = 2$ (solid line connecting circular symbols), $1.5, 1.2, 0.8, 0.5$, and 0.1 (solid line connecting asterisks).

4.7.7 Complex derivative

Equation (4.7.4) expressing a forward finite-difference approximation can be written as

$$f^{(m)}(x) \simeq \frac{1}{\Delta x^m} \sum_{k=0}^{m} (-1)^k \binom{m}{k} f(x - k\Delta x), \tag{4.7.14}$$

where $\Delta x = -h$ is a negative increment and h is a positive interval. In fact, this equation arises from the backward finite-difference approximation (4.6.4) by substituting Δx for h. This observation motivates introducing a small complex increment,

$$\Delta z \equiv -h\, \mathrm{e}^{\mathrm{i}\phi}, \tag{4.7.15}$$

where h is a real increment, i is the imaginary unit, $\mathrm{i}^2 = -1$, and ϕ is an arbitrary phase. The value $\phi = 0$ corresponds to forward differences discussed in this section, whereas the value $\phi = \pi$ corresponds to backward differences discussed in Section 4.6.

Now we may extend our domain of definition to the complex z plane and define the complex derivative of integer order, m,

$$f^{(m)}(z) \simeq \frac{1}{\Delta z^m} \sum_{k=0}^{m} (-1)^k \binom{m}{k} f(z - k\Delta z), \tag{4.7.16}$$

where z is a chosen evaluation point. If the complex function $f(z)$ is analytic, the mth derivative, $f^{(m)}(z)$ is independent of the orientation of Δz in the limit as Δz tends to zero. When the evaluation point z lies on the real x axis, we obtain the real mth derivative of the real function $f(x)$.

The complex Grünwald–Letnikov fractional derivative is given by

$$[\mathrm{D}^{(\alpha)} f]^{\mathrm{GL}}_{z_\mathrm{A}}(z) \simeq \frac{1}{\Delta z^\alpha} \sum_{k=0}^{k_{\max}} (-1)^k \binom{\alpha}{k} f(z - k\,\Delta z) \tag{4.7.17}$$

for positive fractional orders, α, where z is an evaluation point,

$$k_{\max} = \Big[\frac{z - z_\mathrm{A}}{\Delta z} \Big] \tag{4.7.18}$$

is a positive integer, the square brackets indicate the integral part, and z_{A} is a specified point in the complex plane. In terms of the differentiation coefficients $\omega_{\alpha,k}$,

$$ {}_{\mathrm{GL}}[\mathrm{D}^{(\alpha)} f]_{z_{\mathrm{A}}}(z) \simeq \frac{1}{\Delta z^{\alpha}} \sum_{k=0}^{k_{\max}} \omega_{\alpha,k} f(z - k\,\Delta z). \tag{4.7.19} $$

The complex derivative will be discussed further in Section 11.5.

Exercise

4.7.1 Generate the counterpart of Figure 4.7.1 for the first moment of the Gaussian distribution, $f(x) = x\,\exp(-x^2)$.

4.8 Caputo fractional derivative

The Caputo fractional derivative is a variation of the left Riemann–Liouville fractional derivative, defined as

$$ \big[\,\mathrm{D}^{(\alpha)} f\,\big]^{\mathrm{C}}_{x_{\mathrm{A}}}(x) \equiv \frac{1}{\Gamma(n-\alpha)} \int_{x_{\mathrm{A}}}^{x} \frac{f^{(n)}(\xi)}{(x-\xi)^{\alpha+1-n}}\,\mathrm{d}\xi \tag{4.8.1} $$

for $n - 1 \leq \alpha < n$, where x_{A} is a specified lower integration limit. Defining $v = x - \xi$, we obtain

$$ \big[\,\mathrm{D}^{(\alpha)} f\,\big]^{\mathrm{C}}_{x_{\mathrm{A}}}(x) \equiv (-1)^n \frac{1}{\Gamma(n-\alpha)} \int_{0}^{x-x_{\mathrm{A}}} \frac{f^{(n)}(x-v)}{v^{\alpha+1-n}}\,\mathrm{d}v. \tag{4.8.2} $$

The Caputo fractional derivative is preferred over the Riemann–Liouville derivative in applications where the Laplace transform is required.

4.8.1 Relation to the Riemann–Liouville derivative

We recall that the Riemann–Liouville derivative is given by the analogous formula

$$ [\,\mathrm{D}^{(\alpha)} f\,]^{\mathrm{RL}}_{x_{\mathrm{A}}}(x) = \frac{1}{\Gamma(n-\alpha)} \frac{\mathrm{d}^n}{\mathrm{d}x^n} \int_{x_{\mathrm{A}}}^{x} \frac{f(\xi)}{(x-\xi)^{\alpha+1-n}}\,\mathrm{d}\xi \tag{4.8.3} $$

for $n-1 \leq \alpha < n$. Integrating by parts, we derive the following relation between the Caputo and the Riemann–Liouville derivative:

$$\big[\,\mathrm{D}^{(\alpha)} f\,\big]^{\mathrm{C}}_{x_\mathrm{A}}(x) = \big[\mathrm{D}^{(\alpha)} f\big]^{\mathrm{RL}}_{x_\mathrm{A}}(x) - \sum_{k=0}^{n-1} \frac{f^{(k)}(x_\mathrm{A})}{\Gamma(k+1-\alpha)}\,(x-x_\mathrm{A})^{k-\alpha}. \tag{4.8.4}$$

When $n = 1$, that is, for $0 \leq \alpha < 1$, only one term in the sum survives. The sum does not make a contribution when the function $f(x)$ and its derivatives virtually vanish at the chosen end-point, x_A, except when the evaluation point x is very close to x_A.

4.8.2 Limiting values

The limiting values of the Caputo derivative at the end-points of the interval $(n-1, n)$ are given by

$$\lim_{\alpha \to n-1} [\,\mathrm{D}^{(\alpha)} f\,]^{\mathrm{C}}_{x_\mathrm{A}}(x) = f^{(n-1)}(x) - f^{(n-1)}(x_\mathrm{A}) \tag{4.8.5}$$

and

$$\lim_{\alpha \to n} [\,\mathrm{D}^{(\alpha)} f\,]^{\mathrm{C}}_{x_\mathrm{A}}(x) = f^{(n)}(x), \tag{4.8.6}$$

where we recall that $n-1 \leq \alpha < n$. These formulas show that discontinuities generally occur at the end-points of the interval between two integers. When the function $f(x)$ decays as $x \to -\infty$, then, in the limit as $x_\mathrm{A} \to -\infty$, the discontinuities disappear.

4.8.3 Flat function

The Caputo derivative of the flat function, $f(x) = c$ is identically zero for any fractional order, α,

$$[\,\mathrm{D}^{(\alpha)} c\,]^{\mathrm{C}}_{x_\mathrm{A}} = 0, \tag{4.8.7}$$

where c is a constant. The vanishing of this fractional derivative is reassuring and represents an advantage of the the Riemann–Liouville derivative.

4.8.4 Power function

The Caputo derivative of the power function,

$$f(x) = x^p, \tag{4.8.8}$$

where p is zero or a positive integer, is given by

$$[\,\mathrm{D}^{(\alpha)} x^p\,]^{\mathrm{C}}_{x_\mathrm{A}}(x) = 0 \tag{4.8.9}$$

for $p \leq n-1$ and

$$[\,\mathrm{D}^{(\alpha)} x^p\,]^{\mathrm{C}}_{x_\mathrm{A}}(x) = [\,\mathrm{D}^{(\alpha)} x^p\,]^{\mathrm{RL}}_{x_\mathrm{A}}(x) = \frac{\Gamma(p+1)}{\Gamma(p-\alpha+1)}\,(x-x_\mathrm{A})^{p-\alpha} \tag{4.8.10}$$

for $p > n-1$.

Using these relations, we find that the fractional derivative of the Taylor series expansion of a function $f(x)$, is given by

$$[\,\mathrm{D}^{(\alpha)} f\,]^{\mathrm{C}}_{x_\mathrm{A}}(x) = \sum_{k=n}^{\infty} \frac{f^{(k)}(x)}{\Gamma(k-\alpha+1)}\,(x-x_\mathrm{A})^k, \tag{4.8.11}$$

where $n-1 \leq \alpha < n$. Note that summation starts at $k=n$ on the right-hand side.

4.8.5 Computation of the Caputo fractional derivative

To compute the integral in (4.8.1), we subtract out the singularity by writing

$$\begin{aligned}\int_{x_\mathrm{A}}^{x} \frac{f^{(n)}(\xi)}{(x-\xi)^{\alpha+1-n}}\,\mathrm{d}\xi = \int_{x_\mathrm{A}}^{x} \frac{f^{(n)}(\xi)-f^{(n)}(x)}{(x-\xi)^{\alpha+1-n}}\,\mathrm{d}\xi \\ + f^{(n)}(x)\int_{x_\mathrm{A}}^{x} \frac{\mathrm{d}\xi}{(x-\xi)^{\alpha+1-n}}.\end{aligned} \tag{4.8.12}$$

Using a Taylor series expansion, we write

$$f^{(n)}(x) - f^{(n)}(x_0) \simeq f^{(n+1)}(x_0)\,(x-x_0) + \cdots, \tag{4.8.13}$$

and find that the integrand of the first integral on the right-hand side of (4.8.12) behaves like $(x_0 - x)^{n-\alpha}$, and is thus sufficiently regular for the integral to be computed by standard numerical methods, such as the mid-point rule.

The second integral on the right-hand side of (4.8.12) can be computed analytically, and is given by

$$\int_{x_A}^{x} \frac{\mathrm{d}\xi}{(x-\xi)^{\alpha+1-n}} = \frac{1}{n-\alpha}\,(x - x_A)^{n-\alpha} \tag{4.8.14}$$

for $x > x_A$, which is finite provided that $\alpha \leq n$.

4.8.6 Numerical evaluation

The following Matlab function named *caputo_fder*, located in directory FRAC of SEMLIB, evaluates the Caputo fractional integral using this method:

```
function cfd = caputo_fder (alpha,xA,x,nint,menu)

%=======================================
% compute the Caputo fractional derivative
%
% int_xA_x f^(n)(x) /(x-xi)^(alpha+1-n) d x
%
% nint: integration intervals
%=======================================

%---
% prepare
%---

[f,fp,fpp,fppp] = fnc(x,menu);

Dx = (x-xA)/nint;

if(alpha >= 0 & alpha < 1) n = 1; end
if(alpha >= 1 & alpha < 2) n = 2; end
if(alpha >= 2 & alpha < 3) n = 3; end
if(n>3)
```

```
  disp('caputo_fder: this values of alpha not yet available')
  return
end

nexp = alpha+1-n;

%---
% integrate by the mid-point rule
%---

intgrl = 0.0;

for i=1:nint

 xi(i) = xA + (i-0.5)*Dx;
 xr(i) = x-xi(i);

 [fxi(i),fxip(i),fxipp(i),fxippp(i)] = fnc(xi(i),menu);

 if(n==1)
  intgrl = intgrl + (fxip(i)-fp)/xr(i)^nexp;
 elseif(n==2)
  intgrl = intgrl + (fxipp(i)-fpp)/xr(i)^nexp;
 elseif(n==3)
  intgrl = intgrl + (fxippp(i)-fppp)/xr(i)^nexp;
 end

end
%---

cfd = intgrl*Dx;
nal = n-alpha;

 if(n==1)
  cfd = cfd + fp/nal * (x-xA)^nal;
 elseif(n==2)
  cfd = cfd + fpp/nal * (x-xA)^nal;
 elseif(n==3)
  cfd = cfd + fppp/nal * (x-xA)^nal;
```

```
 end

cfd = cfd/gamma(nal);

%---
% done
%---

return
```

4.8.7 Exponential function

The Caputo fractional derivative of the exponential function, $f(x) = \mathrm{e}^x$, evaluated at $x_0 = 1.0$ is shown in Figure 4.8.1 for $x_\mathrm{A} = 0$.

The results confirm that discontinuities occur at integer values of α. In contrast, the Riemann–Liouville fractional derivative, plotted with the broken line in Figure 4.2.1, does not exhibit discontinuities. In this case, substantial differences are observed between the Caputo and the Riemann–Liouville fractional derivative.

Exercise

4.8.1 Prepare the counterpart of Figure 4.8.1 for the Gaussian distribution, $f(x) = \exp(-x^2)$.

4.9 Right Caputo fractional derivative

The right Caputo fractional derivative is a variation of the right Riemann–Liouville fractional derivative, defined as

$$\big[\,\mathrm{D}^{(\alpha)} f\,\big]_\mathrm{C}^{x_\mathrm{B}}(x) \equiv (-1)^n \frac{1}{\Gamma(n-\alpha)} \int_x^{x_\mathrm{B}} \frac{f^{(n)}(\xi)}{(\xi - x)^{\alpha+1-n}}\, \mathrm{d}\xi \qquad (4.9.1)$$

for $n-1 \leq \alpha < n$ and $x \leq x_\mathrm{B}$, where x_B is a specified upper integration limit. Note the alternating factor $(-1)^n$ on the right-hand side.

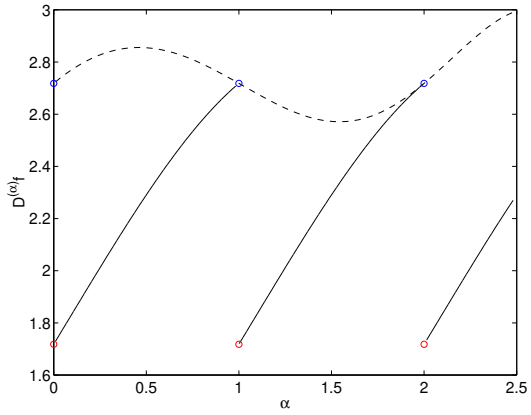


FIGURE 4.8.1 The Caputo fractional derivative of the exponential function, $f(x) = \mathrm{e}^x$ (solid line) evaluated at $x_0 = 1.0$ for $x_\mathrm{A} = 0$. Note that discontinuities occur at integer values of α. The Riemann–Liouville fractional derivative is plotted with the broken line.

4.9.1 Relation to the right Riemann–Liouville derivative

We recall that the right Riemann–Liouville derivative is given by the analogous formula

$$[\,\mathrm{D}^{(\alpha)} f\,]^{x_\mathrm{B}}_{\mathrm{RL}}(x_0) = (-1)^n \frac{1}{\Gamma(n-\alpha)} \frac{\mathrm{d}^n}{\mathrm{d}x^n} \int_x^{x_\mathrm{B}} \frac{f(\xi)}{(x-\xi)^{\alpha+1-n}}\,\mathrm{d}\xi \tag{4.9.2}$$

for $n-1 \leq \alpha < n$ and $x \leq x_\mathrm{B}$. Integrating by parts, we find the following relation between the right Caputo and the right Riemann–Liouville fractional derivative:

$$\begin{aligned}\big[\,\mathrm{D}^{(\alpha)} f\,\big]^{x_\mathrm{B}}_{\mathrm{C}}(x) &= \big[\,\mathrm{D}^{(\alpha)} f\,\big]^{x_\mathrm{B}}_{\mathrm{RL}}(x) \\ &\quad + \sum_{k=0}^{n-1} (-1)^k \frac{f^{(k)}(x_\mathrm{B})}{\Gamma(k+1-\alpha)}\,(x_\mathrm{B}-x)^{k-\alpha}.\end{aligned} \tag{4.9.3}$$

Note the alternating factor $(-1)^n$ inside the sum. When $n = 1$, only one term in the sum survives. The sum does not make a contribution when the function $f(x)$ virtually vanishes at the chosen end-point, x_B.

4.9.2 Limiting values

The limiting values of the Caputo derivative in the interval $(n-1, n)$ are given by

$$\lim_{\alpha \to n-1} [\, \mathrm{D}^{(\alpha)} f \,]_{\mathrm{C}}^{x_{\mathrm{B}}}(x) = (-1)^n \left(f^{(n-1)}(x) - f^{(n-1)}(x_{\mathrm{B}}) \right) \tag{4.9.4}$$

and

$$\lim_{\alpha \to n} [\, \mathrm{D}^{(\alpha)} f \,]_{\mathrm{C}}^{x_{\mathrm{B}}}(x) = (-1)^n f^{(n)}(x), \tag{4.9.5}$$

where $n - 1 \leq \alpha < n$. These formulas show that discontinuities occur at the end-points of the interval. When the function $f(x)$ decays as $x \to \infty$, in the limit as $x_{\mathrm{B}} \to -\infty$, the discontinuities disappear.

4.9.3 Computation of the right Caputo fractional derivative

To compute the integral in (4.9.1), we subtract out the singularity by writing

$$\begin{aligned} \int_x^{x_{\mathrm{B}}} \frac{f^{(n)}(\xi)}{(\xi - x)^{\alpha+1-n}} \, \mathrm{d}\xi = & \int_x^{x_{\mathrm{B}}} \frac{f^{(n)}(\xi) - f^{(n)}(x)}{(\xi - x)^{\alpha+1-n}} \, \mathrm{d}\xi \\ & + f^{(n)}(x) \int_x^{x_{\mathrm{B}}} \frac{\mathrm{d}\xi}{(\xi - x)^{\alpha+1-n}}. \end{aligned} \tag{4.9.6}$$

Using a Taylor series expansion, we write

$$f^{(n)}(\xi) - f^{(n)}(x) \simeq f^{(n+1)}(x) \, (\xi - x) + \cdots, \tag{4.9.7}$$

and find that the integrand of the first integral on the right-hand side of (4.9.6) behaves like $(x_0 - x)^{n-\alpha}$, and is thus sufficiently regular for the integral to be computed by standard numerical methods, such as the mid-point rule.

The second integral on the right-hand side of (4.9.6) can be computed analytically, and is given by

$$\int_x^{x_{\mathrm{B}}} \frac{\mathrm{d}\xi}{(x - \xi)^{\alpha+1-n}} = \frac{1}{n - \alpha} \, (x_{\mathrm{B}} - x)^{n-\alpha} \tag{4.9.8}$$

for $x < x_{\mathrm{B}}$, which is finite provided that $\alpha \leq n$.

4.9.4 Numerical evaluation

The following Matlab function named *caputo_fderr*, located in directory FRAC of SEMLIB, evaluates the Caputo fractional integral using this method:

```
function cfd = caputo_fderr (alpha,xB,x,nint,menu)

%=======================================
% compute the Caputo fractional derivative
%
% (-1)^n int_x_xB f^(n)(xi) /(xi-x)^(alpha+1-n) d xi
%
% nint: integration intervals
%=======================================

%---
% prepare
%---

if(alpha >= 0 & alpha < 1)
  n = 1;
elseif(alpha >= 1 & alpha < 2)
  n = 2;
elseif(alpha >= 2 & alpha < 3)
  n = 3;
else
  disp('caputo_fderr: this values of alpha not yet available')
  return
end

Dx = (xB-x)/nint;
nexp = alpha+1-n;

[f,fp,fpp,fppp] = fnc(x,menu);

%---
% integrate by the mid-point rule
%---
```

```
intgrl = 0.0;

for i=1:nint

 xi(i) = x + (i-0.5)*Dx;
 xr(i) = xi(i)-x;

 [fxi(i),fxip(i),fxipp(i),fxippp(i)]  ...
        = fnc(xi(i),menu);

 if(n==1)
  intgrl = intgrl + (fxip(i)-fp)/xr(i)^nexp;
 elseif(n==2)
  intgrl = intgrl + (fxipp(i)-fpp)/xr(i)^nexp;
 elseif(n==3)
  intgrl = intgrl + (fxippp(i)-fppp)/xr(i)^nexp;
 end

end
%---

cfd = intgrl*Dx;

nal = n-alpha;

 if(n==1)
  cfd = cfd + fp/nal * (xB-x)^nal;
 elseif(n==2)
  cfd = cfd + fpp/nal * (xB-x)^nal;
 elseif(n==3)
  cfd = cfd + fppp/nal * (xB-x)^nal;
 end

cfd = (-1)^n*cfd/gamma(nal);

%---
% done
%---
```

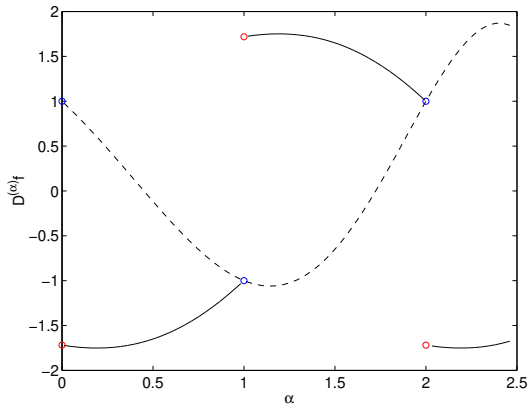


FIGURE 4.9.1 The right Caputo fractional derivative of the exponential function, $f(x) = \mathrm{e}^x$ (solid line) evaluated at $x = 0.0$ for $x_\mathrm{B} = 1.0$. Note that discontinuities occur at integer values of α. The Riemann–Liouville fractional derivative is drawn with the broken line.

```
return
```

The right Caputo fractional derivative of the exponential function, $f(x) = \mathrm{e}^x$ evaluated at $x = 0$ is shown in Figure 4.9.1 for $x_\mathrm{B} = 1$. An expected discontinuity occurs at integer values of α. In contrast, the Riemann–Liouville fractional derivative, plotted with the broken line, does not exhibit discontinuities.

Exercise

4.9.1 Prepare the counterpart of Figure 4.9.1 for the Gaussian distribution, $f(x) = \exp(-x^2)$.

4.10 The fractional Laplacian

The fractional Laplacian of a suitable function, $f(x)$, defined over the entire x axis and decaying sufficiently fast as $x \to \pm\infty$, also known as the Riesz fractional derivative, is defined by the expression

$$ {}_{\mathrm{FL}}\big[\,\mathrm{D}^{(\alpha)} f\,\big](x_0) \equiv c_{1,\alpha} \int_{-\infty}^{\infty}\!\!\!\!\mathrm{pv}\;\; \frac{f(x)-f(x_0)}{|x-x_0|^{1+\alpha}}\,\mathrm{d}x, \qquad (4.10.1) $$

which can be restated as

$$ {}_{\mathrm{FL}}\big[\,\mathrm{D}^{(\alpha)} f\,\big](x_0) \equiv c_{1,\alpha} \int_{-\infty}^{\infty}\!\!\!\!\mathrm{pv}\;\; \frac{f(x_0+v)-f(x_0)}{|v|^{1+\alpha}}\,\mathrm{d}v, \qquad (4.10.2) $$

where x_0 is an arbitrary evaluation point, α is the order of the fractional derivative varying in the range $[0,2)$, and pv denotes the principal-value integral. The coefficient $c_{1,\alpha}$ is given by

$$ c_{1,\alpha} = \alpha\,\frac{2^{\alpha-1}}{\sqrt{\pi}}\,\frac{\Gamma(\frac{1+\alpha}{2})}{\Gamma(\frac{2-\alpha}{2})}, \qquad (4.10.3) $$

where Γ is the Gamma function. An explanation for this value will be given later in this section.

The principal-value integral in (4.10.1) and (4.10.2) is defined as the limit as $\epsilon \to 0$ of the two sums

$$ \int_{-\infty}^{x_0-\epsilon} \frac{f(x_0+v)-f(x_0)}{|v|^{1+\alpha}}\,\mathrm{d}v + \int_{x_0+\epsilon}^{\infty} \frac{f(x_0+v)-f(x_0)}{|v|^{1+\alpha}}\,\mathrm{d}v, \qquad (4.10.4) $$

where ϵ is a small positive parameter. In this limit, a positive infinite contribution cancels a negative infinite contribution, yielding a finite contribution.

Motivation for the preceding definitions will be given later in this section in the context of random flights.

4.10.1 Alternative expression

An alternative expression for the fractional Laplacian is

$$ {}_{\mathrm{FL}}\big[\,\mathrm{D}^{(\alpha)} f\,\big](x_0) \equiv c_{1,\alpha} \int_0^\infty \frac{f(x_0 - v) - 2\,f(x_0) + f(x_0 + v)}{v^{1+\alpha}}\,\mathrm{d}v, \tag{4.10.5} $$

Note the semi-infinite integration domain. It is interesting that the numerator of the fraction inside the integral of (4.10.5) on the right-hand side is the second-order centered difference stencil associated with the ordinary second derivative.

4.10.2 Explanation for $c_{1,\alpha}$

To explain the meaning of the coefficient, $c_{1,\alpha}$, we introduce the Fourier transform of a rapidly decaying function, $f(x)$, denoted by a caret (hat), defined as

$$ \widehat{f}(k) \equiv \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x)\,\mathrm{e}^{\mathrm{i}kx}\,\mathrm{d}x, \tag{4.10.6} $$

where k is the wave number in Fourier space and i is the imaginary unit, $\mathrm{i}^2 = -1$. The inverse Fourier transform provides us with the function under consideration,

$$ f(x) \equiv \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \widehat{f}(k)\,\mathrm{e}^{-\mathrm{i}kx}\,\mathrm{d}k. \tag{4.10.7} $$

If the Fourier transform is available, the function can be reconstructed uniquely, and *vice versa*.

The value of $c_{1,\alpha}$ is computed deliberately so that

$$ \widehat{f''}(k) = -|k|^\alpha\,\widehat{f}(k), \tag{4.10.8} $$

which is hallmark of the fractional Laplacian, where the double prime denotes the fractional Laplacian. As α tends to zero, we obtain the negative of the Fourier transform of the function itself, $\widehat{f}(k)$.

Relation (4.10.8) is a generalization of a familiar relation for the ordinary Laplacian corresponding to $\alpha = 2$,

$$ \widehat{\ddot{f}}(k) = -k^2\,\widehat{f}(k), \tag{4.10.9} $$

where the double dot denotes the ordinary second derivative. The Fourier transform of the fractional Laplacian for $\alpha = 1$ is *not* equal to the Fourier transform of the first derivative, which is given by

$$\widehat{\dot{f}}(k) = -\mathrm{i}\, k\, \widehat{f}(k), \tag{4.10.10}$$

where the dot denotes the ordinary first derivative.

4.10.3 Integer fractional orders

As $\alpha \to 2$, the fractional Laplacian reduces to the the ordinary second derivative. As $\alpha \to 0$, the fractional Laplacian reduces to the negative of the function at the evaluation point, $-f(x_0)$.

As $\alpha \to 1$, the fractional does not necessarily reduce to the ordinary first derivative,

$$\lim_{\alpha \to 1_{\mathrm{FL}}} \big[\, \mathrm{D}^{(\alpha)} f \,\big](x_0) \neq \pm f'(x_0), \tag{4.10.11}$$

that is, the nominal "square root" of the Laplacian is *not* generally equal to the ordinary first derivative, $f'(x_0)$.

4.10.4 Relation to the centered Riemann–Liouville fractional derivative

In Section 4.2, we stated and confirmed by numerical evaluation that the fractional Laplacian is related to the centered Riemann–Liouville fractional derivative by the equation

$$_{\mathrm{R}}\big[\mathrm{D}^{(\alpha)} f\big](x_0) = -\frac{1}{2\,\cos(\dfrac{\alpha\pi}{2})}\, {}_{\mathrm{RL}}\big[\, \mathrm{D}^{(\alpha)} f \,\big]_{-\infty}^{+\infty}(x_0) \tag{4.10.12}$$

for $0 \leq \alpha < 2$. This formula provides us with an indirect method for evaluating the fractional Laplacian by numerical methods.

4.10.5 Direct numerical evaluation

With reference to (4.10.5), we define the integral

$$\mathcal{J}(x_0) \equiv \int_0^\infty \frac{f(x_0 - v) - 2\, f(x_0) + f(x_0 + v)}{v^{1+\alpha}}\, \mathrm{d}v. \tag{4.10.13}$$

To compute the integral with arbitrary accuracy, we introduce a suitable cut-off length, $\omega(x_0) > 0$, and break up the semi-infinite integration domain into two pieces, writing

$$\mathcal{J}(x_0) = \mathcal{J}_1(x_0) + f''(x_0)\,\mathcal{J}_2(x_0) + \mathcal{J}_3(x_0), \tag{4.10.14}$$

where $f''(x_0)$ is the ordinary second derivative evaluated at x_0,

$$\mathcal{J}_1(x_0) \equiv \int_0^{\omega(x_0)} \frac{f(x_0 - v) - 2\,f(x_0) + f(x_0 + v) - f''(x_0)\,v^2}{v^{1+\alpha}}\,\mathrm{d}v \tag{4.10.15}$$

is a regularized integral in a finite interval,

$$\mathcal{J}_2(x_0) \equiv \int_0^{\omega(x_0)} v^{1-\alpha}\,\mathrm{d}v = \frac{1}{2-\alpha}\,\omega(x_0)^{2-\alpha} \tag{4.10.16}$$

is a standard integral, and

$$\mathcal{J}_3(x_0) \equiv \int_{\omega(x_0)}^{\infty} \frac{f(x_0 - v) - 2\,f(x_0) + f(x_0 + v)}{v^{1+\alpha}}\,\mathrm{d}v \tag{4.10.17}$$

is a regular integral over a semi-infinite domain.

The integrand of the first integral, $\mathcal{J}_1(x_0)$, remains nonsingular as the integration variable, v, tends to zero, behaving as

$$\tfrac{1}{12}\,f^{(4)}(x_0)\,v^{3-\alpha}, \tag{4.10.18}$$

where $f^{(4)}(x_0)$ is the ordinary fourth derivative evaluated at x_0. Strictly speaking, the integral exists for a class of functions that fall in the Schwartz space. Consequently, the integral defining $\mathcal{J}_1(x_0)$ can be computed by conventional numerical methods, including the trapezoidal rule.

If we choose $\omega(x_0) \gg |x_0|$, we can approximate the third integral as

$$\mathcal{J}_3(x_0) \simeq -2\,f(x_0) \int_{\omega(x_0)}^{\infty} \frac{\mathrm{d}v}{v^{1+\alpha}}, \tag{4.10.19}$$

yielding

$$\mathcal{J}_3(x_0) \simeq -f(x_0)\,\frac{2}{\alpha}\,\frac{1}{\omega(x_0)^{\alpha}}. \tag{4.10.20}$$

The accuracy of this approximation improves as $\omega(x_0)$ becomes larger.

The method is implemented in the following Matlab code for the Gaussian distribution, $f(x) = \mathrm{e}^{-x^2}$:

```
%---
% integration limits
%---

  a = 0.00001;
  b = x0+ 9.0000;

  Dv = (b-a)/Nint;

%---
% base points and function values
%---

 for i=1:Nint+1
  v = a+(i-1.0)*Dv;

   f(i) =     (    exp(-(x0-v)^2) ...
              -2.0*exp(-    x0^2) ...
              +    exp(-(x0+v)^2) ...
              +2.0*(1.0-2.0*x0^2)*exp(-x0^2)*v^2 )/v^alp;
%---
% compute the v integral
% by the trapezoidal rule
%---

 trapz = 0.5*f(1);

 for i=2:Nint
    trapz = trapz + f(i);
 end
 trapz = trapz + 0.5*f(Nint+1);
```

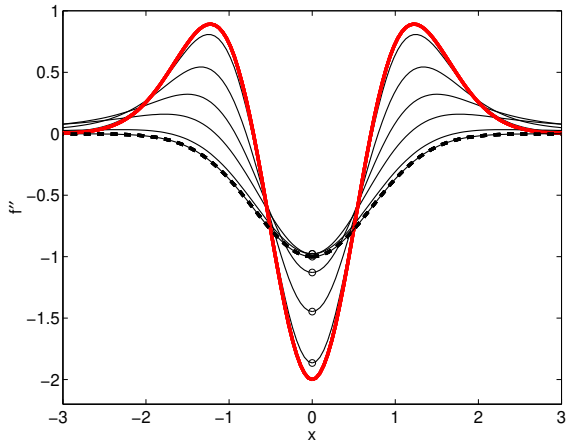


FIGURE 4.10.1 The fractional Laplacian of the Gaussian distribution, $f(x) = e^{-x^2}$, for $\alpha = 2$ (bold red solid line), 1.9, 1.5, 1.0, 0.5, 0.1, and 0 (bold broken line representing the negative of the Gaussian).

```
trapz = trapz*Dv;

trapz = trapz + 2.0*(2.0*x0^2-1.0) ...
        *exp(-x0^2)*(b^alpp+0^alpp)/alpp;
trapz = trapz - 2.0*exp(-x0^2)/(alpha*b^alpha);

c1 = 0.5*alpha/(gamma(1-alpha)*cos(alpha*pi/2));

flapl = c1*trapz;
```

A graph of the fractional Laplacian of the Gaussian distribution is shown in Figure 4.10.1.

Other functions can be implemented in the code by straightforward substitutions, provided that their second ordinary derivative can be computed analytically or by numerical methods.

4.10.6 Green's function

The Green's function of the unsteady fractional diffusion equation, denoted by $\mathcal{G}_\alpha$, satisfies the equation

$$\frac{\partial \mathcal{G}_\alpha}{\partial t} = \kappa_\alpha \, {}_{\rm FL}\big[\, {\rm D}^{(\alpha)} \mathcal{G}_\alpha \,\big](x) + \delta_1(x - x_s)\, \delta_1(t - t_s) \tag{4.10.21}$$

for $t > t_s$ and any x, where δ_1 is the Dirac delta function in one dimension, x_s is a chosen singular point, t_s is a chosen application time, κ_α is the fractional diffusivity with units of ℓ^α/τ, ℓ is a characteristic length and τ is a characteristic time.

