

Lecture Notes in Bioengineering

Socrates Dokos

Modelling Organs, Tissues, Cells and Devices

Using MATLAB and COMSOL
Multiphysics

 Springer

Lecture Notes in Bioengineering

More information about this series at <http://www.springer.com/series/11564>

Socrates Dokos

Modelling Organs, Tissues, Cells and Devices

Using MATLAB and COMSOL Multiphysics

Socrates Dokos
Graduate School of Biomedical Engineering
University of New South Wales
Sydney
Australia

ISSN 2195-271X ISSN 2195-2728 (electronic)
Lecture Notes in Bioengineering
ISBN 978-3-642-54800-0 ISBN 978-3-642-54801-7 (eBook)
DOI 10.1007/978-3-642-54801-7

Library of Congress Control Number: 2017930790

© Springer-Verlag Berlin Heidelberg 2017

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer-Verlag GmbH Germany
The registered company address is: Heidelberger Platz 3, 14197 Berlin, Germany

To my wonderful Anna for her selfless support, inspiration, wisdom, and encouragement over a lifetime in academia; and to our children Vasilina, Elias, George, Timothy, Catherine and Raphael, who make us proud beyond words.

Preface

Computational modeling plays an increasingly important role in biological and medical research, as well as in the medical device industry. Like other industries, successful development of medical devices and implants requires not only extensive testing (bench tests, animal experiments and human trials), but also extensive computational simulations which allow engineers to cost-effectively investigate system behavior and iterate the device design. These approaches are inherently multidisciplinary: biomedical engineers are required to understand not only the fundamental physical principles of their devices, but also how they interact with complex physiological systems at the cellular, tissue, and whole-organ levels. Computational models of such systems are typically multiphysics in nature. In the not too distant future, bioengineering modeling will also be used routinely in the clinic to tailor a range of individual therapies and treatment strategies based on patient-specific models.

It is evident that computational modeling is an important skill for biomedical engineers, and should form an indispensable component of modern curricula in biomedical engineering. This book grew from my own course, ‘Modelling Organs, Tissues and Devices’, taught to late-stage undergraduate and postgraduate engineering students at the Graduate School of Biomedical Engineering, UNSW. It covered a broad range of modeling topics, from electrical stimulation of tissues through to diffusion, biomechanics, heat transfer and fluid dynamics. Numerical software used for teaching the course were MATLAB and COMSOL Multiphysics—the former due to its prevalent use in engineering and the latter due to its multiphysics capabilities and appealing user interface. For their major project, students were required to submit a COMSOL model in any bioengineering field of their choice. Despite the popularity of the course, it became apparent there was no single book which could be used as recommended text which covered the range of topics taught, including numerical/analytical methods for solving ODE/PDEs, an overview of the various multiphysics principles involved, as well as how such models can be practically implemented in MATLAB and COMSOL. Hence, the idea for this book was born!

The text is divided into two parts: Part A covers basic modeling concepts as well as analytical and numerical methods for solving ODE/PDE systems. Part B covers specific physics applications in bioengineering. Each chapter also includes a set of problems, over 50 in total, with detailed answers provided in the solutions section, along with several worked-examples with code listings throughout the main text itself. Appendices A and B provide an introduction to MATLAB and COMSOL, respectively.

All models in the text were solved on a base-model MacBook Air 2013 laptop with 8G RAM. Some of the models took tens of minutes to solve, whilst others took only a few seconds. The MATLAB version employed was R2014b, and the COMSOL version was 5.2 (the latest at the time of writing). Most of these models could be solved using the standard base packages of MATLAB and COMSOL, whilst other models required optional add-ons such as MATLAB's Symbolic Math Toolbox or COMSOL's Nonlinear Structural Materials Module. Such instances whenever they occur are mentioned in the text.

Finally, I wish to acknowledge my family for their unwavering support throughout the four long years it took to complete this text, as well the encouragement and support from my colleagues at UNSW. A special mention to Jan-Philip Schmidt, my editor at Springer Verlag, for his patience and encouragement, as well as my many students and postdocs, past and present, whose MATLAB and/or COMSOL modeling works have inspired many examples throughout this text. They include Amr Al Abed, Siwei Bai, Tianruo Guo, Miganoosh Abramian, Khalid Alonazi, Siniša Sovilj, Yousef Alharbi, Azam Bakir, Chin Neng Leong, Bee Ting Chan, Abdulrahman Alqahtani, Mitra Mohd Addi, Shuhaida Yahud, Shijie Yin, Ben Hui, David Chan and Einly Lim. I have thoroughly treasured and learnt much from our interactions.

Sydney, Australia
May 2016

Socrates Dokos

Contents

Part I Bioengineering Modelling Principles, Methods and Theory

| | |
|--|----|
| 1 Introduction to Modelling in Bioengineering | 3 |
| 1.1 Modelling and Simulation in Medicine and Biology | 3 |
| 1.2 The Modelling Process | 4 |
| 1.3 Mathematical Model Types | 5 |
| 1.3.1 Linear Versus Non-linear | 6 |
| 1.3.2 Dynamic Versus Static | 7 |
| 1.3.3 Deterministic Versus Stochastic | 7 |
| 1.3.4 Continuous Versus Discrete | 9 |
| 1.3.5 Rule-Based | 12 |
| 1.4 Dimensional Analysis | 16 |
| 1.4.1 Dimensions and Units | 16 |
| 1.4.2 Buckingham π -Theorem | 19 |
| 1.5 Model Scaling | 21 |
| Problems | 23 |
| References | 27 |
| 2 Lumped Parameter Modelling with Ordinary Differential Equations | 29 |
| 2.1 Overview of Ordinary Differential Equations | 29 |
| 2.2 Linear ODEs | 31 |
| 2.3 ODE Systems | 35 |
| 2.3.1 Example Model 1: Cardiac Mechanics | 37 |
| 2.3.2 Example Model 2: Hodgkin–Huxley Model of Neural Excitation | 42 |
| 2.4 Further Reading | 46 |
| Problems | 46 |
| References | 53 |

| | | |
|----------|--|-----|
| 3 | Numerical Integration of Ordinary Differential Equations | 55 |
| 3.1 | Taylor's Theorem | 55 |
| 3.2 | One-Step Methods | 60 |
| 3.2.1 | Backward-Euler Method | 63 |
| 3.2.2 | Trapezoidal Method | 65 |
| 3.2.3 | Runge–Kutta Methods | 66 |
| 3.2.4 | The Generalized- α Method | 73 |
| 3.3 | Multistep Methods | 82 |
| 3.3.1 | Predictor-Corrector Methods | 86 |
| 3.3.2 | Backward Differentiation Formulas | 93 |
| 3.3.3 | Numerical Differentiation Formulas | 96 |
| 3.4 | ODE Solver Implementations in Matlab and COMSOL | 97 |
| 3.5 | Further Reading | 100 |
| | Problems | 100 |
| | References | 103 |
| 4 | Distributed Systems Modelling with Partial Differential Equations | 105 |
| 4.1 | Modelling with PDEs | 105 |
| 4.1.1 | The Gradient | 105 |
| 4.1.2 | The Divergence | 108 |
| 4.1.3 | The Curl | 112 |
| 4.1.4 | The Divergence Theorem | 113 |
| 4.1.5 | Conservation Law Formulation | 117 |
| 4.1.6 | The Laplacian | 119 |
| 4.1.7 | PDE Boundary Conditions | 120 |
| 4.2 | Basic Analytical and Numerical Solution Techniques | 123 |
| 4.2.1 | Separation of Variables | 123 |
| 4.2.2 | Finite Difference Method | 139 |
| 4.2.3 | Method of Lines | 147 |
| 4.3 | Further Reading | 153 |
| | Problems | 153 |
| | References | 157 |
| 5 | The Finite Element Method | 159 |
| 5.1 | Finite Elements for 1D Systems | 159 |
| 5.1.1 | Weak Form PDE Equivalent | 160 |
| 5.1.2 | Basis Function Approximation | 164 |
| 5.1.3 | Higher-Order Basis Functions | 175 |
| 5.2 | Finite Elements for 2D/3D Systems | 179 |
| 5.2.1 | Weak Form Description | 180 |
| 5.2.2 | Basis Function Approximation | 183 |

- 5.3 FEM Numerical Implementation. 190
 - 5.3.1 Assembly of System Matrices. 191
 - 5.3.2 Gaussian Quadrature 192
 - 5.3.3 Non-Linear Systems 194
- 5.4 Further Reading 195
- Problems. 196
- References. 197

Part II Bioengineering Applications

- 6 Modelling Electrical Stimulation of Tissue 201**
 - 6.1 Electrical Stimulation 201
 - 6.1.1 Maxwell’s Equations. 201
 - 6.1.2 Electrostatic Formulations 203
 - 6.1.3 Volume Conductor Theory 204
 - 6.1.4 Example: Cell Culture Electric Field Stimulator 207
 - 6.1.5 Example: Access Resistance of Electrode Disc 210
 - 6.2 Modelling Electrical Activity of Tissues. 215
 - 6.2.1 Continuum Models of Excitable Tissues. 215
 - 6.2.2 Example: Modelling Spiral-Wave Reentry
in Cardiac Tissue 217
 - 6.2.3 Modelling PDEs/ODEs on Boundaries,
Edges and Points. 225
 - 6.2.4 Example: Axonal Stimulation Using Nerve Cuff
Electrodes 226
 - 6.3 Further Reading 234
 - Problems. 234
 - References. 235
- 7 Models of Diffusion and Heat Transfer 237**
 - 7.1 Diffusion. 237
 - 7.1.1 Fick’s Laws of Diffusion 237
 - 7.1.2 Example: Diffusion and Uptake into a Spherical Cell 238
 - 7.1.3 Convective Transport 241
 - 7.1.4 Example: Drug Delivery in a Coronary Stent 242
 - 7.2 Heat Transfer 247
 - 7.2.1 Heat Conduction and Convection 248
 - 7.2.2 The Bioheat Equation 250
 - 7.2.3 Example: RF Atrial Ablation 251
 - 7.3 Further Reading 258
 - Problems. 258
 - References. 260

| | |
|--|-----|
| 8 Solid Mechanics | 263 |
| 8.1 Biomechanics | 263 |
| 8.2 Tensor Fundamentals | 263 |
| 8.2.1 Tensor Definition | 263 |
| 8.2.2 Indicial Notation | 265 |
| 8.2.3 Tensor Transformation Law | 266 |
| 8.2.4 Tensor Invariants. | 268 |
| 8.3 Mechanics Principles | 271 |
| 8.3.1 Stress | 271 |
| 8.3.2 Strain | 275 |
| 8.4 Linear Elasticity | 281 |
| 8.4.1 Example: Detecting Tension in a Respirator Strap | 284 |
| 8.5 Linear Viscoelasticity | 290 |
| 8.6 Hyperelastic Materials | 291 |
| 8.6.1 Example: Myocardial Shear | 294 |
| 8.7 Further Reading | 299 |
| Problems. | 300 |
| References. | 302 |
| 9 Fluid Mechanics | 305 |
| 9.1 Fluid Motion | 305 |
| 9.1.1 Example: Laminar Flow Through a Circular Tube | 306 |
| 9.2 Navier-Stokes Equations. | 311 |
| 9.2.1 Example: Drug Delivery in a Coronary Stent Revisited | 314 |
| 9.3 Non-laminar Flow. | 321 |
| 9.4 Modelling Blood Flow | 324 |
| 9.4.1 Electric Circuit Analogues for Blood Flow | 324 |
| 9.4.2 Example: Aortic Blood Flow | 325 |
| 9.4.3 Blood as a Non-newtonian Fluid. | 329 |
| 9.4.4 Example: Axial Streaming of a Blood Cell | 330 |
| 9.5 Further Reading | 339 |
| Problems. | 339 |
| References. | 341 |
| Appendix A: Matlab Fundamentals | 343 |
| Appendix B: Overview of COMSOL Multiphysics | 355 |
| Solutions | 381 |
| Index | 495 |

Acronyms

| | |
|-------|--|
| 0D | Zero-Dimensional |
| 1D | One-Dimensional |
| 2D | Two-Dimensional |
| 3D | Three-Dimensional |
| AE | Algebraic Equation |
| BDF | Backward Differentiation Formula |
| CAD | Computer-Aided Design |
| CFD | Computational Fluid Dynamics |
| DAE | Differential-Algebraic Equation |
| DC | Direct Current |
| ECG | Electrocardiogram |
| FD | Finite Difference |
| FE | Finite Element |
| FEM | Finite Element Method |
| GMRES | Generalized Minimal Residual Method |
| LV | Left Ventricle/Ventricular |
| NDF | Numerical Differentiation Formula |
| ODE | Ordinary Differential Equation |
| PDE | Partial Differential Equation |
| RF | Radio Frequency |
| RMS | Root Mean Square |
| SI | International System of Units (Système International d'Unités) |
| STL | Stereolithography |
| TTX | Tetrodotoxin |
| VRML | Virtual Reality Modeling Language |

Part I

Bioengineering Modelling Principles, Methods and Theory

The first part of this text will cover fundamentals of bioengineering modelling, including computational techniques and relevant principles of physics.

Chapter 1

Introduction to Modelling in Bioengineering

1.1 Modelling and Simulation in Medicine and Biology

In mathematics and engineering, *modelling* is defined as the formulation of a mathematical or computational representation of a physical system. Such representations (or *models*) are typically solved numerically using a computer: a process known as *simulation*. The terms *mathematical modelling* (i.e. representing a system with equations) and *computational modelling* (replicating a system on computer) are largely synonymous insofar that numerical computation is utilised in both.

Modelling is central to science itself. Since the time of the ancient Greeks, mathematical principles have been regarded as integral to understanding the cosmos. Plato for instance, was said to have placed an inscription over the doors of his philosophical Academy: “let no one ignorant of geometry enter under my roof” [16].¹ However the first great triumph of modelling occurred in the 17th century with the publication of Isaac Newton’s “Mathematical Principles of Natural Science” (otherwise known as the “Principia”), in which he expounded his laws of motion and the inverse-square law of gravity, accurately accounting for the complex motions of the planets with a simple and elegant mathematical description. Since then, modelling has continued to enjoy enormous success throughout the fields of physics and engineering.

However the slowest areas of science to succumb to mathematical description have arguably been those of medicine and biology, owing largely to the overwhelming complexity and variation found in living systems. There are no analogous simple laws of biology as in physics which lend themselves to simple mathematical formulation. Despite the complexity inherent in biological systems, the last several decades have seen major advances in the application of mathematics to biology, physiology and medicine. Modelling is increasingly being utilised to study the integrative behaviour of complex biological systems [5, 13, 14, 17], also known as *systems biology*.

Today, computational modelling is used extensively for biomedical research and medical device development in areas as diverse as biomechanics and orthopaedics,

¹μηδείς ἀγεωμέτρητος εἰσὶτω μου τὴν στέγην.

fluid dynamics, drug delivery, tissue ablation, neurostimulation and many others. Computer models are increasingly being utilised for virtual prototyping of medical devices and implants [19]. Standards for reporting medical simulations are under active development [6], as are standards for the representation and exchange of biological models [3, 12]. Major initiatives are underway for developing multiscale physiological models ranging from the molecular through to the whole-organ and organ systems scale [20]. Furthermore, patient-specific computer simulations are increasingly being employed to tailor individualised treatments and therapies [18].