Using the method of Fourier transforms, we find that

$$\mathcal{G}_\alpha(\widehat{x}, \widehat{t}) = \frac{1}{(\kappa_\alpha \widehat{t})^{1/\alpha}} \, \Phi_\alpha(\eta_\alpha), \tag{4.10.22}$$

where

$$\widehat{t} \equiv t - t_s, \qquad \widehat{x} \equiv x - x_s, \qquad \eta_\alpha \equiv \frac{\widehat{x}}{(\kappa_\alpha \widehat{t})^{1/\alpha}} \tag{4.10.23}$$

is a dimensionless similarity variable,

$$\Phi_\alpha(\eta_\alpha) = \frac{1}{2\pi} \int_{-\infty}^{\infty} {\rm e}^{-w^\alpha} \exp(-{\rm i}\, \eta_\alpha w) \, {\rm d}w \tag{4.10.24}$$

is a dimensionless function, and ${\rm i}$ is the imaginary unit, ${\rm i}^2 = -1$. Resolving the complex exponential inside the integral into its real and imaginary parts, we obtain

$$\Phi_\alpha(\eta_\alpha) = \frac{1}{\pi} \int_0^{\infty} {\rm e}^{-w^\alpha} \cos(\eta_\alpha w) \, {\rm d}w. \tag{4.10.25}$$

Analytical expressions for the definite integral on the right-hand side are available only for certain values of α.

For $\alpha = 1$, we find that

$$\mathcal{G}_1(x - x_s, t - t_s) = \frac{1}{\pi \kappa_1 (t - t_s)} \, \frac{1}{1 + \eta_1^2}, \tag{4.10.26}$$

where

$$\eta_1 \equiv \frac{x - x_s}{\kappa_1 (t - t_s)} \tag{4.10.27}$$

is a dimensionless similarity variable.

4.10.7 Physical relevance

Of all fractional derivatives and related operators, the fractional Laplacian plays a prominent role in that it describes macroscopically diffusion-like stochastic processes associated with long displacements identified as flights or jumps. In fact, the fractional Laplacian is the first ordinary derivative of a fractional flux.

Exercise

4.10.1 Duplicate Figure 4.10.1 for the first moment of the Gaussian distribution, $f(x) = x \exp(-x^2)$.

Chapter 5

Complex variables

The invention of the imaginary unit, i, defined by a seemingly nonsensical equation, $\mathrm{i}^2 = -1$, has allowed us to formulate and solve a variety of problems in engineering, including the derivation of the quadratic formula for the roots of a quadratic equation with negative discriminant, the computation of seemingly intractable integrals, and many others.

A host of new results have been derived, known properties and procedures have been recast into compact and elegant forms, and efficient computational techniques have been developed.

Most important, the introduction of the imaginary unit allows us to define a complex variable, z, and then consider functions of a complex variable, $f(z)$, that enjoy a variety of interesting properties. Regular and fractional derivatives can be defined. A selective overview is given in this chapter.

5.1 Complex numbers

The imaginary unit, i, is defined by the equation

$$\mathrm{i}^2 = -1. \tag{5.1.1}$$

Note that the imaginary unit is a square root of a negative number,

$$\mathrm{i} = \pm\sqrt{-1}, \tag{5.1.2}$$

hence the qualifier *imaginary*, which could also be rephrased as *unreal*.

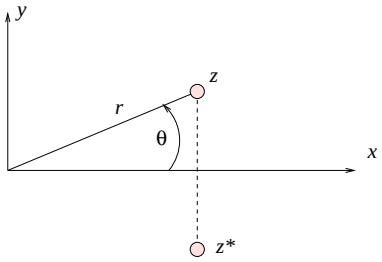


FIGURE 5.1.1 A complex number, z, is represented by a point in the xy plane, where x is the real axis and y is the imaginary axis. The complex conjugate of this number is z^*.

Consider two real numbers or variables, x and y, and introduce a corresponding complex number or variable, z, consisting of a real and an imaginary part,

$$z = x + \mathrm{i}\, y, \tag{5.1.3}$$

This number is represented by a point in the xy plane with coordinates x and y, as shown in Figure 5.1.1.

5.1.1 Polar representation

The distance of a complex number, z, from the origin is

$$r = \sqrt{x^2 + y^2}, \tag{5.1.4}$$

and the associated polar angle, θ, is computed from the equations

$$\tan\theta = y/x, \qquad \cos\theta = x/r, \qquad \sin\theta = y/r, \tag{5.1.5}$$

where θ is usually defined in the interval $(-\pi, \pi]$. Conversely,

$$x = r\, \cos\theta, \qquad y = r\, \sin\theta. \tag{5.1.6}$$

Care must be exercised in evaluating inverse trigonometric functions to prevent unwanted jumps.

The distance from the origin, r, is called the *magnitude* or *modulus* of z, and the angle θ is called the *phase* or *argument* of z, and is usually denoted by $\theta \equiv \arg(z)$.

5.1.2 *Quadratic formula*

Consider the quadratic equation for a variable, z,

$$a\,z^2 + b\,z + c = 0, \tag{5.1.7}$$

where a, b, c are three real numbers. To find the solution, we complete the square by expressing the equation as

$$a\,(z+p)^2 = q. \tag{5.1.8}$$

where p and q are defined in terms of a, b, and c. Expanding the square, we find that

$$2ap = b, \qquad ap^2 - q = c. \tag{5.1.9}$$

The solution of (5.1.8) is

$$z = -p \pm \sqrt{\frac{q}{a}}. \tag{5.1.10}$$

Solving for p and q, we obtain the quadratic formula for the roots,

$$z = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}. \tag{5.1.11}$$

When the discriminant is negative, $\mathcal{D} \equiv b^2 - 4ac < 0$, the roots are imaginary, given by

$$z = \frac{-b \pm \mathrm{i}\sqrt{-b^2 + 4ac}}{2a}. \tag{5.1.12}$$

In the absence of the imaginary unit, the roots would have existed only for a non-negative discriminant.

We see that the imaginary unit provides us with a vehicle for performing *imaginary* calculations and manipulations, and then becoming *real* in the end.

5.1.3 Addition and multiplication

The sum of two complex numbers, z_1 and z_2, is another complex number given by

$$z_1 \pm z_2 = x_1 \pm x_2 + \mathrm{i}\,(y_1 \pm y_2). \tag{5.1.13}$$

The product of two complex numbers, z_1 and z_2, is another complex number given by

$$z_1 z_2 = z_2 z_1 = x_1 x_2 - y_1 y_2 + \mathrm{i}\,(x_1 y_2 + y_1 x_2). \tag{5.1.14}$$

These expressions arise by carrying out the addition, subtraction, or multiplication of $x_1 + \mathrm{i}y_1$ and $x_2 + \mathrm{i}y_2$ in a distributive fashion, and recalling that $\mathrm{i}^2 = -1$.

Complex numbers enjoy the usual associative and distributive properties with regard to multiplication. For example,

$$z_1(z_2 + z_2) = z_1 z_2 + z_1 z_3, \tag{5.1.15}$$

where z_1, z_2, and z_3 are any three complex numbers.

Equation (5.1.14) appears to imply that four real products must be evaluated in order to compute one complex product, including $x_1 x_2$, $y_1 y_2$, $x_1 y_2$, and $y_1 x_2$. However, Gauss showed that we may write

$$z_1 z_2 = x_1 x_2 - y_1 y_2 + \mathrm{i}\,(x_1 + y_1)(x_2 + y_2) - \mathrm{i}\,(x_1 x_2 + y_1 y_2), \tag{5.1.16}$$

which involves three real products. The savings are significant when the computational cost of multiplication is an important consideration.

5.1.4 Complex conjugate

The conjugate of a complex number, z, indicated by an asterisk, is defined as

$$z^* = x - \mathrm{i}\,y. \tag{5.1.17}$$

Graphically, the complex conjugate, z^*, is the reflection of z with respect to the x axis, as shown in Figure 5.1.1. We see that

$$z z^* = x^2 + y^2 = r^2. \tag{5.1.18}$$

Combining equations (5.1.3) and (5.1.17), we find that

$$x = \tfrac{1}{2}\,(z + z^*), \qquad y = \tfrac{1}{2}\,\mathrm{i}\,(z^* - z). \tag{5.1.19}$$

5.1.5 *Inverse*

The reciprocal (inverse) of a complex number is given by

$$z^{-1} = \frac{1}{z} = \frac{1}{x + \mathrm{i}y} = \frac{1}{x^2 + y^2}\,(x - \mathrm{i}y) = \frac{1}{x^2 + y^2}\,z^*. \tag{5.1.20}$$

We have found that

$$\frac{1}{z} = \frac{1}{x^2 + y^2}\,(x - \mathrm{i}\,y) = \frac{z^*}{r^2}. \tag{5.1.21}$$

To divide two complex numbers, we multiply the first number by the reciprocal of the second.

Exercise

5.1.1 Derive an expression for ratio of two complex numbers, z_1/z_2.

5.2 *Functions of a complex variable*

Consider a complex function in the xy plane consisting of a real part (R) and an imaginary part (I),

$$f(x, y) = f_{\mathrm{R}}(x, y) + \mathrm{i}\,f_{\mathrm{I}}(x, y), \tag{5.2.1}$$

where i is the imaginary unit, $\mathrm{i}^2 = -1$. The real part, f_{R}, and the imaginary part, f_{I}, are both real functions that can be plotted separately and individually in a three-dimensional graph over the xy plane.

An example is the function

$$f(x, y) = x^2 - y^2 + \mathrm{i}\,2xy. \tag{5.2.2}$$

The real and imaginary parts of this function are

$$f_{\mathrm{R}} = x^2 - y^2, \qquad f_{\mathrm{I}} = 2xy. \tag{5.2.3}$$

There is no obvious relation between the real and imaginary parts.

Substituting expressions (5.1.19) into $f(x,y)$, we obtain a new function, $\phi(z,z^*)$, such that

$$f(x,y) = \phi(z,z^*), \tag{5.2.4}$$

where an asterisk denotes the complex conjugate. If z^* does not appear in the arguments of the function ϕ, then, not surprisingly, $f(x,y)$ is a function of the complex variable z.

For example, with reference to (5.2.2), we find that $\phi(z) = z^2$. The function $\phi(z^*) = z^*$ is *not* a function of a complex variable, and neither is the function $\phi(z,z^*) = |z| = \sqrt{zz^*}$.

We can take partial derivatives of $\phi(z,z^*)$ with respect to z or z^* keeping the other constant. Using the differentiation chain rule, we find that

$$\frac{\partial\phi}{\partial z} = \tfrac{1}{2}\Big(\frac{\partial f}{\partial x} - \mathrm{i}\,\frac{\partial f}{\partial y}\Big), \qquad \frac{\partial\phi}{\partial z^*} = \tfrac{1}{2}\Big(\frac{\partial f}{\partial x} + \mathrm{i}\,\frac{\partial f}{\partial y}\Big). \tag{5.2.5}$$

If z^* does not appear in the function ϕ, then, the derivative with respect to z^* is identically zero.

5.2.1 Cauchy–Riemann equations

It is evident from expressions (5.2.5) that ϕ depends on z, but not on z^*, that is, $\partial\phi/\partial z^* = 0$, if and only if (iff)

$$\frac{\partial f}{\partial x} = -\mathrm{i}\,\frac{\partial f}{\partial y}. \tag{5.2.6}$$

The real and imaginary parts of this equation are the Cauchy–Riemann equations,

$$\frac{\partial f_{\mathrm{R}}}{\partial x} = \frac{\partial f_{\mathrm{I}}}{\partial y}, \qquad \frac{\partial f_{\mathrm{R}}}{\partial y} = -\frac{\partial f_{\mathrm{I}}}{\partial x}. \tag{5.2.7}$$

These equations define uniquely f_{I} from f_{R}, and *vice versa*, within an arbitrary additive constant. If we know $f_{\mathrm{R}}(x,y)$, then we can uniquely

identify $f_{\mathrm{I}}(x,u)$, and *vice versa*, by integrating the Cauchy–Riemann equations or by using the Kramers–Kronig relations.

A necessary and sufficient condition for a complex function to be an exclusive function of a complex variable z is that its real part, $f_{\mathrm{R}}(x,y)$, and imaginary part, $f_{\mathrm{I}}(x,y)$, satisfy the Cauchy–Riemann equations,

5.2.2 *Power function*

As an example, we consider the nth power function of a complex variable

$$f(z) = z^n = (x + \mathrm{i}\, y)^n = \sum_{m=0}^{n} \binom{n}{m} x^{n-m} (\mathrm{i}y)^m, \tag{5.2.8}$$

where n is an integer and the tall parentheses denote the binomial coefficient. The real and imaginary parts of this function satisfy the Cauchy–Riemann equations.

5.2.3 *Exponential function*

The exponential function is defined in terms of an infinite series,

$$\mathrm{e}^z = 1 + z + \tfrac{1}{2} z^2 + \cdots + \frac{1}{n!} z^n + \cdots , \tag{5.2.9}$$

where $n!$ is the factorial. Based in this definition, we deduce that

$$\mathrm{e}^{z_1} \mathrm{e}^{z_2} = \mathrm{e}^{z_1 + z_2}, \tag{5.2.10}$$

and thus

$$(\mathrm{e}^{z_1})^{z_2} = \mathrm{e}^{z_1 z_2}. \tag{5.2.11}$$

5.2.4 *Logarithmic function*

The logarithmic function is defined such that

$$\ln \mathrm{e}^z = \mathrm{e}^{\ln z} = z. \tag{5.2.12}$$

If $z = r\mathrm{e}^{\mathrm{i}\theta}$, then

$$\ln z = \ln r + \mathrm{i}\, \theta. \tag{5.2.13}$$

Since the argument, θ, is multivalued, a branch cut must be specified.

5.2.5 Trigonometric functions

Trigonometric functions can be introduced in terms of the exponential function

$$\sin z = -\mathrm{i}\,\frac{1}{2}\,(\mathrm{e}^{\mathrm{i}z} - \mathrm{e}^{-\mathrm{i}z}) \tag{5.2.14}$$

and

$$\cos z = \frac{1}{2}\,(\mathrm{e}^{\mathrm{i}z} + \mathrm{e}^{-\mathrm{i}z}). \tag{5.2.15}$$

Hyperbolic functions are defined as

$$\sinh z = \frac{1}{2}\,(\mathrm{e}^{z} - \mathrm{e}^{-z}) \tag{5.2.16}$$

and

$$\cosh z = \frac{1}{2}\,(\mathrm{e}^{z} + \mathrm{e}^{-z}). \tag{5.2.17}$$

The definitions of the hyperbolic functions are consistent with those for z real.

5.2.6 Euler's formula

Euler's formula provides us with an expression for the complex exponential,

$$\mathrm{e}^{z} = \mathrm{e}^{x+\mathrm{i}y} = e^{x}\,e^{\mathrm{i}y} = e^{x}\,(\cos y + \mathrm{i}\sin y). \tag{5.2.18}$$

To show that

$$e^{\mathrm{i}y} = \cos y + \mathrm{i}\sin y, \tag{5.2.19}$$

we write the Taylor series expansion of the left-hand side, separate the real from the imaginary part, and refer to the Taylor series expansions of the sine and cosine.

For example, setting $z = \mathrm{i}\pi$, that is, $x = 0$ and $y = \pi$, and recalling that $e^{0} = 1$, we obtain

$$\mathrm{e}^{\mathrm{i}\pi} = -1. \tag{5.2.20}$$

Using (5.1.6), we derive the polar representation of an arbitrary complex number,

$$z = r\ \mathrm{e}^{\mathrm{i}\theta}. \tag{5.2.21}$$

The complex conjugate is

$$z^* = r\,\mathrm{e}^{-\mathrm{i}\theta}. \tag{5.2.22}$$

If z_1 and z_2 are two complex numbers, then

$$z_1 z_2 = r_1 r_2\,\mathrm{e}^{\mathrm{i}(\theta_1+\theta_2)}. \tag{5.2.23}$$

Consequently,

$$z\,\mathrm{e}^{\mathrm{i}\varphi} = r\ \mathrm{e}^{\mathrm{i}(\theta+\varphi)}, \tag{5.2.24}$$

which shows that multiplying a complex number by a unit complex number, $\mathrm{e}^{\mathrm{i}\varphi}$, rotates the number about the origin by φ.

Moreover, we find that

$$z^n = r^n\,\mathrm{e}^{\mathrm{i}n\theta} = r^n\ (\cos(n\theta) + \mathrm{i}\sin(n\theta)). \tag{5.2.25}$$

We see that raising a complex number to a power amounts to raising the modulus and multiplying the argument. However, we may also write

$$z^n = r^n\,\mathrm{e}^{\mathrm{i}n(\theta+2m\pi)}, \tag{5.2.26}$$

where m is an integer, which is equivalent to

$$z^n = r^n\ \big(\ \cos(n\theta + 2nm\pi) + \mathrm{i}\sin(n\theta + 2nm\pi)\ \big), \tag{5.2.27}$$

which indicates that z^n may not be uniquely defined.

Using Euler's formula, we find that

$$\cos\varphi = \tfrac{1}{2}\,(\mathrm{e}^{i\varphi} + \mathrm{e}^{-i\varphi}) \tag{5.2.28}$$

and

$$\sin\varphi = -\mathrm{i}\,\tfrac{1}{2}\,(\mathrm{e}^{i\varphi} - \mathrm{e}^{-i\varphi}), \tag{5.2.29}$$

for any real φ.

5.2.7 Roots of a complex number

Replacing n with $1/n$ in (5.2.25), we obtain the nth root of z,

$$z^{1/n} = r^{1/n}\,\exp(\,\mathrm{i}\,\frac{\theta + 2m\pi}{n}) \tag{5.2.30}$$

or

$$z^{1/n} = r^{1/n}\,\big(\cos\frac{\theta + 2m\pi}{n} + \mathrm{i}\sin\frac{\theta + 2m\pi}{n}\big) \tag{5.2.31}$$

for $m = 0, \pm 1, \ldots$. The principal nth root corresponds to $m = 0$. When n is an integer, we obtain n roots for $m = 0, 1, \ldots, n-1$.

5.2.8 Harmonic functions

If the Cauchy–Riemann equations (5.2.7) are satisfied, then f_{R} and f_{I} are harmonic functions, that is,

$$\frac{\partial^2 f_{\mathrm{R}}}{\partial x^2} + \frac{\partial^2 f_{\mathrm{R}}}{\partial y^2} = 0, \qquad \frac{\partial^2 f_{\mathrm{I}}}{\partial x^2} + \frac{\partial^2 f_{\mathrm{I}}}{\partial y^2} = 0. \tag{5.2.32}$$

In concise notation,

$$\nabla^2 f_{\mathrm{R}} = 0, \qquad \nabla^2 f_{\mathrm{I}} = 0, \tag{5.2.33}$$

where the Laplacian, ∇^2, is the sum of second derivatives with respect to x and y. The proof follows by straightforward partial differentiation of the Cauchy–Riemann equations with respect to x or y.

Conversely, any real function that satisfies Laplace's equation can be regarded as the real or imaginary part of a function of a complex variable.

5.2.9 Biharmonic functions

The biharmonic equation for a function, ψ, reads

$$\nabla^4 \psi = 0, \tag{5.2.34}$$

where $\nabla^4 = \nabla^2 \nabla^2$ is the biharmonic operator and ∇^2 is the Laplacian operator. Writing

$$\nabla^2 \psi = \frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2}, \tag{5.2.35}$$

we obtain

$$\nabla^4 \psi = \frac{\partial^4 \psi}{\partial^4 x} + 2\,\frac{\partial^4 \psi}{\partial^2 x \partial y^2} + \frac{\partial^4 \psi}{\partial^4 y}. \tag{5.2.36}$$

It can be shown that the general solution of this equation is given by

$$\psi = \mathrm{imag}\big(\, z^* \, f(z) + g(z) \,\big), \tag{5.2.37}$$

where $f(z)$ and $g(z)$ are two complex functions of z.

5.2.10 Complex derivative

The complex derivative of a function of a complex variable, $f(z)$, is given by

$$\frac{\mathrm{d}f}{\mathrm{d}z} = \frac{\partial f_\mathrm{R}}{\partial x} + \mathrm{i}\,\frac{\partial f_\mathrm{I}}{\partial x} = \frac{\partial f_\mathrm{I}}{\partial y} - \mathrm{i}\,\frac{\partial f_\mathrm{R}}{\partial y}, \tag{5.2.38}$$

where

$$\frac{\mathrm{d}f}{\mathrm{d}z} = \lim_{\Delta z \to 0} \frac{f(z + \Delta z)}{\Delta z}, \tag{5.2.39}$$

and the orientation of the small increment Δz in the complex xy plane is arbitrary. This means that

$$\frac{\mathrm{d}f}{\mathrm{d}z} = \frac{1}{1 + \mathrm{i}\,\alpha} \lim_{\Delta x \to 0} \frac{f(z + \Delta x\,(1 + \mathrm{i}\,\alpha)) - f(z)}{\Delta x}, \tag{5.2.40}$$

independent of the choice of any real number, α.

For example, for $f(z) = z^2$, we find

$$\frac{\mathrm{d}z^2}{\mathrm{d}z} = \frac{1}{1+\mathrm{i}\,\alpha} \lim_{\Delta x \to 0} \frac{\big(z + \Delta x\,(1+\mathrm{i}\,\alpha)\big)^2 - z^2}{\Delta x} = 2z, \quad (5.2.41)$$

independent of α.

5.2.11 *Derivative of the logarithmic function*

The complex derivative of the logarithmic function, $f(z) = \ln z$, is

$$\frac{\mathrm{d}\ln z}{\mathrm{d}z} = \frac{1}{1+\mathrm{i}\,\alpha} \lim_{\Delta x \to 0} \frac{\ln\big(z + (1+\alpha\mathrm{i})\,\Delta x\big) - \ln z}{\Delta x}. \quad (5.2.42)$$

The numerator on the right-hand side can be resolved into real and imaginary parts,

$$\mathcal{D} \equiv \ln\big(z + (1+\alpha\mathrm{i})\,\Delta x\big) - \ln z = \mathcal{D}_{\mathrm{R}} + \mathrm{i}\,\mathcal{D}_{\mathrm{I}}, \quad (5.2.43)$$

where

$$\mathcal{D}_{\mathrm{R}} = \ln\sqrt{(x+\Delta x)^2 + (y+\alpha\Delta x)^2} - \ln\sqrt{x^2+y^2} \quad (5.2.44)$$

and

$$\mathcal{D}_{\mathrm{I}} = \arctan\frac{y+\alpha\Delta x}{x+\Delta x} - \arctan\frac{y}{x}. \quad (5.2.45)$$

Rearranging and linearizing, we obtain

$$\mathcal{D}_{\mathrm{R}} = \tfrac{1}{2}\ln\Big(1 + 2\,\frac{x+\alpha y}{x^2+y^2}\,\Delta x\Big) \simeq \frac{x+\alpha y}{x^2+y^2}\,\Delta x \quad (5.2.46)$$

and

$$\mathcal{D}_{\mathrm{I}} = \arctan\Big(\frac{y}{x} + \frac{1}{x}\,\big(\alpha - \frac{y}{x}\big)\Delta x\Big) - \arctan\frac{y}{x}. \quad (5.2.47)$$

Recalling that $\mathrm{d}\arctan(w)/\mathrm{d}w = 1/(1+w^2)$, we obtain

$$\mathcal{D}_{\mathrm{I}} = \frac{x^2}{x^2+y^2}\,\frac{1}{x}\,\big(\alpha - \frac{y}{x}\big)\,\Delta x = \frac{\alpha x - y}{x^2+y^2}\,\Delta x. \quad (5.2.48)$$

Combining these results, we find that

$$\mathcal{D} = \frac{x - \mathrm{i}y + \alpha y + \mathrm{i}\alpha x}{x^2 + y^2}\,\Delta x \simeq \frac{(1 + \mathrm{i}\,\alpha)(x - \mathrm{i}y)}{x^2 + y^2}\,\Delta x. \quad (5.2.49)$$

Substituting this final expression into (5.2.42), we obtain a familiar formula,

$$\frac{\mathrm{d}\ln z}{\mathrm{d}z} = \frac{1}{z}, \quad (5.2.50)$$

which is identical to that for a function of a real variable.

Exercise

5.2.1 Compute the derivative of the function $f(z) = a^z$, where a is a complex number.

5.3 Integral of a complex function

The indefinite integral of a function of a complex variable, $f(z)$, is another function, $F(z)$, defined such that

$$\frac{\mathrm{d}F}{\mathrm{d}z} = f(z). \quad (5.3.1)$$

As in the case of a function of a real variable, we may write

$$F(z) = \int f(z)\,\mathrm{d}z. \quad (5.3.2)$$

The indefinite integral is also known as the anti-derivative.

For example, since $\mathrm{d}\ln z/\mathrm{d}z = 1/z$,

$$\int \frac{1}{z}\,\mathrm{d}z = \ln z, \quad (5.3.3)$$

where the logarithmic function is defined in (5.2.13). An arbitrary constant could be added to the right-hand side of (5.3.3).

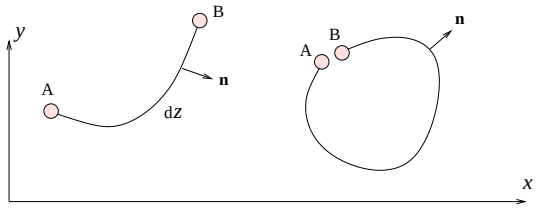


FIGURE 5.3.1 Typical integration paths in the complex plane defined by end points A and B.

5.3.1 Definite integral along a path

The integral of a function, $f(z)$, along an arbitrary integration path connecting two points in the complex plane, $z_{\rm A}$ and $z_{\rm B}$, as shown in Figure 5.3.1, is given by

$$\int_{z_{\rm A}}^{z_{\rm B}} f(z)\,{\rm d}z = F(z_B) - F(z_A). \tag{5.3.4}$$

The definite integral is independent of the shape of the path, so long as the path starts at $z_{\rm A}$ and ends at $z_{\rm B}$.

Now writing ${\rm d}z = {\rm d}x + {\rm i}\,{\rm d}y$, and resolving $f(z)$ into its real and imaginary parts, we find that the path integral is given by

$$\int_{z_{\rm A}}^{z_{\rm B}} f(z)\,{\rm d}z = \int_{z_{\rm A}}^{z_{\rm B}} (f_{\rm R}(z) + {\rm i}\,f_{\rm I}(z)\,)\,({\rm d}x + {\rm i}\,{\rm d}y). \tag{5.3.5}$$

Separating the real from the imaginary parts, we obtain

$$\begin{aligned}&\int_{z_{\rm A}}^{z_{\rm B}} f(z)\,{\rm d}z \qquad\qquad (5.3.6)\\ &\quad = \int_{z_{\rm A}}^{z_{\rm B}} (f_{\rm R}(z)\,{\rm d}x - f_{\rm I}(z)\,{\rm d}y) + {\rm i}\int_{z_{\rm A}}^{z_{\rm B}} (f_{\rm R}(z)\,{\rm d}y + f_{\rm I}(z)\,{\rm d}x).\end{aligned}$$

We see that the computation of a complex integral along a path requires the computation of two real integrals.

If $\mathrm{d}\ell$ is the differential arc length along the integration path, then the components of the unit vector normal to the integration path, $\mathbf{n}$, are

$$n_x = \frac{\mathrm{d}y}{\mathrm{d}\ell}, \qquad n_y = -\frac{\mathrm{d}x}{\mathrm{d}\ell}, \tag{5.3.7}$$

as shown in Figure 5.3.1. Consequently, we may write

$$\int_{z_\mathrm{A}}^{z_\mathrm{B}} f(z)\,\mathrm{d}z = -\int_{z_\mathrm{A}}^{z_\mathrm{B}} \mathbf{q}\cdot\mathbf{n}\,\mathrm{d}\ell + \mathrm{i}\int_{z_\mathrm{A}}^{z_\mathrm{B}} \mathbf{r}\cdot\mathbf{n}\,\mathrm{d}\ell, \tag{5.3.8}$$

where

$$\mathbf{q} = (f_\mathrm{I}, f_\mathrm{R}), \qquad \mathbf{r} = (f_\mathrm{R}, -f_\mathrm{I}) \tag{5.3.9}$$

are two orthogonal real vector functions, $\mathbf{q}\cdot\mathbf{r} = 0$. Using the Cauchy–Riemann equations, we find that

$$\boldsymbol{\nabla}\cdot\mathbf{q} = 0, \qquad \boldsymbol{\nabla}\cdot\mathbf{r} = 0, \tag{5.3.10}$$

which means that the functions $\mathbf{q}$ and $\mathbf{r}$ are solenoidal.

5.3.2 *Integral around a closed loop*

If the integration path is closed, which means that the end points of the integration path, z_A and z_B, coincide, we may use the Gauss divergence theorem to write

$$\oint f(z)\,\mathrm{d}z = \iint (-\boldsymbol{\nabla}\cdot\mathbf{q} + \mathrm{i}\,\boldsymbol{\nabla}\cdot\mathbf{r})\,\mathrm{d}x\,\mathrm{d}y, \tag{5.3.11}$$

where the areal integral on the right-hand side is computed over the interior of the closed integration path, and the integration is performed in the counterclockwise direction around the path. Using equations (5.3.10) to set the right-hand side to zero, we find that

$$\oint f(z)\,\mathrm{d}z = 0, \tag{5.3.12}$$

provided that the Cauchy–Riemann equations are satisfied throughout the interior of the closed integration path. Equation (5.3.12) expresses the Cauchy integral theorem.

5.3.3 Logarithmic function

Consider he logarithmic function, $f(z) = \ln z$. Because the Cauchy–Riemann equations are not satisfied at the origin, the logarithmic function does not satisfy the conditions for (5.3.12) when the closed integration path encloses the origin of the complex plane.

Invoking the indefinite integral shown in (5.3.3) and the definition of the logarithmic function, we find that

$$\oint \frac{1}{z}\,\mathrm{d}z = 2\pi\mathrm{i}, \tag{5.3.13}$$

where the closed path contour encloses the origin.

By extension, we find that

$$\oint \frac{1}{z-z_0}\,\mathrm{d}z = 2\pi\mathrm{i}, \tag{5.3.14}$$

where the closed integration contour encloses an arbitrary point, z_0. Because the function $1/(z-z_0)$ has a pole singularity at z_0, the conditions underlying (5.3.12) are not satisfied.

5.3.4 Cauchy integral formula

Assume that a complex function, $f(z)$, is differentiable throughout the interior of a closed integration path enclosing an arbitrary point, z_0, and write

$$\oint \frac{f(z)}{z-z_0}\,\mathrm{d}z = \oint \frac{f(z)-f(z_0)}{z-z_0}\,\mathrm{d}z + f(z_0)\oint \frac{1}{z-z_0}\,\mathrm{d}z. \tag{5.3.15}$$

The first integral on the right-hand is zero, thanks to its differentiable integrand. The second integral on the right-hand is equal to $2\pi\mathrm{i}$ because of (5.3.14). Rearranging, we obtain the Cauchy integral formula

$$f(z_0) = \frac{1}{2\pi\mathrm{i}}\oint \frac{f(z)}{z-z_0}\,\mathrm{d}z. \tag{5.3.16}$$

The argument of the difference $z-z_0$ changes by 2π as a turn is completed along the closed integration contour. This remarkable formula

can be used to compute otherwise intractable definite integrals by a wise choice if the closed integration path.

When the integration path is chosen to be a circle of radius a centered at z_0, we may write

$$z - z_0 = a\,\mathrm{e}^{\mathrm{i}\theta}, \qquad \mathrm{d}z = \mathrm{i}\,a\,\mathrm{e}^{\mathrm{i}\theta}\,\mathrm{d}\theta, \tag{5.3.17}$$

where θ is the polar angle around z_0. Substituting into (5.3.16), we obtain

$$f(z_0) = \frac{1}{2\pi}\int_0^{2\pi} f(z_0 + a\mathrm{e}^{\mathrm{i}\theta})\,\mathrm{d}\theta, \tag{5.3.18}$$

which identifies $f(z_0)$ with the mean value around the circle.

Exercise

5.3.1 Confirm (5.3.18) for the function $f(z) = \cos z$ at the origin, $z_0 = 0$.

5.4 Cauchy differentiation formula

Applying Cauchy's equation (5.3.16) for the first derivative of a function $f(z)$, denoted by a prime, we obtain

$$f'(z_0) = \frac{1}{2\pi\mathrm{i}}\oint \frac{1}{z - z_0}\,f'(z)\,\mathrm{d}z. \tag{5.4.1}$$

Integrating by parts, we obtain

$$f'(z_0) = -\frac{1}{2\pi\mathrm{i}}\oint f(z)\,\frac{\mathrm{d}}{\mathrm{d}z}\Big(\frac{1}{z - z_0}\Big)\,\mathrm{d}z. \tag{5.4.2}$$

Carrying out the differentiation, we obtain

$$f'(z_0) = \frac{1}{2\pi\mathrm{i}}\oint \frac{f(z)}{(z - z_0)^2}\,\mathrm{d}z. \tag{5.4.3}$$

In fact, this formula can be derived directly by differentiating (5.4.1) with respect to z_0 and transposing differentiation with integration.

5.4.1 nth derivative

Now considering the nth derivative, indicated by the superscript (n), we write

$$f^{(n)}(z_0) = \frac{1}{2\pi \mathrm{i}} \oint \frac{1}{z - z_0} f^{(n)}(z)\, \mathrm{d}z. \tag{5.4.4}$$

Integrating by parts n times, we obtain the Cauchy differentiation formula for the nth derivative,

$$f^{(n)}(z_0) = \frac{n!}{2\pi \mathrm{i}} \oint \frac{f(z)}{(z - z_0)^{n+1}}\, \mathrm{d}z, \tag{5.4.5}$$

where $n\,! \equiv 1 \cdot 2 \cdots n = \Gamma(n+1)$ is the factorial and $\Gamma(p)$ is the Gamma function.