Bioengineering models are simulated using custom-written, open-source or commercial software. In many cases, the models require implementation of highly-specific equations, or utilise complex coupling between multiple systems. As such, they require simulation platforms flexible enough to cope with a range of formulations and physics implementations. In this text, Matlab² and COMSOL Multiphysics³ software will be used to implement a variety of bioengineering models. For the reader unfamiliar with these software, a brief overview of Matlab and COMSOL is provided in Appendices A and B respectively.

1.2 The Modelling Process

The modelling process typically involves iteration of the following four stages: Formulation, Coding, Verification and Validation, as shown in Fig. 1.1.

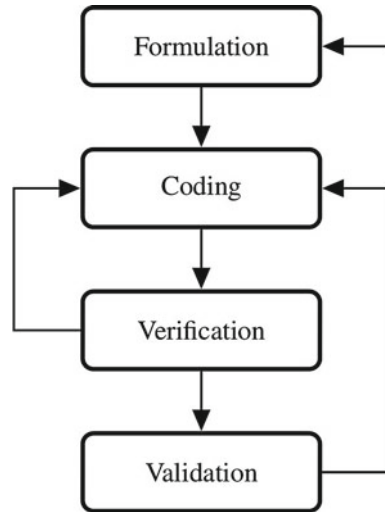
The first step in this process is model *formulation*, which consists in choosing the appropriate equations/rules that describe the system to be modelled. In some cases, this will simply involve selecting the appropriate laws of physics known to apply e.g. conservation of mass, charge, etc. In other cases, specialised formulations characteristic of the system will need to be derived. This latter process will depend to some extent on the level of computational detail required, which in turn will depend on the objectives of the model itself. Typically the model formulations will represent some degree of simplification, as opposed to reproducing the full extent of biological complexity known to be present in the system. To this end, modelling is as much art as it is science.

The next step in the modelling process is to implement the formulations on computer, a process referred to as *coding*. This includes manually programming the model equations or entering these in a modelling software user-interface, along with initial variable values and/or boundary conditions. In addition, numerical algorithms must be implemented for solving the model. The use of specialised mathematical or modelling software such as Matlab or COMSOL can drastically speed-up this coding stage, due to inbuilt user-interfaces for entering equations as well as extensive libraries of numerical solvers from which the modeller can select.

²The Mathworks Inc, Natick, Massachusetts, U.S.A.

³COMSOL AB, Stockholm, Sweden.

Fig. 1.1 The modelling process. The first stage begins with model *formulation* (choosing equations/rules), followed by *coding* (implementation on computer), then *verification* (checking solution accuracy), followed finally by *validation* (checking if model matches observed system behaviour). It may be necessary to visit earlier stages of the process as shown if the model fails the verification or validation steps



Results of computer simulations following the coding procedure must then be subjected to model *verification*. This entails checking whether the simulation results represent accurate solutions of the underlying model formulation. This verification process may involve, for example, reducing the numerical integration step size to verify that the simulations have converged to the same solution to within an acceptable tolerance, irrespective of the step size. If the verification step fails, the user will need to return to the coding step to make adjustments, either by modifying parameters of the numerical solver or the underlying spatial discretization employed, or even switching to another more appropriate solver.

The final step in the modelling process is that of *validation*. Even though the simulation results may have been verified, this of itself provides no guarantee that the model is a sufficiently accurate representation of the system in question. Simulation results must match observed behaviour of the real system. If the model fails the validation step, it may be necessary to revisit the coding step, modifying formulation-independent model features as parameter values, initial variables, and/or boundary conditions. It may even be necessary to reformulate the model again, reiterating the entire modelling process from the beginning.

1.3 Mathematical Model Types

Mathematical models can be categorised into *linear* or *non-linear*, *static* or *dynamic*, *deterministic* or *stochastic*, *continuous* or *discrete*, or *rule-based*, as described below.

1.3.1 Linear Versus Non-linear

A *linear* model is characterised by a set of equations whose general solution (that is, ignoring constraints imposed by initial values or boundary conditions) is a linear combination of other solutions of the same set of equations. That is, if ϕ_1 and ϕ_2 are distinct solutions of a linear model, then for all constants c_1, c_2 , the linear combination $c_1\phi_1 + c_2\phi_2$ is also a solution.

Example 1.1 Consider the following simple model of bacterial growth in a Petri dish:

$$\frac{dN}{dt} = kN \quad (1.1)$$

where N is the total number of bacteria, and k is a parameter. Determine if this equation is linear.

Answer: To ascertain if this equation is linear, we assume that $\phi_1(t)$ and $\phi_2(t)$ are two distinct solutions. We can readily show that $N = c_1\phi_1(t) + c_2\phi_2(t)$ is also a solution:

$$\begin{aligned} \frac{dN}{dt} &= \frac{d}{dt} [c_1\phi_1(t) + c_2\phi_2(t)] \\ &= c_1 \frac{d\phi_1(t)}{dt} + c_2 \frac{d\phi_2(t)}{dt} \\ &= c_1 k\phi_1(t) + c_2 k\phi_2(t) \\ &= k(c_1\phi_1(t) + c_2\phi_2(t)) \\ &= kN \end{aligned}$$

and thus Eq. 1.1 is linear. □

Example 1.2 We now modify Eq. 1.1 such that the Petri dish can only support a maximum bacterial population N_{max} :

$$\frac{dN}{dt} = kN \left(1 - \frac{N}{N_{max}} \right) \quad (1.2)$$

then it is left as an exercise for the reader to show that Eq. 1.2 is *non-linear* (see Problem 1.1b).

In general, linear models are easier to solve for than their non-linear counterparts: numerical solvers often run into convergence issues when attempting to solve highly non-linear models.

1.3.2 Dynamic Versus Static

Models whose dependent variables vary with time are referred to as *dynamic* or *time-dependent*. An example is the 1D diffusion equation:

$$\frac{\partial c}{\partial t} = D \frac{\partial^2 c}{\partial x^2} \quad (1.3)$$

where c is the concentration of a diffusing substance and D is a parameter known as the diffusion coefficient. Clearly variable c will depend on time t as well as space x . However, if instead we are interested in the steady-state concentration profile as $t \rightarrow \infty$, then by setting the time-derivative to zero in Eq. 1.3, we obtain the *static* form:

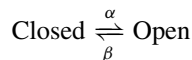
$$D \frac{\partial^2 c}{\partial x^2} = 0 \quad (1.4)$$

where variable c now only depends on x . When solving time-dependent models on computer, solutions to dependent variables must be stored for all required output time points, as well as discrete spatial locations. As a result, storage requirements for time-dependent models are typically much larger than those of their static counterparts, particularly in 3D.

1.3.3 Deterministic Versus Stochastic

Models specified by exact mathematical or rule-based formulations involving no random variables are referred to as *deterministic*. In contrast, models involving random processes are referred to as *stochastic*.⁴

Example 1.3 Consider the following stochastic model of a single ion channel in a cell membrane:



where the channel can be in either the Open or Closed state and α , β determine the transition probabilities between each state as shown. When the channel is in its Open state, ions can enter or leave the cell through the channel, and this can influence the resulting electric potential developed across the membrane. It is therefore of interest to model the kinetics of such channels in order to understand the biophysical mechanisms underlying electrical activity in excitable cells [9]. Using Matlab, simulate the stochastic behaviour of such a channel.

⁴From the Greek $\sigma\tau\omicron\chi\acute{\alpha}\zeta\omicron\mu\alpha\iota$ – I ponder over, guess at, take aim.

Answer: For a sufficiently small time step Δt , the state transition probabilities are governed by the following:

- If the channel is Closed, the probability it will switch to Open is $\alpha \Delta t$.
- If the channel is Open, the probability it will switch to Closed is $\beta \Delta t$.

We can simulate the stochastic behaviour of, for example, $N = 1000$ such channels, assuming $\alpha = 0.1 \text{ ms}^{-1}$, $\beta = 0.01 \text{ ms}^{-1}$ and $\Delta t = 0.001 \text{ ms}$, with all channels initially in the Closed state at $t = 0$. The proportion of all N channels in the Open state can be determined using the following Matlab code:

```
% Stochastic model of ion channel kinetics
alpha = 0.1; % (1/ms)
beta = 0.01; % (1/ms)
Dt = 0.001; % (ms)
N = 1000; % no. of ion channels
time = 0:Dt:40; % (0 --> 40 ms)
state = zeros(length(time),N); % channel states
% generate random variable array:
R = rand(length(time)-1,N);
% generate sequence of states for each channel:
for j = 1:N
    for i = 2:length(time)
        if (state(i-1,j) == 0) % if previously closed
            if (R(i-1,j) <= alpha*Dt)
                state(i,j) = 1;
            else
                state(i,j) = 0;
            end;
        else % if previously open
            if (R(i-1,j) <= beta*Dt)
                state(i,j) = 0;
            else
                state(i,j) = 1;
            end;
        end;
    end;
end;
Open_stochastic = sum(state,2)/N; % open proportion
Open_deterministic = ... % deterministic proportion
    (1-exp(-time*(alpha+beta)))*alpha/(alpha+beta);

% plot result
plot(time, Open_stochastic, 'k-', ...
     time, Open_deterministic, 'k--'), ...
```



```

xlabel('time (ms)'), ...
ylabel('Proportion Open'), ...
title('Ion Channel Dynamics'), ...
legend('Open (stochastic)', ...
'Open (deterministic)', ...
'Location', 'East');

```

with the resulting Matlab plot generated shown in Fig. 1.4. Also shown in the figure is the theoretical proportion of Open channels x as $N \rightarrow \infty$, given by:

$$x = \frac{\alpha}{\alpha + \beta} \left[1 - e^{-(\alpha+\beta)t} \right] \quad (1.5)$$

which is the solution of the deterministic model

$$\frac{dx}{dt} = \alpha(1 - x) - \beta x \quad (1.6)$$

with initial value $x(0) = 0$. Depending on the context, a sufficiently large number of stochastic processes can usually be averaged to yield an equivalent deterministic description of the system, such as Eq. 1.6. The remainder of this text will concern itself primarily with deterministic models, as these are more common in the bioengineering context (Fig. 1.2). \square

1.3.4 Continuous Versus Discrete

When modelling biological tissues, it is often convenient to represent the tissue using a collection of distinct subunits, each representing individual cells or groups of cells. In general, models comprised of a finite number of distinct subunits are referred

Fig. 1.2 Proportion of open ion channels as a function of time, solved using Matlab[®] for $N = 1000$, $\alpha = 0.1 \text{ ms}^{-1}$ and $\beta = 0.01 \text{ ms}^{-1}$. The *solid curve* is the output of the stochastic simulation using $\Delta t = 0.001 \text{ ms}$, whilst the *dashed curve* is the solution of the deterministic equivalent given by Eq. 1.6. The channels are all assumed to be in the Closed state at $t = 0$

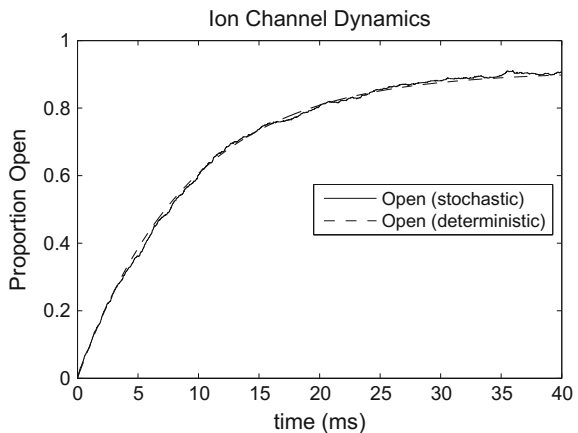
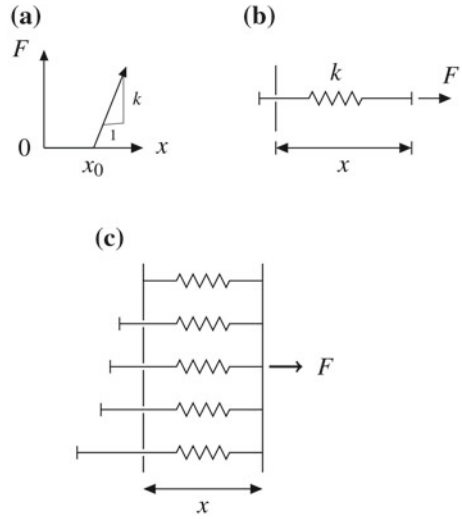


Fig. 1.3 a
 Force-displacement characteristic of non-linear spring unit: x_0 is the slack length and k is the spring constant, **b** Schematic representation of the same spring unit: the spring resists stretch only after its slack length has been exceeded, **c** Schematic representation of entire muscle fibre comprised of multiple parallel spring units, each with varying slack length



to as *discrete*. In the limit of an infinite number of subunits, each one vanishingly small, such a model can be represented by a *continuous* ‘macroscopic’ equivalent description, with all model variables being continuous in time and/or space. The ion channel kinetic model of the previous section is an example of a discrete system when the number of channels N is finite. As N becomes large, the discrete description is well-approximated by the continuous formulation of Eq. 1.6.

Example 1.4 Another example of a discrete model is illustrated in Fig. 1.3, which describes the non-linear mechanical properties of passive muscle. In this model, the muscle is assumed to consist of multiple ‘springs’ in parallel, each with its own slack length x_0 . These spring elements could represent, for example, individual collagen fibres which exhibit slack due to their helical arrangement which ‘uncoils’ when the fibre is stretched. Below the slack length, the collagen fibres offer no resistance to stretch, but after uncoiling, they oppose any further stretch with a force in proportion to their extension beyond the slack length. The force-displacement characteristic of such a spring unit is shown in Fig. 1.3a, with a schematic representation of the unit in Fig. 1.3b. In the discrete muscle model, the multiple spring units are connected in parallel, each with identical stiffness k but varying slack length, as depicted in Fig. 1.3c.

Let N be the number of parallel springs and k_{max} the maximum stiffness of the muscle when all springs have been recruited, that is all springs have all been stretched above their slack length. Since the springs are connected in parallel,

$$k_{max} = Nk \tag{1.7}$$

Furthermore, we assume that the slack lengths of all unit springs follow a lognormal distribution, characterised by parameters μ and σ according to:

$$\phi_x = \frac{1}{\sigma\sqrt{2\pi}} \int_0^x \frac{1}{\xi} e^{-\frac{(\ln\xi-\mu)^2}{2\sigma^2}} d\xi \quad (1.8)$$

where ϕ_x denotes the proportion of all springs having slack lengths between 0 and x . Lognormal distributions are useful for describing strictly positive quantities: the distribution given by Eq. 1.8 is related to the standard normal distribution in that the quantity $(\ln \xi)$ is distributed normally with mean μ and standard deviation σ . The discrete-spring model is therefore characterised by four parameters: k_{max} , N , μ , and σ . Solve this model in Matlab, and determine the force-displacement characteristic for reasonable physiological values of these parameters.

Answer: To represent Eq. 1.8 as a discrete distribution in N springs, we solve for the N values of slack length x which yield $\phi_x = \frac{i}{N+1}$ for $i = 1 \dots N$. To do this, Matlab provides the helpful `logninv` function, as implemented in the following code:

```
N = 10;
k_max = 87;      % N/cm
mu = 3.88;
sigma = 0.027;
x_start = 40;   % cm
x_end = 52;    % cm
x0 = logninv((1/(N+1):1/(N+1):N/(N+1)), mu, sigma);
x = [x_start, x0, x_end];
F = zeros(size(x));
dynamic_k = 0;
for i=2:N+2
    F(i) = F(i-1) + dynamic_k*(x(i)-x(i-1));
    dynamic_k = dynamic_k + k_max/N;
end;
plot(x, F, '-k', x(2:end-1), F(2:end-1), 'ks'), ...
    xlabel('x (cm)'), ylabel('F (N)'), ...
    title('Passive Discrete Muscle Model');
```

where parameters for the discrete model ($N = 10$, $k_{max} = 87 \text{ N cm}^{-1}$, $\mu = 3.88$ and $\sigma = 0.027$) were chosen to match an empirical passive force-displacement relation for human gastrocnemius muscle given by $F = a_g[e^{k_g(x-l_g)} - 1]$ if $x > l_g$, otherwise $F = 0$, using median reported parameter values of $a_g = 0.053 \text{ N}$, $k_g = 0.92428 \text{ cm}^{-1}$ and $l_g = 41.381 \text{ cm}$ [11]. The force-displacement characteristic of this discrete model is shown in Fig. 1.4.

In the limit $N \rightarrow \infty$, the discrete model can be expressed as the continuous distribution of Eq. 1.8. In this case, the dynamic stiffness of the muscle, dF/dx , is simply equal to the proportion of all springs having slack length less than the current displacement x multiplied by the maximum muscle stiffness k_{max} :

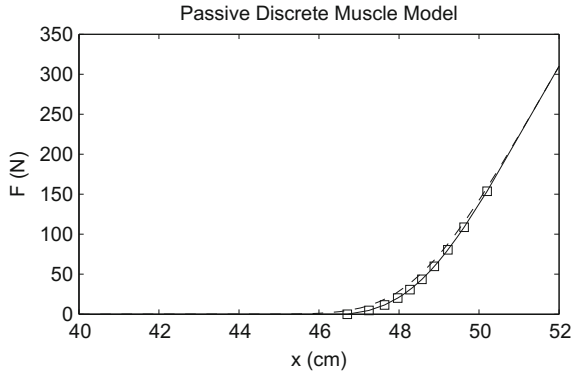


Fig. 1.4 Force-displacement characteristic of discrete 1D muscle spring model with parameters $N = 10$, $k_{max} = 87 \text{ N cm}^{-1}$, $\mu = 3.88$ and $\sigma = 0.027$. The *hollow squares* denote the positions of the piecewise linear nodes of the curve, and have x coordinates corresponding to the discrete spring slack lengths. The *dashed curve* is the solution to the continuous limit as $N \rightarrow \infty$, given by Eq. 1.10

$$\frac{dF}{dx} = k_{max}\phi_x = \frac{k_{max}}{\sigma\sqrt{2\pi}} \int_0^x \frac{1}{\xi} e^{-\frac{(\ln\xi-\mu)^2}{2\sigma^2}} d\xi \quad (1.9)$$

and taking the derivative of both sides with respect to x , we obtain the continuous equivalent:

$$\frac{d^2F}{dx^2} = \frac{k_{max}}{x\sigma\sqrt{2\pi}} e^{-\frac{(\ln x-\mu)^2}{2\sigma^2}} \quad (1.10)$$

the solution of which is also plotted in Fig. 1.4, using the initial conditions $F = 0$, $dF/dx = 0$ at $x = 0$. \square

1.3.5 Rule-Based

Rule-based models are those for which model behaviour is not expressed in terms of algebraic or differential equations, but through algorithms that readily lead to implementation on computer. An important class of such models is *cellular automata* [7], the most famous example being John Conway's *Game of Life* [8]. Such models consist of an array of cellular elements whose states are evolving with time. The current state of each cell and the state of its immediate neighbours defines its state in the next time step, according to a defined set of rules. Such a cellular automata approach has been used, for example, to model the spread of electrical activation in cardiac tissue, including factors underlying ventricular fibrillation [15]. Other types of rule-based models are used to reproduce branching features of tree-like biological

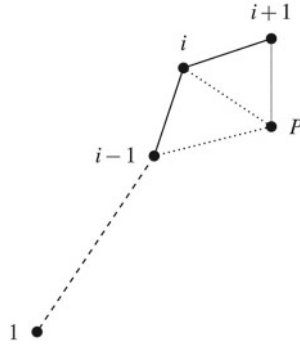


Fig. 1.5 Neuronal branching algorithm. The root node of the neuron (soma) is shown at *bottom left*, labelled as 1, with three additional nodes $i - 1$, i and $i + 1$. Point P represents a seed point. The neuron can grow towards P by forming a new segment connecting P to an existing node that minimizes the cost function $(1 - b)d_{iP} + bs_{iP}$ over all values of i , where b is a fixed ‘balance factor’, d_{iP} is the Euclidean distance between node i and P , and s_{iP} is the total path length along the neuron from the root node through node i to P

structures such as the arteries, airways of the lung, or neuronal dendrites, as described in the below example.

Example 1.5 Considered a neuronal branching model adapted from Cuntz et al. [4], in which a neural dendritic tree ‘grows’ from a single node (the soma) to connect to a set of pre-defined seed points. Consider a seed point P , as shown in Fig. 1.5, adjacent to an existing neural branch, with three representative nodes on the neuron labelled $i - 1$, i and $i + 1$. The neuron will ‘grow’ towards P by forming a new straight-line segment connecting an existing node to P . The node chosen is that which minimizes the following cost function over all nodes i :

$$cost_i = (1 - b)d_{iP} + bs_{iP} \quad (1.11)$$

where d_{iP} is the Euclidean distance between node i and seed point P , s_{iP} is the total path length along the neuron from node 1 through node i terminating at P , and b is a user-defined ‘balance factor’ lying between 0 and 1. Finally, for the optimal node i that minimizes Eq. 1.11, a new node j is appended to the neuron by drawing a straight-line segment not greater than length d_{seg} from node i towards P , where d_{seg} is a parameter defining the length of each neural segment. If the Euclidean distance between i and P is greater than d_{seg} , some random jitter is added to the coordinates of node j , otherwise the segment is connected directly to P . Implement this neural growth model in Matlab.

Answer: To write Matlab code to implement the above algorithm, we introduce the following bookkeeping convention for our neuron: define a $M \times 5$ array `neuron`, where M is the total number of nodes. Each row of the array contains five pieces of information for one node as follows:

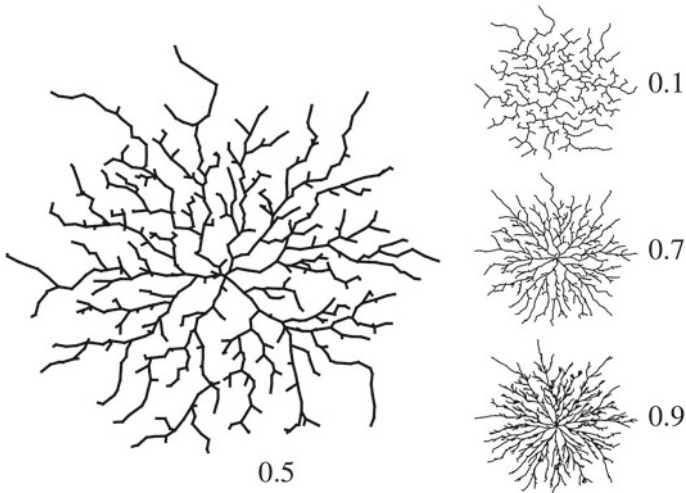


Fig. 1.6 Dendritic tree structures produced by the neuronal branching Matlab code. Numbers next to each structure represent the corresponding balance factor used. The left panel shows the default tree structure generated using the balance factor of 0.5, as per the Matlab code in the text. The right panels show, from top to bottom, corresponding structures for balance factors of 0.1, 0.7 and 0.9, using the same seed points for all four trees. These seed points were randomly placed in a circular annular region surrounding the root node

```
[node number, x coordinate, y coordinate, parent node, ...
path length to root node]
```

The array is initialised to $[1 \ 0 \ 0 \ 0 \ 0]$ representing root node 1 located at $(0, 0)$. Each time a new node is added, a row is appended to the array until all seed points have been reached. The Matlab code implementing this algorithm is given below for random set of seed points located within a circular annular region, with the resulting dendritic tree shown in Fig. 1.6 for a range of balance factors. Using this algorithm, it is possible to generate a wide range of realistic neuron morphologies by adjusting the seed point placement and number, as well as the balance factor used [4].

```
N = 500;    % total no. of seed pts
mu = 100;   % mean annular radius (um)
sd = 40;    % annular standard deviation (um)
jitt = 2;   % jitter in generated node positions (um)
seg = 10;   % neuron segment length (um)
bf = 0.5;   % balance factor

% generate random seed points
theta = 2*pi*rand(1,N);
r = mu + sd*randn(1,N);
```

```

x = r.*cos(theta);
y = r.*sin(theta);

% initialize root node (soma)
neuron = [1, 0, 0, 0, 0];
points_left = N; % seed points left to connect
nodes = 1;

% generate neural segments
while (points_left > 0)
    best_dist = inf;
    for ii = 1:nodes
        dist_1 = sqrt((x-neuron(ii,2)).^2 + ...
            (y-neuron(ii,3)).^2);
        dist_2 = dist_1 + neuron(ii,5);
        dist = (1-bf)*dist_1 + bf*dist_2;
        [min_dist, seed_pt] = min(dist);
        if (min_dist < best_dist)
            best_dist = min_dist;
            best_dist_1 = dist_1(seed_pt);
            best_pt = seed_pt;
            best_node = ii;
        end;
    end;
    if (best_dist_1 <= seg)
        node_x = x(best_pt);
        node_y = y(best_pt);
        a = find(neuron(:,1)==best_node);
        neuron = [neuron; [nodes+1, node_x, node_y, ...
            best_node, neuron(a,5)+best_dist_1]];
        x(best_pt) = [];
        y(best_pt) = [];
        points_left = points_left-1;
    else
        jitter = jitt*randn(2,1);
        a = find(neuron(:,1)==best_node);
        x0 = neuron(a,2);
        y0 = neuron(a,3);
        node_x = x0+seg*(x(best_pt)-x0)/best_dist_1 + ...
            jitter(1);
        node_y = y0+seg*(y(best_pt)-y0)/best_dist_1 + ...
            jitter(2);
        neuron = [neuron; [nodes+1, node_x, node_y, ...
            best_node, seg+neuron(a,5)]];
    end;
end;

```

```

        nodes = nodes+1;
    end;

    % plot neuron
    x1 = neuron(2,2);
    y1 = neuron(2,3);
    plot([0 x1], [0 y1], 'k', 'LineWidth', 2), axis('square');
    if (nodes > 2)
        hold on;
        for jj = 3:nodes
            x1 = neuron(jj,2);
            y1 = neuron(jj,3);
            a = find(neuron(:,1)==neuron(jj,4));
            x2 = neuron(a,2);
            y2 = neuron(a,3);
            plot([x1 x2], [y1 y2], 'k', 'LineWidth', 2), ...
                axis('square');
            set(gca,'xticklabel',[]);
            set(gca,'yticklabel',[]);
        end;
        hold off;
    end;
end;

```

1.4 Dimensional Analysis

Useful information on a physical system can often be obtained purely through analysis of the dimensions of its physical properties, a process referred to as *dimensional analysis*. This process can readily lead to mathematical relationships between these physical properties in the absence of other detailed information on physical principles involved.

1.4.1 Dimensions and Units

Physical properties of a system, expressed as variables or parameters, are associated with a set of seven *fundamental dimensions*, each of which has its own base unit of measure. A list of the fundamental dimensions and base units according to the International System of Units (SI)⁵ is given in Table 1.1. The dimensions of any other physical quantity can be expressed in terms of these fundamental quantities. Thus,

⁵Système International d'Unités.

Table 1.1 Fundamental dimensions and SI base units

| Dimension | Symbol | SI unit name | SI symbol |
|---------------------|----------|--------------|-----------|
| Length | L | metre, meter | m |
| Mass | M | kilogram | kg |
| Time | T | second | s |
| Electric current | I | ampere | A |
| Temperature | Θ | kelvin | K |
| Amount of substance | N | mole | mol |
| Luminous intensity | J | candela | cd |

for example, the dimensions of area are L^2 and volume L^3 , where L is the symbol for the dimension of length (Table 1.1).

In any equation that links physical quantities together, all terms must be consistent in terms of their dimensionality. In practice, this means that (a) both sides of an equation must have the same dimension, (b) each term appearing in a sum must have the same dimension and (c) arguments of transcendental and special functions such as exponentials, logarithms and trigonometric functions must be dimensionless. Knowledge of the relationship between various physical quantities allows their dimension to be readily determined.

Example 1.6 Find the dimensions of force.

Answer: To find the dimensions of force F , which we denote by $[F]$, we can use the equation $F = ma$, where m is mass and a acceleration, to obtain:

$$\begin{aligned}
 [F] &= [m][a] \\
 &= (M) \left(\frac{L}{T^2} \right) \\
 &= MLT^{-2} \quad \square
 \end{aligned}$$

Example 1.7 Determine the dimensions of pressure.

Answer: We can use the relation $P = F/A$, where P is pressure, F is force and A area to obtain:

$$\begin{aligned}
 [P] &= [F]/[A] \\
 &= (MLT^{-2})/(L^2) \\
 &= ML^{-1}T^{-2} \quad \square
 \end{aligned}$$

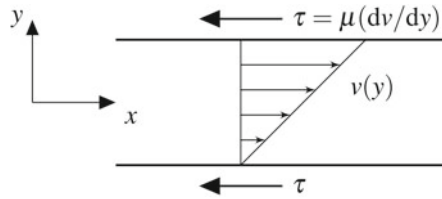


Fig. 1.7 Fluid with linearly varying velocity profile. Shear stress τ (in Nm^{-2}) is defined as the viscosity μ multiplied by the velocity gradient dv/dy

Note that some physical quantities are themselves dimensionless. This includes the radian measure of an angle, defined as the arc length subtended on a circle by the angle divided by the radius. Hence its dimension is $L/L = 1$.

Example 1.8 A final example relates to the dimension of *viscosity*, loosely defined as the internal friction of a fluid moving past itself. To more formalise its definition, consider the case of a fluid moving in a medium such that its velocity v is linearly varying with spatial position y , as shown in Fig. 1.7. The relative motion of fluid layers generates an internal shear stress τ , defined as force per unit area, which opposes the fluid motion and is given by:

$$\tau = \mu \left(\frac{dv}{dy} \right)$$

where μ is the viscosity. Using this expression, determine the dimension of μ .

Answer:

$$\begin{aligned} [\tau] &= [\mu] \left[\frac{dv}{dy} \right] \\ \frac{[F]}{[A]} &= [\mu] \frac{[v]}{[y]} \\ \therefore [\mu] &= \frac{[F][y]}{[A][v]} \\ &= \frac{(MLT^{-2})(L)}{(L^2)(LT^{-1})} \\ &= ML^{-1}T^{-1} \end{aligned}$$

□

Dimensions of some common physical quantities are summarised in Table 1.2.