Cauchy's formula (5.4.5) can be expressed in the alternative form

$$f^{(n)}(z_0) = (-1)^{n+1} \frac{n\,!}{2\pi \mathrm{i}} \oint \frac{f(z)}{(z_0 - z)^{n+1}}\, \mathrm{d}z, \tag{5.4.6}$$

where n is zero or an integer. The argument of the difference $z_0 - z$ changes by 2π as one turn is completed along the closed integration contour.

5.4.2 Confirmation by code

Cauchy's formula (5.4.5) can be confirmed by performing the integration numerically around an arbitrary closed contour described by a sequence of contour points.

As a preliminary, we introduce a function named *fncc*, located in directory CAUCHY of SEMLIB, that receives z and returns the value $f(z)$ and its first few derivatives offered through an integer menu:

```
function  [f,fp,fpp,fppp] = fncc(z,menu)

%=======================================
% evaluation of a complex function, f,
% its first, second, and third derivatives
```

```
%
% all functions implemented are arbitrary
%=========================================

  im = sqrt(-1);

  if(menu==1)

   dz = z - 2.3 - 3.2*im;
   f = 1/dz;
   fp = -1/dz^2;
   fpp = 2/dz^3;
   fppp =-6/dz^4;

  elseif(menu==2)

   a = 0.48;

   f = sin(a*z);
   fp = a*cos(a*z);
   fpp = -a*a*f;
   fppp = -a*a*a*fp;

  elseif(menu==3)

   f = z^3;
   fp = 3.0*z^2;
   fpp = 6.0*z;
   fppp = 6.0;

  elseif(menu==4)

   a = 1.0;
   f = exp(a*z);
   fp = a*exp(a*z);
   fpp = a*a*exp(a*z);
   fppp = a*a*a*exp(a*z);

  elseif(menu==5)
```

```
   t = exp(-z^2);
   f = t;
   fp = -2*z*t;
   fpp = 2*(2*z^2-1)*t;
   fppp = 4*z*(3-2*z^2)*t;

  elseif(menu==6)

   t = exp(-z^2);
   f = z*t;
   fp = (1-2.0*z^2)*t;
   fpp = 2.0*z*(2.0*z^2-3.0)*t;
   fppp = 2.0*(4.0*z^4-12.0*z^2+3.0)*t;

  end

return
```

The following Matlab code named *cauchy*, located in directory CAUCHY of SEMLIB, confirms the Cauchy differentiation formula by integrating around an arbitrary contour enclosing the evaluation point using the trapezoidal rule:

```
%===========
% confirm the Cauchy differentiation formula
%===========

n = 2; % differentiation order

N = 4*32; % points around the contour

x0 = 0.389;  % arbitrary
y0 =-1.2; % arbitrary

menu = 1;

%---
% prepare
%---
```

```
im = sqrt(-1);

z0 = x0+im*y0;

%---
% generate contour nodes (arbitrary)
%---

dthet = 2*pi/N;

thetoff = 0.1234*pi;  % arbitrary

for i=1:N+1
 theta(i) = (i-1)*dthet + thetoff;
 r = 1.0 + 0.5*sin(theta(i))^2  ...
         - 1.3*sin(theta(i))^3;  % arbitrary
 x(i) = r*cos(theta(i));
 y(i) = r*sin(theta(i));
end

%---
% plot
%---

figure(1)
hold on
plot(x,y,'-ko');
plot(x(1),y(1),'-r*');
plot([x0,x(1)],[y0,y(1)],'--k');
plot(x0,y0,'ro');

%---
% contour node distance from z0
%---

for i=1:N+1
 Xr(i) = x(i)-x0;
 Yr(i) = y(i)-y0;
```

```
 Rr(i) = sqrt(Xr(i)^2+Yr(i)^2);
end

%---
% angles
%---

Tr(1) = acos(Xr(1)/Rr(1));

if(Yr(1)<0) Tr(1)=-Tr(1); end

for i=1:N
 crossprd = Xr(i)*Yr(i+1) - Xr(i+1)*Yr(i);
 den = Rr(i)*Rr(i+1);
 dt = asin(crossprd/den);
 innerprd = Xr(i)*Xr(i+1)+Yr(i)*Yr(i+1);
 if(innerprd<0)
  dt = dt*(pi/abs(dt)-1.0);
 end
 Tr(i+1) = Tr(i)+dt;
end

%---
% function and integrand evaluation
%---

for i=1:N+1
 invp(i) = exp(-(n+1)*im*Tr(i))/Rr(i)^(n+1);
 z(i) = x(i)+im*y(i);
 [f(i),fp(i),fpp(i),fppp(i)] = fncc(z(i),menu);
end

%---
% integrate by the trapezoidal rule
%---

integral = 0.0;

for i=1:N
```

```
 integral = integral + 0.5*(f(i)  *invp(i) ...
                           +f(i+1)*invp(i+1)) ...
                           *(z(i+1)-z(i));
end

integral = gamma(n+1)*integral/(2*pi*im);
integral

[f0,fp0,fpp0,fppp0] = fncc(z0,menu);
n
f0_fp0_fpp0_fppp0 = [f0;fp0;fpp0;fppp0]
```

The location of the coincident end points of the closed contour is determined by the numerical parameter `thetoff`. The results are independent of this parameter. The polar angle around the evaluation point, z_0, is computed incrementally by adding contributions to the value corresponding to the first contour point.

Running the code generates the graph shown in Figure 5.4.2, where the dashed line connects the evaluation point inside the closed contour to the first contour point, and displays the following output:

```
integral =

  -0.0503 + 0.1362i

n =
     2

f0_fp0_fpp0_fppp0 =

   0.2173 - 0.5978i
   0.5521 + 0.0542i
  -0.0501 + 0.1377i
  -0.0611 - 0.0060i
```

We see that the numerical result are consistent with the exact value for the second derivative, $f''(z_0)$. The small difference is attributed to numerical integration error.

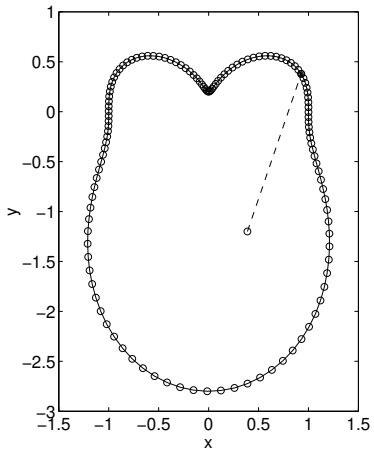


FIGURE 5.4.1 An arbitrary closed contour described by a sequence of nodes, is used to confirm the Cauchy differentiation formula at point, z_0, inside the contour. The dashed line connects the evaluation point, z_0, with the coincident end points of the closed integration contour. The integration is performed in the counterclockwise direction.

Exercise

5.4.1 Compute the fourth derivative of the function $f(z) = (3z + 5)/(6z + 7)$ at the point $z_0 = 1.4 + 3.9\mathrm{i}$ by contour integration using Cauchy's differentiation formula.

5.5 Fractional integrals and derivatives

Riemann–Liouville fractional integrals and derivatives can be defined in the complex plane working by analogy with corresponding integrals and derivatives on the real axis, as discussed in Chapter 4. In addition, Cauchy fractional derivatives can be defined based on the Cauchy differentiation formula discussed in Section 5.4.

We will see that the Riemann–Liouville and the Grünwald–Letnikov derivatives discussed in this section coincide with the Cauchy fractional derivative discussed in Section 5.4, subject to certain conventions.

5.5.1 Riemann–Liouville fractional integral

Working as in Section 4.1, we consider a function of a complex variable, $f(z)$, and introduce the Riemann–Liouville fractional integral of positive order, β, defined by the formula

$$[\mathrm{I}^{(\beta)} f]^{\mathrm{RL}}_{z_\mathrm{A}}(z_0) \equiv \frac{1}{\Gamma(\beta)} \int_{z_\mathrm{A}}^{z_0} \frac{f(z)}{(z_0 - z)^{1-\beta}} \, \mathrm{d}z, \tag{5.5.1}$$

where $\Gamma(\beta)$ is the gamma function and the integration path is a *straight line* in the complex plane connecting a specified fixed point, z_A, to the evaluation point, z_0.

Since $\beta > 0$, the integrand exhibits an integrable singularity and the fractional integral is well defined. Note the order of the difference $z_0 - z$ in the denominator.

5.5.2 Evaluation of the Riemann–Liouville fractional integral

To compute the integral in (5.5.1), we subtract out the singularity by writing

$$\begin{aligned} &\int_{z_\mathrm{A}}^{z_0} \frac{f(z)}{(z_0 - z)^{1-\beta}} \, \mathrm{d}z \\ &\qquad = \int_{z_\mathrm{A}}^{z} \frac{f(z) - f(z_0)}{(z_0 - z)^{1-\beta}} \, \mathrm{d}z + f(z_0) \int_{z_\mathrm{A}}^{z_0} \frac{\mathrm{d}z}{(z_0 - z)^{1-\beta}}. \end{aligned} \tag{5.5.2}$$

Using a Taylor series expansion, we write

$$f(z) - f(z_0) \simeq f'(z_0) \, (z - z_0), \tag{5.5.3}$$

and find that the first integral on the right-hand side of (5.5.2) is sufficiently regular to be computed by standard numerical methods, including the mid-point rule. The second integral can be evaluated analytically, and is given by

$$\int_{z_\mathrm{A}}^{z_0} \frac{\mathrm{d}z}{(z_0 - z)^{1-\beta}} = \frac{(z_0 - z_\mathrm{A})^\beta}{\beta} = \frac{|z_0 - z_\mathrm{A}|^\beta}{\beta} \, \mathrm{e}^{\mathrm{i}\beta\phi}, \tag{5.5.4}$$

where

$$\phi = \arg(z_0 - z_A), \tag{5.5.5}$$

under the stipulation that $0 \leq \phi < 2\pi$.

5.5.3 Computation of the Riemann–Liouville fractional integral

The following Matlab function named *rl_fint*, located in directory CAUCHY of SEMLIB, evaluates the Riemann–Liouville fractional integral using this method. The regularized integral on the right-hand side of (5.5.2) is computed by the mid-point rule:

```
function rli = rl_fint (bet,zA,z0,nint,menu)

%==============================
% compute the Riemann--Liouville
% integral along a straight path
%
% int_zA_z f(z) /(z0-z)^bet d z
%==============================

im = sqrt(-1);
nint = 4*8*64;

%---
% prepare
%---

x0 = real(z0);
y0 = imag(z0);

xA = real(zA);
yA = imag(zA);

[f0,fp0,fpp0,fppp0] = fncc(z0,nint,menu);

Dz = (z0-zA)/nint;

%---
% integrate by the mid-point rule
```

```
%---

intgrl = 0.0;

%---
for i=1:nint

 z(i) = zA + (i-0.5)*Dz;

 x(i)  = real(z(i));
 y(i)  = imag(z(i));
 Xr(i) = x0-x(i);
 Yr(i) = y0-y(i);
 Rr(i) = sqrt(Xr(i)^2+Yr(i)^2);

 [f(i),fp(i),fpp(i),fppp(i)] = fncc(z(i),menu);

 intgrl = intgrl + (f(i)-f0)/Rr(i)^(1-bet);

end
%---

ang = acos(Xr(1)/Rr(1));
if(Yr(1)<0) ang=2*pi-ang; end

rli = intgrl*exp(-im*(1-bet)*ang)*Dz;

d = sqrt((x0-xA)^2+(y0-yA)^2);

rli = rli + f0/bet * d^bet * exp(im*beta*ang);

rli = rli/gamma(bet);

%---
% done
%---

return
```

A call to the function *fncc* listed in Section 5.4 is made to evaluate $f(z)$ through a menu. To prevent confusion with the beta function, the parameter β is named `bet` in the code.

5.5.4 *Riemann–Liouville fractional derivative*

A fractional derivative of positive order, α, can be defined in terms of whole derivatives of the Riemann–Liouville fractional integral of order β introduced in (5.5.1), where $0 < \beta \leq 1$ and $z_{\rm A}$ is a specified point.

When $0 \leq \alpha < 1$, the Riemann–Liouville fractional derivative is defined as a first derivative,

$$[\mathrm{D}^{(\alpha)} f]^{\rm RL}_{z_{\rm A}}(z_0) \equiv \frac{\mathrm{d}}{\mathrm{d}z_0} \left[\mathrm{I}^{(1-\alpha)} f \right]_{z_{\rm A}}. \tag{5.5.6}$$

Making substitutions, we find that

$$[\mathrm{D}^{(\alpha)} f]^{\rm RL}_{z_{\rm A}}(z_0) = \frac{1}{\Gamma(1-\alpha)} \frac{\mathrm{d}}{\mathrm{d}z_0} \int_{x_{\rm A}}^{z_0} \frac{f(z)}{(z_0 - z)^{\alpha}} \, \mathrm{d}z. \tag{5.5.7}$$

The integration path is a straight line connecting the arbitrary point, $z_{\rm A}$, to the evaluation point, z_0.

When $1 \leq \alpha < 2$, the Riemann–Liouville fractional derivative is defined as a second derivative,

$$\left[\mathrm{D}^{(\alpha)} f\right]^{\rm RL}_{z_{\rm A}}(z_0) \equiv \frac{\mathrm{d}^2}{\mathrm{d}z_0^2} \left[\mathrm{I}^{(2-\alpha)} f \right]_{z_{\rm A}}. \tag{5.5.8}$$

Making substitutions, we find that

$$\left[\mathrm{D}^{(\alpha)} f\right]^{\rm RL}_{z_{\rm A}}(z_0) = \frac{1}{\Gamma(2-\alpha)} \frac{\mathrm{d}^2}{\mathrm{d}z_0^2} \int_{z_{\rm A}}^{x_0} \frac{f(z)}{(z_0 - z)^{\alpha-1}} \, \mathrm{d}z. \tag{5.5.9}$$

The integration path is a straight line connecting the arbitrary point, $z_{\rm A}$, to the evaluation point, z_0.

More generally, when $n - 1 \leq \alpha < n$, where $n = 1, 2, \ldots$, the Riemann–Liouville fractional derivative is defined as an nth derivative,

$$\left[\mathrm{D}^{(\alpha)} f\right]^{\rm RL}_{z_{\rm A}}(z_0) \equiv \frac{\mathrm{d}^n}{\mathrm{d}z_0^n} \, {}_{\rm RL}[\mathrm{I}^{(n-\alpha)} f]_{z_{\rm A}}. \tag{5.5.10}$$

Making substitutions, we find that

$$\left[\mathrm{D}^{(\alpha)} f\right]_{z_\mathrm{A}}^{\mathrm{RL}}(z_0) = \frac{1}{\Gamma(n-\alpha)} \frac{\mathrm{d}^n}{\mathrm{d}z_0^n} \int_{z_\mathrm{A}}^{z_0} \frac{f(z)}{(z_0 - z)^{\alpha+1-n}}\, \mathrm{d}z. \quad (5.5.11)$$

The integration path is a straight line in the complex plane connecting the arbitrary point z_A to the evaluation point z_0. Note that the exponent in the denominator of the integrand satisfies the inequality

$$0 \leq \alpha + 1 - n < 1, \quad (5.5.12)$$

which ensures the integrability of the integrand.

5.5.5 *Computation of the Riemann–Liouville fractional derivative*

In practice, the derivatives in (5.5.11) with respect to z_0 can be computed by finite-difference approximations using a small real or complex numerical differentiation step.

The following Matlab function named *rl_fder*, located in directory CAUCHY of SEMLIB, evaluates the fractional derivative by numerical differentiation of the fractional integral using centered differences:

```
function rlip = rl_fder(alpha,zA,z0,eps,nint,menu)

%====================================
% computation of the Riemann-Liouville
% fractional derivative
%
% eps: numerical differentiation step
%====================================

if(alpha<0)

 bet = -alpha;
 rlip = riem_liouv_fint (bet,zA,z0,nint,menu);

elseif(alpha > 0 & alpha <= 1)

 bet = 1.0-alpha;
 rli_2 = rl_fint (bet,zA,z0+eps,nint,menu);
```

```
 rli_1 = rl_fint (bet,zA,z0-eps,nint,menu);
 rlip = (rli_2-rli_1)/(2*eps);

elseif(alpha > 1 & alpha <= 2)

 bet = 2.0-alpha;
 rli_0 = rl_fint (bet,zA,z0,nint,menu);
 rli_1 = rl_fint (bet,zA,z0-eps,nint,menu);
 rli_2 = rl_fint (bet,zA,z0+eps,nint,menu);
 rlip = (rli_2 - 2.0*rli_0 + rli_1)/eps^2;

elseif(alpha > 2 & alpha <= 3)

 bet = 3.0-alpha;
 rli_A = rl_fint (bet,zA,z0-2*eps,nint,menu);
 rli_1 = rl_fint (bet,zA,z0-eps,nint,menu);
 rli_2 = rl_fint (bet,zA,z0+eps,nint,menu);
 rli_B = rl_fint (bet,zA,z0+2*eps,nint,menu);
 rlip = (-rli_A + 2*rli_1 - 2*rli_2 + rli_B)/(2*eps^3);

else

 disp('rl_fder: this choice of alpha not yet implemented')

end

%---
% done
%---

return
```

A small yet arbitrary real, imaginary, or complex differentiation parameter, ϵ, named eps in the code, can be used in the centered difference approximations.

5.5.6 Grunwald–Letnikov fractional derivative

The Grünwald–Letnikov fractional derivative of a real function was discussed in Sections 4.6 and 4.7. In the case of a function of a complex

variable, the Grünwald–Letnikov fractional derivative is computed from the formula

$$[\,\mathrm{D}^{(\alpha)} f\,]^{\mathrm{GL}}_{z_\mathrm{A}}(z_0) \equiv \frac{1}{\Delta z^\alpha} \sum_{k=0}^{k_{\max}} \omega_{\alpha,k} f(z_0 - k\Delta z), \tag{5.5.13}$$

where $k_{\max}$ is a specified integer,

$$\Delta z = \frac{z_0 - z_\mathrm{A}}{k_{\max}} \tag{5.5.14}$$

is a complex increment whose argument lies in the range $[0, 2\pi)$ and

$$\omega_{\alpha,k} \equiv (-1)^k \binom{\alpha}{k} \tag{5.5.15}$$

are differentiation coefficients. We find that

$$\omega_{\alpha,k} = \frac{1}{\Gamma(-\alpha)} \frac{\Gamma(k-\alpha)}{\Gamma(k+1)} = (-1)^k \frac{\Gamma(\alpha+1)}{\Gamma(\alpha-k+1)\Gamma(k+1)}, \tag{5.5.16}$$

where Γ is the gamma function, and derive the recursion formula

$$\omega_{\alpha,k} = \frac{k-\alpha-1}{k}\, \omega_{\alpha,k-1}, \tag{5.5.17}$$

starting with $\omega_{\alpha,0} = 1$, as discussed in Section 10.6.

In the limit as $k_{\max}$ tends to infinity while z_A is kept fixed, the Grünwald–Letnikov fractional derivative reduces precisely to the Riemann–Liouville fractional derivative discussed earlier in this section.

5.5.7 Caputo fractional derivative

The Caputo fractional derivative is defined as

$$\big[\,\mathrm{D}^{(\alpha)} f\,\big]^{\mathrm{C}}_{z_\mathrm{A}}(z_0) \equiv \frac{1}{\Gamma(n-\alpha)} \int_{z_\mathrm{A}}^{z_0} \frac{f^{(n)}(z)}{(z_0 - z)^{1+\alpha-n}}\, \mathrm{d}z \tag{5.5.18}$$

for $n-1 \le \alpha < n$, where n is an integer. Note that $0 \le 1+\alpha-n < 1$, which ensures the integrability of the integrand. When $f^{(n)}(z) = 0$, the Caputo derivative is identically zero for $\alpha \ge n$.

The Caputo fractional derivative differs from the Riemann–Liouville and the Grünwald–Letnikov fractional derivatives discussed earlier in this section. For example, the Caputo fractional derivative of a constant function is zero, whereas the Riemann–Liouville fractional derivative of a constant function is not zero, as discussed in Chapter 4.

5.5.8 Right fractional integrals and derivatives

In addition to the fractional integrals and derivatives introduced in this section, corresponding right fractional derivatives can be defined, as discussed in Chapter 4. A summary is given in Table 5.5.1.

Exercise

5.5.1 Add one more option in function *rl_frac* the handle the case $2 \leq \alpha < 3$.

5.6 Cauchy fractional integrals and derivatives

Cauchy's differentiation formula (5.4.5) is repeated below for convenience,

$$f^{(n)}(z_0) = \frac{\Gamma(n+1)}{2\pi \mathrm{i}} \oint \frac{f(z)}{(z-z_0)^{n+1}} \, \mathrm{d}z. \tag{5.6.1}$$

Note the order $z - z_0$ in the denominator of the integrand, which can be contrasted with the order $z_0 - z$ that appears in the left Riemann–Liouville fractional integral and derivative but is consistent with the right Riemann–Liouville fractional integral and derivative discussed in Section 5.5.

It might appear that Cauchy's differentiation formula can be generalized simply by replacing the integer derivative order, n, with a positive non-integer derivative order, α where the closed integration contour is traced in a counterclockwise direction. However, for non-integer α, the contour integral on the right-hand side of (5.6.1) depends on our choice of the coincident closed contour integration end points, z_{A} and

(Left) Riemann–Liouville fractional integral:

$$[\,\mathrm{I}^{(\beta)} f\,]^{\mathrm{RL}}_{z_\mathrm{A}}(z_0) \equiv \frac{1}{\Gamma(\beta)} \int_{z_\mathrm{A}}^{z_0} \frac{f(z)}{(z_0 - z)^{1-\beta}}\,\mathrm{d}z \qquad \text{for} \qquad \beta > 0$$

Right Riemann–Liouville fractional integral:

$$[\,\mathrm{I}^{(\beta)} f\,]^{z_\mathrm{B}}_{\mathrm{RL}}(z_0) \equiv \frac{1}{\Gamma(\beta)} \int_{z_0}^{z_\mathrm{B}} \frac{f(z)}{(z - z_0)^{1-\beta}}\,\mathrm{d}z \qquad \text{for} \qquad \beta > 0$$

(Left) Riemann–Liouville fractional derivative:

$$\left[\mathrm{D}^{(\alpha)} f\right]^{\mathrm{RL}}_{z_\mathrm{A}}(z_0) = \frac{1}{\Gamma(n-\alpha)} \frac{\mathrm{d}^n}{\mathrm{d}z_0^n} \int_{z_\mathrm{A}}^{z_0} \frac{f(z)}{(z_0 - z)^{\alpha+1-n}}\,\mathrm{d}z$$

where $n - 1 \le \alpha < n$.

Right Riemann–Liouville fractional derivative:

$$\left[\mathrm{D}^{(\alpha)} f\right]^{z_\mathrm{B}}_{\mathrm{RL}}(z_0) = \frac{(-1)^n}{\Gamma(n-\alpha)} \frac{\mathrm{d}^n}{\mathrm{d}z_0^n} \int_{z_0}^{z_\mathrm{B}} \frac{f(z)}{(z - z_0)^{\alpha+1-n}}\,\mathrm{d}z$$

where $n - 1 \le \alpha < n$

TABLE 5.5.1 Fractional integrals and derivatives defined with respect to two arbitrary points in the complex plane, z_A and z_B, where z_0 is an evaluation point, $\beta > 0$. In all cases, the integration path is a straight line in the complex plane.

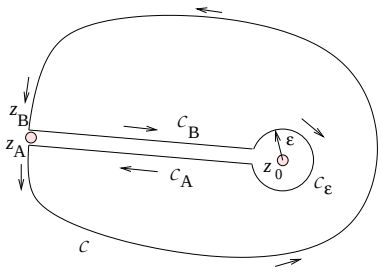


FIGURE 5.6.1 Integration path around a closed contour, $\mathcal{C}$, starting at $z_{\rm A}$ and ending at the coincident point, $z_{\rm B}$, using for evaluating the ordinary or fractional derivatives at an interior point, z_0.

$z_{\rm B}$, where, by convention,

$$-\pi \leq \arg(z_{\rm A} - z_0) < \pi, \tag{5.6.2}$$

as shown in Figure 5.6.1.

To be clear, we denote the Cauchy fractional derivative at a point, z_0, parametrized by the contour end point z_A, as

$$f^{(\alpha)}_{z_{\rm A}}(z_0) = \frac{\Gamma(\alpha+1)}{2\pi {\rm i}} \oint_{z_{\rm A}} \frac{f(z)}{(z-z_0)^{\alpha+1}}\, {\rm d}z, \tag{5.6.3}$$

for any real, positive or negative α. Even though the first end point $z_{\rm A}$ coincides with the last end point $z_{\rm B}$ on a closed contour, the integrand of (5.6.3) takes different values at $z_{\rm A}$ and $z_{\rm B}$, as shown in Figure 5.6.2. We find that

$$\frac{1}{(z_{\rm B}-z_0)^{\alpha+1}} - \frac{1}{(z_{\rm A}-z_0)^{\alpha+1}} = \frac{{\rm e}^{-{\rm i}(\alpha+1)(\phi+2\pi)} - {\rm e}^{-{\rm i}(\alpha+1)\phi}}{|z_{\rm A}-z_0|^{\alpha+1}}, \tag{5.6.4}$$

where $\phi = \arg(z_{\rm A} - z_0)$. Simplifying, we obtain

$$\Big[\, \frac{1}{(z-z_0)^{\alpha+1}} \,\Big]_{{\rm A}\to{\rm B}} = \frac{{\rm e}^{-2\pi\alpha{\rm i}} - 1}{(z_{\rm A}-z_0)^{\alpha+1}}, \tag{5.6.5}$$

where

$$[\, Q \,]_{{\rm A}\to{\rm B}} \equiv Q_{\rm B} - Q_{\rm A} \tag{5.6.6}$$

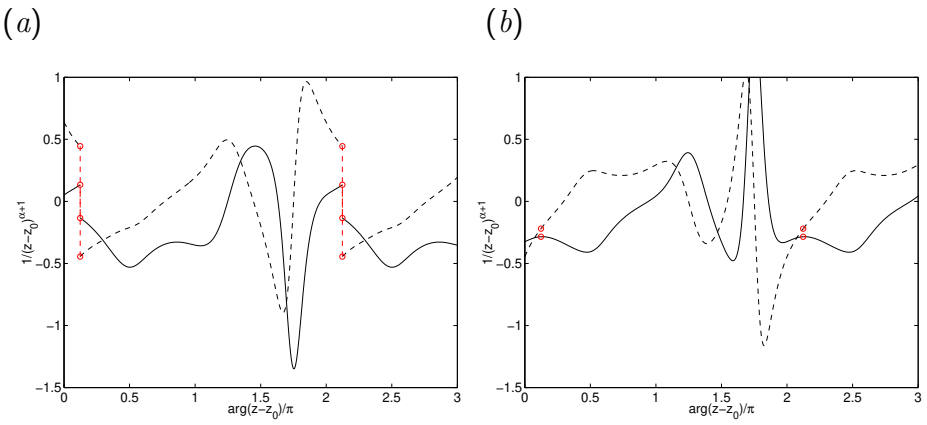


FIGURE 5.6.2 Real part (solid lines) and imaginary part (broken lines) of the kernel $1/(z-z_0)^{\alpha+1}$ shown in (5.6.3) for the configuration shown in Figure 5.4.1 for fractional derivative order (a) $\alpha = 0.5$ and (b) 1.0.

is the jump of a suitable property, Q. Note that the jump shown in (5.6.5) vanishes when α is an integer.

5.6.1 *Computation of the Cauchy fractional integral or derivative*

The following Matlab code named *cauchy_frac*, located in directory CAUCHY of SEMLIB, computes the Cauchy fractional integral of a specified negative or positive order α by integrating around an arbitrary contour enclosing the evaluation point using the trapezoidal rule, and also generates the graphs shown in Figure 5.6.2:

```
N = 2*8*32; % points around the contour

x0 = 0.389;  % arbitrary
y0 =-1.2;    % arbitrary

menu = 2;

%---
% prepare
```

```
%---

im = sqrt(-1);

z0 = x0+im*y0;

%---
% generate contour nodes (arbitrary)
%---

dthet = 2*pi/N;

thetoff = 0.1234*pi;  % arbitrary

for i=1:N+1
 theta(i) = (i-1)*dthet + thetoff;
 r = 1.0 + 0.5*sin(theta(i))^2 ...
         - 1.3*sin(theta(i))^3;  % arbitrary
 x(i) = r*cos(theta(i));
 y(i) = r*sin(theta(i));
end

%---
% contour node distances from x0
%---

for i=1:N+1
 Xr(i) = x(i)-x0;
 Yr(i) = y(i)-y0;
 Rr(i) = sqrt(Xr(i)^2+Yr(i)^2);
end

%---
% angles
%---

Tr(1) = acos(Xr(1)/Rr(1));

if(Yr(1)<0) Tr(1)=-Tr(1); end
```

```
for i=1:N
 crossprd = Xr(i)*Yr(i+1) - Xr(i+1)*Yr(i);
 den = Rr(i)*Rr(i+1);
 dt = asin(crossprd/den);
 innerprd = Xr(i)*Xr(i+1)+Yr(i)*Yr(i+1);
 if(innerprd<0)
  dt = dt*(pi/abs(dt)-1.0);
 end
 Tr(i+1) = Tr(i)+dt;
end

%---
% function evaluation
%---

for i=1:N+1
 z(i) = x(i)+im*y(i);
 [f(i),fp(i),fpp(i),fppp(i)] = fncc(z(i),menu);
end

alpha = -0.333;

%---
% integrand evaluation
%---

for i=1:N+1
 invp(i) = exp(-(alpha+1)*im*Tr(i))/Rr(i)^(alpha+1);
end

%---
% integrate by the trapezoidal rule
%---

integral = 0.0;

for i=1:N
 integral = integral + 0.5*(f(i)  *invp(i) ...
```

```
                          +f(i+1)*invp(i+1)) ...
                          *(z(i+1)-z(i));
end

integral = gamma(alpha+1)*integral/(2*pi*im);

[f0,fp0,fpp0,fppp0] = fncc(z0,menu);
f0_fp0_fpp0_fppp0 = [f0;fp0;fpp0;fppp0]

%---
% jump
%---

jumparg = exp(-2*pi*alpha*im)-1;
jumpinvp = jumparg*invp(1);
jumpR = real(jumpinvp);
jumpI = imag(jumpinvp);

%---
% plot
%---

figure(3)
hold on
axis([0 3 -1.5 1])
xlabel('arg(z-z_0)/\pi','fontsize',15)
ylabel('1/(z-z_0)^{\alpha+1}','fontsize',15)
set(gca,'fontsize',15)
box on

plot(theta/pi,real(invp),'k')
plot(theta/pi+2,real(invp),'k')
plot(theta/pi-2,real(invp),'k')
plot(theta/pi,imag(invp),'k--')
plot(theta/pi+2,imag(invp),'k--')
plot(theta/pi-2,imag(invp),'k--')

plot([theta(1)/pi+0,theta(1)/pi+0] ...
    ,[real(invp(1)),real(invp(1))+jumpR],'--or')
```

```
plot([theta(1)/pi+0,theta(1)/pi+0] ...
    ,[imag(invp(1)),imag(invp(1))+jumpI],'--or')
plot([theta(1)/pi+2,theta(1)/pi+2] ...
    ,[real(invp(1)),real(invp(1))+jumpR],'--or')
plot([theta(1)/pi+2,theta(1)/pi+2] ...
    ,[imag(invp(1)),imag(invp(1))+jumpI],'--or')
```

The code makes calls to the function *fncc* listed in Section 5.4 to evaluate the function $f(z)$ through a menu.

5.6.2 Equivalence to the Riemann–Liouville fractional integral and derivative

We will show that, in fact, the Cauchy fractional derivative is identical to the (left) Riemann–Liouville fractional integral when $\alpha < 0$ or to the (left) Riemann–Liouville fractional derivative when $\alpha > 0$, as discussed in Section 5.5.

5.6.3 Cauchy fractional integral

To demonstrate the equivalence, we first consider a negative Cauchy fractional derivative, $\alpha < 0$, and note that

$$\oint_{\mathcal{C},\mathcal{C}_\mathrm{A},\mathcal{C}_\mathrm{B},\mathcal{C}_\epsilon} \frac{f(z)}{(z-z_0)^{\alpha+1}}\,\mathrm{d}z = 0, \tag{5.6.7}$$

where the contours $\mathcal{C}$, $\mathcal{C}_\mathrm{A}$, $\mathcal{C}_\mathrm{B}$, and $\mathcal{C}_\epsilon$ are defined in Figure 5.6.1. The reason is that the integrand is regular in the area enclosed by these four contours.