Table 1.2 Dimensions and SI units of common physical quantities

| Quantity | Dimension | SI unit |
|--------------------|--------------------|---|
| Area | L^2 | m^2 |
| Volume | L^3 | m^3 |
| Velocity | LT^{-1} | m s^{-1} |
| Acceleration | LT^{-2} | m s^{-2} |
| Mass density | ML^{-3} | kg m^{-3} |
| Force | MLT^{-2} | $\text{kg m s}^{-2} = \text{N (newton)}$ |
| Pressure | $ML^{-1}T^{-2}$ | $\text{N m}^{-2} = \text{Pa (pascal)}$ |
| Work, energy | ML^2T^{-2} | $\text{N m} = \text{J (joule)}$ |
| Electric charge | IT | $\text{A s} = \text{C (coulomb)}$ |
| Electric potential | $ML^2T^{-3}I^{-1}$ | $\text{kg m}^2 \text{s}^{-3} \text{A}^{-1} = \text{V (volt)}$ |

1.4.2 Buckingham π -Theorem

The Buckingham π -theorem [2] is a powerful tool in dimensional analysis that allows mathematical relationships to be derived between physical quantities from dimensional considerations only, without detailed knowledge of the physical principles involved. Consider a system involving n dimensionally independent physical quantities q_1, q_2, \dots, q_n , such that there exists a non-trivial mathematical relationship between them given by

$$f(q_1, q_2, \dots, q_n) = 0 \quad (1.12)$$

If the dimensions of the n quantities can be expressed using m fundamental dimensions, then the Buckingham π -theorem states that $n - m$ independent dimensionless quantities $\pi_1, \pi_2, \dots, \pi_{n-m}$ can be defined from the products and quotients of the original quantities such that a non-trivial mathematical relationship corresponding to Eq. 1.12 exists between these dimensionless quantities:

$$f^*(\pi_1, \pi_2, \dots, \pi_{n-m}) = 0 \quad (1.13)$$

The theorem gives no information of the nature of f^* , only that it exists. The precise form of f^* must be derived from theoretical principles or from experiment.

Example 1.9 Use Buckingham's- π theorem to derive an expression for blood flow in a circular-cylindrical artery.

Answer: We assume that the vessel is a circular cylinder of diameter D and length L , and that flow arises due to a pressure differential ΔP across the ends of the tube. For this problem, the physical quantities to be included are vessel diameter D ($[D] = L$), pressure difference per unit length of vessel $\Delta P/L$ ($[\Delta P/L] = ML^{-2}T^{-2}$), viscosity of blood μ ($[\mu] = ML^{-1}T^{-1}$), and flow Q ($[Q] = L^3T^{-1}$). Since there are four physical quantities in three fundamental dimensions, it follows that there will only

a single ($4 - 3 = 1$) independent dimensionless quantity π_1 that can be formed. Taking the product of powers of all four original physical quantities, we form the new quantities

$$\pi_i = D^a \left(\frac{\Delta P}{L} \right)^b \mu^c Q^d \quad (1.14)$$

where the powers a, b, c, d will be chosen to make these quantities dimensionless. Inserting the fundamental dimensions into Eq. 1.14, we obtain:

$$\begin{aligned} [\pi_i] &= [L]^a [ML^{-2}T^{-2}]^b [ML^{-1}T^{-1}]^c [L^3T^{-1}]^d \\ &= L^{a-2b-c+3d} M^{b+c} T^{-2b-c-d} \end{aligned}$$

For these quantities to be dimensionless, we require that

$$\begin{aligned} a - 2b - c + 3d &= 0 \\ b + c &= 0 \\ -2b - c - d &= 0 \end{aligned}$$

which has infinitely many solutions of the form

$$\begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} -4 \\ -1 \\ 1 \\ 1 \end{pmatrix} \alpha$$

where α may be freely chosen. Choosing $\alpha = 1$, we have

$$\pi_1 = D^{-4} \left(\frac{\Delta P}{L} \right)^{-1} \mu Q$$

and from the theorem of Buckingham (Eq. 1.13), there exists a non-trivial function f^* such that

$$f^* \left(D^{-4} \left(\frac{\Delta P}{L} \right)^{-1} \mu Q \right) = 0$$

which implies

$$D^{-4} \left(\frac{\Delta P}{L} \right)^{-1} \mu Q = c$$

where c is a dimensionless constant. Hence, on re-arranging:

$$Q = c \left(\frac{D^4}{\mu} \right) \left(\frac{\Delta P}{L} \right)$$

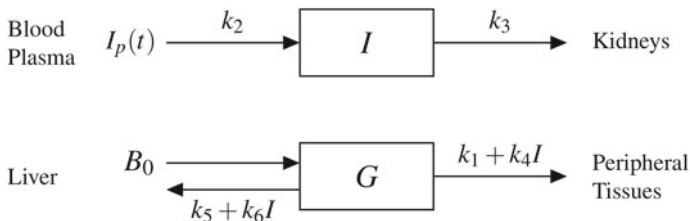


Fig. 1.8 Two-compartment pharmacokinetic model of glucose-insulin interaction in the human body. G denotes glucose concentration in a lumped glucose compartment and I is insulin concentration in another lumped remote compartment. $I_p(t)$ represents input insulin injected into the blood plasma, and B_0 is the basal production of glucose by the liver. Insulin is extracted into the kidneys, whilst glucose is uptaken by the peripheral tissues and the liver. The latter rates of glucose uptake are dependant on I . k_1 - k_6 are rate parameters

From dimensional analysis alone, we have deduced that blood flow in a circular vessel is proportional to both the pressure gradient along the vessel and the fourth power of its diameter, as well as inversely proportional to the viscosity, without invoking any advanced principles of fluid mechanics! □

1.5 Model Scaling

For mathematical models involving multiple variables and parameters, it is often an advantage to apply an appropriate *scaling* of variables to reduce the total number of parameters characterising the system. Scaling is often used to expedite mathematical analysis of model behaviour, but is rarely used in computational modelling of biological and physiological systems, due to the complexity of the models involved. Scaling, however, is useful in reducing the total number of parameters in a given model, and is particularly useful when fitting models to experimental data.

The aim therefore of model scaling is to reduce the number of parameters characterising a system. Typically, although not necessarily, the scaling of variables is chosen to produce dimensionless quantities.

Example 1.10 Consider a minimal model of glucose-insulin kinetics⁶ in the human body, as shown in Fig. 1.8. The concentrations of insulin and glucose are denoted by I and G respectively, representing an insulin ‘remote’ compartment and a glucose compartment. The input to this system is a time-dependent insulin concentration $I_p(t)$, injected into the blood plasma. Model parameters are B_0 , the basal rate of glucose production by the liver, and rates k_1 - k_6 , representing a total of seven parameters. Can this number of parameters be reduced?

Answer: The system shown in Fig. 1.8 can be represented by the following pair of equations:

⁶Adapted from Bergman et al. [1], model VI.

$$\frac{dI}{dt} = k_2 I_p - k_3 I \quad (1.15)$$

$$\frac{dG}{dt} = B_0 - (k_1 + k_4 I + k_5 + k_6 I)G \quad (1.16)$$

Now, when $I_p = 0$ (i.e. no insulin injected), the steady-state values of I and G can be determined by setting their corresponding time-derivatives to zero, namely:

$$\begin{aligned} k_2 I_p - k_3 I &= 0 \\ B_0 - (k_1 + k_4 I + k_5 + k_6 I)G &= 0 \end{aligned}$$

to yield $I_\infty = 0$ and $G_\infty = \frac{B_0}{k_1 + k_5}$. In particular, the latter G_∞ represents a ‘characteristic concentration’ of the system, that can be used to scale I , G and I_p , to yield the new dimensionless quantities:

$$I^* = \frac{I}{G_\infty} = \frac{(k_1 + k_5)I}{B_0} \quad (1.17)$$

$$G^* = \frac{G}{G_\infty} = \frac{(k_1 + k_5)G}{B_0} \quad (1.18)$$

$$I_p^* = \frac{I_p}{G_\infty} = \frac{(k_1 + k_5)I_p}{B_0} \quad (1.19)$$

Furthermore, we note that the dimensions of rate parameter k_3 in Eq. 1.15 are T^{-1} (see Problem 1.3), which can be used to scale t to yield the dimensionless time variable:

$$t^* = k_3 t \quad (1.20)$$

The time-derivatives in Eqs. 1.15 and 1.16 can now be related to derivatives of the dimensionless variables:

$$\frac{dI}{dt} = \frac{dI}{dI^*} \frac{dI^*}{dt^*} \frac{dt^*}{dt} = G_\infty \frac{dI^*}{dt^*} k_3 = \left(\frac{B_0 k_3}{k_1 + k_5} \right) \frac{dI^*}{dt^*} \quad (1.21)$$

$$\frac{dG}{dt} = \frac{dG}{dG^*} \frac{dG^*}{dt^*} \frac{dt^*}{dt} = G_\infty \frac{dG^*}{dt^*} k_3 = \left(\frac{B_0 k_3}{k_1 + k_5} \right) \frac{dG^*}{dt^*} \quad (1.22)$$

Substituting Eqs. 1.17–1.22 into 1.15 and 1.16, after a little algebraic manipulation we obtain the scaled system of equations:

$$\frac{dI^*}{dt^*} = p_1 I_p^* - I^* \quad (1.23)$$

$$\frac{dG^*}{dt^*} = p_2 (1 - G^*) - p_3 I^* G^* \quad (1.24)$$

with $p_1 = \frac{k_2}{k_3}$, $p_2 = \frac{k_1+k_5}{k_3}$ and $p_3 = \frac{B_0(k_4+k_6)}{k_3(k_1+k_5)}$, representing only three parameters that fully characterise system behaviour. Note that other choices of scaling are also possible. \square

Problems

1.1 (a) Verify from first principles that the following 1D diffusion equation is linear:

$$\frac{\partial c}{\partial t} = D \frac{\partial^2 c}{\partial x^2}$$

where c is concentration and D is a fixed diffusion coefficient.

(b) Verify using first principles that the following equation governing bacterial population N in a Petri dish is non-linear:

$$\frac{dN}{dt} = kN \left(1 - \frac{N}{N_{max}} \right)$$

where k and N_{max} are fixed parameters.

1.2 Determine the dimensions of the following quantities:

- (a) Weight
- (b) Diffusion coefficient (see Problem 1.1a)
- (c) Capacitance
- (d) Resistance
- (e) Cardiac output
- (f) Heart rate
- (g) Radian angle measure

1.3 For the minimal glucose-insulin kinetic model given by

$$\begin{aligned} \frac{dI}{dt} &= k_2 I_p - k_3 I \\ \frac{dG}{dt} &= B_0 - (k_1 + k_4 I + k_5 + k_6 I) G \end{aligned} \quad (1.25)$$

the dimensions of I , G and I_p are all $N L^{-3}$. Find the corresponding dimensions of B_0 , k_1 , k_2 , k_3 , k_4 , k_5 , and k_6 .

1.4 (a) In a left ventricular assist pump, the pressure head ΔP developed across the device is given by the following empirical relationship:

$$\Delta P = c_0 + c_1 Q_p^3 + c_2 \omega^2$$

where ω is the angular velocity of the pump impeller in units of rad s^{-1} and Q_P is the pump flow rate in $\text{m}^3 \text{s}^{-1}$. If ΔP is in units of Pa ($=\text{N m}^{-2}$), find the dimensions and units of the coefficients c_0 , c_1 and c_2 .

(b) The opening (α) and closing (β) rate coefficients for potassium ionic channels in a particular excitable cell membrane are given by the following empirical expressions:

$$\alpha = \frac{a(V+b)}{1 - \exp\left[\frac{-(V+b)}{c}\right]} \quad \beta = A \exp\left[\frac{-(V+B)}{C}\right]$$

where V is the transmembrane potential in mV, and α , β are in units of s^{-1} . Find the units of parameters a , b , c , A , B and C .

1.5 (a) The electrical resistance across a block of material of resistivity ρ is given by $R = \rho L/A$, where A is the cross-sectional area and L is the length of the block. Find the dimensions of resistivity.

(b) A monopolar disk stimulating electrode of diameter D is embedded in a medium of resistivity ρ . Assuming the medium is an infinite hemisphere centred around the disk, use dimensional analysis to find a formula for the access resistance of the electrode, defined as the resistance between the electrode disk and the hemispherical boundary at infinity.

1.6 The 1D momentum balance equation for the velocity u of a fluid may be written as

$$\rho \left(\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} \right) = -\frac{\partial p}{\partial x} + \mu \frac{\partial^2 u}{\partial x^2}$$

where x is the spatial position, t is time, ρ is fluid density, μ the viscosity and p is the pressure. Assuming the system has characteristic velocity, length and frequency of V , L , and ω respectively, scale all variables to dimensionless quantities and obtain the dimensionless form of this equation. How many parameters characterise the dimensionless system?

1.7 The Hodgkin–Huxley [10] equations governing electrical activity in neuronal axons may be written as a coupled system of four differential equations as follows:

$$\begin{aligned} \frac{dV}{dt} &= -\frac{1}{C} [g_{Na}m^3h(V - V_{Na}) + g_Kn^4(V - V_K) + g_L(V - V_L)] \\ \frac{dn}{dt} &= \alpha_n(1 - n) - \beta_n n \\ \frac{dm}{dt} &= \alpha_m(1 - m) - \beta_m m \\ \frac{dh}{dt} &= \alpha_h(1 - h) - \beta_h h \end{aligned}$$

with

$$\begin{aligned}\alpha_n &= \frac{A_n(V+V_{an})}{1-\exp\left[\frac{-(V+V_{an})}{s_{an}}\right]} & \beta_n &= B_n \exp\left[\frac{-(V+V_{bn})}{s_{bn}}\right] \\ \alpha_m &= \frac{A_m(V+V_{am})}{1-\exp\left[\frac{-(V+V_{am})}{s_{am}}\right]} & \beta_m &= B_m \exp\left[\frac{-(V+V_{bm})}{s_{bm}}\right] \\ \alpha_h &= A_h \exp\left[\frac{-(V+V_{ah})}{s_{ah}}\right] & \beta_h &= \frac{B_h}{1+\exp\left[\frac{-(V+V_{bh})}{s_{bh}}\right]}\end{aligned}$$

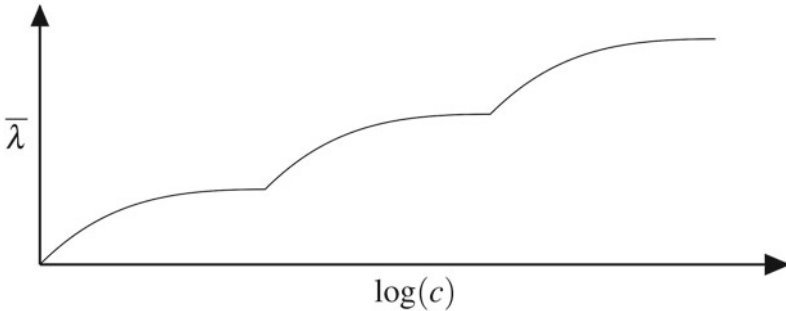
where V is the transmembrane potential (typically in units of mV), n, m, h are dimensionless gating variables, and $i_{\text{stim}}(t)$ is the input stimulus current. The remaining terms represent 25 parameters, namely: $C, g_{Na}, V_{Na}, g_K, V_K, g_L, V_L, A_n, V_{an}, s_{an}, B_n, V_{bn}, s_{bn}, A_m, V_{am}, s_{am}, B_m, V_{bm}, s_{bm}, A_h, V_{ah}, s_{ah}, B_h, V_{bh}$ and s_{bh} .