Since $\alpha < 0$, the contour integral over $\mathcal{C}_\epsilon$ vanishes in the limit as ϵ tends to zero. Consequently,

$$\mathcal{J}^{(\alpha)} \equiv \oint_{\mathcal{C}} \frac{f(z)}{(z-z_0)^{\alpha+1}}\,\mathrm{d}z = -\int_{z_\mathrm{A}}^{z_0} \Big[\frac{f(z)}{(z-z_0)^{\alpha+1}}\Big]_{+\to-}\,\mathrm{d}z, \tag{5.6.8}$$

where the square brackets signify the upward jump across the line connecting z_A to z_0, as shown in (5.6.6). Using the counterpart of expression (5.6.5) for the jump of the integrand, we find that

$$\mathcal{J}^{(\alpha)} = (1-\mathrm{e}^{-2\pi\alpha\mathrm{i}}) \int_{z_\mathrm{A}}^{z_0} \frac{f(z)}{(z-z_0)^{\alpha+1}}\,\mathrm{d}z. \tag{5.6.9}$$

Now changing $z - z_0$ to $z_0 - z$ in the denominator of the integrand, we obtain

$$\mathcal{J}^{(\alpha)} = (1 - \mathrm{e}^{-2\alpha\pi\mathrm{i}})\,\mathrm{e}^{(\alpha+1)\pi\mathrm{i}} \int_{z_\mathrm{A}}^{z_0} \frac{f(z)}{(z_0 - z)^{\alpha+1}}\,\mathrm{d}z. \tag{5.6.10}$$

We note that

$$(1 - \mathrm{e}^{-2\alpha\pi\mathrm{i}})\,\mathrm{e}^{(\alpha+1)\pi\mathrm{i}} = (\mathrm{e}^{\alpha\pi\mathrm{i}} - \mathrm{e}^{-\alpha\pi\mathrm{i}})\,\mathrm{e}^{\pi\mathrm{i}} = -2\,\mathrm{i}\sin(\alpha\pi), \tag{5.6.11}$$

and obtain

$$\mathcal{J}^{(\alpha)} = -2\,\mathrm{i}\sin(\alpha\pi) \int_{z_\mathrm{A}}^{z_0} \frac{f(z)}{(z_0 - z)^{\alpha+1}}\,\mathrm{d}z. \tag{5.6.12}$$

Now we recall that

$$f_{z_\mathrm{A}}^{(\alpha)}(z_0) \equiv \frac{\Gamma(\alpha+1)}{2\pi\mathrm{i}}\,\mathcal{J}^{(\alpha)}, \tag{5.6.13}$$

and find that

$$f_{z_\mathrm{A}}^{(\alpha)}(z_0) = -\frac{\Gamma(\alpha+1)\sin(\alpha\pi)}{\pi} \int_{z_\mathrm{A}}^{z_0} \frac{f(z)}{(z_0 - z)^{\alpha+1}}\,\mathrm{d}z. \tag{5.6.14}$$

Using the properties of the Gamma function, we find that the negative of the fraction before the last integral is equal to $1/\Gamma(-\alpha)$. Comparing the resulting equation with (5.5.1), we find that

$$f_{z_\mathrm{A}}^{(\alpha)}(z_0) = [\,\mathrm{I}^{(-\alpha)} f\,]_{z_\mathrm{A}}^{\mathrm{RL}}(z_0) \tag{5.6.15}$$

for $\alpha \leq 0$, where $[\,\mathrm{I}_{z_\mathrm{A}}^{(\alpha)} f\,]_{\mathrm{RL}}(z_0)$ is the Riemann–Liouville fractional integral.

5.6.4 *Cauchy fractional derivative*

For α in the range $[0, 1]$, we express formula (5.6.3) as

$$f_{z_\mathrm{A}}^{(\alpha)}(z_0) = -\frac{\Gamma(\alpha+1)}{2\alpha\pi\mathrm{i}} \frac{\mathrm{d}}{\mathrm{d}z_0} \oint_{z_\mathrm{A}} \frac{f(z)}{(z - z_0)^{\alpha}}\,\mathrm{d}z, \tag{5.6.16}$$

which can be restated as

$$f_{z_\mathrm{A}}^{(\alpha)}(z_0) = -\frac{\Gamma(\alpha+1)}{2\alpha\pi\mathrm{i}} \frac{\mathrm{d}}{\mathrm{d}z_0} \mathcal{J}^{(\alpha-1)}. \tag{5.6.17}$$

Using expression (5.6.12), we obtain

$$f^{(\alpha)}_{z_\mathrm{A}}(z_0) = \frac{\Gamma(\alpha+1)\sin(\alpha\pi)}{\alpha\pi}\frac{\mathrm{d}}{\mathrm{d}z_0}\int_{z_\mathrm{A}}^{z_0}\frac{f(z)}{(z_0-z)^\alpha}\,\mathrm{d}z. \qquad (5.6.18)$$

Using the properties of the Gamma function, we find that the fraction after the equal sign is equal to $1/\Gamma(1-\alpha)$, and thereby recover precisely expression (5.5.7) for the Riemann–Liouville fractional derivative.

Similar results are obtained for higher values of α, involving second and higher-order derivatives with respect to z_0. We conclude that the Cauchy fractional derivative is the same as the Riemann–Liouville fractional derivative under the conditions and conventions described in this section.

5.6.5 *Cauchy–Caputo fractional derivative*

Another fractional derivative can be defined based on Cauchy's differentiation formula (5.4.4). We define the Cauchy–Caputo (CC) fractional derivative as

$$[\mathrm{D}^{(\alpha)}f]^{\mathrm{CC}}_{z_\mathrm{A}}(z_0) = \frac{\Gamma(\alpha+1-n)}{2\pi\mathrm{i}}\oint\frac{f^{(n)}(z)}{(z-z_0)^{1+\alpha-n}}\,\mathrm{d}z \qquad (5.6.19)$$

for $n-1 \le \alpha < n$, where the superscript (n) denotes the nth derivative. The Gamma function in the numerator of the fraction before the integral is unity when $\alpha = n-1$. We see that, when $f(z)$ is a constant, the fractional derivative is identically zero.

For non-integer α, the contour integral on the right-hand side of (5.6.19) depends on our choice of the coincident integration contour end points, z_A and z_B, represented by convention by z_A, as discussed earlier in this section.

Working as we did earlier in this section for the Riemann–Liouville derivative, we find that

$$\begin{aligned}&[\mathrm{D}^{(\alpha)}f]^{\mathrm{CC}}_{z_\mathrm{A}}(z_0) \qquad (5.6.20)\\ &= -\Gamma(\alpha+1-n)\,\frac{\sin(\alpha\pi)}{\pi}\,\mathrm{e}^{-n\pi\mathrm{i}}\int_{z_\mathrm{A}}^{z_0}\frac{f^{(n)}(z)}{(z_0-z)^{1+\alpha-n}}\,\mathrm{d}z.\end{aligned}$$

Now invoking the properties of the Gamma function, we find that

$$\Gamma(\alpha+1-n)\,\Gamma(n-\alpha)=\frac{\pi}{\sin((n-\alpha)\pi)}=-(-1)^n\frac{\pi}{\sin(\alpha\pi)}. \tag{5.6.21}$$

Combining the last two equations, we recover precisely the expression for the Caputo derivative given in (5.5.18). In conclusion,

$$[\mathrm{D}^{(\alpha)}f]^{\mathrm{CC}}_{z_{\mathrm{A}}}(z_0)=[\mathrm{D}^{(\alpha)}f]^{\mathrm{C}}_{z_{\mathrm{A}}}(z_0), \tag{5.6.22}$$

where C denotes the Cauchy–Caputo derivative and C denotes the Caputo derivative.

Exercises

5.6.1 Reproduce the graphs shown in Figure 5.6.2 for $\alpha=1.5$.

5.6.2 Confirm that the fraction after the equal sign in (5.6.18) is equal to $1/\Gamma(1-\alpha)$.

5.6.3 Confirm (5.6.21).

5.7 Solution of a linear ordinary differential equation

Consider an Nth-order linear ordinary differential equation for a function, $f(t)$, with a constant forcing term,

$$a_1 f^{(N)}(t)+a_2 f^{(N-1)}(t)+\cdots+a_N f'(t)+a_{N+1}f(t)=b, \tag{5.7.1}$$

where $f^{(i)}(t)$ is the ith derivative, a_i are complex coefficients, and b is a constant. We will assume that $a_{N+1}\neq 0$.

The complex-variable formulation allows us to derive a compact solution given by

$$f(t)=\frac{b}{a_{N+1}}+\sum_{j=1}^{N} c_j\, t^{m_j}\exp(-\mathrm{i}\,\sigma_j t), \tag{5.7.2}$$

where c_j are arbitrary complex coefficients, and σ_j are complex growth rates computed such that the arguments of the exponential, $-\mathrm{i}\,\sigma_j$, are roots of the Nth-degree polynomial

$$\mathcal{Q}_N(z) = a_1 z^N + a_2 z^{N-1} + \cdots + a_N z + a_{N+1}. \tag{5.7.3}$$

The integer exponents, $m_j = 0, 1, \ldots$, are determined by the multiplicity of each root. For a single root, $m_j = 0$; for a double root, $m_j = 0, 1$. An example for $N = 6$ is shown below:

j	σ_j	m_j
1	1.3	0
2	1.3	1
3	0.1	0
4	0.7	0
5	0.7	1
6	0.7	2

(5.7.4)

As an example, we consider the second-order equation

$$f''(t) - 2f'(t) + f(t) = b, \tag{5.7.5}$$

where prime denotes a derivative with respect to t. In this case, $a_1 = 1$, $a_2 = -2$, and $a_3 = 1$. The polynomial defined in (5.7.3) becomes

$$\mathcal{Q}_2(z) = z^2 - 2z + 1 = (z-1)^2. \tag{5.7.6}$$

We find that $-\mathrm{i}\,\sigma_1 = 1$ is a double root.

5.7.1 *Modulated sine waves*

Resolving σ_j into their real and imaginary parts, denoted by the subscripts R and I,

$$\sigma_j = \sigma_{\mathrm{R}_j} + \mathrm{i}\,\sigma_{\mathrm{I}_j}, \tag{5.7.7}$$

we obtain

$$f(t) = \frac{b}{a_{N+1}} + \sum_{j=1}^{N} c_j\, t^{m_j} \exp(-\mathrm{i}\,\sigma_{\mathrm{R}_j} t)\, \exp(\sigma_{\mathrm{I}_j} t). \tag{5.7.8}$$

Each term in this sum is a normal mode exhibiting sinusoidal oscillations with angular frequency σ_{R_j}, while increasing or decreasing exponentially in t with growth or decay rate equal to $\sigma_{I,j}$.

Exercise

5.7.1 Confirm that (5.7.2) satisfies the differential equation (5.7.1).

5.8 Joukowski transformation

Cross-sectional airfoil shapes can be generated by mapping a closed contour in an auxiliary parametric (ξ, η) plane to the airfoil contour in the physical xy plane using an appropriate mapping function.

In theoretical aerodynamics, the mapping function arises from a function of a complex variable, $f(\zeta)$, by setting

$$z = f(\zeta), \tag{5.8.1}$$

where

$$z \equiv x + \mathrm{i}\, y, \qquad \zeta \equiv \xi + \mathrm{i}\, \eta \tag{5.8.2}$$

are two complex variables, and i is the imaginary unit satisfying $\mathrm{i}^2 = -1$.

Joukowski's transformation employs the mapping function

$$f(\zeta) = \zeta + \frac{\sigma^2}{\zeta}, \tag{5.8.3}$$

where σ is a specified length. A circle in the $\xi\eta$ plane passing through the singular point $(-\sigma, 0)$ and enclosing the reflected singular point $(\sigma, 0)$ is mapped to a cusped airfoil whose camberline and camber are determined by the location of the center of the circle in the $\xi\eta$ plane, as shown in Figure 5.8.1. The airfoil cusp is located at the image of the first singular point.

Different airfoil shapes can be generated by fixing the value of σ and varying the position of the center of the circle, $\mathbf{x}_c$, in the $\xi\eta$ plane.

The following Matlab code generates airfoil shapes:

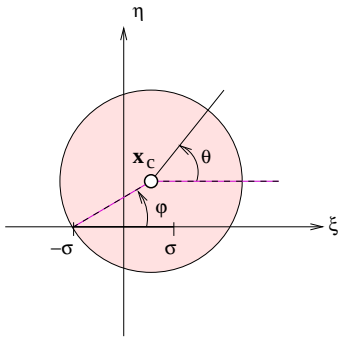


FIGURE 5.8.1 A Joukowski airfoil is generated by mapping conformally a circle in the $\xi\eta$ plane to an airfoil contour in the xy plane. The first singular point, located at the ξ axis at $\xi = -\sigma$, is mapped to the airfoil trailing edge. The second singular point, located at $\xi = \sigma$, is mapped inside the airfoil.

```
xc = 0.1;
yc = 0.2;
n = 32;
sigma = 1.0;

%---
% prepare
%---

rad = sqrt((xc+sigma)^2+yc*yc);
phi = acos((xc+sigma)/rad);
thstart = -pi+phi;
dth = 2*pi/n;

%---
% run around
%---

for i=1:n+1
  th = thstart+(i-1)*dth;
  xi = xc+rad*cos(th);
```

```
  et = yc+rad*sin(th);
  r2 = xi*xi+et*et;
  x(i) = xi*(1.0+sigma^2/r2);
  y(i) = et*(1.0-sigma^2/r2);
end

%---
% plotting
%---

figure(1)
hold on
axis equal
axis(sigma*[-2.2 2.2 -0.5 1])
xlabel('x','fontsize',13)
ylabel('y','fontsize',13)
set(gca,'fontsize',13)
box on

plot(-x,y,'ko-')
```

The graphics generated by the code is shown in Figure 5.8.2 where starting angle, ϕ, is defined in Figure 5.8.1.

The first and last points, labeled 1 and $n+1$ are mapped to the trailing edge. Intermediate points are distributed in the counterclockwise direction around the airfoil. Note that the airfoil contour has been reflected at the plotting stage to place the leading edge on the left.

Exercise

5.8.1 Generate and discuss the airfoil shape when the center, $\mathbf{x}_c$ lies on the η axis.

5.9 Pulsating flow inside a tube

Consider unidirectional pulsating flow of a viscous fluid inside a circular tube of radius a due to an imposed oscillatory pressure gradient along

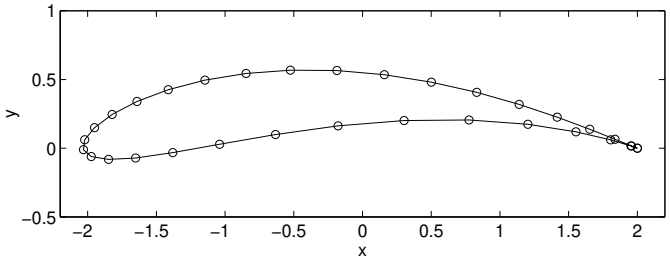


FIGURE 5.8.2 A Joukowski airfoil generated by the code joukowski listed in the text for the parameter values listed in the code.

the x axis. The axial pressure gradient is given by

$$\frac{\partial p}{\partial x} = \rho g_x - \zeta \sin(\Omega t), \tag{5.9.1}$$

where ρ is fluid density, g_x is the axial component of the acceleration of gravity, ζ is the amplitude of the pressure gradient and Ω is the angular frequency of the oscillations.

We introduce the distance from the centerline, σ, and find that the streamwise velocity component, u_x, satisfies the partial differential equation

$$\rho \frac{\partial u_x}{\partial t} = \zeta \sin(\Omega t) + \mu \left(\frac{\partial^2 u_x}{\partial \sigma^2} + \frac{1}{\sigma} \frac{\partial u_x}{\partial \sigma} \right), \tag{5.9.2}$$

which can be regarded as an unsteady forced heat conduction equation.

5.9.1 Wall slip

The fluid may slip over the tube wall so that the wall velocity is proportional to the wall shear stress or wall shear rate,

$$u_x = -\ell \left(\frac{\partial u_x}{\partial \sigma} \right)_{\sigma=a}, \tag{5.9.3}$$

where ℓ is a specified slip length. The no-slip boundary condition requiring that the velocity is zero at the tube wall, $u_x(\sigma = a) = 0$, arises as ℓ tends to zero.

5.9.2 Expression of the velocity in complex form

The velocity can be expressed in the form

$$u_x(\sigma,t) = \frac{\zeta}{\rho\,\Omega}\,\big(\,f_{\rm R}(\sigma)\,\cos(\Omega t) + f_{\rm I}(\sigma)\,\sin(\Omega t)\,\big), \tag{5.9.4}$$

where $f_{\rm R}$ and $f_{\rm I}$ are two real functions. It is expedient to introduce the complex function

$$f(\sigma) \equiv f_{\rm R}(\sigma) + {\rm i}\, f_{\rm I}(\sigma), \tag{5.9.5}$$

where ${\rm i}$ is the imaginary unit, ${\rm i}^2 = -1$. The velocity, u_x, is given by the real part of the complex function

$$w(\sigma,t) = \frac{\zeta}{\rho\Omega}\, f(y)\, {\rm e}^{-{\rm i}\,\Omega t}, \tag{5.9.6}$$

satisfying the governing equation

$$\rho\,\frac{\partial w}{\partial t} = {\rm i}\,\zeta {\rm e}^{-{\rm i}\Omega t} + \mu\,\big(\,\frac{\partial^2 w}{\partial \sigma^2} + \frac{1}{\sigma}\,\frac{\partial w}{\partial \sigma}\,\big). \tag{5.9.7}$$

The real part of this complex equation is the real equation (5.9.2).

5.9.3 Bessel equation

Substituting expression (5.9.6) into (5.9.7), and simplifying, we obtain an ordinary differential equation,

$$\frac{{\rm d}^2 f}{{\rm d}\sigma^2} + \frac{1}{\sigma}\,\frac{{\rm d}f}{{\rm d}\sigma} = -\frac{{\rm i}\,\Omega}{\nu}\,(f+1). \tag{5.9.8}$$

The general solution of this equation is

$$f(\sigma) = c\,{\rm J}_0^*\big(\,\frac{\sigma}{a}\,{\rm Wo}\,\sqrt{-{\rm i}}\,\big) - 1, \tag{5.9.9}$$

where ${\rm J}_0(z)$ a Bessel function of the first kind, an asterisk denotes the complex conjugate, c is a complex constant, and

$${\rm Wo} \equiv a\,\sqrt{\frac{\Omega}{\nu}}. \tag{5.9.10}$$

is the Womersley number, In the human circulation regarding blood flow, the Womersley number varies between nearly zero at the capillaries to nearly 10 in the large vessels.

If β is a real positive number, then

$$\mathrm{J}_0(\beta\sqrt{-\mathrm{i}}) = \mathrm{J}_0(\beta\,\mathrm{e}^{3\pi\mathrm{i}/4}) = \mathrm{ber}_0(\beta) + \mathrm{i}\,\mathrm{bei}_0(\beta), \tag{5.9.11}$$

where ber_0, bei_i are the zeroth-order Kelvin functions. Consequently,

$$f(\sigma) = c\,\big(\,\mathrm{ber}_0(\frac{\sigma}{a}\,\mathrm{Wo}) - \mathrm{i}\,\mathrm{bei}_0(\frac{\sigma}{a}\,\mathrm{Wo})\,\big) - 1, \tag{5.9.12}$$

where c will be determined by the wall slip condition.

5.9.4 *Wall slip*

The slip boundary condition at the wall requires that

$$f(a) = -\ell\,\Big(\frac{\partial f}{\partial\sigma}\Big)_{\sigma=a}. \tag{5.9.13}$$

Substituting expression (5.9.12), we obtain

$$\begin{aligned} c\,\big(\,\mathrm{ber}_0(\mathrm{Wo}) - \mathrm{i}\,\mathrm{bei}_0(\mathrm{Wo})\,\big) - 1 \\ = -c\,\lambda\,\mathrm{Wo}\,\big(\mathrm{ber}_0'(\mathrm{Wo}) - \mathrm{i}\,\mathrm{bei}_0'(\mathrm{Wo})\,\big), \end{aligned} \tag{5.9.14}$$

where $\lambda \equiv \ell/a$. Solving for c, we obtain

$$c = \frac{1}{\mathrm{ber}_0(\mathrm{Wo}) - \mathrm{i}\,\mathrm{bei}_0(\mathrm{Wo}) + \lambda\,\mathrm{Wo}\,\big(\mathrm{ber}_0'(\mathrm{Wo}) - \mathrm{i}\,\mathrm{bei}_0'(\mathrm{Wo})\,\big)}, \tag{5.9.15}$$

thereby completing the solution. We have found that the velocity is given by the real part of the complex function

$$w(\sigma,t) = \frac{\zeta}{\rho\Omega}(F-1)\,\mathrm{e}^{-\mathrm{i}\Omega t}, \tag{5.9.16}$$

where

$$F = \frac{\mathrm{ber}_0(\frac{\sigma}{a}\,\mathrm{Wo}) - \mathrm{i}\,\mathrm{bei}_0(\frac{\sigma}{a}\,\mathrm{Wo})}{\mathrm{ber}_0(\mathrm{Wo}) + \lambda\,\mathrm{Wo}\,\mathrm{ber}_0'(\mathrm{Wo}) - \mathrm{i}\,\big(\,\mathrm{bei}_0(\mathrm{Wo}) + \lambda\,\mathrm{Wo}\,\mathrm{bei}_0'(\mathrm{Wo})\,\big)} \tag{5.9.17}$$

The slip parameter λ appears in the denominator of the fraction on the right-hand side.

5.9.5 Computation of the Kelvin functions

The following Matlab code, located in directory TUBE_PULSE of SEMLIB, computes the Kelvin functions and their derivatives for argument in the range $[0, 8]$:

```
function [ber_0,bei_0,ber_0_p,bei_0_p] ...
...
      = ber_bei_0 (Iopt,X)

%-----------------------------------------
% Evaluation of the Kelvin functions:
%            ber_0, bei_0
%            and their derivatives
%
%  If Iopt.ne.1 derivatives are not computed
%-----------------------------------------

ber_0_p = 0;
bei_0_p = 0;

%---
 if(X<8.00)
%---

%---
% use formulas (9.11.1) and (9.11.2) of A&S, p. 384
%---

  XH  = 0.5*X;  Y   = X/8.0;  Y2  = Y*Y;    Y4  = Y2 *Y2;
  Y6  = Y4 *Y2; Y8  = Y6 *Y2; Y10 = Y8 *Y2; Y12 = Y10*Y2;
  Y14 = Y12*Y2; Y16 = Y14*Y2; Y18 = Y16*Y2; Y20 = Y18*Y2;
  Y22 = Y20*Y2; Y24 = Y22*Y2; Y26 = Y24*Y2; Y28 = Y26*Y2;

  ber_0 = 1.0 ...
          - 64.0        * Y4  + 113.77777774 * Y8 ...
          - 32.36345652 * Y12 +   2.64191397 * Y16 ...
```

```
          -  0.08349609 * Y20 +   0.00122552 * Y24 ...
          -  0.00000901 * Y28;
  bei_0 =    16.0        * Y2  - 113.77777774 * Y6 ...
          + 72.81777742 * Y10 -  10.56765779 * Y14 ...
          +  0.52185615 * Y18 -   0.01103667 * Y22 ...
          +  0.00011346 * Y26;
%---
   if(Iopt~=0)

%---
% use formulas (9.11.5) and (9.11.6) of A&S, p. 384
%---

    ber_0_p = X*(- 4.0        * Y2  + 14.22222222 * Y6 ...
            - 6.06814810 * Y10 +  0.66047849 * Y14 ...
            - 0.02609253 * Y18 +  0.00045957 * Y22 ...
            - 0.00000394 * Y26 );
    bei_0_p = X*(    0.5              - 10.66666666 * Y4 ...
            + 11.37777772 * Y8  -  2.31167514 * Y12 ...
            +  0.14677204 * Y16 -  0.00379386 * Y20 ...
            +  0.00004609 * Y24 );
   end

%---------
  else
%---------

  disp ' Sorry, ber_bei_0 not yet implemented for X>8'
  return

%---
 end
%---

return
```

Graphs of the zeroth-order Kelvin functions are shown in Figure 5.9.1.

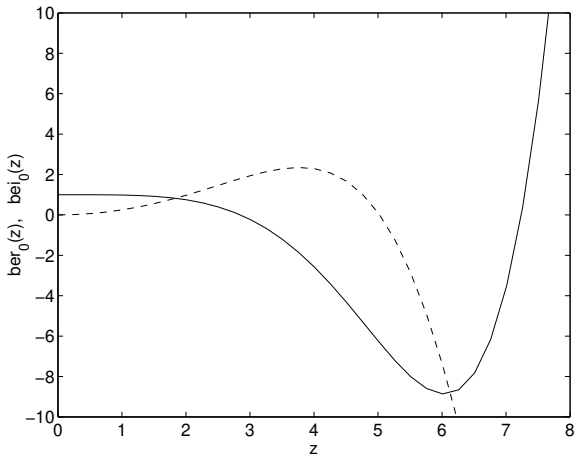


FIGURE 5.9.1 Graphs of the Kelvin functions $\mathrm{ber}_0(z)$ and $\mathrm{bei}_0(z)$ plotted, respectively, with the solid and broken line, arising in the computation of pulsating tube flow.

5.9.6 *Computation of the velocity profile*

The following Matlab code, located in directory TUBE_PULSE of SEMLIB, computes the velocity profile:

```
function u = tube_crc_wom ...
  ...
  (a ...
  ,visc ...
  ,rho ...
  ,pgamp ...
  ,omega ...
  ,ell ...
  ,ot ...
  ,s ...
  )

%-----------------------------------------
% Unsteady unidirectional flow in a circular tube
% due to an oscillating pressure gradient
% with wall slip
```

```
%
% The pressure gradient is:
%
% dp/dx = - pgamp * sin(omega*time)
%
% SYMBOLS:
% -------
%
% a:     tube radius
% omega: angular frequency
% ot:    omega*time
% visc:  fluid viscosity
% rho:   fluid density
% s:     sigma, distance from centerline
% u:     axial velocity
% ell:   slip length
%-----------------------------------------

%--------
% prepare
%--------

  nu  = visc/rho;  %   kinematic viscosity

  Wo = a*sqrt(omega/nu);
  soa = s/a;
  lambda = ell/a;

  Iopt = 1;     % need derivatives: ber0' and ber0'

%---
% denominator
%---

 [ber,bei,berp,beip] ...
  ...
  = ber_bei_0 (Iopt,Wo);

  denr =  ber + lambda*Wo*berp;
```

```
  deni = -bei - lambda*Wo*beip;
  dens = denr*denr + deni*deni;

  aar  = denr/dens;
  aai  =-deni/dens;

%---
% numerator
%---

 [ber,bei,berp,beip] ...
  ...
  = ber_bei_0 (Iopt,soa*Wo);

 bbr =  ber;
 bbi = -bei;

%----
% real part of the product
%---

 Ur = aar*bbr - aai*bbi - 1.0;
 Ui = aar*bbi + aai*bbr;

 cs = cos(ot);
 sn = sin(ot);

 u = pgamp*(cs*Ur + Ui*sn)/(rho*omega);

%-----
% done
%-----

return
```

Evolving velocity profiles for $\mathrm{Wo} = 2$ and scaled slip length $\lambda = 0.1$ are shown in Figure 5.9.2.

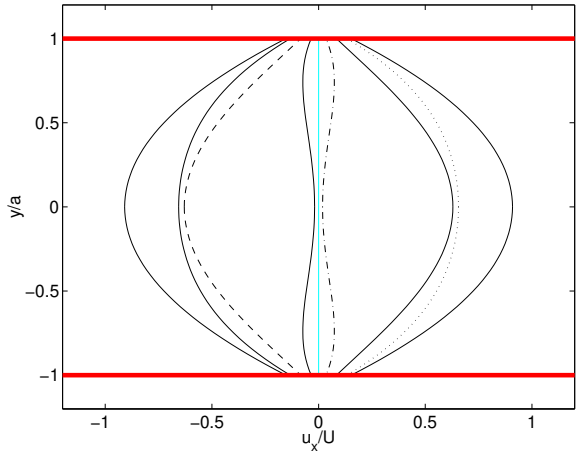


FIGURE 5.9.2 Evolving velocity profiles at different phase angles for $\mathrm{Wo} = 2$ and $\lambda = 0.1$. The velocity is scaled by the centerline velocity of Poiseuille flow in the absence of slip.

5.9.7 Small frequencies

For small values of their arguments, the Kelvin functions exhibit the asymptotic behavior

$$\begin{aligned} \mathrm{ber}_0(\beta) &= 1 - \tfrac{1}{64}\,\beta^4 + \cdots, \\ \mathrm{bei}_0(\beta) &= \tfrac{1}{4}\,\beta^2 - \tfrac{1}{2304}\,\beta^6 + \cdots, \end{aligned} \tag{5.9.18}$$

yielding

$$f(\sigma) \simeq \frac{1 - \frac{1}{4}\mathrm{i}\,\frac{\sigma^2}{a^2}\,\mathrm{Wo}^2}{1 - \frac{1}{4}\mathrm{i}\,\mathrm{Wo}^2 - \frac{1}{2}\,\mathrm{i}\lambda\mathrm{Wo}^2} - 1, \tag{5.9.19}$$

and then

$$f(\sigma) \simeq \big(1 - \mathrm{i}\,\tfrac{1}{4}\,\frac{\sigma^2}{a^2}\,\mathrm{Wo}^2\big)\big(1 + \mathrm{i}\,\tfrac{1}{4}\,\mathrm{Wo}^2 + \tfrac{1}{2}\,\mathrm{i}\lambda\mathrm{Wo}^2\,\big) - 1. \tag{5.9.20}$$

Simplifying, we obtain

$$f(\sigma) \simeq \mathrm{i}\,\tfrac{1}{4}\,(a^2 + 2a\ell - \sigma^2)\,\frac{\Omega}{\nu}. \tag{5.9.21}$$

Substituting $f_{\rm R} = 0$ and $f_{\rm I} = \frac{1}{4}\,(a^2 + 2a\ell - \sigma^2)\,\frac{\Omega}{\nu}$ into (5.9.4), we recover the quasi-steady parabolic profile of Hagen–Poiseuille flow,

$$u_x(\sigma, t) = \frac{\zeta}{4\mu}\,\sin(\Omega t)\,(a^2 + 2a\ell - \sigma^2) + \cdots . \tag{5.9.22}$$

We recall that the no-slip boundary condition arises when $\ell = 0$.

Exercise

5.9.1 Discuss the effect of wall slip on the shape of the velocity profile for ${\rm Wo} = 2$.

5.10 Solving the Poisson equation in a plane

Consider the Poisson equation for an unknown function, $f(x, y)$, in the xy plane, written in the form

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = -s(x, y), \tag{5.10.1}$$

where $s(x, y)$ is a specified source. The solution is to be found inside an arbitrary region, $\mathcal{D}$, subject to the Dirichlet boundary condition,

$$f = \phi \tag{5.10.2}$$

around the boundary of $\mathcal{D}$, where ϕ is a given distribution.

5.10.1 Particular and homogeneous solutions

It is useful to decompose the solution into a particular solution and a homogeneous solution,

$$f = f^{\rm p} + f^{\rm h}. \tag{5.10.3}$$

The particular solution satisfies the given Poisson equation,

$$\nabla^2 f^{\rm p} + s(x, y) = 0, \tag{5.10.4}$$

while the homogeneous solution satisfies the Laplace equation,

$$\nabla^2 f^{\mathrm{h}} = 0. \tag{5.10.5}$$

The boundary conditions require that

$$f^{\mathrm{p}} + f^{\mathrm{h}} = \phi \tag{5.10.6}$$

around the boundary of the solution domain. Consequently, the homogeneous solution must satisfy the boundary condition

$$f^{\mathrm{h}} = \phi - f^{\mathrm{p}} \tag{5.10.7}$$

around the boundary of the solution domain.

5.10.2 *Solution inside a circular disk*

The solution at a point $\mathbf{x}$ inside a circular disk of radius a is given in terms of the Poisson integral by the expression

$$f(\mathbf{x}) = f^{\mathrm{p}}(\mathbf{x}) + \frac{a^2 - |\mathbf{x}|^2}{2\pi a} \oint_{\circ} \frac{(\phi - f^{\mathrm{p}})_{(\mathbf{x}')}}{|\mathbf{x} - \mathbf{x}'|^2} \, \mathrm{d}\ell(\mathbf{x}'), \tag{5.10.8}$$

where $\circ$ in the lower integration limit denotes the circular boundary and ℓ is the arc length along the circular boundary.

For a circular boundary centered at the origin, we introduce polar coordinates, (r, θ), such that $x = r\cos\theta$, $y = r\sin\theta$, and $\mathrm{d}\ell = a\,\mathrm{d}\theta$, and obtain the expression

$$f(r,\theta) = f^{\mathrm{p}}(r,\theta) + \frac{a^2 - r^2}{2\pi} \int_0^{2\pi} \frac{(\phi - f^{\mathrm{p}})_{(\theta')}}{a^2 + r^2 - 2ar\cos(\theta - \theta')} \, \mathrm{d}\theta' \tag{5.10.9}$$

for $r \leq a$.

In the absence of a source term, $f^{\mathrm{p}} = 0$, and when $\phi = \phi_0$ for $-\alpha < \theta < \alpha$, where ϕ_0 is a constant, we obtain

$$f(r,\theta) = \phi_0 \, \frac{1}{2\pi} \, (a^2 - r^2) \int_{-\alpha}^{\alpha} \frac{\mathrm{d}\theta'}{a^2 + r^2 - 2ar\cos(\theta - \theta')}. \tag{5.10.10}$$

Performing the integration, we obtain

$$f(r,\theta) = \frac{\phi_0}{\pi}\Big[\arctan\big(\frac{a+r}{a-r}\tan\frac{\alpha-\theta}{2}\big) + \arctan\big(\frac{a+r}{a-r}\tan\frac{\alpha+\theta}{2}\big)\Big]. \tag{5.10.11}$$

In the limit as α tends to π, we obtain a uniform solution, $f = f_0$, where f_0 is a constant.

5.10.3 *Solution inside an arbitrary domain*

Now we proceed to consider the solution in a domain with arbitrary shape. The theory of functions of a complex variable guarantees that the harmonic function $f^{\rm h}$ can be regarded as the real part of an analytic function, $\mathcal{G}(z)$, where

$$z = x + {\rm i}y = r\,{\rm e}^{{\rm i}\theta} \tag{5.10.12}$$

is the complex variable in the xy plane and ${\rm i}$ is the imaginary unit, so that

$$f^{\rm h} = {\rm real}\{G(z)\}. \tag{5.10.13}$$

The imaginary part of $G(z)$ is also a harmonic function.

To compute the solution, we introduce a new complex variable,

$$\zeta = \xi + {\rm i}\,\eta = \sigma {\rm e}^{{\rm i}\varphi}, \tag{5.10.14}$$

where ξ and η are the real and imaginary parts of ζ, σ is the distance from the origin, and φ is the polar angle in the ζ plane, $0 \le \varphi < 2\pi$.

Next, we introduce a complex function that maps a disk of radius ϱ centered at the origin of the ζ plane to the solution domain in the z complex plane,

$$z = \mathcal{F}(\zeta). \tag{5.10.15}$$

The boundary of the solution domain in the z plane is described parametrically by the functions

$$x_c(\varphi) = {\rm real}\big\{\mathcal{F}(\varrho {\rm e}^{{\rm i}\varphi})\big\} \tag{5.10.16}$$

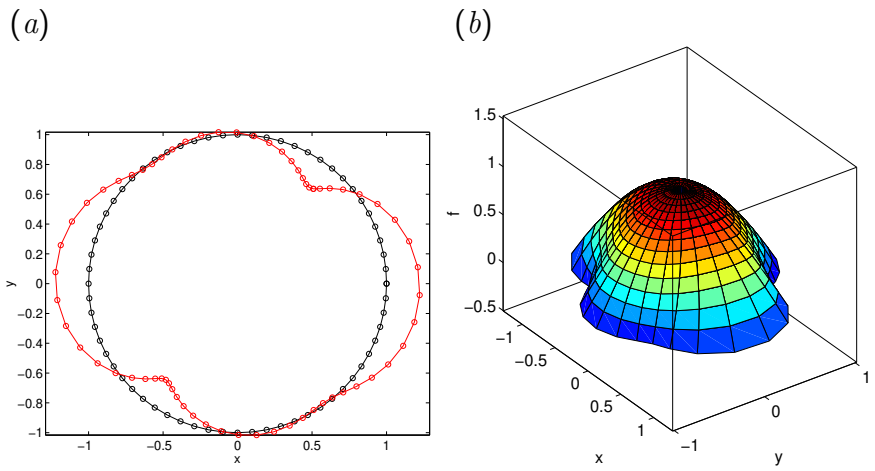


FIGURE 5.10.1 (a) Mapping of the unit disk (black line) to an non-circular domain (red line) using the mapping function (5.10.18). (b) Solution of the Poisson equation with a uniform source term.

and

$$y_c(\varphi) = \operatorname{imag}\{\mathcal{F}(\varrho e^{i\varphi})\}. \tag{5.10.17}$$

As an example, the shape corresponding to the mapping function

$$F(\zeta) = \frac{1}{\sqrt{2}}\,(1 + i - 0.2\,\zeta^2)\,(1 - 0.1\,\zeta^4)\,\zeta \tag{5.10.18}$$

is shown in Figure 5.10.1(a).

5.10.4 *Poisson integral*

Now using the Poisson integral, we obtain the solution in terms of an integral around the circular boundary of the disk in the ζ plane,

$$f(z) = f^{\mathrm{P}}(z) + \frac{\varrho^2 - \sigma^2}{2\pi}\int_0^{2\pi} \frac{(\phi - f^{\mathrm{P}})_{(\varphi')}}{\varrho^2 + \sigma^2 - 2\varrho\sigma\cos(\varphi - \varphi')}\,\mathrm{d}\varphi', \tag{5.10.19}$$

where the numerator of the fraction inside the integral is regarded as a function of the integration variable, φ', and (x, y) is mapped to (σ, ϕ) through $\mathcal{F}(\zeta)$.

5.10.5 Evaluation code

The following Matlab function, located in directory POISSON_CF of SEMLIB, evaluates the right-hand side of (5.10.19) at a given radial distance, σ, and polar angle, ϕ, in the ζ plane. The homogeneous boundary condition, $\phi = 0$, and a constant source term that is equal to unity are assumed. The chosen particular solution is given by

$$f^{\rm p} = -\tfrac{1}{4}\,(x^2 + y^2). \tag{5.10.20}$$

The integral with respect to φ' is computed using the mid-point rule:

```
function [x,y,f] = pintegral(Nphp,rho,sigma,phi)

%=====================================================
% Compute the Poisson integral using the mid-point rule
%
% sigma, phi: evaluation point in the zeta domain
% x, y:       evaluation point in the z domain
% phip :      phi' (integration angle)
% Nphp :      number of integration divisions
% fp   :      particular solution
%=====================================================

%---
% prepare
%---

   im = sqrt(-1);
   Dphp = 2*pi/Nphp;

%---
% evaluation point in the z plane
% by mapping
%---

   zeta = sigma*exp(im*phi);
   z = (1.0+1.0*im-0.2*zeta^2)/sqrt(2) ...
       *(1.0-0.1*zeta^4)*zeta;
   x = real(z); y = imag(z);
```

```
%---
% integrate
%---

  fc = Dphp*(rho^2-sigma^2)/(2.0*pi);

  f = 0.0;

  for i=1:Nphp;
     phip = (i-0.5)*Dphp;
     zeta = rho*exp(im*phip);
     zp = (1.0+1.0*im-0.2*zeta^2)/sqrt(2) ...
           *(1.0-0.1*zeta^4)*zeta;
     xp = real(zp);
     yp = imag(zp);
     fp = - 0.25*(xp^2+yp^2);   % particular solution
     dp = rho^2+sigma^2-2*rho*sigma*cos(phi-phip);
     f = f - fp/dp;
  end

  f = fc*f;
  f = f - 0.25*(x^2+y^2);

%---
% done
%---

return
```

The solution profile computed using this function is shown in Figure 5.10.1(b).

Exercise

5.10.1 Generate the counterpart of Figure 5.10.1 for a different mapping function of your choice and discuss the results.

5.11 Unit imaginary matrix

Consider a 2×2 back-diagonal matrix, $\mathbf{J}$, with nonzero off-diagonal elements $J_{1,2} = 1$ and $J_{2,1} = -1$, called the *unit imaginary matrix*. Written explicitly,

$$\mathbf{J} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}. \tag{5.11.1}$$

The trace of this matrix is zero and the determinant is equal to unity. For comparison, we display the 2×2 identity matrix,

$$\mathbf{I} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{5.11.2}$$

The matrix $\mathbf{I}$ is diagonal and isotropic, whereas the matrix $\mathbf{J}$ is skew-symmetric.

5.11.1 Association with the imaginary unit

Computing successive powers of the unit imaginary matrix, we obtain

$$\mathbf{J}^2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} = -\mathbf{I} \tag{5.11.3}$$

for the second power,

$$\mathbf{J}^3 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \cdot \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} = -\mathbf{J}, \tag{5.11.4}$$

for the third power, and

$$\mathbf{J}^4 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \mathbf{I} \tag{5.11.5}$$

for the fourth power.

We observe that the powers of this matrix behave similarly to the imaginary unit i, in that they satisfy

$$\mathrm{i}^2 = -1, \qquad \mathrm{i}^3 = -\mathrm{i}, \qquad \mathrm{i}^4 = 1, \qquad \ldots \qquad . \tag{5.11.6}$$

That is, $\mathbf{J}^3 = -\mathbf{J}$, just as $\mathrm{i}^3 = -\mathrm{i}$, and $\mathbf{J}^4 = -\mathbf{J}^2$, just as $\mathrm{i}^4 = -\mathrm{i}^2$.

5.11.2 Complex numbers

These observations suggest that, if a and b are two real numbers, then we can define a 2×2 complex matrix,

$$\mathbf{Z}_{a,b} = a\mathbf{I} + b\,\mathbf{J} = \begin{bmatrix} a & b \\ -b & a \end{bmatrix}, \tag{5.11.7}$$

whose rules of multiplication mimic those of the complex number $z = a + \mathrm{i}b$. Note that

$$\det(\mathbf{Z}_{a,b}) = |z|^2 = a^2 + b^2 \tag{5.11.8}$$

and

$$\mathrm{trace}(\mathbf{Z}_{a,b}) = 2\,a. \tag{5.11.9}$$

The matrix inverse is

$$\mathbf{Z}_{a,b}^{-1} = \frac{1}{a^2 + b^2} \begin{bmatrix} a & -b \\ b & a \end{bmatrix}. \tag{5.11.10}$$

The inverse matrix exist, unless both a and b are zero.

5.11.3 Eigenvalues and eigenvectors

The eigenvalues of the matrix $\mathbf{Z}_{a,b}$ are the complex number $a + \mathrm{i}b$ and its complex conjugate denoted by an asterisk,

$$\lambda_1 = a + \mathrm{i}\,b, \qquad \lambda_2 = \lambda_1^* = a - \mathrm{i}\,b. \tag{5.11.11}$$

The corresponding eigenvectors are

$$\mathbf{v}_1 = \begin{bmatrix} 1 \\ \mathrm{i} \end{bmatrix}, \qquad \mathbf{v}_2 = \begin{bmatrix} 1 \\ -\mathrm{i} \end{bmatrix}. \tag{5.11.12}$$

Note that the eigenvectors are independent of a and b.

5.11.4 Complex conjugate

The complex conjugate of a complex number, z, is represented by the matrix transpose,

$$z^* \to \mathbf{Z}_{a,b}^{\mathrm{T}}. \tag{5.11.13}$$

We find that

$$\mathbf{Z}_{a,b}\,\mathbf{Z}_{a,b}^{\mathrm{T}} = \begin{bmatrix} a & b \\ -b & a \end{bmatrix} \begin{bmatrix} a & -b \\ b & a \end{bmatrix} = (a^2+b^2)\,\mathbf{I}, \tag{5.11.14}$$

where $\mathbf{I}$ is the 2×2 identity matrix.

5.11.5 Polar representation

We may write

$$\mathbf{Z}_{a,b} = \sqrt{a^2+b^2}\,\mathbf{exp}(\theta\mathbf{J}), \tag{5.11.15}$$

where $\tan\theta = b/a$ and

$$\mathbf{exp}(\theta\mathbf{J}) = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \tag{5.11.16}$$

according to the usual rules of matrix exponentiation. The matrix inverse is

$$\mathbf{Z}_{a,b}^{-1} = \frac{1}{\sqrt{a^2+b^2}}\,\mathbf{exp}(-\theta\mathbf{J}). \tag{5.11.17}$$

5.11.6 Multiplication

With regard to multiplication, we note that

$$\mathbf{Z}_{a,b}\cdot\mathbf{Z}_{c,d} = \mathbf{Z}_{g,h}, \tag{5.11.18}$$

where

$$g = ac - bd, \qquad h = ad + bg, \tag{5.11.19}$$

consistent with the rules of multiplication of complex numbers.

5.11.7 Cayley–Hamilton theorem

The Cayley–Hamilton theorem for a 2×2 matrix requires that

$$\mathbf{Z}_{a,b}^2 = -\det(\mathbf{Z}_{a,b})\,\mathbf{I} + \operatorname{trace}(\mathbf{Z}_{a,b})\,\mathbf{Z}_{a,b}, \tag{5.11.20}$$

which implies the identity

$$z^2 = -|z|^2 + 2a\,z, \tag{5.11.21}$$

where a is the real part of z.

5.11.8 Function of a complex variable

A function of a complex variable, $f(z)$, maps a complex matrix, $\mathbf{Z}_{a,b}$, to another complex matrix,

$$\mathbf{Z}_{c,d} = \begin{bmatrix} f_{\mathrm{R}}(a,b) & f_{\mathrm{I}}(a,b) \\ -f_{\mathrm{I}}(a,b) & f_{\mathrm{R}}(a,b) \end{bmatrix}, \tag{5.11.22}$$

where f_{R} is the real part and f_{I} is the imaginary part of f. The Cauchy–Riemann equations specify that

$$\frac{\partial f_{\mathrm{R}}}{\partial a} = \frac{\partial f_{\mathrm{I}}}{\partial b}, \qquad \frac{\partial f_{\mathrm{R}}}{\partial b} = -\frac{\partial f_{\mathrm{I}}}{\partial a}, \tag{5.11.23}$$

which guarantee that the derivative $\mathrm{d}\mathbf{Z}_{c,d}/\mathrm{d}z$ is well defined.

5.11.9 Resolution of a 2×2 matrix

An arbitrary 2×2 matrix,

$$\mathbf{A} \equiv \begin{bmatrix} a & b \\ c & d \end{bmatrix} \tag{5.11.24}$$

can be resolved into a sum of four terms, as

$$\mathbf{A} = \tfrac{1}{2}(a+d)\,\mathbf{I} + \tfrac{1}{2}(a-d)\,\mathbf{L} + \tfrac{1}{2}(b-c)\,\mathbf{J} + \tfrac{1}{2}(b+c)\,\mathbf{M}, \tag{5.11.25}$$

where

$$\mathbf{I} \equiv \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad \mathbf{L} \equiv \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{5.11.26}$$

are diagonal matrices,

$$\mathbf{J} \equiv \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \qquad \mathbf{M} \equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \tag{5.11.27}$$

are back-diagonal matrices, $\mathbf{I}$ is the unit matrix, and $\mathbf{J}$ is the unit imaginary matrix.

We find that

$$\mathbf{I}^2 = \mathbf{I}, \qquad \mathbf{L}^2 = \mathbf{I}, \qquad \mathbf{J}^2 = -\mathbf{I}, \qquad \mathbf{M}^2 = \mathbf{I}, \tag{5.11.28}$$

and conclude that the matrices $\mathbf{L}$ and $\mathbf{M}$ are two square roots of the identity matrix. Moreover, we find that

$$\mathbf{LJ} = -\mathbf{JL} = \mathbf{M}, \quad \mathbf{JM} = -\mathbf{MJ} = \mathbf{L}, \quad \mathbf{ML} = -\mathbf{LM} = -\mathbf{J}. \tag{5.11.29}$$

These properties are summarized in the following multiplication table:

	$\mathbf{I}$	$\mathbf{L}$	$\mathbf{J}$	$\mathbf{M}$
$\mathbf{I}$	$\mathbf{I}$	$\mathbf{L}$	$\mathbf{J}$	$\mathbf{M}$
$\mathbf{L}$	$\mathbf{L}$	$\mathbf{I}$	$\mathbf{M}$	$\mathbf{J}$
$\mathbf{J}$	$\mathbf{J}$	$-\mathbf{M}$	$-\mathbf{I}$	$\mathbf{L}$
$\mathbf{M}$	$\mathbf{M}$	$-\mathbf{J}$	$-\mathbf{L}$	$\mathbf{I}$

(5.11.30)

Using this multiplication table, we find that, if

$$\mathbf{B} \equiv \mathbf{A} - \tfrac{1}{2}(a+d)\,\mathbf{I}, \tag{5.11.31}$$

then

$$\mathbf{B}^2 = \xi\,\mathbf{I}, \tag{5.11.32}$$

where

$$\xi = \tfrac{1}{4}\left((a-d)^2 - (b-c)^2 + (b+c)^2\right) = \tfrac{1}{4}(a-d)^2 + bc. \tag{5.11.33}$$

We conclude that

$$\mathbf{A} = \tfrac{1}{2}(a+d)\,\mathbf{I} + \sqrt{\xi}\,\mathbf{I}^{1/2} \tag{5.11.34}$$

for any square root of the 2×2 identity matrix.

Exercise

5.11.1 Confirm equation (5.11.21).

5.12 Quaternions

Consider a collection of four 2×2 matrices,

$$\mathbf{I} \equiv \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad \mathbf{N} \equiv \begin{bmatrix} \mathrm{i} & 0 \\ 0 & -\mathrm{i} \end{bmatrix} = \mathrm{i}\mathbf{L},$$
$$\mathbf{J} \equiv \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \qquad \mathbf{K} \equiv \begin{bmatrix} 0 & \mathrm{i} \\ \mathrm{i} & 0 \end{bmatrix} = \mathrm{i}\mathbf{M}, \tag{5.12.1}$$

where $\mathbf{I}$ is the identity matrix, $\mathbf{J}$ is the unit imaginary matrix, i is the imaginary unit, $\mathrm{i}^2 = -1$, and the matrices $\mathbf{L}$ and $\mathbf{M}$ are defined in (5.11.26) and (5.11.27).

The squares of these matrices are found to be

$$\mathbf{I}^2 = \mathbf{I}, \qquad \mathbf{N}^2 = -\mathbf{I}, \qquad \mathbf{J}^2 = -\mathbf{I}, \qquad \mathbf{K}^2 = -\mathbf{I}. \tag{5.12.2}$$

We observe that $\mathbf{N}$, $\mathbf{J}$, and $\mathbf{K}$ are three square roots of the negative of the identity matrix,

$$\mathbf{N} = \sqrt{-\mathbf{I}}, \qquad \mathbf{J} = \sqrt{-\mathbf{I}}, \qquad \mathbf{K} = \sqrt{-\mathbf{I}}, \tag{5.12.3}$$

and may thus be considered as being imaginary.

Moreover, we find that

$$\mathbf{N}\cdot\mathbf{J} = -\mathbf{J}\cdot\mathbf{N} = \mathbf{K}, \qquad \mathbf{J}\cdot\mathbf{K} = -\mathbf{K}\cdot\mathbf{J} = \mathbf{N},$$
$$\mathbf{K}\cdot\mathbf{N} = -\mathbf{N}\cdot\mathbf{K} = \mathbf{J}, \tag{5.12.4}$$

and thus

$$\mathbf{N}\cdot\mathbf{J}\cdot\mathbf{K}=-\mathbf{I}. \tag{5.12.5}$$

We see that the ordered triplet $(\mathbf{N},\,\mathbf{J},\,\mathbf{K})$ comprises a three-member ring.

The preceding multiplication properties are summarized in the following multiplication table:

	$\mathbf{I}$	$\mathbf{N}$	$\mathbf{J}$	$\mathbf{K}$
$\mathbf{I}$	$\mathbf{I}$	$\mathbf{N}$	$\mathbf{J}$	$\mathbf{K}$
$\mathbf{N}$	$\mathbf{N}$	$-\mathbf{I}$	$\mathbf{K}$	$-\mathbf{J}$
$\mathbf{J}$	$\mathbf{J}$	$-\mathbf{K}$	$-\mathbf{I}$	$\mathbf{N}$
$\mathbf{K}$	$\mathbf{K}$	$\mathbf{J}$	$-\mathbf{N}$	$-\mathbf{I}$

(5.12.6)

The first diagonal element is the identity matrix, $\mathbf{I}$, whereas the last three diagonal elements are the negative of the identity matrix.

5.12.1 *Matrix expansion*

An arbitrary matrix with real or complex elements, a, b, c, d,

$$\mathbf{A}\equiv\begin{bmatrix} a & b \\ c & d \end{bmatrix}, \tag{5.12.7}$$

can be resolved as

$$\mathbf{A}=q_1\,\mathbf{I}+q_2\,\mathbf{N}+q_3\,\mathbf{J}+q_4\,\mathbf{K}, \tag{5.12.8}$$

where q_i are expansion coefficients. We find that

$$\begin{aligned} a&=q_1+\mathrm{i}\,q_2, \qquad & b&=q_3+\mathrm{i}\,q_4,\\ c&=-q_3+\mathrm{i}\,q_4, \qquad & d&=q_1-\mathrm{i}\,q_2, \end{aligned} \tag{5.12.9}$$

and thus

$$\mathbf{A}\equiv\begin{bmatrix} q_1+\mathrm{i}\,q_2 & q_3+\mathrm{i}\,q_4 \\ -q_3+\mathrm{i}\,q_4 & q_1-\mathrm{i}\,q_2 \end{bmatrix}. \tag{5.12.10}$$

Conversely,

$$\begin{aligned} q_1 &= \tfrac{1}{2}\,(a+d), \quad q_2 = -\mathrm{i}\,\tfrac{1}{2}\,(a-d), \\ q_3 &= \tfrac{1}{2}\,(b-c), \quad q_4 = -\mathrm{i}\,\tfrac{1}{2}\,(b+c). \end{aligned} \tag{5.12.11}$$

If $d = a^*$ and $c = -b^*$, so that

$$\mathbf{A} \equiv \begin{bmatrix} a & b \\ -b^* & a^* \end{bmatrix}, \tag{5.12.12}$$

the four expansion coefficients, q_i, are real, where an asterisk denotes the complex conjugate.

5.12.2 *Matrix Hermitian*

The Hermitian of a matrix, $\mathbf{A}$, is the complex conjugate transpose, denoted by the superscript H. We note that

$$\mathbf{I}^{\mathrm{H}} = \mathbf{I}, \quad \mathbf{N}^{\mathrm{H}} = -\mathbf{N}, \quad \mathbf{J}^{\mathrm{H}} = -\mathbf{J}, \quad \mathbf{K}^{\mathrm{H}} = -\mathbf{K}, \tag{5.12.13}$$

and find that

$$\mathbf{A}^{\mathrm{H}} = q_1^*\,\mathbf{I} - q_2^*\,\mathbf{N} - q_3^*\,\mathbf{J} - q_4^*\,\mathbf{K}. \tag{5.12.14}$$

where an asterisk denotes the complex conjugate.

5.12.3 *Matrix mapping*

The decomposition (5.12.8) maps an arbitrary real or complex 2×2 matrix to a quaternion regarded as an ordered quadruplet,

$$\mathbf{A} \to q \equiv \big(\, q_1, \quad q_2, \quad q_3, \quad q_4 \,\big). \tag{5.12.15}$$

The Hermitian of $\mathbf{A}$ is mapped to the quaternion

$$\mathbf{A}^{\mathrm{H}} \to q^{\mathrm{H}} \equiv \big(\, q_1^*, \quad -q_2^*, \quad -q_3^*, \quad -q_4^* \,\big), \tag{5.12.16}$$

where an asterisk denotes the complex conjugate.

5.12.4 Determinant

It can be shown by direct substitution that the determinant of a matrix, $\mathbf{A}$, is the square of the norm of a quaternion defined as

$$\det(\mathbf{A}) = |q|^2 \equiv q_1^2 + q_2^2 + q_3^2 + q_4^2. \tag{5.12.17}$$

This expression does *not* imply that the matrix $\mathbf{A}$ is singular only if all components of the quaternion are zero. The reason is that these components may be imaginary or complex.

5.12.5 Matrix inverse

The inverse of a matrix, $\mathbf{A}$, is given by

$$\mathbf{A}^{-1} = \frac{1}{\det(\mathbf{A})} \begin{bmatrix} d & -c \\ -b & a \end{bmatrix}. \tag{5.12.18}$$

We find that the inverse is mapped to

$$\mathbf{A}^{-1} \to \frac{1}{|q|^2} \big(q_1, \quad -q_2, \quad -q_3, \quad -q_4 \big), \tag{5.12.19}$$

where the components q_i pertain to $\mathbf{A}$.

5.12.6 Addition, subtraction, and multiplication

With regard to addition and subtraction, it is evident that the sum or difference of any pair of 2×2 matrices, $\mathbf{A}$ and $\mathbf{B}$, is mapped to

$$\mathbf{A} \pm \mathbf{B} \to \big(q_1^{\mathrm{A}} \pm q_1^{\mathrm{B}}, \quad q_2^{\mathrm{A}} \pm q_2^{\mathrm{B}}, \quad q_3^{\mathrm{A}} \pm q_3^{\mathrm{B}}, \quad q_4^{\mathrm{A}} \pm q_4^{\mathrm{B}} \big). \tag{5.12.20}$$

With regard to multiplication, we find that

$$\mathbf{C} \equiv \mathbf{A} \cdot \mathbf{B} \to \big(q_1^{\mathrm{C}}, \quad q_2^{\mathrm{C}}, \quad q_3^{\mathrm{C}}, \quad q_4^{\mathrm{C}} \big), \tag{5.12.21}$$

where

$$\begin{aligned} q_1^{\mathrm{C}} &= q_1^{\mathrm{A}} q_1^{\mathrm{B}} - q_2^{\mathrm{A}} q_2^{\mathrm{B}} - q_3^{\mathrm{A}} q_3^{\mathrm{B}} - q_4^{\mathrm{A}} q_4^{\mathrm{B}}, \\ q_2^{\mathrm{C}} &= q_1^{\mathrm{A}} q_2^{\mathrm{B}} + q_2^{\mathrm{A}} q_1^{\mathrm{B}} + q_3^{\mathrm{A}} q_4^{\mathrm{B}} - q_4^{\mathrm{A}} q_3^{\mathrm{B}}, \\ q_3^{\mathrm{C}} &= q_4^{\mathrm{A}} q_3^{\mathrm{B}} - q_2^{\mathrm{A}} q_4^{\mathrm{B}} + q_3^{\mathrm{A}} q_1^{\mathrm{B}} + q_4^{\mathrm{A}} q_2^{\mathrm{B}}, \\ q_1^{\mathrm{C}} &= q_1^{\mathrm{A}} q_4^{\mathrm{B}} + q_2^{\mathrm{A}} q_3^{\mathrm{B}} - q_3^{\mathrm{A}} q_2^{\mathrm{B}} + q_4^{\mathrm{A}} q_1^{\mathrm{B}}. \end{aligned} \tag{5.12.22}$$

Using the expression for the determinant given in (5.12.17), we find that

$$|q^{\mathrm{C}}|^2 = |q^{\mathrm{A}}|^2 \, |q^{\mathrm{B}}|^2, \tag{5.12.23}$$

which ensures the multiplicative property of the norms.

5.12.7 Projection

The inner (dot) product of two quaternions, A and B, is a scalar defined as

$$q^{\mathrm{A}} \cdot q^{\mathrm{B}} \equiv q_1^{\mathrm{A}} q_1^{\mathrm{B}} + q_2^{\mathrm{A}} q_2^{\mathrm{B}} + q_3^{\mathrm{A}} q_3^{\mathrm{B}} + q_4^{\mathrm{A}} q_4^{\mathrm{B}}. \tag{5.12.24}$$

When $q_{\mathrm{A}} = q_{\mathrm{B}}$, we obtain the square of the norm of q_{A}.

With regard to the inner product, we find that

$$q^{\mathrm{A}} \cdot q^{\mathrm{B}} = \tfrac{1}{2} \frac{1}{\det(\mathbf{A})} \operatorname{trace}(\mathbf{A}\mathbf{B}^{-1}) = \tfrac{1}{2} \frac{1}{\det(\mathbf{B})} \operatorname{trace}(\mathbf{B}\mathbf{A}^{-1}). \tag{5.12.25}$$

5.12.8 Confirmation by code

The following Matlab code named *quaternions*, located in directory QUATERNIONS of SEMLIB, confirms these transformation rules:

```
im = sqrt(-1);

A = [3.0+1.0*im 4.0+0.3*im;   % arbitrary
     5.0-0.4*im 6.0+2.0*im];

B = [1.0+1.0*im 3.0+0.4*im; % arbitrary
     2.0-0.8*im 7.0-1.3*im];

C = A*B;

q1A =     0.5*(A(1,1)+A(2,2));
q2A = -im*0.5*(A(1,1)-A(2,2));
q3A =     0.5*(A(1,2)-A(2,1));
q4A = -im*0.5*(A(1,2)+A(2,1));

q1B =     0.5*(B(1,1)+B(2,2));
q2B = -im*0.5*(B(1,1)-B(2,2));
q3B =     0.5*(B(1,2)-B(2,1));
q4B = -im*0.5*(B(1,2)+B(2,1));

q1AB = q1A*q1B - q2A*q2B - q3A*q3B - q4A*q4B;
q2AB = q1A*q4B + q2A*q3B - q3A*q2B + q4A*q1B;
q3AB = q1A*q2B + q2A*q1B + q3A*q4B - q4A*q3B;
q4AB = q1A*q3B - q2A*q4B + q3A*q1B + q4A*q2B;

q1C =     0.5*(C(1,1)+C(2,2));
q2C = -im*0.5*(C(1,1)-C(2,2));
q3C =     0.5*(C(1,2)-C(2,1));
q4C = -im*0.5*(C(1,2)+C(2,1));

[q1C q1AB;
 q2C q2AB;
 q3C q4AB;
 q4C q4AB;
]
```

Running the code generates the following output:

```
  35.0000 + 4.2000i  35.0000 + 4.2000i
  -2.8000 +24.7600i  -2.8000 +24.7600i
```

```
8.9950 - 1.3500i   8.9950 - 1.3500i
2.4500 -27.9950i   2.4500 -27.9950i
```

The two columns of complex numbers representing q_1^{C}, q_2^{C}, q_3^{C}, and q_4^{C}, are identical.

5.12.9 *Fundamental quaternion units*

The multiplication rules listed in Table (5.12.6) are identical to those satisfied by the fundamental quaternion units,

$$1, \quad \mathrm{i}, \quad \mathrm{j}, \quad \mathrm{k}, \tag{5.12.26}$$

summarized in the following multiplication table:

	1	i	j	k
1	1	i	j	k
i	i	-1	k	$-\mathrm{j}$
j	j	$-\mathrm{k}$	-1	i
k	k	j	$-\mathrm{i}$	-1

(5.12.27)

We observe that i, j, and k are three square roots of -1. The second quaternion unit, i, is the familiar imaginary unit defining complex numbers and functions of a complex variable. An arbitrary quaternion is represented as

$$q = q_1 + q_2\,\mathrm{i} + q_3\,\mathrm{j} + q_4\,\mathrm{k}, \tag{5.12.28}$$

where q_1, q_2, q_3, q_4 are real components, q_1 is the real part, and q_2, q_3, q_4 are imaginary components. We see that a quaternion extends the notion of a complex number from two-dimensional to four-dimensional space.

5.12.10 *Summation, subtraction and multiplication*

Summation and subtraction follow the usual rules of component summation.

$$\begin{aligned} q^{\mathrm{A}} \pm q^{\mathrm{B}} = (q_1^{\mathrm{A}} \pm q_1^{\mathrm{B}}) + (q_2^{\mathrm{A}} \pm q_2^{\mathrm{B}})\,\mathrm{i} \\ +(q_3^{\mathrm{A}} \pm q_3^{\mathrm{B}})\,\mathrm{j} + (q_4^{\mathrm{A}} \pm q_4^{\mathrm{B}})\,\mathrm{k}. \end{aligned} \tag{5.12.29}$$

To add or subtract two quaternions, we perform corresponding operations on their components.

Multiplication follows the rules stated in (5.12.22), where $q^{\rm C} = q^{\rm A} q^{\rm B}$. The rules follow from those listed in (5.12.27) for the fundamental quaternion units.

5.12.11 Conjugate and reciprocal

The conjugates of the fundamental quaternion units, denoted by an asterisk, are

$$1^* = 1, \qquad {\rm i}^* = -{\rm i}, \qquad {\rm j}^* = -{\rm j}, \qquad {\rm k}^* = -{\rm k}. \tag{5.12.30}$$

The conjugate of an arbitrary quaternion is

$$q^* = q_1 - q_2 {\rm i} - q_3 {\rm j} - q_4 {\rm k}. \tag{5.12.31}$$

The square of the norm of the quaternion is given by

$$|q|^2 \equiv q\, q^* = q^* q = q_1^2 + q_2^2 + q_3^2 + q_4^2. \tag{5.12.32}$$

The inverse (reciprocal) of the quaternion is given by

$$q^{-1} = \frac{1}{|q|^2}\, q. \tag{5.12.33}$$

Division is defined in terms of multiplication by the reciprocal.

5.12.12 Scalar and vector parts

An arbitrary quaternion can be regarded as a doublet consisting of a scalar and a vector part,

$$q = q_1 + (\, q_2\, {\rm i} + q_3\, {\rm j} + q_4\, {\rm k}\,), \tag{5.12.34}$$

where the vector part is enclosed by the parentheses. We define $v_1 = q_2$, $v_2 = q_3$, and $v_3 = q_4$, and write

$$\mathbf{v} \equiv q_2\, {\rm i} + q_3\, {\rm i} q_3 + q_4\, {\rm k} = \, v_1\, {\rm i} + v_2\, {\rm j} + v_3\, {\rm k}. \tag{5.12.35}$$

The vector $\mathbf{v}$ is now regarded as the position vector in three-dimensional space.