Variables V and t may be transformed into corresponding dimensionless variables using

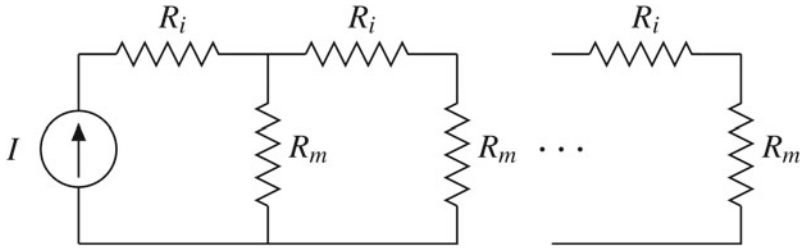
$$V^* = \frac{V - V_K}{V_{Na} - V_K} \quad t^* = B_n t$$

Using these, find the dimensionless form of the above Hodgkin–Huxley equations. How many parameters are needed to characterise this dimensionless system?

1.8 A hydrogel-based optical sensor responds to concentrations of an analyte (c) through changes in the mean wavelength ($\bar{\lambda}$) of its emitted reflectance. The wavelength – log-concentration profile consists of three sequential levels of saturation, as shown below. Formulate a possible model of this behaviour using a system of differential equations.



1.9 Steady-state passive electrical behaviour of an unmyelinated nerve cell axon can be approximated by a cylinder of length 10 mm, radius 25 μm , filled with axoplasmic medium of resistivity 0.2 Ωm , and surrounded by a membrane of resistance 0.1 Ωm^2 . If a current of 1 mA is injected into one end of the axon, and assuming the extracellular potential is set everywhere to ground, this system can be represented as N discrete circuit elements shown below:



where the value of each R_i and R_m will depend on N . The membrane voltage across the R_m 's will monotonically decrease from the current source, and the distance at which it has decayed to $e^{-1} \approx 0.368$ of its maximum value is known as the *length constant* of the axon. Determine the length constant for $N = 5, 10, 20, 40, 80$ and 160 . If the length constant for $N = 160$ is assumed to be the exact value, what minimum value of N in this list is sufficient to guarantee a length constant accuracy of 1%?

1.10 A simplified model of cardiac ventricular electrical activity, based on the cellular automata model of Mitchell et al. [15], consists of a 50×50 square grid, each square representing a small $2 \times 2 \text{ mm}^2$ region of electrical activity in the heart. The grid is wrapped around into a cylinder, so that the left and right edges are assumed to be in contact. Every region has exactly eight neighbours, with the exception of those squares on the top and bottom edges alone. Each square in the grid can be in one of four states: 0 (quiescent), 1 (relative refractory), 2 (absolute refractory) and 3 (excited). Once a region is excited, it will move through states $3 \rightarrow 2 \rightarrow 1 \rightarrow 0$ in that order, unless it is prematurely excited again.

| State | Description | Behaviour |
|-------|----------------------|--|
| 0 | quiescent | the region is excited on the next time step if at least one of its eight neighbours is currently excited |
| 3 | excited | the region is excited, and can excite neighbouring regions on the next time step |
| 2 | absolute refractory | the region cannot be excited, nor can it excite any of its neighbours |
| 1 | relative refractory | the region is excited on the next time step if more than one of its eight neighbours is currently excited. The number of excited neighbours required is dependent on the time elapsed since entering this state, according to the following: |
| | Time elapsed (in ms) | Excited neighbours required |
| | ≤ 2 | 8 |
| | ≤ 4 | 7 |
| | ≤ 6 | 6 |
| | ≤ 8 | 5 |
| | ≤ 12 | 4 |
| | ≤ 20 | 3 |
| | ≤ 50 | 2 |

The duration of the excited state is fixed at T_{ex} and the duration of the relative refractory period is fixed at 50ms. To account for the inhomogeneous refractory properties of the tissue, the duration of the total refractory period (absolute + relative) is a random variable for each region, drawn from a normal distribution of mean T_{rp} and standard deviation σ_{rp} . The ventricles are excited by a periodic impulse from the atrioventricular (AV) node, of period T , located in a small square on the top border of the grid one quarter of the way in from the left edge. Whenever applied, this impulse can only excite the small square if its current state is not absolute refractory. All state behaviours and parameter values are given on the following page.

| Parameter | Description | Value |
|---------------|---|--------|
| T_{ex} | Duration of excited state | 70 ms |
| T_{rp} | Mean total refractory period | 250 ms |
| σ_{rp} | Standard deviation of total refractory period | 100 ms |
| T | AV node pacing period | 0.8 s |
| d_T | Time step increment | 2 ms |

- (a) Implement this model in Matlab, and plot a snapshot of the heart's electrical activity at $t = 1.65$ s.
- (b) Decrease the pacing period to 0.2 s, and plot the electrical activity again at $t = 1.7$ s. You should observe chaotic activation akin to ventricular fibrillation.

References

1. Bergman RN, Ider YZ, Bowden CR, Cobelli C (1979) Quantitative estimation of insulin sensitivity. *Am J Physiol* 236:E667–E677
2. Buckingham E (1914) On physically similar systems: illustrations of the use of dimensional equations. *Phys Rev* 4:345–376
3. Cuellar AA, Lloyd CM, Nielsen PF, Bullivant DP, Nickerson DP, Hunter PJ (2003) An overview of CellML 1.1, a biological model description language. *Simulation* 79(12):740–747
4. Cuntz H, Forstner F, Borst A, Häusser A (2010) One rule to the grow them all: a general theory of neuronal branching and its practical application. *PLoS Comput Biol* 6(8):e1000877
5. Edelstein-Keshet L (2005) *Mathematical models in biology*. SIAM, Philadelphia
6. Erdemir A, Guess TM, Halloran J, Tadepalli SC, Morrison TM (2012) Considerations for reporting finite element studies in biomechanics. *J. Biomech.* 45:625–633
7. Ermentrout GB, Edelstein-Keshet L (1993) Cellular automata approaches to biological modeling. *J. Theor. Biol.* 160:97–113
8. Gardner M (1970) Mathematical games - the fantastic combinations of John Conway's new solitaire game "life". *Sci Am* 223:120–123
9. Hille B (2001) *Ion channels of excitable membranes*, 3rd edn. Sinauer, Sunderland
10. Hodgkin AL, Huxley AF (1952) A quantitative description of membrane current and its application to conduction and excitation in nerve. *J Physiol (Lond)* 117:500–544
11. Hoang PD, Herbert RD, Todd G, Gorman RB, Gandevia SC (2007) Passive mechanical properties of human gastrocnemius muscle-tendon units, muscle fascicles and tendons in vivo. *J Exp Biol* 210:4159–4168

12. Hucka M, Finney A, Sauro HM, Bolouri H, Doyle JC, Kitano H, Arkin AP, Bornstein BJ, Bray D, Cornish-Bowden A, Cuellar AA, Dronov S, Gilles ED, Ginkel M, Gor V, Goryanin II, Hedley WJ, Hodgman TC, Hofmeyr J-H, Hunter PJ, Juty NS, Kasberger JL, Kremling A, Kummer U, Le Novère A, Loew LM, Lucio D, Mendes P, Minch E, Mjolsness ED, Nakayama Y, Nelson MR, Nielsen PF, Sakurada T, Schaff JC, Shapiro BE, Shimizu TS, Spence HD, Stelling J, Takahashi K, Tomita M, Wagner JM, Wang J (2003) The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics* 19:524–531
13. Keener J, Sneyd J (2009) *Mathematical physiology: I. cellular physiology*. Springer, New York
14. Keener J, Sneyd J (2009) *Mathematical physiology: II. systems physiology*. Springer, New York
15. Mitchell RH, Bailey AH, Anderson J (1992) Cellular automaton model of ventricular fibrillation. *IEEE Trans Biomed Eng* 39:253–259
16. Smith DE (1958) *History of mathematics*, vol 1, Dover edn. Dover, New York
17. Stewart I (2012) *Mathematics of life: unlocking the secrets of existence*. Profile Books, London
18. Suzuki S, Eto K, Hattori A, Yanaga K, Suzuki N (2007) Surgery simulation using patient-specific models for laparoscopic colectomy. *Stud Health Technol Inform* 125:464–466
19. Thomas R (2012) Improving medical devices using computational modeling. <https://www.asme.org/engineering-topics/articles/performance-test-codes/improving-medical-devices-using-computational-mode>. Accessed on 3 Aug 2013
20. Viceconti M, Clapworthy G, Van Sint JS (2008) The virtual physiological human - a European initiative for in silico human modelling. *J Physiol Sci* 58(7):441–446

Chapter 2

Lumped Parameter Modelling with Ordinary Differential Equations

2.1 Overview of Ordinary Differential Equations

An *ordinary differential equation* (ODE) is used to express a relationship between a function of one independent variable (typically time) and its derivatives. If no derivatives are present, the relationship is characterised by an *algebraic equation* (AE). ODEs are often used in *lumped parameter* modelling to approximate the behaviour a physical system by separating it into discrete parts, each characterised by one or more dependent variables. An example of a simple ODE is:

$$\frac{dN}{dt} = \frac{rN(K - N)}{K}, \quad N(0) = N_0 \tag{2.1}$$

where N represents the population of, say, bacteria in a Petri-dish, r is the growth rate when $N = 0$, and K is the maximum population capacity of the system. For this simple example, it is possible to obtain an exact closed-form solution for N as a function of t using the method of *separation of variables*, in which the variables are grouped on each side of the equality. Thus, we can rewrite Eq. 2.1 in the form:

$$\frac{dN}{rN(K - N)} = \frac{dt}{K}$$

Integrating both sides, we obtain

$$\int \frac{dN}{rN(K - N)} = \int \frac{dt}{K} = \frac{t}{K} + C_0 \tag{2.2}$$

where C_0 is a constant of integration. To integrate the left-hand side, we rewrite the integrand using the partial fraction expansion:

$$\frac{1}{rN(K - N)} = \frac{A_1}{rN} + \frac{A_2}{K - N} \tag{2.3}$$

where A_1, A_2 are constants to be determined. Multiplying both sides of Eq. 2.3 by rN , then setting $N = 0$, yields $A_1 = 1/K$. Similarly, multiplying both sides of Eq. 2.3 by $K - N$, then setting $N = K$, yields $A_2 = 1/rK$. Hence, the left-hand side of Eq. 2.2 can be written as:

$$\begin{aligned} \int \frac{dN}{rN(K-N)} &= \frac{1}{K} \int \frac{dN}{rN} + \frac{1}{rK} \int \frac{dN}{(K-N)} \\ &= \frac{\ln N}{rK} - \frac{\ln(K-N)}{rK} \\ &= \frac{1}{rK} \ln \left[\frac{N}{K-N} \right] \end{aligned}$$

Substituting this into the left-hand side of Eq. 2.2 and multiplying both sides by rK yields:

$$\ln \left[\frac{N}{K-N} \right] = rt + C_0 r K$$

and since $N = N_0$ when $t = 0$, we have $C_0 = \frac{1}{rK} \ln \left[\frac{N_0}{K-N_0} \right]$. Thus,

$$\ln \left[\frac{N}{K-N} \right] = rt + \ln \left[\frac{N_0}{K-N_0} \right]$$

and taking the exponential of both sides:

$$\frac{N}{K-N} = \left[\frac{N_0}{K-N_0} \right] e^{rt}$$

Finally, after a little algebraic manipulation, we obtain the closed-form solution for N as:

$$N = \frac{K e^{rt}}{\left[\frac{K-N_0}{N_0} + e^{rt} \right]}$$

which is known as the *logistic equation*. In general, however, when modelling with ODEs we must numerically-integrate to obtain an approximate solution.

When more than one ODE is involved, the set of equations is known as a system of ordinary differential equations. If the multiple set of equations includes a combination of ODEs and AEs, it is termed a differential-algebraic equation (DAE) system. If any of the differential equations involves multiple independent variables (such as time and space), then it is referred to a partial differential equation (or PDE). PDEs are discussed further in the next chapter.

Example 2.1 Consider a mass m connected to a spring, moving in the presence of a damping resistance, as shown in Fig. 2.1. Derive an ODE governing the motion of the mass.

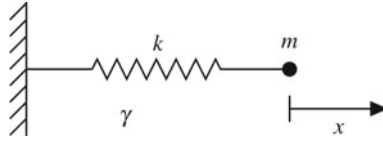


Fig. 2.1 Damped oscillator. Mass m is connected to a linear spring k in the presence of a damping medium γ . The other end of the spring is connected to a fixed support. The force exerted by the spring on the mass is $-kx$, where x is the displacement of the mass. The damping force exerted by the medium is equal to $-\gamma v$, where v is the velocity of the mass

Answer: The motion of the mass can be determined from the following relations:

- Total force acting on mass = ma , where a is the acceleration.
- Damping force = $-\gamma v$, where v is the velocity.
- Elastic force of spring = $-kx$, where x is the displacement of the mass from its resting position.

This yields the following equation for the motion of the mass:

$$ma = -\gamma v - kx$$

$$ma + \gamma v + kx = 0$$

Substituting the relationships $a = \frac{d^2x}{dt^2}$ and $v = \frac{dx}{dt}$, we obtain the following ODE:

$$m \frac{d^2x}{dt^2} + \gamma \frac{dx}{dt} + kx = 0$$

This represents a 2nd order ODE, since the highest derivative is of order 2. It can be reduced to a coupled system of 1st order ODEs by introducing the velocity variable v to obtain:

$$\frac{dx}{dt} = v$$

$$\frac{dv}{dt} = -\frac{k}{m}x - \frac{\gamma}{m}v$$

□

2.2 Linear ODEs

Ignoring constraints imposed by initial or boundary values, we assume that a given ODE is satisfied by two distinct solutions, $\phi_1(t)$ and $\phi_2(t)$. If the combination of solutions $c_1\phi_1(t) + c_2\phi_2(t)$ is also a solution, where c_1 and c_2 are any arbitrary

constants, then the ODE is *linear*. Otherwise, it is *non-linear*. In general, a linear ODE of order N is given by:

$$a_N(t) \frac{d^N x}{dt^N} + a_{N-1}(t) \frac{d^{N-1} x}{dt^{N-1}} + \cdots + a_2(t) \frac{d^2 x}{dt^2} + a_1(t) \frac{dx}{dt} + a_0(t)x = F(t) \quad (2.4)$$

where $F(t)$ is the *forcing term* and along with the coefficients $a_i(t)$ ($i = 0 \cdots N$), are all functions only of the independent variable. If $F(t) = 0$, the linear ODE is said to be *homogeneous*, and its solution is known as the *homogeneous solution*. If $F(t) \neq 0$, then the ODE is *non-homogeneous*. If one solution can be found for the ODE (i.e. a *particular solution*), then the *general solution* is given by the sum of the particular and homogeneous solutions.