5.12.13 Rotations

We introduce Euler's formula,

$$\mathrm{e}^{\mathrm{i}\theta} = \cos\theta + \mathrm{i}\,\sin\theta, \tag{5.12.36}$$

and compute the product

$$\mathrm{e}^{\mathrm{i}\theta}\,q = (\cos\theta + \mathrm{i}\,\sin\theta)(q_1 + q_2\,\mathrm{i} + q_3\,\mathrm{j} + q_4\,\mathrm{k}). \tag{5.12.37}$$

Expanding the product, we obtain

$$\begin{aligned}\mathrm{e}^{\mathrm{i}\theta}\,q = {}& c\,q_1 + c\,q_2\,\mathrm{i} + c\,q_3\,\mathrm{j} + c\,q_4\,\mathrm{k} \\ & + s\,q_1\,\mathrm{i} - s\,q_2 + s\,q_3\,(\mathrm{i}\times\mathrm{j}) + s\,q_4\,(\mathrm{i}\times\mathrm{k}),\end{aligned} \tag{5.12.38}$$

where $c = \cos\theta$, $s = \sin\theta$, and $\times$ indicates regular scalar multiplication. Setting $\mathrm{i}\times\mathrm{j} = \mathrm{k}$ and $\mathrm{i}\times\mathrm{k} = -\mathrm{j}$, and grouping similar terms we obtain

$$\begin{aligned}\mathrm{e}^{\mathrm{i}\theta}\,q = {}& c\,q_1 - s\,q_2 + (c\,q_2 + s\,q_1)\,\mathrm{i} + \\ & (c\,q_3 - s\,q_4)\,\mathrm{j} + (c\,q_4 + s\,q_3)\,\mathrm{k}.\end{aligned} \tag{5.12.39}$$

Working in a similar function, we find that

$$\begin{aligned}&\mathrm{e}^{\mathrm{i}\theta}\,q\,\mathrm{e}^{-\mathrm{i}\theta} = (\cos\theta + \mathrm{i}\,\sin\theta)\times q\times(\cos\theta - \mathrm{i}\,\sin\theta) \\ &= c^2 q_1 - cs\,q_2 + (c^2 q_2 + cs\,q_1)\,\mathrm{i} + (c^2 q_3 - cs\,q_4)\,\mathrm{j} + (c^2\,q_4 + cs\,q_3)\,\mathrm{k} \\ &-(cs\,q_1 - s^2 q_2)\,\mathrm{i} + (cs\,q_2 + s^2 q_1) + (cs\,q_3 - s^2 q_4)\,\mathrm{k} - (cs\,q_4 + s^2 q_3)\,\mathrm{j}.\end{aligned} \tag{5.12.40}$$

Simplifying, we find that

$$\begin{aligned}\mathrm{e}^{\mathrm{i}\theta}\,q\,\mathrm{e}^{-\mathrm{i}\theta} = {}& q_1 + q_2\,\mathrm{i} + \big((c^2 - s^2)\,q_3 - 2cs\,q_4\big)\,\mathrm{j} \\ & + \big((c^2 - s^2)\,q_4 + 2cs\,q_3\big)\,\mathrm{k}.\end{aligned} \tag{5.12.41}$$

Using trigonometric identities, we find that

$$\begin{aligned}\mathrm{e}^{\mathrm{i}\theta}\,q\,\mathrm{e}^{-\mathrm{i}\theta} = {}& q_1 + q_2\,\mathrm{i} + \big(\cos(2\theta)\,q_3 - \sin(2\theta)\,q_4\big)\,\mathrm{j} \\ & + \big(\cos(2\theta)\,q_4 + \sin(2\theta)\,q_3\big)\,\mathrm{k},\end{aligned} \tag{5.12.42}$$

which can be written as

$$\mathrm{e}^{\mathrm{i}\theta}\, q\, \mathrm{e}^{-\mathrm{i}\theta} = q_1 + \boldsymbol{\mathcal{R}}^{(\mathrm{i})}(2\,\theta)\cdot\mathbf{v}, \tag{5.12.43}$$

where

$$\boldsymbol{\mathcal{R}}^{(\mathrm{i})}(\varphi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \sin\varphi & \cos\varphi \end{bmatrix}, \tag{5.12.44}$$

is a rotation matrix around the i axis.

In summary, the operation $\mathrm{e}^{\mathrm{i}\theta}\, q\, \mathrm{e}^{-\mathrm{i}\theta}$ rotates the vector part of q around the i axis by an angle 2θ.

Working in a similar fashion, we find that

$$\mathrm{e}^{\mathrm{j}\theta}\, q\, \mathrm{e}^{-\mathrm{j}\theta} = q_1 + \boldsymbol{\mathcal{R}}^{(\mathrm{j})}(2\theta)\cdot\mathbf{v} \tag{5.12.45}$$

and

$$\mathrm{e}^{\mathrm{k}\theta}\, q\, \mathrm{e}^{-\mathrm{k}\theta} = q_1 + \boldsymbol{\mathcal{R}}^{(\mathrm{k})}(2\theta)\cdot\mathbf{v}, \tag{5.12.46}$$

where

$$\boldsymbol{\mathcal{R}}^{(\mathrm{j})}(\varphi) = \begin{bmatrix} \cos\varphi & 0 & \sin\varphi \\ 0 & 1 & 0 \\ -\sin\varphi & 0 & \cos\varphi \end{bmatrix} \tag{5.12.47}$$

and

$$\boldsymbol{\mathcal{R}}^{(\mathrm{k})}(\varphi) = \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.12.48}$$

are corresponding rotation matrices. The operation $\mathrm{e}^{\mathrm{j}\theta}\, q\, \mathrm{e}^{-\mathrm{j}\theta}$ rotates the vector component of q around the j axis by an angle 2θ, and the operation $\mathrm{e}^{\mathrm{k}\theta}\, q\, \mathrm{e}^{-\mathrm{k}\theta}$ rotates the vector component of q around the k axis by an angle 2θ.

Assume that α_{i}, α_{j}, and α_{k} are arbitrary real numbers satisfying

$$\alpha_{\mathrm{i}}^2 + \alpha_{\mathrm{j}}^2 + \alpha_{\mathrm{k}}^2 = 1. \tag{5.12.49}$$

By extending Euler's formula, we write

$$\mathrm{e}^{\boldsymbol{\alpha}\theta} \equiv \cos\theta + \sin\theta\,(\,\alpha_{\mathrm{i}}\,\mathrm{i} + \alpha_{\mathrm{j}}\,\mathrm{j} + \alpha_{\mathrm{k}}\,\mathrm{k}\,) \tag{5.12.50}$$

is a quaternion of unit length. The operation

$$\mathrm{e}^{\boldsymbol{\alpha}\theta}\, q\, \mathrm{e}^{-\boldsymbol{\alpha}\theta} = q_1 + \boldsymbol{\mathcal{R}}^{(\boldsymbol{\alpha})}(\varphi)\cdot\mathbf{v}, \tag{5.12.51}$$

rotates the vector component of q around the $\boldsymbol{\alpha}$ axis by an angle equal to $\varphi = 2\theta$, where

$$\begin{aligned}\boldsymbol{\mathcal{R}}^{(\boldsymbol{\alpha})}(\varphi) &= \cos\varphi\,\mathbf{I} \\ &+(1-\cos\varphi)\begin{bmatrix} \alpha_{\mathrm{i}}^2 & \alpha_{\mathrm{i}}\alpha_{\mathrm{j}} & \alpha_{\mathrm{i}}\alpha_{\mathrm{k}} \\ \alpha_{\mathrm{j}}\alpha_{\mathrm{i}} & \alpha_{\mathrm{j}}^2 & \alpha_{\mathrm{j}}\alpha_{\mathrm{k}} \\ \alpha_{\mathrm{i}}\alpha_{\mathrm{k}} & \alpha_{\mathrm{j}}\alpha_{\mathrm{k}} & \alpha_{\mathrm{k}}^2 \end{bmatrix} + \sin\varphi \begin{bmatrix} 0 & -\alpha_{\mathrm{k}} & \alpha_{\mathrm{j}} \\ \alpha_{\mathrm{k}} & 0 & -\alpha_{\mathrm{i}} \\ -\alpha_{\mathrm{j}} & \alpha_{\mathrm{i}} & 0 \end{bmatrix}\end{aligned} \tag{5.12.52}$$

is a rotation matrix, as discussed in Section 8.10.

5.12.14 *Encoding rotation*

We may introduce three real numbers, β_{i}, β_{j}, and β_{k}, such that

$$\alpha_{\mathrm{i}} = \frac{\beta_{\mathrm{i}}}{\beta}, \qquad \alpha_{\mathrm{j}} = \frac{\beta_{\mathrm{j}}}{\beta}, \qquad \alpha_{\mathrm{k}} = \frac{\beta_{\mathrm{k}}}{\beta}, \tag{5.12.53}$$

and

$$\beta \equiv \sqrt{\beta_{\mathrm{i}}^2 + \beta_{\mathrm{j}}^2 + \beta_{\mathrm{k}}^2} \tag{5.12.54}$$

so that (5.12.49) is satisfied, and another real number, β_{r}, subject to the restriction

$$\beta_{\mathrm{r}}^2 + \beta_{\mathrm{i}}^2 + \beta_{\mathrm{j}}^2 + \beta_{\mathrm{k}}^2 = 1, \tag{5.12.55}$$

such that

$$\sin\frac{\varphi}{2} = \sqrt{\beta_{\mathrm{i}}^2 + \beta_{\mathrm{j}}^2 + \beta_{\mathrm{k}}^2}, \qquad \cos\frac{\varphi}{2} = \beta_{\mathrm{r}}. \tag{5.12.56}$$

From trigonometry, we know that

$$\sin\varphi = 2\sin\frac{\varphi}{2}\cos\frac{\varphi}{2} = 2\beta_{\rm r}\sqrt{\beta_{\rm i}^2+\beta_{\rm j}^2+\beta_{\rm k}^2} \qquad (5.12.57)$$

and

$$\cos\varphi = 1 - 2\,\sin^2\frac{\varphi}{2} = 1 - 2\,\beta_{\rm i}^2 - 2\,\beta_{\rm j}^2 - 2\,\beta_{\rm k}^2. \qquad (5.12.58)$$

Substituting these expressions into (5.12.52) and simplifying, we obtain

$$\boldsymbol{\mathcal{R}}^{(\boldsymbol{\alpha})}(\varphi) = (1 - 2\,\beta_{\rm i}^2 - 2\,\beta_{\rm j}^2 - 2\,\beta_{\rm k}^2)\,\mathbf{I} \qquad (5.12.59)$$
$$+2\begin{bmatrix} \beta_{\rm i}^2 & \beta_{\rm i}\beta_{\rm j} & \beta_{\rm i}\beta_{\rm k} \\ \beta_{\rm j}\beta_{\rm i} & \beta_{\rm j}^2 & \beta_{\rm j}\beta_{\rm k} \\ \alpha_{\rm i}\beta_{\rm k} & \beta_{\rm j}\beta_{\rm k} & \beta_{\rm k}^2 \end{bmatrix} + 2\beta_{\rm r}\begin{bmatrix} 0 & -\beta_{\rm k} & \beta_{\rm j} \\ \beta_{\rm k} & 0 & -\beta_{\rm i} \\ -\beta_{\rm j} & \beta_{\rm i} & 0 \end{bmatrix}.$$

The first and second term on the right-hand side can be consolidated.

Exercise

5.12.1 Confirm the inverse given in (5.12.19).

Chapter 6

Numerical methods

Numerical methods offer invaluable tools for tackling small-size problems and for solving large-size problems that are intractable by analytical methods. The importance of numerical methods above and beyond the inner workings of numerical computational frameworks, software packages, and toolboxes cannot be overstated. Selected concepts, topics, and methods are discussed in this chapter.

6.1 Order of a method: operational count

Suppose that we need to compute a high power of an integer, such as 3^{128}. We can multiply 3 by 3, then multiply the product by 3, and repeat the process 127 times. The operation count of this method is $n-1$, where n is the exponent; in our example, $n = 128 = 2^7$.

A much faster method involves computing the second and fourth power of 3 by successive squaring,

$$3, \qquad 3^2 = 3 \times 3, \qquad 3^4 = 3^2 \times 3^2, \tag{6.1.1}$$

which involves 2 multiplications, and then setting $3^7 = 3 \times 3^2 \times 3^4$, which involves 2 additional multiplications, for a total of 4 multiplications.

6.1.1 Al-Kashi algorithm

To generalize the method, we consider computing the nth power of an arbitrary real number a,

$$b = a^n, \tag{6.1.2}$$

where n is an arbitrary positive integer. The Al-Kashi algorithm allows us to compute this power with the least number of multiplications according to the following pseudocode:

```
b = 1.0

while(n>0)
  if(n is even) then
    n → n/2
    a = a*a
  else
    n → n-1
    b = b*a
  end
end
```

The method is implemented in the following Matlab code involving the *modulus* function:

```
b = 1.0;

while(n>0)

  if(mod(n,2)==0)
   n = n/2;
   a = a*a;
  else
   n = n-1;
   b = b*a;
   end

end
```

The operational count of the method, n_c, is plotted in Figure 6.1.1 on a log-liner scale against the exponent, n. An operation is a multiplication or division. Because additions and subtractions are cheap, they are not counted.

The data in Figure 6.1.1 show that

$$n_c = 2 \ln_2 n + 1, \tag{6.1.3}$$

which is extremely competitive when compared to that of carrying out $n - 1$ multiplications. For $n = 2^7 = 128$, we may compare 15 against 127 operations necessary with the direct method.

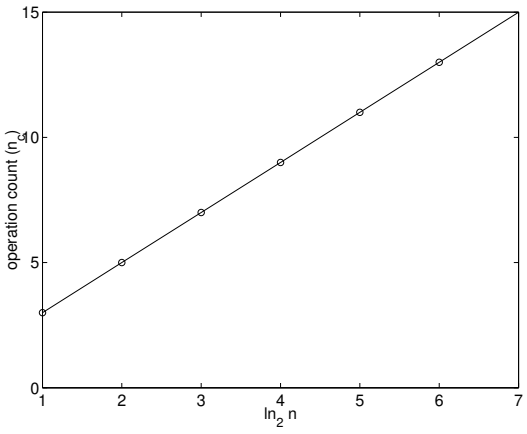


FIGURE 6.1.1 Operational count of the Al-Kashi algorithm including multiplications and divisions for raising a real number to an integer, p.

6.1.2 General guidelines

In a typical application, we identify an appropriate integer expressing the problem size, n, and consider the operation count involving multiplications and divisions. Depending on the available computer hardware, an operation count may also include additions and subtractions. For example, n can be the number of unknowns in a balanced system of linear algebraic equations.

The following general guidelines apply:

- If an algorithm involves a loop, and the range of the iteration loop variable is divided by a number of order one in each iteration, the operation count is logarithmic, that is, it is proportional to $\ln n$.
- If an algorithm involves q nested loops with fixed range, the operation count scales with n^q.
- Algorithms with order $n^q \ln n$ operations are possible.

It is evident that algorithms with logarithmic operation count are highly desirable.

For example, in Section 6.3 we will see that tridiagonal systems of n linear equations can be solved by the Thomas algorithm with an operation count on the order of n. General methods for solving systems of linear equations carry on operation count on the order of n^3. Fast converging iterative methods for solving systems of linear equations carry an operation count on the order of n^2.

6.1.3 Binary search

The method of binary search carries a logarithmic or lower operation count. In this method, we search through a sorted list of numbers or names with the objective of finding a particular entry and possibly attached information according to the following steps:

- Compare a target number or name with the number or name in the middle of the list.
- If the comparison places the target at the upper half of the list, abandon the bottom half of the list and repeat the process with the upper half.
- Otherwise, do the converse.
- Repeat.

The catch is that the list must be ordered before the search can be applied. However, once the list has been ordered, it can be searched multiple times with different targets.

Exercise

6.1.1 Compute the sum

$$\sum_{n=0}^{5} 3^n \tag{6.1.4}$$

with the least number of multiplications.

6.2 Systems of linear algebraic equations

In a wide variety of applications, we need to solve a system of N linear algebraic equations for N scalar unknowns, x_1, x_2, $\ldots$, x_N,

$$
\begin{aligned}
&A_{1,1}\,x_1 + A_{1,2}\,x_2 + \cdots + A_{1,N-1}\,x_{N-1} + A_{1,N}\,x_N = b_1,\\
&A_{2,1}\,x_1 + A_{2,2}\,x_2 + \cdots + A_{2,N-1}\,x_{N-1} + A_{2,N}\,x_N = b_2,\\
&\ldots\\
&A_{N-1,1}\,x_1 + A_{N-1,2}\,x_2 + \cdots + A_{N-1,N-1}\,x_{N-1} + A_{N-1,N}\,x_N = b_{N-1},\\
&A_{N-1,1}\,x_1 + A_{N-1,2}\,x_2 + \cdots + A_{N-1,N-1}\,x_{N-1} + A_{N-1,N}\,x_N = b_{N-1},\\
&A_{N,1}\,x_1 + A_{N,2}\,x_2 + \cdots + A_{N,N-1}\,x_{N-1} + A_{N,N}\,x_N = b_N,
\end{aligned}
\tag{6.2.1}
$$

where $A_{i,j}$ are given coefficients and b_i are given constants, for $i, j = 1, \ldots, N$.

In vector-matrix notation, the system (6.2.1) takes the compact form

$$
\mathbf{A}\cdot\mathbf{x} = \mathbf{b}, \tag{6.2.2}
$$

where $\mathbf{A}$ is the $N \times N$ coefficient matrix

$$
\mathbf{A} = \begin{bmatrix}
A_{1,1} & A_{1,2} & \ldots & A_{1,N-1} & A_{1,N}\\
A_{2,1} & A_{2,2} & \ldots & A_{2,N-1} & A_{2,N}\\
\ldots & \ldots & \ldots & \ldots & \ldots\\
A_{N-1,1} & A_{N-1,2} & \ldots & A_{N-1,N-1} & A_{N-1,N}\\
A_{N,1} & A_{N,2} & \ldots & A_{N,N-1} & A_{N,N}
\end{bmatrix}, \tag{6.2.3}
$$

$\mathbf{x}$ is the N-dimensional solution vector, and $\mathbf{b}$ is a specified N-dimensional vector defined as

$$
\mathbf{x} = \begin{bmatrix} x_1\\ x_2\\ \vdots\\ x_{N-1}\\ x_N \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} b_1\\ b_2\\ \vdots\\ b_{N-1}\\ b_N \end{bmatrix}. \tag{6.2.4}
$$

In practice, the matrix, $\mathbf{A}$, the solution vector, $\mathbf{x}$, and the right-hand side, $\mathbf{b}$, may have a variety of interpretations depending on the physical or mathematical context underlying the linear equations.

6.2.1 Numerical methods

A variety of direct and iterative, general and specialized methods are available for solving systems of linear equations. Direct methods generate the exact solution within computer round-off, whereas iterative methods improve an alleged solution up to a specified tolerance.

As a rule of thumb, the operational count of general methods scales with N^3, whereas the operational count of specialized methods scales with N. The operational count of general methods is lower for coefficient matrices, $\mathbf{A}$, with a special structure.

6.2.2 Lower triangular systems and forward substitution

When the coefficient matrix, $\mathbf{A}$, is lower triangular, denoted by $\mathbf{L}$, the linear system takes the form

$$\mathbf{L} \cdot \mathbf{x} = \mathbf{b}, \tag{6.2.5}$$

where

$$\mathbf{L} = \begin{bmatrix} L_{1,1} & 0 & 0 & \cdots & 0 & 0 & 0 \\ L_{2,1} & L_{2,2} & 0 & \cdots & 0 & 0 & 0 \\ L_{3,1} & L_{3,2} & L_{3,3} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ L_{N-2,1} & L_{N-2,2} & L_{N-2,3} & \cdots & L_{N-2,N-2} & 0 & 0 \\ L_{N-1,1} & L_{N-1,2} & L_{N-1,3} & \cdots & L_{N-1,N-2} & L_{N-1,N-1} & 0 \\ L_{N,1} & L_{N,2} & L_{N,3} & \cdots & L_{N,N-2} & L_{N,N-1} & L_{N,N} \end{bmatrix} \tag{6.2.6}$$

The first equation contains only the first unknown, the second equation contains the first and second unknowns, the ith equation contains the first i unknowns, and the last equation contains all unknowns.

Accordingly, we solve the first equation for the first unknown, which then becomes a known. At the second stage, we solve the second equation for the second unknown, which also becomes a known.

The process is repeated according to the forward substitution algorithm based on the formula

$$x_1 = \frac{b_1}{L_{N,N}}, \qquad x_i = \frac{1}{L_{i,i}} \left(b_i - \sum_{j=1}^{i-1} L_{i,j} x_j \right) \tag{6.2.7}$$

for $i = 2, \ldots, N$. The double loop implied in the second formula reveals that the operational count of the method is on the order of N^2.

6.2.3 *Upper triangular systems and backward substitution*

When the coefficient matrix, $\mathbf{A}$, is upper triangular, denoted by $\mathbf{U}$, the linear system takes the form

$$\mathbf{U} \cdot \mathbf{x} = \mathbf{b}, \tag{6.2.8}$$

where

$$\mathbf{U} = \begin{bmatrix} U_{1,1} & U_{1,2} & U_{1,3} & \cdots & U_{1,N-2} & U_{1,N-1} & U_{1,N} \\ 0 & U_{2,2} & U_{2,3} & \cdots & U_{2,N-2} & U_{2,N-1} & U_{2,N} \\ 0 & 0 & U_{3,3} & \cdots & U_{3,N-2} & U_{3,N-1} & U_{3,N} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & U_{N-2,N-2} & U_{N-2,N-1} & U_{N-2,N} \\ 0 & 0 & 0 & \cdots & 0 & U_{N-1,N-1} & U_{N-1,N} \\ 0 & 0 & 0 & \cdots & 0 & 0 & U_{N,N}. \end{bmatrix} \tag{6.2.9}$$

The last equation involves only the last unknown, the penultimate equation contains the last two unknowns, the ith equation contains the last i unknowns, and the first equation contains all unknowns. Accordingly, we solve the last equation for the last unknown, which then becomes a known. At the second stage, we solve the penultimate equation for the penultimate unknown, which also becomes a known.

The process is repeated according to the backward substitution algorithm based on the formula

$$x_N = \frac{b_N}{U_{n,N}}, \qquad x_i = \frac{1}{U_{i,i}} \left(b_i - \sum_{j=i+1}^{N} U_{i,j}\, x_j \right), \tag{6.2.10}$$

for $i = N-1, \ldots, 1$. The double loop implied in the second formula reveals that the operational count of the method is on the order of N^2.

6.2.4 LU decomposition

In the $\mathbf{LU}$ decomposition method, the coefficient matrix, $\mathbf{A}$, is first resolved into a product

$$\mathbf{A} = \mathbf{L} \cdot \mathbf{U}, \tag{6.2.11}$$

where $\mathbf{L}$ is a lower triangular matrix and $\mathbf{U}$ is an upper triangular matrix. Once the matrices $\mathbf{L}$ and $\mathbf{U}$ are available, the solution of the linear system $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$ or, equivalently,

$$\mathbf{L} \cdot \mathbf{U} \cdot \mathbf{x} = \mathbf{b}, \tag{6.2.12}$$

is found in two stages. First, we define

$$\mathbf{w} \equiv \mathbf{U} \cdot \mathbf{x}, \tag{6.2.13}$$

and solve the system

$$\mathbf{L} \cdot \mathbf{w} = \mathbf{b} \tag{6.2.14}$$

for $\mathbf{w}$ by forward substitution. Second, we solve the system

$$\mathbf{U} \cdot \mathbf{x} = \mathbf{w} \tag{6.2.15}$$

for $\mathbf{x}$ by backward substitution. Thus, combining an $\mathbf{LU}$ decomposition algorithm with the forward or backward substitution algorithms allows us to tackle arbitrary systems.

6.2.5 Matlab/Octave solvers

Matlab solves the linear system (6.2.2) using an efficient algorithm instructed by a function invoked by the statement

```
x = b/A';
```

or the equivalent statement

```
x = b\A;
```

In standard Matlab notation, a prime denotes the matrix transpose.

Exercise

6.2.1 Solve a linear system of your choice using the internal Matlab function and confirm the solution by ensuring that $\mathbf{A} \cdot \mathbf{x} - \mathbf{b} = \mathbf{0}$.

6.3 Thomas algorithm for tridiagonal systems

Consider a linear system of K algebraic equations in K unknowns encapsulated in a vector, $\mathbf{x}$,

$$\mathbf{D} \cdot \mathbf{x} = \mathbf{s}, \tag{6.3.1}$$

where $\mathbf{D}$ is a given $K \times K$ coefficient matrix and $\mathbf{s}$ is a given N-dimensional array. The coefficient matrix is assumed to have a tridiagonal form,

$$\mathbf{D} = \begin{bmatrix} a_1 & b_1 & 0 & \cdots & 0 & 0 & 0 \\ c_2 & a_2 & b_2 & \cdots & 0 & 0 & 0 \\ 0 & c_3 & a_3 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a_{K-2} & b_{K-2} & 0 \\ 0 & 0 & 0 & \cdots & c_{K-1} & a_{K-1} & b_{K-1} \\ 0 & 0 & 0 & \cdots & 0 & c_K & a_K \end{bmatrix}. \tag{6.3.2}$$

Note that only the diagonal, superdiagonal, and subdiagonal elements of $\mathbf{D}$ are nonzero. Such tridiagonal matrices arise from the numerical discretization of differential equations involving second derivatives, as discussed in Chapter 9. For convenience and to save computer memory, only the three nonzero diagonal lines of $\mathbf{T}$ are stored in one-dimensional arrays.

Thomas's algorithm proceeds in two stages. In the first stage, the tridiagonal system (6.3.1) is transformed into an upper bidiagonal system,

$$\mathbf{D}' \cdot \mathbf{x} = \mathbf{y}, \tag{6.3.3}$$

involving an upper bidiagonal coefficient matrix with *ones* along the diagonal,

$$\mathbf{D}' = \begin{bmatrix} 1 & d_1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & d_2 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & d_{K-2} & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 & d_{K-1} \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{bmatrix}, \tag{6.3.4}$$

where $\mathbf{y}$ is an intermediate solution vector.

In the second stage, the upper bidiagonal system (6.3.3) is solved by backward substitution, which involves solving the last equation for the last unknown, $x_K = y_K$, and then moving upward to compute the rest of the unknowns in a sequential fashion.

To develop the Thomas algorithm, we assume that we have managed to put the ith equation into the bidiagonal form

$$x_i + d_i\, x_{i+1} = y_i. \tag{6.3.5}$$

Solving for x_i and substituting the resulting expression in the $i+1$ equation of the original system,

$$c_{i+1}\, x_i + a_{i+1}\, x_{i+1} + b_{i+1}\, x_{i+2} = s_{i+1}, \tag{6.3.6}$$

we obtain

$$c_{i+1}\, (y_i - d_i\, x_{i+1}) + a_{i+1}\, x_{i+1} + b_{i+1}\, x_{i+2} = s_{i+1}. \tag{6.3.7}$$

Rearranging, we obtain

$$x_{i+1} + \frac{b_{i+1}}{a_{i+1} - d_i\, c_{i+1}}\, x_{i+2} = \frac{s_{i+1} - c_{i+1}\, y_i}{a_{i+1} - d_i\, c_{i+1}}. \tag{6.3.8}$$

Finally we recast this equation into the form

$$x_{i+1} + d_{i+1}\, x_{i+2} = y_{i+1}, \tag{6.3.9}$$

Reduction to bidiagonal :

$$\begin{bmatrix} d_1 \\ y_1 \end{bmatrix} = \frac{1}{a_1} \begin{bmatrix} b_1 \\ s_1 \end{bmatrix}$$

Do $i = 1, K-1$

$$\begin{bmatrix} d_{i+1} \\ y_{i+1} \end{bmatrix} = \frac{1}{a_{i+1} - c_{i+1} d_i} \begin{bmatrix} b_{i+1} \\ s_{i+1} - c_{i+1} y_i \end{bmatrix}$$

End Do

Backward substitution :

$x_K = y_K$
Do $i = K-1, 1 \quad (\text{step} = -1)$
$\quad x_i = y_i - d_i \, x_{i+1}$
End Do

TABLE 6.3.1 Thomas algorithm for solving a system of K linear equations with a tridiagonal coefficient matrix.

and derive the relations

$$d_{i+1} = \frac{b_{i+1}}{a_{i+1} - d_i \, c_{i+1}}, \qquad y_{i+1} = \frac{s_{i+1} - c_{i+1} \, y_i}{a_{i+1} - d_i \, c_{i+1}}. \tag{6.3.10}$$

Combining the reduction to the bidiagonal form with the backward substitution algorithm, we obtain the Thomas algorithm shown in Table 6.3.1.

The method is implemented in the following function named *thomas* located in directory THOMAS of SEMLIB:

```
function x = thomas (n,a,b,c,s)

%================================================
% Thomas algorithm for a tridiagonal system
%
% n:  system size
% a,b,c:  diagonal, superdiagonal,
%         and subdiagonal elements
% s:  right-hand side
%================================================

%------------------------------
% reduction to upper bidiagonal
%------------------------------

d(1) = b(1)/a(1);
y(1) = s(1)/a(1);

for i=1:n-2
  i1 = i+1;
  den = a(i1)-c(i1)*d(i);
  d(i1) = b(i1)/den;
  y(i1) = (s(i1)-c(i1)*y(i))/den;
end

den = a(n)-c(n)*d(n-1);
y(n) = (rhs(n)-c(n)*y(n-1))/den;

%-----------------
% back substitution
%-----------------

x(n) = y(n);

for i=n-1:-1:1
  x(i) = y(i)-d(i)*x(i+1);
end

%-----
```

```
% done
%-----

return;
```

An operation count and the observation that the algorithm involves single non-nested loops reveal that the necessary computational time scales linearly with the system size, K. By contrast, general purpose non-iterative methods requires computational time on the order of K^3, which can be demanding or even prohibitive.

Exercise

6.3.1 Use the Thomas algorithm to solve a system of 1024 equations of your choice and discuss the necessary computational time.

6.4 Data regression

A set of data obtained from an experiment or numerical computation involving random error can be fit to an Mth-degree approximating polynomial, denoted by

$$\Pi_M(x) = a_1\, x^M + a_2\, x^{M-1} + a_3\, x^{M-2} + \cdots + a_M\, x + a_{M+1}, \tag{6.4.1}$$

where a_i is a set of $M+1$ polynomial coefficients, as shown in Figure 6.4.1. The order of the polynomial, M, is selected according to physical intuition. I small value for M is chosen in practical applications. The process of fitting is called *data regression*.

In linear regression, we employ a first-degree polynomial,

$$\Pi_1(x) = a_1\, x + a_2, \tag{6.4.2}$$

involving two coefficients. In the case of quadratic regression, we employ a second-degree polynomial,

$$\Pi_2(x) = a_1\, x^2 + a_2\, x + a_3, \tag{6.4.3}$$

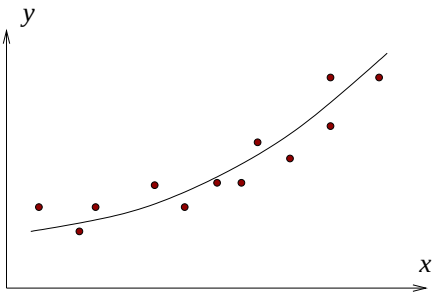


FIGURE 6.4.1 In data regression with least squares, the approximated function is represented by a polynomial or another suitable functional form.

involving three coefficients.

6.4.1 *Overdetermined system of equations*

Requiring that the polynomial interpolates the data, $\Pi_M(x_k) = y_k$, we obtain a system of $N+1$ linear equations for the $M+1$ polynomial coefficients,

$$\mathbf{A} \cdot \begin{bmatrix} a_{M+1} \\ a_M \\ \vdots \\ a_2 \\ a_1 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \\ y_{N+1} \end{bmatrix}, \tag{6.4.4}$$

where

$$\mathbf{A} = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^M & x_1^{M+1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^M & x_2^{M+1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_N & x_N^2 & \cdots & x_N^M & x_N^{M+1} \\ 1 & x_{N+1} & x_{N+1}^2 & \cdots & x_{N+1}^M & x_{N+1}^{M+1} \end{bmatrix} \tag{6.4.5}$$

is an $(N+1)\times(M+1)$ matrix. Since typically $N \geq M$, the linear system is overdetermined, that is, we have more equations than unknowns.

6.4.2 Normal equations

To solve a rectangular overdetermined linear system,

$$\mathbf{A} \cdot \mathbf{a} = \mathbf{y}, \tag{6.4.6}$$

we multiply both sides by the transpose of the $(N+1) \times (M+1)$ coefficient matrix $\mathbf{A}$ to obtain a system of $M+1$ equations in $M+1$ unknowns,

$$\mathbf{B} \cdot \mathbf{a} = \mathbf{A}^{\mathrm{T}} \cdot \mathbf{y}, \tag{6.4.7}$$

where

$$\mathbf{B} \equiv \mathbf{A}^{\mathrm{T}} \cdot \mathbf{A} \tag{6.4.8}$$

is a symmetric matrix, $\mathbf{a}$ is the vector on the left-hand side and $\mathbf{x}$ is the vector on the right-hand side of (6.4.4). Equation (6.4.7) encapsulates $M+1$ linear equations, known as the *normal equations*.

The solution of the normal equations is given by

$$\mathbf{a} = \mathbf{A}^{+} \cdot \mathbf{y}, \tag{6.4.9}$$

where

$$\mathbf{A}^{+} = \mathbf{B}^{-1} \cdot \mathbf{A}^{\mathrm{T}} = (\mathbf{A}^{\mathrm{T}} \cdot \mathbf{A})^{-1} \cdot \mathbf{A}^{\mathrm{T}} \tag{6.4.10}$$

is an $(M+1) \times (N+1)$ matrix.

An implied assumption is that the $(M+1) \times (M+1)$ matrix $\mathbf{A}^{\mathrm{T}} \cdot \mathbf{A}$ is nonsingular so that its inverse can be found. In turn, this requires that the $M+1$ columns of the matrix $\mathbf{A}$ are linearly independent. In our application, this will be true as long as the abscissas, x_i, are distinct.