If the coefficients a_i ($i = 0 \cdots N$) in Eq. 2.4 are constant, then the homogeneous ODE can be solved analytically. Consider the following N^{th} order ODE:

$$\frac{d^N x}{dt^N} + a_{N-1} \frac{d^{N-1} x}{dt^{N-1}} + \cdots + a_2 \frac{d^2 x}{dt^2} + a_1 \frac{dx}{dt} + a_0 x = 0 \quad (2.5)$$

where $a_{N-1} \cdots a_0$ are constant. To solve this ODE, substitute $x = e^{mt}$ into Eq. 2.5, where m is constant to be determined, to obtain:

$$a_{N-1} m^{N-1} e^{mt} + \cdots + a_1 m e^{mt} + a_0 e^{mt} = 0$$

Dividing throughout by e^{mt} , we obtain the *characteristic equation*:

$$a_{N-1} m^{N-1} + \cdots + a_1 m + a_0 = 0$$

which has N roots: m_1, \dots, m_N . The solution to the ODE is then given by the linear combination:

$$x(t) = C_1 e^{m_1 t} + C_2 e^{m_2 t} + \cdots + C_N e^{m_N t}$$

where the C_i ($i = 1 \cdots N$) are N integration constants whose values can be determined from the ODE *boundary conditions*. Two types of boundary conditions for N^{th} order ODEs, both linear and non-linear, are defined:

- *Initial-value problem*, where N initial conditions are specified at the start of the interval. For example:

$$\begin{aligned} \frac{d^2 x}{dt^2} + 2 \frac{dx}{dt} + 3x &= 0 \\ x(0) &= 0 \\ x'(0) &= 1 \end{aligned}$$

- *Boundary-value problem*, where N conditions are specified at either end of the interval, as in:

$$\begin{aligned} \frac{d^2x}{dt^2} + 2\frac{dx}{dt} + 3x &= 0 \\ x(0) &= 1 \\ x(1) &= -1 \end{aligned}$$

Example 2.2 Solve the ODE

$$\begin{aligned} \frac{d^2x}{dt^2} + 4\frac{dx}{dt} + 3x &= 0 \\ x(0) &= 1 \\ x'(0) &= 1 \end{aligned}$$

Answer: The characteristic equation is

$$\begin{aligned} m^2 + 4m + 3 &= 0 \\ (m + 3)(m + 1) &= 0 \\ m &= -3 \quad \text{or} \quad -1 \end{aligned}$$

Hence the solution is of the form

$$x(t) = C_1e^{-3t} + C_2e^{-t}$$

To find C_1, C_2 , we use the given initial values $x(0) = x'(0) = 1$. Noting that $x'(t) = -3C_1e^{-3t} - C_2e^{-t}$, we obtain

$$\begin{aligned} C_1 + C_2 &= 1 \\ -3C_1 - C_2 &= 1 \end{aligned}$$

which yields $C_1 = -1, C_2 = 2$. Hence, the solution to the initial-value problem is

$$x(t) = -e^{-3t} + 2e^{-t}$$

□

If the characteristic equation contains r repeated roots

$$\overbrace{m_1, m_1, \dots, m_1, \dots, m_{N-r+1}}^{r \text{ times}}$$

then the form of the solution is

$$x(t) = C_1e^{m_1t} + \underbrace{C_2te^{m_1t} + \dots + C_r t^{r-1}e^{m_1t}}_{\text{note extra powers of } t} + \dots + C_{N-r+1}e^{m_{N-r+1}t}$$

where extra powers of the dependent variable are present for the repeated roots.

Example 2.3 Find the solution to the ODE:

$$\begin{aligned}\frac{d^2x}{dt^2} + 2\frac{dx}{dt} + x &= 0 \\ x(0) &= 1 \\ x'(0) &= 0\end{aligned}$$

Answer: The ODE has a repeated root of -1 in its characteristic equation, and has the solution

$$x(t) = e^{-t} + te^{-t}$$

□

If the characteristic equation contains complex roots, then we make use of *Euler's formula*:¹

$$e^{i\theta} = \cos \theta + i \sin \theta$$

where $i = \sqrt{-1}$.

Example 2.4 Solve the ODE

$$\begin{aligned}\frac{d^2x}{dt^2} + \omega^2x &= 0 \\ x(0) &= 1 \\ x'(0) &= 0\end{aligned}$$

Answer: The characteristic equation has roots of $\pm i\omega$, hence

$$\begin{aligned}x(t) &= C_1e^{-i\omega t} + C_2e^{i\omega t} \\ x'(t) &= -i\omega C_1e^{-i\omega t} + i\omega C_2e^{i\omega t}\end{aligned}$$

Substituting the initial values at $t = 0$ yields

$$\begin{aligned}C_1 + C_2 &= 1 \\ -i\omega C_1 + i\omega C_2 &= 0\end{aligned}$$

which can be solved to obtain $C_1 = C_2 = 0.5$. Hence,

¹Named after Leonhard Euler (1707–1783), influential Swiss mathematician, physicist and engineer who made important discoveries in mathematics, mechanics, fluid mechanics, optics and astronomy.

$$\begin{aligned}
 x(t) &= 0.5e^{-i\omega t} + 0.5e^{i\omega t} \\
 &= 0.5[\cos(-\omega t) + \sin(-\omega t)] + 0.5[\cos(\omega t) + \sin(\omega t)] \\
 &= 0.5[\cos(\omega t) - \sin(\omega t)] + 0.5[\cos(\omega t) + \sin(\omega t)] \\
 &= \cos(\omega t)
 \end{aligned}$$

□

2.3 ODE Systems

A system of ODEs can be expressed in terms of 1st order ODEs expressed in the general form:

$$\begin{aligned}
 \frac{dy_1}{dt} &= f_1(t, y_1, y_2, \dots, y_N) \\
 \frac{dy_2}{dt} &= f_2(t, y_1, y_2, \dots, y_N) \\
 &\vdots \\
 \frac{dy_N}{dt} &= f_N(t, y_1, y_2, \dots, y_N)
 \end{aligned}$$

$$y_1(0) = y_{1,0}, \quad y_2(0) = y_{2,0}, \dots, \quad y_N(0) = y_{N,0}$$

where f_1, f_2, \dots, f_N represent linear or non-linear functions, and the $y_{i,0}$ ($i = 1 \dots N$) are the initial variable values. The above system can be more compactly written as

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(t, \mathbf{y}), \quad \mathbf{y}(0) = \mathbf{y}_0 \tag{2.6}$$

where $\mathbf{y} = (y_1, \dots, y_N)^T$, $\mathbf{f} = (f_1, \dots, f_N)^T$, and \mathbf{y}_0 is the initial value of \mathbf{y} at $t = 0$. Note that for any instant in time, variable \mathbf{y} contains enough information to completely characterise the system. This information is known as the system *state*, and the \mathbf{y} are known as *state-variables*.

To solve such ODE systems using Matlab, two .m files are required. One is a function that evaluates $\mathbf{f}(t, \mathbf{y})$ of Eq. 2.6, returning the state-variable derivatives at a given time and state. The other file is a script that calls one of Matlab's in-built ODE solvers, for which the previous function will be an input argument. Matlab provides the following ODE solvers, all of which use the same syntax:

```
[Tout, Yout] = odexxx('user_fun', t_span, init);
```

where odexxx stands for one of ode45, ode23, ode113, ode15s, ode23s, ode23t, and ode23tb. user_fun(t, Y) is a user-defined function to compute

the ODE derivatives as a function of the system variable array Y and the current time-value t . Note that when this function is used as an argument to the ODE solvers, its name must be enclosed in single quotes. Also note that its first argument must be the independent variable (in this case, t), irrespective if this variable is present or not in the f function of Eq. 2.6. t_span is an array of time-values specifying the output times. Alternately, t_span can also consist of just two values specifying the start and end times of integration, such as $[0 \ 100]$. Finally, the `init` argument specifies an array of initial values for Y . Note that the solver outputs an array of time values (`Tout`), as well as the calculated state-variables (`Yout`) corresponding to these times. Each column of `Yout` corresponds to one state-variable.

It is also possible to include an additional `options` argument following `init` to specify non-default settings for the ODE solver. When used, the `options` argument is initialised using the `odeset` command. For example, to specify a maximum time step of 0.001 and a relative tolerance of 10^{-4} (i.e. 0.01% accuracy), use:

```
options = odeset('MaxStep', 0.001, 'RelTol', 1e-4);
[Tout,Yout] = odexxx('user_fun', t_span, init, options);
```

Example 2.5 The Van der Pol oscillator, defined by the coupled pair of ODEs

$$\begin{aligned}\frac{dv}{dt} &= u \\ \frac{du}{dt} &= \mu(1 - v^2)u - v \\ v(0) &= 2, \quad u(0) = 0\end{aligned}$$

has been used to model many biological oscillators, including the heartbeat [11] as well as neural spiking activity, referred to as *action potentials* [2]. If parameter $\mu \geq 0$, the system will undergo stable oscillations, known as a *limit cycle*. Using Matlab, solve the Van der Pol ODE system.

Answer: To solve the Van der Pol Oscillator equations in Matlab, first define a function to output the state-variable derivatives:

```
function dy = vdp(t,y)
dy = zeros(2,1); % defines dy as a 2x1 column vector
mu = 1000;
dy(1) = y(2);
dy(2) = mu*(1 - y(1)^2)*y(2) - y(1);
```

and save to `vdp.m`. Then implement the following separate script to solve the ODEs from $t = 0$ to 3000:

```
[T,Y] = ode15s('vdp',[0 3000],[2 0]);
plot(T,Y(:,1),'k-', legend('v'));
```

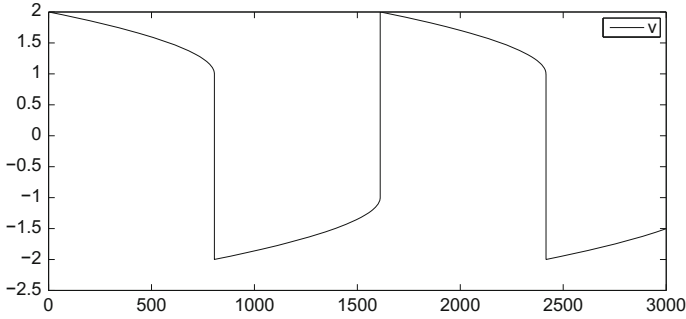


Fig. 2.2 Van der Pol oscillator output

which produces the plot shown in Fig. 2.2. □

2.3.1 Example Model 1: Cardiac Mechanics

A lumped parameter mechanics model of the cardiac left ventricle coupled to the systemic circulation [9] is shown in Fig. 2.3. Here, the various elements of the circulation are represented using electric circuit analogues:

- voltage is analogous to pressure
- current is analogous to volumetric flow rate
- resistance equals pressure across an element divided by flow through it (the hydraulic equivalent of Ohm's law)
- diodes are analogous to valves, allowing only one-way flow
- capacitance is analogous to vessel compliance (C), and equals the volume of fluid (V) stored in the vessel divided by the pressure (P). Writing

$$V = CP$$

and taking derivatives of both sides, we obtain:

$$\frac{dV}{dt} = C \frac{dP}{dt}$$

or $Q = C \frac{dP}{dt}$

where Q is the volumetric flow rate (i.e. fluid volume per unit time).

Left-ventricular pressure (P_v) is given by

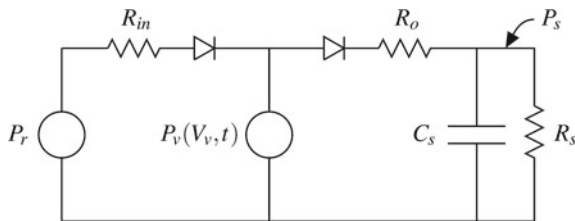


Fig. 2.3 Time-varying elastance model of left ventricle. P_v is the left ventricular pressure, which is a function of left ventricular volume (V_v) and time. P_r represents a fixed filling pressure from a venous reservoir, R_{in} is the input filling resistance, R_o is the resistance of the aorta, P_s is the lumped systemic circulation pressure and R_s , C_s represent the systemic resistance and compliance respectively

$$P_v = a(V_v - b)^2 + (cV_v - d)f(t)$$

where a , b , c , d are parameters, and $f(t)$ describes a time-varying elastance given by:

$$f(t) = \begin{cases} \sin^2\left(\frac{\pi t}{2t_p}\right) & 0 \leq t < t_p \\ \cos^2\left(\frac{\pi(t-t_p)}{2(t_s-t_p)}\right) & t_p \leq t < t_s \\ 0 & t_s \leq t < T \end{cases}$$

$$f(t+T) = f(t)$$

where T is the heart period and t_p , t_s refer respectively to peak contraction time and total contraction time (systole). All model parameters are given in Table 2.1. The state-variables for this model are the systemic pressure P_s and the ventricular volume V_v . From the circuit diagram of Fig. 2.3, we can readily write expressions for the flow Q_{in} entering the ventricle from P_r , as well as the flow Q_{out} exiting through R_o as follows:

$$Q_{in} = \begin{cases} \frac{P_r - P_v}{R_{in}} & P_r > P_v \\ 0 & P_r \leq P_v \end{cases} \quad (\text{due to input valve})$$

$$Q_{out} = \begin{cases} \frac{P_v - P_s}{R_o} & P_v > P_s \\ 0 & P_v \leq P_s \end{cases} \quad (\text{due to output valve})$$

and since Q_{out} flows through the parallel systemic compliance and resistive branches, we can write:

$$Q_{out} = C_s \frac{dP_s}{dt} + \frac{P_s}{R_s}$$

Hence, the ODEs for this model are:

Table 2.1 Parameter values of ventricular elastance model

| Parameter | Value | Parameter | Value |
|-----------|---------------------------------------|-----------|-------------------------------|
| R_o | $0.06 \text{ mmHg s cm}^{-3}$ | a | $0.0007 \text{ mmHg cm}^{-6}$ |
| C_s | $2.75 \text{ cm}^3 \text{ mmHg}^{-1}$ | b | 8 cm^3 |
| R_s | 1 mmHg s^{-3} | c | 1.5 mmHg cm^{-3} |
| R_{in} | $0.001 \text{ mmHg s cm}^{-3}$ | d | 0.9 mmHg |
| P_r | 10 mmHg | t_p | 0.35 s |
| T | 1 s | t_s | 0.8 s |

$$\frac{dP_s}{dt} = \frac{Q_{out}}{C_s} - \frac{P_s}{R_s C_s}$$

$$\frac{dV_v}{dt} = Q_{in} - Q_{out}$$

where P_v , Q_{in} and Q_{out} are defined above. Our task is to implement this model in Matlab, and in particular, obtain the steady-state pressure-volume loop for the left ventricle.

The first step is to code the user-defined function to return the derivatives of the state-variables for any given state and time. This is shown below for the function file `heart_prime.m`:

```
function Y_prime = heart_prime(t,Y)
global Ro Cs Rs R_in Pr T a b c d tp ts;

Y_prime = zeros(2,1); % to ensure a column vector

% extract states
Ps = Y(1);
Vv = Y(2);

% determine elastance
tt = mod(t,T);
if (tt < tp)
    f = sin(pi*tt/(2*tp))^2;
elseif (tt < ts)
    f = cos(pi*(tt-tp)/(2*(ts-tp)))^2;
else
    f = 0;
end;

% determine Pv
Pv = a*(Vv-b)^2+(c*Vv-d)*f;
```

```

% determine flows
if (Pr > Pv)
    Q_in = (Pr-Pv)/R_in;
else
    Q_in = 0;
end;
if (Pv > Ps)
    Q_out = (Pv-Ps)/Ro;
else
    Q_out = 0;
end;

%evaluate derivatives
Y_prime(1) = Q_out/Cs - Ps/(Rs*Cs);
Y_prime(2) = Q_in - Q_out;

```

Note the use of the `global` keyword to declare model parameters as global variables. This allows Matlab to assign and access their values outside of the current function. Also, note the use of the `mod` (modulus) function to implement a periodic elastance. `mod(t, T)` returns the remainder on division of `t` by `T`.

Once the user-defined function has been coded, the following script can be used to solve the model, and extract and plot the steady-state ventricular pressure loop:

```

global Ro Cs Rs R_in Pr T a b c d tp ts;

% assign parameters
a = 0.0007;           % mmHg/cm^6
b = 8;                % cm^3
c = 1.5;              % mmHg/cm^3
d = 0.9;              % mmHg
tp = 0.35;           % s
ts = 0.8;             % s
Ro = 0.06;            % mmHg.s/cm^3
Cs = 2.75;            % cm^3/mmHg
Rs = 1;               % mmHg.s/cm^3
R_in = 0.001;         % mmHg.s/cm^3
Pr = 10;              % mmHg
T = 1;                % s

% solve ODEs
[Tout, Yout] = ode15s('heart_prime',[0 10*T],[50 50]);

% extract Pv, Vv

```



```

f = zeros(size(Tout));
tt = mod(Tout,T);
for ii = 1:length(f)
    if (tt(ii) < tp)
        f(ii) = sin(pi*tt(ii)/(2*tp))^2;
    elseif (tt(ii) < ts)
        f(ii) = cos(pi*(tt(ii)-tp)/(2*(ts-tp)))^2;
    else
        f(ii) = 0;
    end;
end;
Vv = Yout(:,2);
Pv = a*(Vv-b).^2+(c*Vv-d).*f;

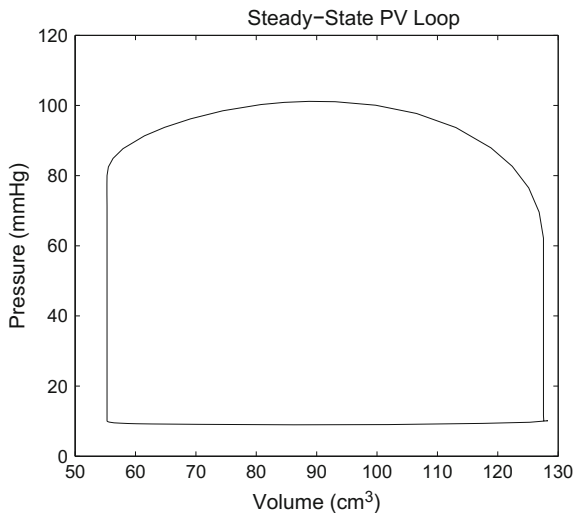
% plot the final heart period PV loop (steady state)
index = find(Tout>=9*T);
A = index(1)-1;

plot(Vv(A:end),Pv(A:end),'k-'), xlabel('Volume (cm^3)'), ...
     ylabel('Pressure (mmHg)'), title('Steady-State PV Loop');

```

which produces the steady-state pressure-volume loop plot shown in Fig. 2.4. Note that to produce this plot, it is necessary to re-extract the ventricular pressure P_v , which is calculated inside the function `heart_prime`. Once the Matlab ODE solver has completed its execution, only the state-variables at the output time values are returned. To extract any other ancillary quantities, these must be re-evaluated.

Fig. 2.4 Steady-state pressure-volume loop for time-varying elastance left ventricular model



For this model, the state variables are P_s and V_v : knowing the associated time value t allows any other model quantity (in this case P_v) to be determined. Also note that the steady-state pressure-volume loop represents a stable limit cycle of the ODE system, independent of the initial values chosen for P_s and V_v (in the above code these initial values were 50 mmHg and 50 cm³ respectively).

2.3.2 Example Model 2: Hodgkin–Huxley Model of Neural Excitation

Sir Alan Hodgkin (1914–1998) and Sir Andrew Huxley (1917–2012) shared the 1963 Nobel Prize in Physiology or Medicine (jointly with Sir John Eccles) “for their discoveries concerning the ionic mechanisms involved in excitation and inhibition in the peripheral and central portions of the nerve cell membrane”.² Their work culminated in the publication of a mathematical model of the neural electrical impulse known as the action potential, based on their electrophysiological experiments in the giant axon of the squid [5]. This model significantly advanced our understanding of active ionic mechanisms in excitable tissues, and most computational models of nerves, muscle and heart electrical activity are based on the formalism Hodgkin and Huxley pioneered several decades ago.

The space-clamped electric analogue of the Hodgkin–Huxley model is shown Fig. 2.5. In this version, the neuron is represented as a single compartment, ignoring any spatial propagation in electrical activity. The neural membrane is described by a capacitance C_m in parallel with three conductance branches representing transmembrane ionic currents. Denoting the transmembrane potential as V , and noting that current flowing through C_m returns through the conductance branches, we can write:

$$C_m \frac{dV}{dt} = - (i_{Na} + i_K + i_L)$$

where i_{Na} , i_K and i_L denote Na⁺, K⁺, and non-specific leakage currents through the conductance pathways. These currents in turn are given by

$$\begin{aligned} i_{Na} &= g_{Na} (V - V_{Na}) \\ i_K &= g_K (V - V_K) \\ i_L &= g_L (V - V_L) \end{aligned}$$

where V_{Na} , V_K and V_L correspond to the *reversal potential* for each of the i_{Na} , i_K and i_L membrane currents respectively. These potentials correspond to the membrane voltage which exactly balances ionic diffusion in each channel with ion flow due to the electric field. The membrane conductances g_{Na} and g_K follow voltage-dependent

²http://www.nobelprize.org/nobel_prizes/medicine/laureates/1963/.

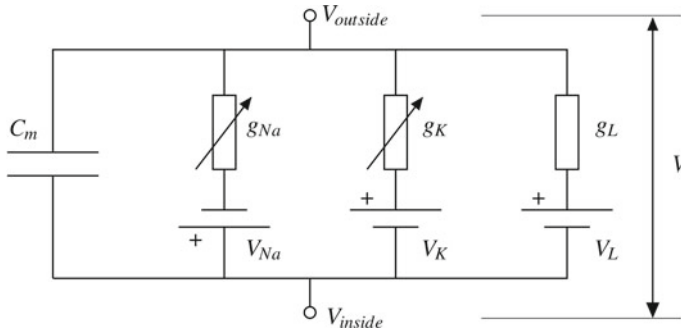


Fig. 2.5 Hodgkin–Huxley equivalent-circuit model of neural electrical activity. The neural cell membrane is represented as a capacitance C_m in parallel with conductance pathways representing transmembrane channels for Na^+ ions (g_{Na}), K^+ ions (g_K) and a non-specific leakage (g_L). Each conductance is in series with a voltage source, representing the reversal potential for that channel. g_{Na} and g_K are variable, obeying voltage-dependent kinetics. V_{inside} and $V_{outside}$ denote the voltages inside and outside the neuron, with their difference equal to the transmembrane potential V

kinetics, so that the complete ODE system is given by:

$$\begin{aligned} \frac{dV}{dt} &= -\frac{1}{C_m} [\bar{g}_{Na} m^3 h (V - V_{Na}) + \bar{g}_K n^4 (V - V_K) + \bar{g}_L (V - V_L) - i_{stim}] \\ \frac{dn}{dt} &= \alpha_n (1 - n) - \beta_n n \\ \frac{dm}{dt} &= \alpha_m (1 - m) - \beta_m m \\ \frac{dh}{dt} &= \alpha_h (1 - h) - \beta_h h \end{aligned}$$

where \bar{g}_{Na} , \bar{g}_K , \bar{g}_L are the maximum membrane conductances of each channel, n , m , h are ‘gating’ variables governed by first-order kinetics, and i_{stim} is an applied intracellular stimulus current. Using a square-wave profile, this stimulus current is given by

$$i_{stim} = \begin{cases} I_s & t_{on} \leq t < t_{on} + t_{dur} \\ 0 & \text{otherwise} \end{cases}$$

where I_s , t_{on} and t_{dur} represent the stimulus current amplitude, onset time and duration respectively. The n , m , h gating variables lie between 0 and 1 and have voltage-dependent forward (α) and reverse (β) rates (in s^{-1}) according to:

$$\alpha_n = \frac{10(V+50)}{1-\exp\left[\frac{-(V+50)}{10}\right]} \quad \beta_n = 125 \exp\left[\frac{-(V+60)}{80}\right]$$

$$\alpha_m = \frac{100(V+35)}{1-\exp\left[\frac{-(V+35)}{10}\right]} \quad \beta_m = 4000 \exp\left[\frac{-(V+60)}{18}\right]$$

$$\alpha_h = 70 \exp\left[\frac{-(V+60)}{20}\right] \quad \beta_h = \frac{1000}{1 + \exp\left[\frac{-(V+30)}{10}\right]}$$

where V in units of mV. All model parameter values are given in Table 2.2.³

To solve this model in Matlab, a user-defined derivative function, `HH_prime.m` can be written as follows:

```
function y_out = HH_prime(t,y)
% returns state-variable derivatives for HH neuron model

% initialise parameters and state-variables
y_out = zeros(4,1);
Cm = 1;
g_Na = 120000;
g_K = 36000;
g_L = 300;
V_Na = 55;
V_K = -72;
V_L = -49.387;
I_s = 60000;
t_on = 0.001;
t_dur = 0.001;
V = y(1);
n = y(2);
m = y(3);
h = y(4);

% calculate rates
alpha_n = 10*(V+50)/(1-exp(-(V+50)/10));
beta_n = 125*exp(-(V+60)/80);
alpha_m = 100*(V+35)/(1-exp(-(V+35)/10));
beta_m = 4000*exp(-(V+60)/18);
alpha_h = 70*exp(-(V+60)/20);
beta_h = 1000/(1+exp(-(V+30)/10));
```

³These parameters were modified from the original Hodgkin–Huxley formulation to yield a resting potential of -60 mV and outward currents positive in accordance with modern electrophysiological convention.

```

% determine membrane and stimulus currents
i_Na = g_Na*m^3*h*(V-V_Na);
i_K = g_K*n^4*(V-V_K);
i_L = g_L*(V-V_L);
if (t >= t_on)&&(t<t_on+t_dur)
    i_stim = I_s;
else
    i_stim = 0;
end;

% calculate derivatives
y_out(1) = -(i_Na+i_K+i_L-i_stim)/Cm;
y_out(2) = alpha_n*(1-n)-beta_n*n;
y_out(3) = alpha_m*(1-m)-beta_m*m;
y_out(4) = alpha_h*(1-h)-beta_h*h;

```

This function is then called upon in the following script, which produces the membrane potential plot shown in Fig. 2.6:

```

Y_init = [-60, 0.3177, 0.0529, 0.5961];
[time,Y] = ode15s('HH_prime', [0 0.02], Y_init);
plot(time,Y(:,1),'k-'), xlabel('time(s)'), ylabel('V (mV)');

```

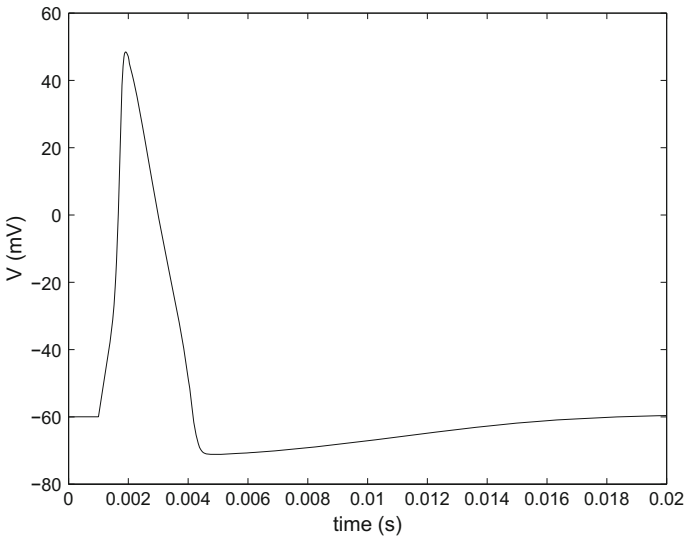


Fig. 2.6 Hodgkin–Huxley neuron model response to a brief stimulus. Shown is the membrane potential V against time

Table 2.2 Parameter values used for Hodgkin–Huxley neuron model

| Parameter | Value | Parameter | Value |
|----------------|-------------------------------|-----------|------------------------------|
| C_m | $1 \mu\text{Fcm}^{-2}$ | V_K | -72 mV |
| \bar{g}_{Na} | $120,000 \mu\text{S cm}^{-2}$ | V_L | -49.387 mV |
| \bar{g}_K | $36,000 \mu\text{S cm}^{-2}$ | I_s | $60,000 \mu\text{A cm}^{-2}$ |
| g_L | $300 \mu\text{S cm}^{-2}$ | t_{on} | 1 ms |
| V_{Na} | 55 mV | t_{dur} | 1 ms |

2.4 Further Reading

Further interesting examples of ODE models in physiological systems and bioengineering can be found in the texts of King and Mody [7], Ottesen et al. [9] and Izhikevitch [6]. A good general text on ODE systems is that of Rabenstein [10].

Problems

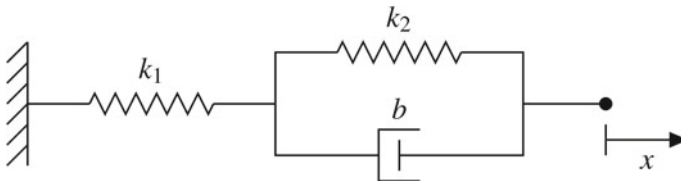
2.1 In the Hodgkin–Huxley formulation of neural activation, three gating variables n, m, h are employed, satisfying the ODE:

$$\frac{dx}{dt} = \alpha_x(V)(1 - x) - \beta_x(V)x$$

where $x \equiv n, m, h$ and $\alpha_x(V)$ and $\beta_x(V)$ are known functions of membrane voltage V . Assuming a voltage-clamp experiment is performed, whereby the membrane voltage is stepped suddenly from a value V_{hold} to a new value V_{clamp} and held at this value via a feedback mechanism. α_x and β_x are now constant.

- (a) Solve this equation analytically for x , with initial value $x(0) = x_0$, stating the homogeneous, particular and general solutions.
- (b) What is the steady-state value of x ? Hence, what is a reasonable estimate for x_0 ?

2.2 The passive mechanical behaviour of skeletal muscle can be modelled using a simplified lumped parameter representation consisting of a linear spring k_1 in series with a parallel linear spring-dashpot combination k_2, b , as shown below:



One end of the muscle is fixed, and the displacement of the other end is x . If x_1 denotes the change in length from rest in spring k_1 , then the forces in each element are given by:

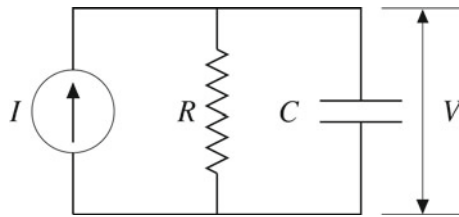
$$F_1 = k_1x_1 \quad F_2 = k_2x_2 \quad F_b = b \frac{dx_2}{dt}$$

where F_1 , F_2 and F_b refer to elements k_1 , k_2 and b respectively, and $x_2 = x - x_1$.

(a) If the length x of the muscle is suddenly stepped and held from 0 to X_m at $t = 0$, solve the system analytically for the applied force, F .

(b) If the applied force on the muscle F is suddenly stepped and held from 0 to F_m at $t = 0$, find the analytical solution for the change in length, x .

2.3 A simple model of neuronal excitation represents the cell membrane as a resistance R in parallel with a capacitance C . An applied stimulus current I depolarises the membrane to a potential of V , as shown below. If V exceeds a pre-defined threshold V_{th} , the neuron will fire.