From (6.4.10), we find that

$$\mathbf{A}^{+} \cdot \mathbf{A} = (\mathbf{A}^{\mathrm{T}} \cdot \mathbf{A})^{-1} \cdot \mathbf{A}^{\mathrm{T}} \cdot \mathbf{A} = \mathbf{I}, \tag{6.4.11}$$

where $\mathbf{I}$ is the $(M+1) \times (M+1)$ identity matrix. It would appear that the matrix $\mathbf{A}^{+}$ is the inverse of $\mathbf{A}$, except that the matrix $\mathbf{A}$ is not necessarily square.

With the polynomial coefficients computed in this fashion, as the order of the approximating polynomial M becomes higher, the approximating polynomial converges to the approximated function represented by the data uniformly inside the approximation interval.

6.4.3 Moore–Penrose inverse

The matrix $\mathbf{A}^+$ shown in (6.4.10) is the Moore–Penrose inverse of the matrix $\mathbf{A}$. Any rectangular or square singular or non-singular matrix, $\mathbf{A}$, has a unique Moore–Penrose inverse that can be defined as a certain limit,

$$\mathbf{A}^+ = \lim_{\epsilon \to 0} (\mathbf{A}^{\mathrm{T}} \cdot \mathbf{A} + \epsilon \mathbf{I})^{-1} \cdot \mathbf{A}^{\mathrm{T}}, \tag{6.4.12}$$

where $\mathbf{I}$ is the appropriate identity matrix. When the matrix $\mathbf{A}$ is square and invertible, the Moore–Penrose matrix inverse reduces to the familiar matrix inverse, $\mathbf{A}^{-1}$.

6.4.4 pinv

Matlab generates the Moore–Penrose inverse using the internal function `pinv`.

6.4.5 Data fitting code

The following Matlab code named *regression*, located in directory REGRET of SEMLIB, generates the Moore–Penrose inverse and solves a typical linear regression problem:

```
x = [ 0.03 0.3 0.43 0.55 0.59 0.67 0.95 ];
y = [ 0.07 0.4 1.03 1.20 1.50 1.3  1.60 ];

S = size(x');
N = S(1)-1;

M = 1;

%---
% generate the matrix A
%---
```

```
for i=1:N+1
 for j=1:M+1
  A(i,j) = x(i)^(j-1);
 end
end

%---
% Moore--Penrose inverse
%---

MP = inv(A'*A)*A';
MP

P = pinv(A);
P
sol = P*y';

%---
% plot
%---

figure(1)
plot(x,y,'ko')
if(M==1)
 a(1) = sol(2);
 a(2) = sol(1);
 yA = a(1)*x(1)+a(2);
 yB = a(1)*x(N+1)+a(2);
 plot([x(1),x(N+1)],[yA,yB],'k-')
end
```

Running the code generates the following output:

```
MP =
    0.6112  0.3438  0.2150  0.0962  0.0566 -0.0227 -0.3000
   -0.9313 -0.3995 -0.1435  0.0928  0.1716  0.3292  0.8806

P =
    0.6112  0.3438  0.2150  0.0962  0.0566 -0.0227 -0.3000
   -0.9313 -0.3995 -0.1435  0.0928  0.1716  0.3292  0.8806
```

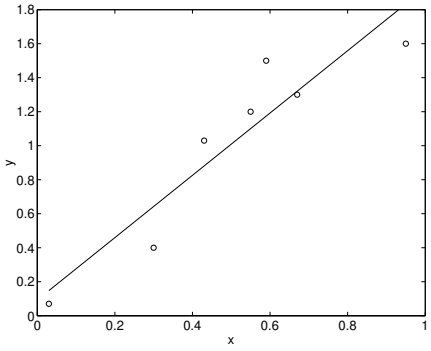


FIGURE 6.4.1 Typical linear regression of data computed by way of the Moore–Penrose inverse.

We confirm that the Moore–Penrose inverse generated by two different ways is the same. The graph generated by the code is shown in Figure 6.4.1.

6.4.6 Least-squares

In the least-squares method, the polynomial coefficients are computed by minimizing the sum of the squares of the differences between the $N+1$ data pairs, (x_k, y_j), and the corresponding polynomial values,

$$\mathcal{E} \equiv \sum_{k=1}^{N+1} \big(y_k - \Pi_M(x_k) \big)^2. \tag{6.4.13}$$

Substituting the expression for the polynomial, we obtain

$$\mathcal{E} \equiv \sum_{k=1}^{N+1} \big(y_k - a_1\, x_k^M - a_2\, x_k^{M-1} - a_3\, x_k^{M-2} - \cdots - a_M\, x_k - a_{M+1}\big)^2, \tag{6.4.14}$$

which can be written as

$$\mathcal{E} = \Big(y_k - \sum_{i=0}^{M} a_{M+1-i}\, x_k^i \Big)^2. \tag{6.4.15}$$

The last term in (6.4.14) corresponds to $i = 0$.

To minimize $\mathcal{E}$, we require that

$$\frac{\partial\,\mathcal{E}}{\partial\,a_{M+1-j}} = 0 \tag{6.4.16}$$

for $j = 0, \ldots, M$, yielding

$$\sum_{k=1}^{N+1} \frac{\partial}{\partial\,a_{M+1-j}} \Big(y_k - \sum_{i=0}^{M} a_{M+1-i}\,x_k^i \Big)^2 = 0, \tag{6.4.17}$$

where j is a free index.

Carrying out the differentiation, we find that

$$\begin{aligned} 2 \sum_{k=1}^{N+1} \Big[\Big(y_k - \sum_{i=0}^{M} a_{M+1-i}\,x_k^i \Big) \\ \times \frac{\partial}{\partial\,a_{M+1-j}} \Big(y_k - \big(\sum_{i=0}^{M} a_{M+1-i}\,x_k^i \big) \Big) \Big] = 0, \end{aligned} \tag{6.4.18}$$

and then

$$\sum_{k=1}^{N+1} \Big(y_k - \sum_{i=0}^{M} a_{M+1-i}\,x_k^j \Big)\, x_i^p = 0. \tag{6.4.19}$$

Rearranging, we derive a linear system of $M + 1$ equations for the polynomial coefficients,

$$\sum_{k=0}^{M} a_{M+1-i}\,x_k^{i+j} = \sum_{k=0}^{M} x_k^j y_k \tag{6.4.20}$$

for $j = 0, \ldots, M$. Explicitly, the linear system is given by

$$\mathbf{B} \cdot \begin{bmatrix} a_{M+1} \\ a_M \\ \vdots \\ a_2 \\ a_1 \end{bmatrix} = \begin{bmatrix} \sum_{k=1}^{N+1} y_k \\ \sum_{k=1}^{N+1} x_k\, y_k \\ \vdots \\ \sum_{k=1}^{N+1} x_k^{M-1}\, y_k \\ \sum_{k=1}^{N+1} x_k^{M}\, y_k \end{bmatrix}, \tag{6.4.21}$$

where

$$\mathbf{B} = \begin{bmatrix} N+1 & \sum_{k=1}^{N+1} x_k & \sum_{k=1}^{N+1} x_k^2 & \ldots & \sum_{k=1}^{N+1} x_k^{M} \\ \sum_{k=1}^{N+1} x_k & \sum_{k=1}^{N+1} x_k^2 & \sum_{k=1}^{N+1} x_k^3 & \cdots & \sum_{k=1}^{N+1} x_k^{M+1} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \sum_{k=1}^{N+1} x_k^{M-1} & \sum_{k=1}^{N+1} x_k^{M} & \sum_{k=1}^{N+1} x_k^{M+1} & \cdots & \sum_{k=1}^{N+1} x_k^{2M-1} \\ \sum_{k=1}^{N+1} x^{M} & \sum_{k=1}^{N+1} x_k^{M+1} & \sum_{k=1}^{N+1} x_k^{M+2} & \cdots & \sum_{k=1}^{N+1} x_k^{2M} \end{bmatrix}. \tag{6.4.22}$$

is a $(M+1)\times(M+1)$ square matrix. In fact, the linear system (6.4.21) is precisely the same as that shown in (6.4.7).

6.4.7 Choosing M

We have mentioned that linear and quadratic regression correspond, respectively, to polynomial degree $M = 1$ and 2. Although the method of least squares works well for small and moderate values of M, the linear system for the polynomial coefficients becomes nearly singular when M is higher than about five. In practice, oscillations are avoided by choosing M to be significantly less than the size of the data set, N.

Exercise

6.4.1 Perform a quadratic data regression for a data set of your choice.

6.5 Newton–Raphson method for one equation

A function of one independent variable, x, is a virtual machine that receives a number, x, and generates another number, $f(x)$. The number generation can be implemented by a mathematical formula, such as

$$f(x) = \exp(x), \tag{6.5.1}$$

or else arise from a simple or involved numerical computation, physical measurement, or quantitative deduction.

In quantum chemistry and other advanced computational science disciplines, generating a single function value may require months or years of computational time even on the fastest available supercomputer.

6.5.1 Roots

In many instances, we are interesting in finding a root of a nonlinear algebraic equation,

$$f(x) = 0, \tag{6.5.2}$$

where $f(x)$ is a given function. To find a root by the Newton–Raphson method, we make a guess for the root, $x^{(0)}$, and then apply the formula

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})} \tag{6.5.3}$$

for $k = 0, 1, \ldots$, where a prime denotes the derivative. We will see that the iterations are guaranteed to converge provided that the guess is close enough to the root.

When the function $f(x)$ is linear, $f(x) = ax + b$, where a and b are given coefficients, the Newton–Raphson formula provides us with the exact root after only one iteration.

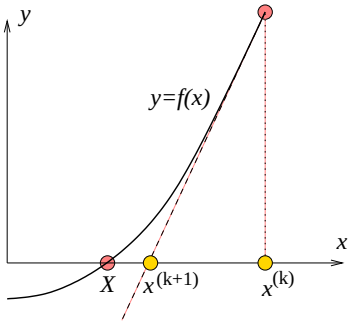


FIGURE 6.5.1 Graphical illustration of Newton's method. The graph of the function $f(x)$ is approximated with a straight line that is tangential at the current guess, $x^{(k)}$.

To derive formula (6.5.3), we draw the graph of the function $f(x)$ in the xy plane, and approximate the graph with a tangential straight line at the point $x^{(k)}$, as shown in Figure 6.5.1. Using elementary calculus, we find that the tangential line is described by the equation

$$y = f(x^{(k)}) + f'(x^{(k)})\,(x - x^{(k)}). \tag{6.5.4}$$

Next, we identify $x^{(k+1)}$ with the intersection of this line and the x axis by setting $y = 0$, and rearrange to obtain formula (6.5.3).

6.5.2 Convergence

To assess whether the method provides us with a convergent sequence, we assume that $x^{(k)}$ is close enough to the root, $x = r$, and rearrange (6.5.3) using Taylor series expansions into

$$x^{(k+1)} - r \simeq x^{(k)} - r - \frac{f(r) + f'(r)(x^{(k)} - r) + \frac{1}{2}f''(r)(x^{(k)} - r)^2 + \cdots}{f'(r) + f''(r)(x^{(k)} - r) + \cdots}. \tag{6.5.5}$$

Recalling that, by definition, $f(r) = 0$, and rearranging, we obtain

$$x^{(k+1)} - r \simeq (x^{(k)} - r)\left(1 - \frac{f'(r) + \frac{1}{2}f''(r)(x^{(k)} - r) + \cdots}{f'(r) + f''(r)(x^{(k)} - r) + \cdots}\right). \tag{6.5.6}$$

Simplifying the term inside the tall parentheses, we obtain

$$x^{(k+1)} - r \simeq (x^{(k)} - r)^2 \left(\frac{\frac{1}{2} f''(r) + \cdots}{f'(r) + f''(r)(x^{(k)} - r) + \cdots} \right), \quad (6.5.7)$$

yielding

$$x^{(k+1)} - r \simeq (x^{(k)} - r)^2 \, \frac{1}{2} \frac{f''(r)}{f'(r)}. \quad (6.5.8)$$

If $f'(r)$ is not zero, the current difference between the numerical solution and the exact solution, $x^{(k)} - r$, is squared and then multiplied by the numerical factor

$$\frac{1}{2} \frac{f''(r)}{f'(r)} \quad (6.5.9)$$

to yield the next difference between the numerical and the exact solution, $x^{(k+1)} - r$. This quadratic dependence guarantees that the sequence will converge as long as the initial guess is sufficiently close to the root.

6.5.3 *Computer code*

The following Matlab function entitled *newton*, located in directory NEWTON of SEMLIB, implements the method. The initial guess is provided in the input and the derivative of the function is computed by a finite-difference approximation with a sufficiently small differentiation step:

```
function [x,f,Iflag] = newton ...
   ...
  (Niter ...
  ,eps ...
  ,x ...
  ,italk ...
  )

%=====================
% Solve one nonlinear equation f(x) = 0
% by the second-order Newton method
```

```
%
% eps: numerical differentiation step
% Df:  derivative f'
% tol: tolerance
%======================

   tol = 0.0000001;

%---------------------
% start the iterations
%---------------------

    for i=1:Niter

       f  = newton_fun(x);

       x = x + eps;     % derivative by finite differences
       f1 = newton_fun(x1);
       x = x - eps;     % reset
       Df = (f1-f)/eps;

       Dx = -f/Df;
       x  = x+Dx;

       if(italk==1)
         format long; disp([x,f]); format short
       end

       iescape = 1;
       if(abs(Dx) > tol) iescape = 0; end

       if(iescape==1)
        Iflag = 0;
        f = newton_fun(x);
        return
       end

    end
```

```
%---
% done
%---

 return
```

This function is accompanied by another function named *newton_fun* that evaluates $f(x)$. An example is the function:

```
 function f = newton_fun(x)

  f = e^x + x - 0.1;

 return
```

Parameters can be passed by global statements, though the use of globals is considered anathema by many.

The following Matlab code calls *newton* to solve the equation implemented in function *newton_fun* listed above:

```
%===========
% Driver for one nonlinear equation
% using the second-order Newton's method
%===========

%---
% parameters
%---

  Niter = 10;
  eps = 0.00001;
  italk = 0;

  x = 1.0;  % initial guess

  [x,f,Iflag] = newton ...
     ...
    (Niter ...
    ,eps ...
```

```
    ,x ...
    ,italk ...
    );

%---
% print the root
%---

x
```

Running the code generates the following output:

```
x = -0.50407
```

Thus, $x = -0.50407$ is a root of the function $f(x) = e^x + x - 0.1$.

6.5.4 Improvements and extensions

Various improvements and interesting applications of the basic method discussed in this section are possible. For example, the method can be used to implement number division in terms of number multiplication.

6.5.5 Division by multiplication

To compute the ratio of two numbers, $r = a/b$, we rewrite it as $r = ax$, where $x = 1/b$, and identify x with the solution of the nonlinear algebraic equation $f(x) \equiv b - 1/x = 0$. The corresponding Newton iteration function is

$$g(x) = x - \frac{b - 1/x}{1/x^2} = x\,(2 - b\,x). \tag{6.5.10}$$

Thus, the inverse of the number b arises as the limit of the sequence

$$x^{(k+1)} = x^{(k)}\,(2 - b\,x^{(k)}), \tag{6.5.11}$$

whose computation does not involve division. The iterations will converge as long as

$$|x^{(0)} - \frac{1}{b}| < \frac{1}{b}. \tag{6.5.12}$$

Thus, if b is positive, a necessary and sufficient condition for the sequence to convergence is that the initial guess, $x^{(0)}$, lies in the interval $(0, 2/b)$.

Exercises

6.5.1 Prove that, for a linear function, $f(x) = ax + b$, where a and b are given coefficients, the Newton–Raphson formula provides us with the exact root after only one iteration.

6.5.2 Use the Newton–Raphson method to find the smallest positive root of the function $f(x) = \sin(x) - x^2$.

6.6 Newton–Raphson method for two equations

We are interesting in finding a solution of two nonlinear algebraic equations,

$$f_1(x_1, x_2) = 0, \qquad f_2(x_1, x_2) = 0, \tag{6.6.1}$$

where $f_1(x_1, x_2)$ and $f_2(x_1, x_2)$ are two given functions of the independent variables x_1 and x_2.

6.6.1 Newton–Raphson algorithm

To find a solution by the Newton–Raphson method, we make a guess for the root, $(x_1^{(0)}, x_2^{(0)})$, and then make corrections according to the formulas

$$\begin{aligned} x_1^{(k+1)} &= x_1^{(k)} + \Delta x_1^{(k)}, \\ x_2^{(k+1)} &= x_2^{(k)} + \Delta x_2^{(k)} \end{aligned} \tag{6.6.2}$$

for $k = 0, 1, \ldots$. The corrections are computed by solving a system of two linear equations,

$$\mathbf{J}^{(k)} \cdot \Delta\mathbf{x}^{(k)} = -\mathbf{f}^{(k)}. \tag{6.6.3}$$

where $\mathbf{J}^{(k)}$ is the Jacobian matrix with elements

$$J_{ij} = \frac{\partial f_i}{\partial x_j} \tag{6.6.4}$$

for $i, j = 1, 2$ evaluated at $\mathbf{x}^{(k)}$,

$$\Delta\mathbf{x}^{(k)} = \big(\Delta x_1^{(k)}, \Delta x_2^{(k)} \big) \tag{6.6.5}$$

is a correction vector, and

$$\mathbf{f}^{(k)} = \big(f_1^{(k)}, f_2^{(k)} \big). \tag{6.6.6}$$

The iterations are guaranteed to converge, provided that the initial guess is sufficiently close to the root doublet.

6.6.2 Derivation

The derivation of formula (6.6.3) is similar to that discussed in Section 6.5 for one equation. First, we expand each function in a Taylor series about the current guess,

$$f_1(x_1, x_2) \simeq f_1(x_1^{(k)}, x_2^{(k)}) + \Big(\frac{\partial f_1}{\partial x_1}\Big)_{x_1^{(k)}, x_2^{(k)}} (x_1 - x_1^{(k)}) + \Big(\frac{\partial f_1}{\partial x_2}\Big)_{x_1^{(k)}, x_2^{(k)}} (x_2 - x_2^{(k)}) \tag{6.6.7}$$

and

$$f_2(x_1, x_2) \simeq f_2(x_1^{(k)}, x_2^{(k)}) + \Big(\frac{\partial f_2}{\partial x_1}\Big)_{x_1^{(k)}, x_2^{(k)}} (x_1 - x_1^{(k)}) + \Big(\frac{\partial f_2}{\partial x_2}\Big)_{x_1^{(k)}, x_2^{(k)}} (x_2 - x_2^{(k)}). \tag{6.6.8}$$

Next, we set $f_1(x_1, x_2)$ and $f_2(x_1, x_2)$ on the left-hand sides to zero and identify x_1 with $x_1^{(k+1)}$ and x_2 with $x_2^{(k+1)}$ on the right-hand sides. Rearranging, we obtain the two equations in (6.6.3).

6.6.3 Computer code

The following Matlab function, located in directory NEWTON2 of SEMLIB, implements the method. The initial guess is provided in the input arguments. The Jacobian matrix is computed by a finite-difference approximation:

```
 function [x,f,Iflag] = newton2 ...
     ...
    (Niter ...
    ,eps ...
    ,x ...
    )

%-------------------------------------------------
%  Newton's method for two nonlinear equations
%
%  SYMBOLS:
%  --------
%
%  eps:   small interval for computing the Jacobian
%      by numerical differentiation
%  Dx:    correction vector
%  tol:   accuracy
%  Iflag: will set equal to 1 if something is wrong
%-------------------------------------------------

   tol = 0.0000001;

%-----------
% initialize
%-----------

   Iflag = 1;

%---------------------
% start the iterations
%---------------------

   for Iter=1:Niter

    f = fun2(x);

%---------------------
% compute the Jacobian
% by numerical differentiation
```

```
%---------------------

   for j=1:2
     x(j) = x(j)+eps;      % perturb
     f1 = fun2(x);
     x(j) = x(j)-eps;      % reset
     for i=1:2
        Jac(i,j) = (f1(i)-f(i))/eps;
     end
   end

%---
% solve the equation: Jac . Dx = - f
% for the correction vector Dx
% by Cramer's rule
%---

      b1  = -f(1);
      b2  = -f(2);
      Det = Jac(1,1)*Jac(2,2)-Jac(1,2)*Jac(2,1);
      dx(1) = (b1*Jac(2,2)-Jac(1,2)*b2)/Det;
      dx(2) = (b2*Jac(1,1)-Jac(2,1)*b1)/Det;

%--------
% correct
%--------

     x(1) = x(1) + dx(1);
     x(2) = x(2) + dx(2);

%-------
% escape
%-------

      iescape = 1;
      if(abs(dx(1)) > tol) iescape = 0; end
      if(abs(dx(2)) > tol) iescape = 0; end

      if(iescape==1)
```

```
       Iflag = 0;
       f = fun2(x);
       return
      end

%----
 end  % of iterations
%----

 return
```

This function must be accompanied by another function entitled *fun2* that evaluates $f_1(x_1, x_2)$. and $f_2(x_1, x_2)$. An example is the function:

```
function f = fun2(x)

  f(1) = x(1)^2+2.0*x(2)^2 -9.0;
  f(2) = x(1)*x(2) -2.0;

return
```

The following Matlab code, located in directory NEWTON2 of SEMLIB, calls this function to compute a root doublet:

```
%---------
% driver for newton's method
% for two equations
%---------

Niter = 8;
eps = 0.001;

%---
% initial guess
%---

x(1) = 0.1;
x(2) = 0.2;

%---
```

```
% newton
%---

 [x,f,Iflag] = newton2 ...
     ...
     (Niter ...
     ,eps ...
     ,x ...
     )

%---
% done
%---
```

Running the code generates the following output:

```
x =
   1.00000   2.00000
f =
   6.4162e-12  -4.4409e-16
Iflag = 0
```

We may confirm that $x_1 = 1$ and $x_2 = 1$ are roots of the two functions implemented in *fun2*.

6.6.4 Zero of a function of a complex variable

We may identity f_1 with the real part, $f_{\rm R}$, and f_2 with the imaginary part, $f_{\rm I}$, of a function of a complex variable, $f(z)$, where $z = x + {\rm i}\, y$. These functions satisfy the Cauchy–Riemann equations,

$$\frac{\partial f_{\rm R}}{\partial x} = \frac{\partial f_{\rm I}}{\partial y}, \qquad \frac{\partial f_{\rm R}}{\partial y} = -\frac{\partial f_{\rm I}}{\partial x}. \tag{6.6.9}$$

The Jacobian matrix is given by

$$\mathbf{J} = \begin{bmatrix} \dfrac{\partial f_{\rm R}}{\partial x} & \dfrac{\partial f_{\rm R}}{\partial y} \\ \\ -\dfrac{\partial f_{\rm R}}{\partial y} & \dfrac{\partial f_{\rm R}}{\partial x} \end{bmatrix}. \tag{6.6.10}$$

The determinant of the Jacobian matrix is given by

$$\det(\mathbf{J}) = \left(\frac{\partial f_{\mathrm{R}}}{\partial x}\right)^2 + \left(\frac{\partial f_{\mathrm{R}}}{\partial y}\right)^2 = \left(\frac{\partial f_{\mathrm{I}}}{\partial x}\right)^2 + \left(\frac{\partial f_{\mathrm{I}}}{\partial y}\right)^2. \quad (6.6.11)$$

Exercises

6.6.1 Show that, for a system of two linear equations, the Newton–Raphson method provides us with the exact solution after only one iteration.

6.6.2 Use the code discussed in the text to find a solution of the equations

$$\begin{aligned} f_1(x_1x_2) &= \sin(x_1) - x_2^2 = 0, \\ f_2(x_1x_2) &= \sin(x_2) - x_1^2 = 0. \end{aligned} \quad (6.6.12)$$

6.7 Fourth-order Runge–Kutta method

Consider N functions, $x_i(t)$ for $i = 1, \ldots, N$, governed by a system of N linear ordinary differential equations,

$$\frac{\mathrm{d}x_i}{\mathrm{d}t} = A_{ij}x_j - b_i, \quad (6.7.1)$$

where A_{ij} and b_i are constants and summation is implied over the repeated index, j. In vector notation,

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{A}\cdot\mathbf{x} - \mathbf{b}. \quad (6.7.2)$$

The fixed point of this system is the vector, $\mathbf{X}$, satisfying

$$\mathbf{A}\cdot\mathbf{X} = \mathbf{b}. \quad (6.7.3)$$

6.7.1 Exact solution

The exact solution of the linear differential equations is given by

$$\mathbf{x}(t) = \mathbf{X} + \mathbf{exp}(t\,\mathbf{A}) \cdot (\mathbf{x}^{(0)} - \mathbf{X}), \tag{6.7.4}$$

where $\mathbf{x}^{(0)} \equiv \mathbf{x}(t=0)$ is the initial condition. The computation of the matrix exponential, $\mathbf{exp}(t\,\mathbf{A})$, requires the availability of all eigenvalues and eigenvectors or principal vectors (generalized eigenvectors) of $\mathbf{A}$.

As an alternative, we consider the solution at time, t, and note that, after a time interval has elapsed, Δt, the solution will be given by

$$\mathbf{x}(t+\Delta t) = \mathbf{X} + \mathbf{exp}(\Delta t\,\mathbf{A})\,\mathbf{exp}(t\,\mathbf{A}) \cdot (\mathbf{x}^{(0)} - \mathbf{X}), \tag{6.7.5}$$

which is equivalent to

$$\mathbf{x}(t+\Delta t) = \mathbf{X} + \mathbf{exp}(\Delta t\,\mathbf{A}) \cdot (\mathbf{x}(t) - \mathbf{X}). \tag{6.7.6}$$

To simplify the notation, we denote

$$t = t_k, \quad t + \Delta t = t_{k+1}, \quad \mathbf{x}(t) = \mathbf{x}^{(k)}, \quad \mathbf{x}(t+\Delta t) = \mathbf{x}^{(k+1)}, \tag{6.7.7}$$

where the superscript (k) is not an exponent, but signifies instead the kth step of the numerical computation. Equation (6.7.6) takes the simpler form

$$\mathbf{x}^{(k+1)} = \mathbf{X} + \mathbf{exp}(\Delta t\,\mathbf{A}) \cdot (\mathbf{x}^{(k)} - \mathbf{X}). \tag{6.7.8}$$

When Δt is sufficiently small, the matrix $\mathbf{exp}(\Delta t\,\mathbf{A})$ can be approximated with a polynomial or rational function. Equation (6.7.8) then allows us to compute $\mathbf{x}(t+\Delta t)$ in terms of $\mathbf{x}(t)$ by carrying out a matrix-vector multiplications.

6.7.2 Approximation of the exponential

Approximating the exponential with a fourth-order Maclaurin sum,

$$\mathbf{exp}(\Delta t\,\mathbf{A}) \simeq \mathbf{P}_4, \tag{6.7.9}$$

where

$$\mathbf{P}_4 \equiv \mathbf{I} + \Delta t\,\mathbf{A} + \tfrac{1}{2}\,\Delta t^2\,\mathbf{A}^2 + \tfrac{1}{6}\,\Delta t^3\,\mathbf{A}^3 + \tfrac{1}{24}\,\Delta t^4\,\mathbf{A}^4, \quad (6.7.10)$$

we obtain the fourth-order Runge–Kutta method implemented according to the formula

$$\mathbf{x}^{(k+1)} = \mathbf{X} + \mathbf{P}_4 \cdot (\mathbf{x}^{(k)} - \mathbf{X}), \quad (6.7.11)$$

Writing $\mathbf{X} = \mathbf{A}^{-1} \cdot \mathbf{b}$, we obtain

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \Delta\mathbf{x}, \quad (6.7.12)$$

where

$$\Delta\mathbf{x} \equiv \Delta t\,(\mathbf{I} + \tfrac{1}{2}\,\Delta t\,\mathbf{A} + \tfrac{1}{6}\,\Delta t^2\mathbf{A}^2 + \tfrac{1}{24}\,\Delta t^3\mathbf{A}^3) \cdot (\mathbf{A}\,\mathbf{x}^{(k)} - \mathbf{b}). \quad (6.7.13)$$

The accumulated error is on the order of Δt^4.

6.7.3 *Predictions and corrections*

The same result is obtained by proceeding in a predictor–corrector sense, making three provisional steps and one final educated step according to the fourth-order Runge-Kutta algorithm:

1. Compute: $\mathbf{f}(\mathbf{x}^{(k)}) = \mathbf{A}\,\mathbf{x}^{(k)} - \mathbf{b}$
2. Set: $\mathbf{x}^{tmp1} = \mathbf{x}^{(k)} + \tfrac{1}{2}\,\Delta t\,\mathbf{f}(\mathbf{x}^{(k)})$
3. Compute: $\mathbf{f}^{tmp1} = \mathbf{f}(\mathbf{x}^{tmp1}) = \mathbf{A}\mathbf{x}^{tmp1} - \mathbf{b}$
4. Set: $\mathbf{x}^{tmp2} = \mathbf{x}^{(k)} + \tfrac{1}{2}\,\Delta t\,\mathbf{f}^{tmp1}$
5. Compute: $\mathbf{f}^{tmp2} = \mathbf{f}(\mathbf{x}^{tmp2}) = \mathbf{A}\,\mathbf{x}^{tmp2} - \mathbf{b}$
6. Set: $\mathbf{x}^{tmp3} = \mathbf{x}_{(k)} + \Delta t\,\mathbf{f}^{tmp2}$
7. Compute: $\mathbf{f}^{tmp3} = \mathbf{f}(\mathbf{x}^{tmp3}) = \mathbf{A}\,\mathbf{x}^{tmp3} - \mathbf{b}$
8. Set: $\mathbf{f}^{final} = \tfrac{1}{6}\,(\,\mathbf{f}(\mathbf{x}^{(k)}) + 2\,\mathbf{f}^{tmp1} + 2\,\mathbf{f}^{tmp2} + \mathbf{f}^{tmp3}\,)$

9. Advance: $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \Delta t\, \mathbf{f}^{final}$

6.7.4 Generalization

The method can be generalized to nonlinear and non-autonomous systems of differential equations,

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}, t), \tag{6.7.14}$$

where $\mathbf{f}$ is an arbitrary function of $\mathbf{x}$ and t. The generalized algorithm involves the following steps:

1. Compute: $\mathbf{f}(\mathbf{x}^{(k)}, t^{(k)})$
2. Set: $\mathbf{x}^{tmp1} = \mathbf{x}^{(k)} + \frac{1}{2}\Delta t\, \mathbf{f}(\mathbf{x}^{(k)}, t^{(k)})$
3. Compute: $\mathbf{f}^{tmp1} = \mathbf{f}(\mathbf{x}^{tmp1}, t^{(k)} + \frac{1}{2}\Delta t)$
4. Set: $\mathbf{x}^{tmp2} = \mathbf{x}^{(k)} + \frac{1}{2}\Delta t\, \mathbf{f}^{tmp1}$
5. Compute: $\mathbf{f}^{tmp2} = \mathbf{f}(\mathbf{x}^{tmp2}, t^{(k)} + \frac{1}{2}\Delta t)$
6. Set: $\mathbf{x}^{tmp3} = \mathbf{x}_{(k)} + \Delta t\, \mathbf{f}^{tmp2}$
7. Compute: $\mathbf{f}^{tmp3} = \mathbf{f}(\mathbf{x}^{tmp3}, t^{(k)} + \Delta t)$
8. Set: $\mathbf{f}^{final} = \frac{1}{6}\,(\,\mathbf{f}(\mathbf{x}^{(k)}, t^{(k)}) + 2\,\mathbf{f}^{tmp1} + 2\,\mathbf{f}^{tmp2} + \mathbf{f}^{tmp3}\,)$
9. Advance: $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \Delta t\, \mathbf{f}^{final}$

6.7.5 Computer code

The following Matlab code, located in directory RK4 of SEMLIB, implements the method:

```
function [t, xarray] = RK4 ...
  ...
   (menu ...
   ,N ...
   ,Nstep ...
```

```
   ,Dt ...
   ,time0 ...
   ,x0 ...
   )

%========================
% solve a system of N odes
% by RK4
%
% N: number of equations
% Nstep: number of steps
%========================

%---
% prepare
%---

 Dth = 0.5*Dt;

%---
% initial condition
%---

  for i=1:N
   x(i) = x0(i);
  end

%---
% integrate
%---

 time = time0;
 t(1) = time;

 for i=1:N
    xarray(i,1) = x0(i);
 end
```

```
for i=1:Nstep

     for j=1:N
      xsave(j) = x(j);
     end

     f0 = RK4_vel (N,menu,x,time);

     for j=1:N
       x(j) = xsave(j) + Dth*f0(j);
     end

     f1 = RK4_vel (N,menu,x,time+Dth);

     for j=1:N
       x(j) = xsave(j) + Dth*f1(j);
     end

     f2 = RK4_vel (N,menu,x,time+Dth);

     for j=1:N
       x(j) = xsave(j) + Dt*f2(j);
     end

     f3 = RK4_vel (N,menu,x,time+Dt);

     for j=1:N
       vel = (f0(j) + 2.0*f1(j) + 2.0*f2(j) + f3(j))/6.0;
       x(j) = xsave(j) + Dt*vel;
     end

     for j=1:N
      xarray(j,i+1) = x(j);
     end

     time = time+Dt;
```

```
      t(i+1) = time;

 end

%---
% done
%---

return
```

The code makes a call to the following function that evaluates the function $\mathbf{f}$ on the right-hand side of (6.7.14) for a typical case:

```
function f = RK4_vel (N,menu,x,t);

 if(menu==1)
   N = 3;
   f(1) = x(1)*t;
   f(2) = x(1)*x(2)*t^2;
   f(3) = x(1)*x(2)*x(3)*t^3;
 end

%---
% done
%---

return
```

The following code calls the function *RK4* to integrate a system of three ODEs and plot the solution:

```
menu = 1;
N = 3;

time0 = 0.0;

x0(1) = 0.8;
x0(2) = 1.2;
```

```
x0(3) = 1.5;

Nstep = 64;
Dt = 0.02;

[time, solution] = RK4 ...
  ...
   (menu ...
   ,N ...
   ,Nstep ...
   ,Dt ...
   ,time0 ...
   ,x0 ...
   );

%---
% plot the solution
%---

figure(1)
hold on
plot(time,solution(1,:),'ko-')
plot(time,solution(2,:),'bs-')
plot(time,solution(3,:),'rd-')
```

The three functions constituting the solution are plotted in Figure 6.7.1.

6.7.6 Deflated spheres and capsules

Consider a deflated hollow sphere or liquid capsule enclosed by an axisymmetric membrane whose shape is generated by rotating a curve around the x axis. To describe the profile of the membrane, we introduce cylindrical polar coordinates consisting of the axial position, x, the distance from the x axis, σ, and the azimuthal angle measured around the x axis with origin in the xy plane, φ, as shown in Figure 5.18.1.

The axisymmetric shape can be parametrized in terms of the arc length measured along the contour of the membrane in an azimuthal

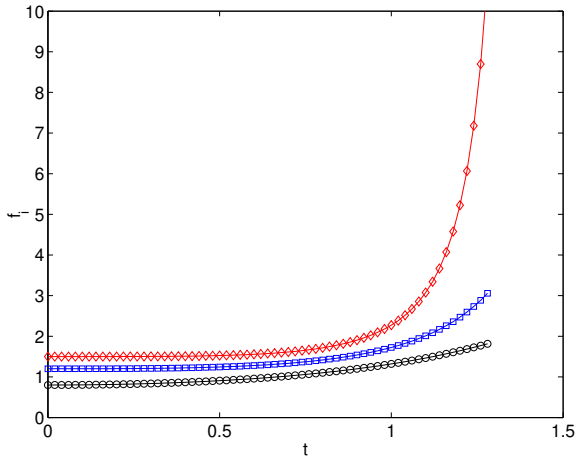


FIGURE 6.7.1 Solution of a system of three ordinary differential equations by the fourth-order Runge–Kutta method.

plane, ℓ. The principal curvatures of the interface in an azimuthal and its conjugate plane are denoted by κ_ℓ and κ_φ.

To compute the capsule profile from a specified curvature distribution, $\kappa_\ell(\ell)$, we regard the x and σ coordinates of point particles along the trace of the membrane in an azimuthal plane as functions of the arc length ℓ, writing

$$x = x_1(\ell) \qquad \sigma = x_2(\ell). \tag{6.7.15}$$

By definition then, $x_1'^2 + x_2'^2 = 1$, which can be differentiated to yield $x_1'\, x_1'' = -x_2'\, x_2''$, where a prime denotes a derivative with respect to arc length, ℓ. Using elementary differential geometry, we derive the relations

$$\kappa_\ell = -x_1'' x_2' + x_1' x_2'' = -\frac{x_1''}{x_2'} = \frac{x_2''}{x_1'}. \tag{6.7.16}$$

Next, we introduce the functions $x_3 \equiv x_1'$ and $x_4 \equiv x_2'$ satisfying $x_3^2 + x_4^2 = 1$, and obtain the following system of four nonlinear

differential equations:

$$\frac{\mathrm{d}x_1}{\mathrm{d}\ell} = x_3, \qquad \frac{\mathrm{d}x_2}{\mathrm{d}\ell} = x_4, \qquad \frac{\mathrm{d}x_3}{\mathrm{d}\ell} = -\kappa_\ell\, x_4, \qquad \frac{\mathrm{d}x_4}{\mathrm{d}\ell} = \kappa_\ell\, x_3. \tag{6.7.17}$$

The second pair of equations is decoupled from the first pair and can be integrated independently. Once the solution has been found, the first pair can be integrated to produce the shell shape. The meridional curvature is given by

$$\kappa_\varphi = -\frac{x_3}{x_2}. \tag{6.7.18}$$

The right-hand sides of equations (6.7.17) comprise the components of the *phase space velocity*. The following Matlab code, located in directory BIC of SEMLIB, returns these velocities:

```
  function f = bic_vel (x,ell)

  global delta

%----------------------------
% phase space velocity
%----------------------------

  fc  = delta * cos(2.0*ell);
  kell = 1.0 - fc;

  f(1) = x(3)
  f(2) = x(4)
  f(3) = -kell * x(4);
  f(4) =  kell * x(3);

 return
```

As an application, we compute oblate or biconcave shapes with left-to-right symmetry with respect to the mid-plane, $x = 0$, by specifying the curvature distribution

$$\kappa_\ell = \frac{\pi}{L}\,(1 - \delta\, \cos\frac{2\pi\ell}{L}\,) \tag{6.7.19}$$

for $0 \le \ell \le L$, where L is the total arc length of the cell contour in an azimuthal plane and δ is a specified dimensionless amplitude. We may then integrate system (6.7.17) with initial conditions

$$x_1(0) = x_0, \qquad x_2(0) = 0, \qquad x_3(0) = 0, \qquad x_4(0) = 1, \quad (6.7.20)$$

where x_0 is an arbitrary position. The integration can be carried out numerically using, for example, a Runge–Kutta method.

The following Matlab code, located in directory BIC of SEMLIB, integrates the differential equations by the fourth-order Runge–Kutta method:

```
 function xall = bic_ode (Dell,Nstp)

%--------------------
% compute capsule profile
%--------------------

  global delta

%--------
% prepare
%--------

   Delh = 0.5*Dell;

%-------------------
% initial conditions
%-------------------

    x(1) = 0.0;  x(2) = 0.0; x(3) = 0.0; x(4) = 1.0;

    for j=1:4
       xall(j,1) = x(j);
    end

%-----------
% integrate
%----------
```

```
ell = 0.0

for i=1:Nstp

   for j=1:4
    xsave(j) = x(j);
   end

   g0 = bic_vel (x,ell);

   for j=1:4
     x(j) = xsave(j) + Delh*g0(j);
   end

   g1 = bic_vel(x,ell+Delh);

   for j=1:4
     x(j) = xsave(j) + Delh*g1(j);
   end

   g2 = bic_vel(x,ell+Delh);

   for j=1:4
     x(j) = xsave(j) + Dell*g2(j);
   end

   g3 = bic_vel(x,ell+Dell);

   for j=1:4
     vel = (g0(j)+2.0*g1(j)+2.0*g2(j)+g3(j))/6.0;
     x(j) = xsave(j) + Dell*vel;
   end

   ell = ell + Dell;

   for j=1:4
    xall(j,i+1) = x(j);
   end
```

```
   end  % over steps

%-----
% done
%-----

return
```

The following code, located in directory BIC of SEMLIB, calls this function to generate capsule profiles:

```
global delta

%---
% parameters
%---

Nstp = 32;
a = 1.0;  % equivalent radius;
L = a*pi;
Dell  = L/(2.0*Nstp);

%---
% prepare
%---

figure(1)
hold on

delta = 2.3;

%---
% integrate the ode
%---

xall = bic_ode (Dell,Nstp);

%---
```

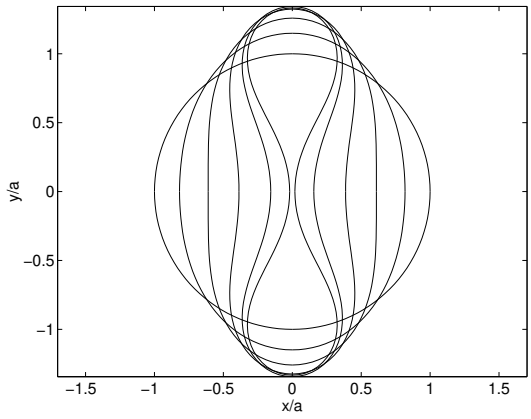


FIGURE 6.7.2 Deflated sphere and biconcave axisymmetric capsule profiles computed in terms of the curvature.

```
% plot profile
%---

xshift = -xall(1,Nstp+1);

for i=1:Nstp+1
 xplot(i) = xall(1,i)+xshift;
 yplot(i) = xall(2,i);
end

figure(1)
plot( xplot, yplot,'k')
plot(-xplot,-yplot,'k')
plot(-xplot, yplot,'k')
plot( xplot,-yplot,'k')
axis equal
box on
```

Computed profiles for $\delta = 0$ (sphere), 0.1, 0.5, 1.0, 1.5, 2.0, and 2.3 are displayed in Figure 6.7.2. The shape for $\delta = 2.0$ is similar to the average shape of a normal red blood cell.

Exercise

6.7.1 Use the code listed in the text to integrate a system of ordinary differential equations of your choice.

6.8 Numerical integration

Consider the definite integral of a function, $f(x)$, between two specified limits, a and b,

$$\mathcal{J} \equiv \int_a^b f(x)\,\mathrm{d}x. \tag{6.8.1}$$

The integral is the area below the graph of the function $f(x)$, above the x axis, to the right of the vertical line $x = a$, and to the left of the vertical line $x = b$ in the xy plane. If the graph of the function falls below the x axis, the area is counted as negative.

It the integral exists, that is, if the aforementioned area is finite (non-infinite), it can be computed by numerical methods.

6.8.1 Mid-point rule

The mid-point rule is implemented by dividing the integration domain, $[a, b]$, into N intervals of length, Δx_i, where

$$\sum_{i=1}^{N} \Delta x_i = b - a, \tag{6.8.2}$$

and approximating the graph of the function with a flat function inside each interval to obtain

$$\mathcal{J} \simeq \sum_{i=1}^{N} f(x_i^{\mathrm{M}})\Delta x_i, \tag{6.8.3}$$

where x_i^{M} is the mid-point of the ith interval. When the integral is defined and the integrand is sufficiently regular, the accuracy of the computation improves as a higher number of intervals are employed.

6.8.2 Trapezoidal rule

In the trapezoidal rule, the mid-point function value is replaced by the arithmetic mean of the two end point values, yielding

$$\mathcal{J} \simeq \tfrac{1}{2} \sum_{i=1}^{N} \big(f(x_i) + f(x_{i+1}) \big) \Delta x_i, \tag{6.8.4}$$

where $x_1 = a$ is the lower integration limit and $x_{N+1} = b$ is the upper integration limit.

6.8.3 Gaussian quadrature

The theory of numerical integration suggests that an integral is best approximated by using an appropriate Gauss integration quadrature. The methodology amounts to replacing the integral with a weighted sum of integrand values at a chosen number of N_Q base points, x_i for $i = 1, \ldots, N_Q$,

$$\mathcal{J} \simeq c \sum_{i=1}^{N_Q} f(x_i)\, w_i, \tag{6.8.5}$$

where c is an appropriate coefficient and w_i are integration weights.

The base points are distributed in some optimal fashion in the integration domain of interest, $[a, b]$. Having chosen at will the number of base points, N_Q, we compute their precise location and associated weights guided by the theory of orthogonal polynomials. Tables of base points and corresponding weights and associated codes are broadly available.

6.8.4 An assortment of quadratures

Different quadratures have been developed for cases where the lower and upper integration limits, a and b, define a finite, semi-infinite, or infinite integration domain. Quadratures for functions that exhibit integrable singularities are also available. The accuracy of the computation improves dramatically as a higher number of base points are employed.

6.8.5 Gauss–Legendre quadrature

When both integration limits, a and b, are finite, that is, they are not infinite, we may apply the Gauss–Legendre quadrature as follows:

- The base points are given by

$$x_i = \tfrac{1}{2}\,(a+b) + \tfrac{1}{2}\,(b-a)\,z_i, \tag{6.8.6}$$

  where z_i are the known zeros of the N_Q-degree Legendre polynomial.

- The coefficient c in front of the quadrature sum is given by

$$c = \tfrac{1}{2}\,(b-a), \tag{6.8.7}$$

  that is, it is equal to half the length of the integration domain.

- The weights w_i are available from tables.

The Gauss–Legendre quadrature is used routinely in scientific computing.

6.8.6 Monte Carlo integration

Consider the definite integral of the product of a sufficiently regular function, $\psi(x)$, and a probability density function (pdf), $\phi(x)$, defined in an interval of interest, $[x_{\min}, x_{\max}]$,

$$\mathcal{J} \equiv \int_{x_{\min}}^{x_{\max}} \psi(x)\,\phi(x)\,\mathrm{d}x. \tag{6.8.8}$$

Using the *law of large numbers*, we find that the integral can be approximated with the mean value of the function $\psi(x)$ at N random points, x_i, sampled according to the pdf $\phi(x)$,

$$\mathcal{J} \simeq \frac{1}{N}\sum_{i=1}^{N} \psi(x_i). \tag{6.8.9}$$

The accuracy improves as a higher number of points are employed. The algorithm involves the following steps:

1. Generate a sequence of random numbers with a flat pdf (uniform random deviates), ϱ_i for $i = 1, \dots N$, using, for example, the Matlab function *rand*.

2. Find the corresponding sequence, x_i, by solving the equation

$$\int_a^{x_i} \phi(u)\,\mathrm{d}u = \varrho_i \quad \text{or} \quad \int_a^{x_i} \phi(u)\,du = 1 - \varrho_i, \qquad (6.8.10)$$

   where u is a dummy integration variable. This process is called inversion.

3. Approximate the integral using (6.8.9).

If a computer function that generates random numbers with a specified pdf is available, the first two steps are skipped. For example, Matlab encapsulates an internal function, *randn* that generates random data obeying the normal (Gaussian) distribution.

6.8.7 A numerical example

To illustrate the Monte Carlo method, we consider the integral

$$\mathcal{J} \equiv \int_0^\infty x^{2m} \mathrm{e}^{-\alpha x^2}\,\mathrm{d}x, \qquad (6.8.11)$$

where m is a specified integer and α is a specified positive parameter. The exact value of the integral is known to be

$$\mathcal{J} = \frac{1 \cdot 3 \cdots (2m-1)}{2\,(2\alpha)^m} \sqrt{\frac{\pi}{\alpha}}. \qquad (6.8.12)$$

Introducing the transformation $x = \xi/\sqrt{2\alpha}$, we obtain

$$\mathcal{J} = \frac{\sqrt{2\pi}}{(2\alpha)^{m+1/2}} \int_0^\infty \xi^{2m} \Big(\frac{1}{\sqrt{2\pi}}\, \mathrm{e}^{-\frac{1}{2}\xi^2} \Big)\,\mathrm{d}\xi. \qquad (6.8.13)$$

The function enclosed by the tall parentheses inside the integral is the normal (Gaussian) distribution with zero mean and unit standard deviation. Consequently, we may approximate

$$\mathcal{J} \simeq \tfrac{1}{2}\, \frac{\sqrt{2\pi}}{(2\alpha)^{m+1/2}}\, \frac{1}{N} \sum_{i=1}^N \xi_i^{2m}, \qquad (6.8.14)$$

where the coefficient $\frac{1}{2}$ accounts for the infinite domain of definition of the Gaussian distribution and the points x_i are sampled from the Gaussian distribution.

The method is implemented in the following Matlab code in terms of the internal function *randn*:

```
m = 4;
alpha = 0.3;
N = 2^20;
monte_carlo = sqrt(pi/2) ...
/ (2*alpha)^(m+0.5) * mean(randn(1,N).^(2*m))
exact = 1*3*5*7/2/(2*alpha)^m * sqrt(pi/alpha)
```

A Monte Carlo integration session with $N = 2^{20} = 1,048,576$ points yields the value $1,287.0$; another session with the same number of points yields the value $1,341.5$. Both values are reasonably close to the exact value, $1,310.9$. However, we note that a high number of sample points, N, are employed.

To compute the integral of an arbitrary function, $Q(x)$,

$$\mathcal{J} \equiv \int_a^b Q(x)\,\mathrm{d}x, \tag{6.8.15}$$

we apply Monte Carlo integration with the function $\psi(x) = Q(x)/\phi(x)$.

Exercise

6.8.1 Compute the integral

$$\mathcal{J} \equiv \int_0^\infty \theta^8 \mathrm{e}^{-\theta^2+\theta}\,\mathrm{d}\theta \tag{6.8.16}$$

using the Monte Carlo integration method.

6.9 Aitken extrapolation of an infinite sum

Consider the computation of an infinite sum,

$$\mathcal{S} = \sum_{i=1}^{\infty} r_i, \tag{6.9.1}$$

and assume that, as i tends to infinity, the ith summed term, r_i, decays like $1/i^s$

$$r_i \sim \frac{1}{i^s}, \tag{6.9.2}$$

where $s > 1$ so that the sum is convergent. For high i, a log-log plot of r_i against i appears as a straight line whose slope tends to $-s$. For example, we may have

$$r_i = a\,\frac{i^m}{b + i^{m+s}}, \tag{6.9.3}$$

where a, b, and m are three constants.

6.9.1 Geometric truncation sequence

To compute the sum expeditiously, we choose an arbitrary positive integer, N, and another arbitrary positive integer, $p > 1$, introduce a sequence of integers,

$$M_k = Np^k \tag{6.9.4}$$

for $k = 0, 1, 2, \ldots$, and consider the sequence

$$b_k = -\tfrac{1}{2}\, r_{M_k} + \sum_{i=1}^{M_k} r_i. \tag{6.9.5}$$

To explain the motivation for introducing the sequence b_k, we consider the associated sequence of integrals

$$\mathcal{I}_k \equiv \int_{M_k}^{\infty} f(x)\,\mathrm{d}x, \tag{6.9.6}$$

where the function $f(x)$ is the apparition of r_i satisfying the interpolation condition $f(i) = r_i$. For large values of i,

$$f(i) = r_i \sim \frac{1}{i^s}. \tag{6.9.7}$$

If the function $f(x)$ is well behaved, $f(x) \simeq c/x^s$ for large x and

$$\mathcal{I}_k \simeq \int_{M_k}^{\infty} \frac{c}{x^s}\,\mathrm{d}x = \frac{c}{s-1}\,\frac{1}{M_k^{s-1}}, \tag{6.9.8}$$

with an error on the order of $1/M_k^2$, where c is a constant.

Now we approximate the integral with a sum using the trapezoidal rule, obtaining

$$\mathcal{I}_k \simeq -\tfrac{1}{2}\,f(M_k) + \sum_{i=M_k}^{\infty} f(i) = -\tfrac{1}{2}\,r_{M_k} + \sum_{i=M_k}^{\infty} r_i, \tag{6.9.9}$$

which can be written as

$$\mathcal{I}_k \simeq \mathcal{S} - b_k, \tag{6.9.10}$$

with an error on the order of $1/M_k^2$.

Combining equations (6.9.8) and (6.9.10), we obtain

$$\mathcal{S} - b_k \simeq \mathcal{I}_k \simeq \frac{c}{s-1}\,\frac{1}{M_k^{s-1}}. \tag{6.9.11}$$

Substituting the definition of M_k, we obtain

$$\mathcal{S} - b_k \simeq \frac{c}{s-1}\,\frac{1}{N^{s-1}}\,\frac{1}{p^{k(s-1)}}, \tag{6.9.12}$$

which shows that

$$\frac{b_k - \mathcal{S}}{b_{k+1} - \mathcal{S}} \simeq p^{s-1}. \tag{6.9.13}$$

Increasing k by one unit, we obtain

$$\frac{b_{k-1} - \mathcal{S}}{b_k - \mathcal{S}} \simeq p^{s-1}. \tag{6.9.14}$$

Combining the last two equations, we obtain

$$(b_k - \mathcal{S})^2 = (b_{k-1} - \mathcal{S})(b_{k+1} - \mathcal{S}). \tag{6.9.15}$$

Solving for $\mathcal{S}$, we obtain

$$\mathcal{S} \simeq b_{k+1} - \frac{(b_{k+1} - b_k)^2}{b_{k+1} - 2\,b_k + b_{k-1}} \equiv a_k. \tag{6.9.16}$$

Simplifying, we obtain

$$a_k = \frac{b_{k-1}\,b_{k+1} - b_k^2}{b_{k+1} - 2\,b_k + b_{k-1}} \tag{6.9.17}$$

for $k = 1, 2, \ldots$.

If the primary sequence, b_k, converges, the extrapolated sequence, a_k, will converge even faster. The numerical method involves computing the sequence a_k, while simultaneously computing the sequence b_k at a minimal computational cost.

6.9.2 Computer code

The method is implemented in the following Matlab code named *sum_fast*, located in directory AITKEN of SEMLIB:

```
function [Sum,a,b,kmax] = sum_fast (menu,s,N,p,kmax,tol)

%==============================
% Sum an infinite series whose
% terms decay like 1/i^s
% for i=1, 2, ...
% using Aitken extrapolation
%==============================

L = 0;
M = N;

Sum = 0.0;

%-------------
```

```
for k=1:kmax+2
%-------------

   for i=L+1:M

        if(menu==1)
          term = 1.0/i^s;
        elseif(menu==2)
          term = 1.0/(i*(i+2.0));
          s = 2;
        elseif(menu==3)
          is = i^2;
          term = 1.0/(1.0/is+is);
          s = 2;
        elseif(menu==4)
          term = 1.0/(2.0*i-1.0)^2;
          s = 2;
        else
           disp('menu option is not available')
        end

    Sum = Sum + term;

   end

   b(k) = Sum - 0.5/M^s;

   if(k>2)
    a(k-1) = (b(k-2)*b(k)-b(k-1)*b(k-1)) ...
             /(b(k)-2.0*b(k-1)+b(k-2));
   end

   if(k>3)
   %---
   if(abs(a(k-1)-a(k-2))<tol)
    Sum = a(k-1);
    kmax = k-2;
    break;
   end
```

```
   %---
   end

   L = M;
   M = M*p;

%---
end
%--

Sum = a(k-1);

%---
% done
%---

return
```

The following driver code calls this function to compute a particular sum:

```
N = 1;
p = 2;

kmax = 5;
kmax = 15;

tol = 0.00001;

menu = 1;
s = 1.1;

%---
% sum
%---

[Sum,a,b,kmax] = sum_fast (menu,s,N,p,kmax,tol);

%===
% verbose
```

```
%===

dsp(1,1)=b(1);

 for k=2:kmax
   dsp(k,1)=b(k);
   dsp(k,2)=a(k);
 end

 dsp(kmax+1,1) = b(kmax+1);
 dsp(kmax+1,2) = a(kmax+1);

format long
dsp
```

Running the code generates the following output:

```
dsp =
    0.500000000000000    0.000000000000000
    1.233258247884202    6.310841780573343
    1.873988136127340    8.842102367604978
    2.460763076256344   10.082737998241887
    3.005594378663794   10.460078223115744
    3.513317360842163   10.554974769994439
    3.986894146475962   10.577545906761697
    4.428722954190673   10.582836769937241
    4.840955886611950   10.584072429415023
    5.225580964880243   10.584360746055538
    5.584448421240750   10.584428003474271
    5.919283497071050   10.584443692121663
    6.231695646091956   10.584447351650040
```

We see that the accelerated sum converges much faster than the primary sum.

Exercise

6.9.1 Compute the sum of $1/r^{1.01}$ accurate to the sixth decimal place.

6.10 Centered differences

The first few derivatives of a function, $f(x)$, at a point, x_i, can be approximated with the following centered finite-difference formulas:

$$\begin{bmatrix} f_i^{(0)} \\ 2\,h f_i^{(1)} \\ h^2 f_i^{(2)} \\ 2\,h^3 f_i^{(3)} \\ h^4 f_i^{(4)} \\ 2\,h^5 f_i^{(5)} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 & 0 \\ 0 & -1 & 2 & 0 & -2 & 1 & 0 \\ 0 & 1 & -4 & 6 & -4 & 1 & 0 \\ -1 & 4 & -5 & 0 & 5 & -4 & 1 \end{bmatrix} \cdot \begin{bmatrix} f_{i-3} \\ f_{i-2} \\ f_{i-1} \\ f_i \\ f_{i+1} \\ f_{i+2} \\ f_{i+3} \end{bmatrix}, \tag{6.10.1}$$

involving function values at evenly-spaced neighboring points, $x_{i+m} = x_i + mh$, where h is a specified small interval and $f_j \equiv f(x_j)$. The associated error is on the order of h^2. Note the alternating factors of *two* on the left-hand side.

For example, the fifth derivative can be approximated as

$$f_i^{(5)}(x) \simeq \frac{1}{2h^5}\big(-f_{i-3} + 4f_{i-2} - 5f_{i-1} + 5f_{i+1} - 4f_{i+2} + f_{i+3}\big). \tag{6.10.2}$$

Note the absence of the central function value, f_i, on the right-hand side

6.10.1 Even-order derivatives

When the order of a derivative, m, is even, we set $m = 2p$, where $p = 0, 1\ldots$, and obtain the following centered difference formula for the qth derivative involving $q+1$ terms:

$$f_i^{(2p)} \simeq (-1)^p \frac{1}{h^{2p}} \sum_{k=-p}^{p} (-1)^k\, \phi_{q,k}\, f_{i+k}, \tag{6.10.3}$$

where

$$\phi_{q,k} \equiv \binom{m}{k+p} \tag{6.10.4}$$

for $k = -p, \ldots, p$, where the tall parentheses denote the binomial coefficient, as discussed in Appendix A. For example, when $q = 4$, we set $p = 2$ and obtain the following approximation to the fourth derivative:

$$f_i^{(4)} \simeq \frac{1}{h^4} \left(\binom{4}{0} f_{i-2} - \binom{4}{1} f_{i-1} + \binom{4}{2} f_i - \binom{4}{3} f_{i+1} + \binom{4}{4} f_{i+2} \right). \tag{6.10.5}$$

Substituting the values of the binomial coefficients provides us with the corresponding formula in (6.10.1).

The binomial coefficient can be expressed in terms of the Gamma function using the relation

$$\binom{\alpha}{\beta} = \frac{\Gamma(\alpha+1)}{\Gamma(\beta+1)\,\Gamma(\alpha-\beta+1)}, \tag{6.10.6}$$

as discussed in Appendix A. Using this relation, we obtain

$$\phi_{m,k} = \frac{\Gamma(m+1)}{\Gamma(k+\frac{m+2}{2})\,\Gamma(-k+\frac{m+2}{2})} \tag{6.10.7}$$

for $k = -p, \ldots, p$, where $m = 2p$.

Substituting into (6.10.3) the shifted parameter $k = \ell - p$, and setting $\ell = k + p$, we obtain the equivalent formula

$$f_i^{(2p)} \simeq \frac{1}{h^{2p}} \sum_{\ell=0}^{2p} (-1)^\ell \binom{m}{\ell} f_{i-p+\ell}. \tag{6.10.8}$$

The first term in the sum for $\ell = 0$ carries f_{i-p}, and the last sum for $\ell = 2p$ carries f_{i+p}.

6.10.2 *Odd-order derivatives*

When the derivative order m is odd, we set $m = 2p+1$ and introduce the following consistent central difference approximation:

$$f_i^{(2p+1)} = \frac{1}{2h} \left(f_{i+1}^{(2p)} - f_{i-1}^{(2p)} \right). \tag{6.10.9}$$

Substituting expansion (6.10.3) with k renamed as r, we obtain

$$f_i^{(2p+1)} = (-1)^p \frac{1}{2h^{2p+1}} \sum_{r=-p}^{p} (-1)^r \binom{2p}{r+p} \big(f_{i+r+1} - f_{i+r-1} \big). \tag{6.10.10}$$

Substituting into (6.10.9) expansion (6.10.8), we obtain the equivalent expression

$$f_i^{(2p+1)} = \frac{1}{2h^{2p+1}} \sum_{\ell=0}^{2p} (-1)^\ell \binom{2p}{\ell} \big(f_{i-p+\ell+1} - f_{i-p+\ell-1} \big). \tag{6.10.11}$$

Equation (6.10.10) can be rearranged into

$$\begin{aligned} f_i^{(2p+1)} = (-1)^p \frac{1}{2\,h^{2p+1}} \Big(& \sum_{k=-p+1}^{p+1} (-1)^{k-1} \binom{2p}{k+p-1} f_{i+k} \\ & - \sum_{k=-p-1}^{p-1} (-1)^{k+1} \binom{2p}{k+p+1} f_{i+k} \Big), \end{aligned} \tag{6.10.12}$$

which can be restated as

$$\begin{aligned} f_i^{(2p+1)} = \frac{1}{2\,h^{2p+1}} \Big(& - f_{i-p-1} + 2\,p\,f_{i-p} \\ & + (-1)^p \sum_{k=-(p-1)}^{p-1} (-1)^k \psi_k \, f_{i+k} - 2\,p\,f_{i+p} + f_{i+p+1} \Big), \end{aligned} \tag{6.10.13}$$

where

$$\psi_k = \binom{2p}{k+p+1} - \binom{2p}{k+p-1}, \tag{6.10.14}$$

which is equivalent to

$$\psi_k = \binom{2p}{k+p+1} - \binom{2p}{-k+p+1} \tag{6.10.15}$$

for $k = -(p-1), \cdots, p-1$. We find that $\psi_0 = 0$.

Using the properties of the binomial coefficient, we find that

$$\binom{2p}{k+p-1} + \binom{2p}{k+p} = \binom{2p+1}{k+p} \tag{6.10.16}$$

and

$$\binom{2p}{k+p} + \binom{2p}{k+p+1} = \binom{2p+1}{k+p+1}, \tag{6.10.17}$$

which shows that

$$\psi_k = \binom{2p+1}{k+p+1} - \binom{2p+1}{k+p}, \tag{6.10.18}$$

which is equivalent to

$$\psi_k = \binom{m}{k+\frac{m+1}{2}} - \binom{m}{k+\frac{m-1}{2}}. \tag{6.10.19}$$

Expressing the binomial coefficient in terms of the Gamma function using relation (6.10.6), we obtain

$$f_i^{(2p+1)} = (-1)^p \frac{1}{h^{2p+1}} \sum_{k=-(p+1)}^{p+1} (-1)^k \phi_{m,k} f_{i+k}, \tag{6.10.20}$$

where

$$\phi_{m,k} = \frac{1}{2}\left(\frac{\Gamma(m+1)}{\Gamma(k+\frac{m+3}{2})\,\Gamma(-k+\frac{m+1}{2})} - \frac{\Gamma(m+1)}{\Gamma(k+\frac{m+1}{2})\,\Gamma(-k+\frac{m+3}{2})}\right). \tag{6.10.21}$$

6.10.3 Coefficients by code

The following Matlab code named *tablec*, located in directory TABLEC of SEMLIB, generates the matrix on the right-hand side of (6.10.1):

```
pmax = 4;

table = zeros(2*pmax+2,2*pmax+2);

%---
% loop over p
%---

Ic = 0;
ishift = pmax+1;

for p=0:pmax

fc = (-1)^p;

%---
% even order
%---

m= 2*p;

Ic = Ic+1;

for k=-p:p

 comb = gamma(m+1)/gamma(k+(m+2)/2)/gamma(-k+(m+2)/2);
 dfc = fc*(-1)^k * comb;
 table(Ic,k+p+1+ishift)=dfc;

end

%---
% odd order
%---

m = 2*p+1;
p1 = p+1;
```

```
ishift = ishift-1;

Ic = Ic+1;

for k = -p1:p1

 combA = gamma(m+1)/gamma(k+(m+3)/2)/gamma(-k+(m+1)/2);
 combB = gamma(m+1)/gamma(k+(m+1)/2)/gamma(-k+(m+3)/2);
 dfc = fc*(-1)^k * (combA-combB);
 table(Ic,k+p1+1+ishift)=dfc;

 end

end % of loop over p

table
```

Running the code generates the following output:

```
 0   0   0   0    0    0    1    0    0    0   0   0  0
 0   0   0   0    0   -1    0    1    0    0   0   0  0
 0   0   0   0    0    1   -2    1    0    0   0   0  0
 0   0   0   0   -1    2    0   -2    1    0   0   0  0
 0   0   0   0    1   -4    6   -4    1    0   0   0  0
 0   0   0  -1    4   -5    0    5   -4    1   0   0  0
 0   0   0   1   -6   15  -20   15   -6    1   0   0  0
 0   0  -1   6  -14   14    0  -14   14   -6   1   0  0
 0   0   1  -8   28  -56   70  -56   28   -8   1   0  0
 0  -1   8 -27   48  -42    0   42  -48   27  -8   1  0
 0   1 -10  45 -120  210 -252  210 -120   45 -10   1  0
-1  10 -44 110 -165  132    0 -132  165 -110  44 -10  1
```

This table is an extension of that shown in (6.10.1).

Exercise

6.10.1 Derive (6.10.13).

Index

Some Engineering Mathematics

C. Pozrikidis

A good way of learning, teaching, and refamiliarizing oneself with methods of applied mathematics and scientific computing is to study a selected set of topics related to practical science and engineering applications that are a blend of the old, the new, and the emerging. This book discusses a pertinent collection of carefully selected topics, problems, algorithms, and complete computer codes, including mainstream and original material. The discussion of the individual topics provides opportunities for recalling, clarifying, and learning methods and developing skills.

- The practical application of methods of applied mathematics and numerical algorithms are discussed by example.
- Topics are discussed on an introductory level assuming a junior or even lower level background on linear algentra and calculus, and elementary familiarity with computer programming.
- Numerical methods and computer codes are developed following the mathematical formulation of the individual problems discussed.
- The choice of topics and problems includes a fair amount of original material not found elsewhere.
- The accompanying software library SEMLIB incorporates elementary and advanced scripts, functions, and simulation codes.

Chester & Bennington