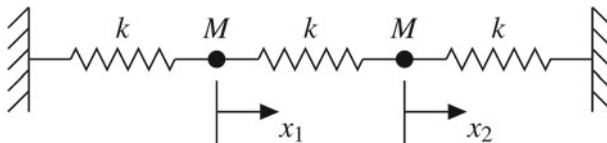
The currents i_R and i_C flowing through the R and C branches are given by

$$i_R = \frac{V}{R} \quad i_C = C \frac{dV}{dt}$$

(a) Assuming the neuron is initially at rest with $V = 0$, and a constant stimulus current I is applied at $t = 0$, find the time taken T to depolarize the membrane to V_{th} . Determine the corresponding stimulus strength-duration characteristic for neuronal activation, i.e. I as a function of T .

(b) Defining the *rheobase* as the minimum current necessary to activate the neuron, and the *chronaxie* as the required stimulus duration for an applied current of twice the rheobase, determine both quantities from the above strength-duration characteristic.

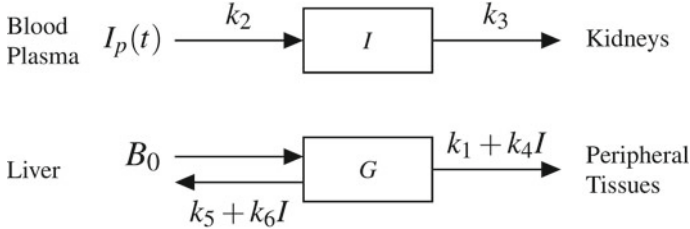
2.4 Consider the system below of two coupled masses M , connected to each other and to fixed supports via three linear springs with spring constants k :



- (a) Determine the pair of ODEs describing the motion of this system.
- (b) Solve this system analytically for the displacements x_1 and x_2 , assuming the masses are initially at rest and displaced by amounts u_1 and u_2 .

Hint: Use the variable substitutions $y_1 = x_1 + x_2, y_2 = x_1 - x_2$.

2.5 A simplified two-compartment model of glucose-insulin kinetics in a human subject proposed by Berman et al. [1]⁴ is shown below. $I_p(t)$ represents insulin injected intravenously into the blood, I is the concentration of insulin in a remote body compartment, and G is the glucose concentration in the blood plasma.



I_p, I and G are all in units of mM, with model parameters given below:

| Parameter | Value | Parameter | Value |
|-----------|---|-----------|---|
| k_1 | 0.015 min^{-1} | k_5 | 0.035 min^{-1} |
| k_2 | 1 min^{-1} | k_6 | $0.02 \text{ mM}^{-1} \text{ min}^{-1}$ |
| k_3 | 0.09 min^{-1} | B_0 | 0.5 mM min^{-1} |
| k_4 | $0.01 \text{ mM}^{-1} \text{ min}^{-1}$ | | |

An intravenous dose of insulin is administered as a square-pulse waveform according to:

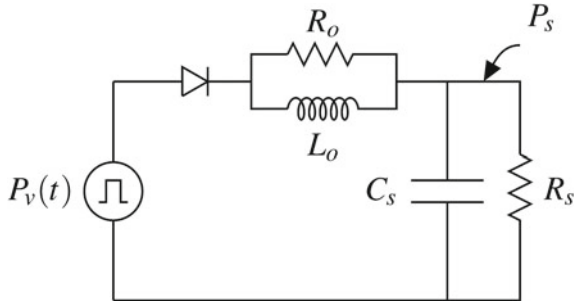
$$I_p(t) = \begin{cases} 200 \text{ mM} & 0 \leq t < 0.1 \text{ min} \\ 0 & t \geq 0.1 \text{ min} \end{cases}$$

The initial values of I and G at $t = 0$ are 0 and 10 mM respectively. Solve for I and G using Matlab over the time interval $0 \leq t \leq 60 \text{ min}$.

2.6 A simplified lumped parameter electric-analogue model of the heart and systemic circulation⁵ is shown below:

⁴Model VI in their paper.

⁵This is an example of a four-element *windkessel* model, translated from German as “air-chamber”. Early German fire engines incorporated an air-filled elastic reservoir between the water pump and outflow hose to dampen any intermittent interruptions to hand-pump water supply. Such damping can be modelled by an electric circuit comprised of resistive, capacitive and inductive elements.



where P_s is the systemic pressure and L_o represents the blood inductance within the aortic root such that the pressure drop across this element is given $L_o \frac{dQ_L}{dt}$, where Q_L is the flow (in $\text{cm}^3 \text{s}^{-1}$) through it. $P_v(t)$ represents the developed ventricular pressure as a function of time, given by the following simplified periodic square-wave profile:

$$P_v(t + T) = P_v(t)$$

$$P_v(t) = \begin{cases} P & 0 \leq t < t_c \\ 0 & t_c \leq t \leq T \end{cases}$$

All other elements are similar to those given earlier in the example of Sect. 2.3.1. Remaining model parameters and descriptions are given below:

| Parameter | Description | Value |
|-----------|---------------------------------------|--|
| R_o | Aortic root resistance | $0.06 \text{ mmHg s cm}^{-3}$ |
| L_o | Aortic root blood inductance | $0.2 \text{ mmHg s}^2 \text{ cm}^{-3}$ |
| C_s | Systemic compliance | $1 \text{ cm}^3 \text{ mmHg}^{-1}$ |
| R_s | Systemic resistance | 1.4 mmHg s^{-3} |
| P | Peak ventricular pressure | 120 mmHg |
| T | Heart period | 1 s |
| t_c | Active contraction interval (systole) | 0.35 s |

- (a) Determine the ODEs describing this system.
- (b) Using an appropriate choice of initial values, solve this system using Matlab for the steady-state oscillations in systemic pressure P_s .

2.7 The following set of ODEs modified from McSharry et al. [8] reproduce a synthetic electrocardiogram (ECG) waveform in variable z :

$$\frac{dx}{dt} = \alpha x - \omega y$$

$$\frac{dy}{dt} = \alpha y + \omega x$$

$$\frac{dz}{dt} = - \sum_{i=1}^5 a_i (\theta - \theta_i) \exp\left(-\frac{(\theta - \theta_i)^2}{2b_i^2}\right) - (z - z_0)$$

where $\alpha = 1 - \sqrt{x^2 + y^2}$, $\omega = 2\pi \text{ rad s}^{-1}$, $z_0 = 0$ and $\theta = \text{atan2}(y, x)$, where atan2 represents the four-quadrant inverse tangent implemented by the Matlab function atan2 . Remaining model parameters are given below:

| Index i | 1 | 2 | 3 | 4 | 5 |
|---|-------------------|--------------------|-----|-------------------|------------------|
| θ_i (rad) | $-\frac{1}{3}\pi$ | $-\frac{1}{12}\pi$ | 0 | $\frac{1}{12}\pi$ | $\frac{1}{2}\pi$ |
| a_i ($\text{mV s}^{-1} \text{ rad}^{-1}$) | 1.2 | -5 | 30 | -7.5 | 0.75 |
| b_i (rad) | 0.25 | 0.1 | 0.1 | 0.1 | 0.4 |

Given the initial values, $x(0) = -1$, $y(0) = 0$, $z(0) = 0$, numerically solve this ECG model in Matlab from $t = 0$ to 1 s, plotting z against t , where z , t are in units of mV and s respectively.

2.8 The Frankenhaeuser-Huxley neural action potential model [3] consists of the following ODE system:

$$\begin{aligned}\frac{dV}{dt} &= -\frac{1}{C_m} [i_{Na} + i_K + i_P + i_L - i_{stim}] \\ \frac{dm}{dt} &= \alpha_m (1 - m) - \beta_m m \\ \frac{dh}{dt} &= \alpha_h (1 - h) - \beta_h h \\ \frac{dn}{dt} &= \alpha_n (1 - n) - \beta_n n \\ \frac{dp}{dt} &= \alpha_p (1 - p) - \beta_p p\end{aligned}$$

with membrane ionic currents given by

$$\begin{aligned}i_{Na} &= m^2 h \bar{P}_{Na} \left(\frac{EF^2}{RT} \right) \left[\frac{[\text{Na}]_o - [\text{Na}]_i \exp\left(\frac{EF}{RT}\right)}{1 - \exp\left(\frac{EF}{RT}\right)} \right] \\ i_K &= n^2 \bar{P}_K \left(\frac{EF^2}{RT} \right) \left[\frac{[\text{K}]_o - [\text{K}]_i \exp\left(\frac{EF}{RT}\right)}{1 - \exp\left(\frac{EF}{RT}\right)} \right] \\ i_P &= p^2 \bar{P}_P \left(\frac{EF^2}{RT} \right) \left[\frac{[\text{Na}]_o - [\text{Na}]_i \exp\left(\frac{EF}{RT}\right)}{1 - \exp\left(\frac{EF}{RT}\right)} \right] \\ i_L &= g_L (V - V_L)\end{aligned}$$

where E is the transmembrane potential, V is the membrane potential displacement from its resting value E_r ($V = E - E_r$), F is Faraday's constant, R is the gas constant, and T is the absolute temperature. $[\text{Na}]_o$, $[\text{Na}]_i$, $[\text{K}]_o$ and $[\text{K}]_i$ represent outside (extracellular) and intracellular Na^+ and K^+ concentrations, whilst i_{Na} , i_K ,

i_p and i_L represent the membrane Na^+ , K^+ , non-specific (mainly Na^+), and leakage currents respectively. The membrane permeabilities of i_{Na} , i_K and i_p are \bar{P}_{Na} , \bar{P}_K and \bar{P}_p respectively, with the kinetics of these currents determined from the m , h , n and p gating variables. i_{stim} is an applied intracellular stimulus current given by

$$i_{stim} = \begin{cases} I_s & t_{on} \leq t < t_{on} + t_{dur} \\ 0 & \text{otherwise} \end{cases}$$

where I_s , t_{on} and t_{dur} represent the stimulus current amplitude, onset time and duration respectively. The voltage-dependent forward (α) and reverse (β) rates (in ms^{-1}) are given by:

$$\begin{aligned} \alpha_m &= \frac{0.36(V-22)}{1-\exp\left[\frac{(22-V)}{3}\right]} & \beta_m &= \frac{0.4(13-V)}{1-\exp\left[\frac{(V-13)}{20}\right]} \\ \alpha_h &= \frac{0.1(-10-V)}{1-\exp\left[\frac{(V+10)}{6}\right]} & \beta_h &= \frac{4.5}{1+\exp\left[\frac{(45-V)}{10}\right]} \\ \alpha_n &= \frac{0.02(V-35)}{1-\exp\left[\frac{(35-V)}{10}\right]} & \beta_n &= \frac{0.05(10-V)}{1-\exp\left[\frac{(V-10)}{10}\right]} \\ \alpha_p &= \frac{0.006(V-40)}{1-\exp\left[\frac{(40-V)}{10}\right]} & \beta_p &= \frac{0.09(-25-V)}{1-\exp\left[\frac{(V+25)}{20}\right]} \end{aligned}$$

where V is in units of mV. Initial values are $V = 0$ mV, $m = 0.0005$, $h = 0.8249$, $n = 0.0268$ and $p = 0.0049$.

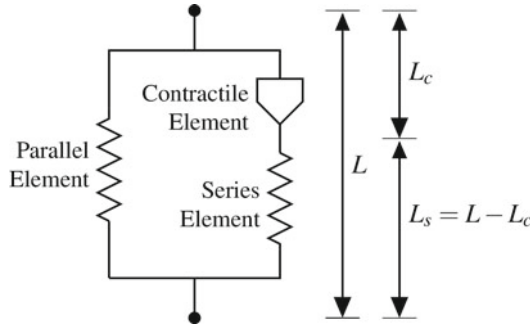
The drug tetrodotoxin (TTX) is known to selectively block the membrane i_{Na} current. Assuming that at one given dosage, TTX reduces parameter \bar{P}_{Na} to 20% of its original value. Solve this model using Matlab over the time interval $t = 0$ to 5 ms, plotting the transmembrane potentials E (in mV) on the same graph for the following two cases:

- (1) control (i.e. no TTX) and
- (2) TTX applied.

All model parameters are given in the following table:

2.9 A simple three-element model of active cardiac muscle contraction consists of a passive non-linear spring in parallel with a contractile and passive series element, as shown in the diagram. The model structure and equations have been modified from Fung [4].

| Parameter | Value | Parameter | Value |
|-----------------|-----------------------------|----------------|--|
| C_m | $2 \mu\text{F cm}^{-2}$ | $[\text{K}]_o$ | 2.5 mM |
| \bar{P}_{Na} | 0.008 cm s^{-1} | $[\text{K}]_i$ | 120 mM |
| \bar{P}_K | 0.0012 cm s^{-1} | I_s | 1 mA cm^{-2} |
| \bar{P}_P | $0.00054 \text{ cm s}^{-1}$ | t_{on} | 1 ms |
| gL | 30.3 ms cm^{-2} | t_{dur} | 0.12 ms |
| V_L | 0.026 mV | F | $96.49 \text{ C mmol}^{-1}$ |
| $[\text{Na}]_o$ | 114.5 mM | R | $8.31 \text{ J mol}^{-1} \text{ K}^{-1}$ |
| $[\text{Na}]_i$ | 13.74 mM | T | 310 K |
| E_r | -70 mV | | |



The total tension T in the muscle is given by the sum of tensions in the parallel and series elements:

$$T = T_p + T_s$$

where T_p and T_s , the tensions in the parallel and series elements respectively, are given by:

$$T_p = \beta (e^{\alpha(L-L_0)} - 1), \quad T_s = \beta (e^{\alpha L_s} - 1)$$

where α , β are parameters and L_0 denotes the resting length of the muscle. For the contractile element, the velocity of its shortening is described by:

$$\frac{dL_c}{dt} = \frac{a [T_s - S_0 f(t)]}{T_s + \gamma S_0}$$

where a , γ , S_0 are muscle parameters and $f(t)$ is the muscle activation function given by

$$f(t) = \begin{cases} \sin\left(\frac{\pi}{2} \left[\frac{t+t_0}{t_{ip}+t_0}\right]\right) & 0 \leq t < 2t_{ip} + t_0 \\ 0 & t \geq 2t_{ip} + t_0 \end{cases}$$

with t_0 and t_{ip} constants defining activation phase offset and the time to peak isometric contraction respectively.

Model parameters for a cardiac papillary muscle specimen are given in the table below:

| Parameter | Value | Parameter | Value |
|-----------|------------------------|-----------|--------|
| L_0 | 10 mm | S_0 | 4 mN |
| α | 15 mm | γ | 0.45 |
| β | 5 mN | t_0 | 0.05 s |
| a | 0.66 mm^{-1} | t_{ip} | 0.2 s |

Note that in the relaxed state, the length of the series element L_s is 0.

- (a) Using Matlab, solve for and plot the total tension T against time during an *isometric* contraction in which the muscle is clamped at its resting length L_0 .
- (b) Solve for and plot muscle length L against time during an *isotonic* contraction in which the muscle is allowed to freely contract with no imposed load (i.e. $T = 0$).

References

1. Bergman RN, Ider YZ, Bowden CR, Cobelli C (1979) Quantitative estimation of insulin sensitivity. *Am J Physiol* 236:E667–E677
2. FitzHugh R (1961) Impulses and physiological states in theoretical models of nerve membrane. *Biophys J* 1:445–466
3. Frankenhaeuser B, Huxley AF (1964) The action potential in the myelinated nerve fibre of *Xenopus laevis* as computed on the basis of voltage clamp data. *J Physiol* 171:302–315
4. Fung YC (1970) Mathematical representation of the mechanical properties of the heart muscle. *J Biomech* 3:381–404
5. Hodgkin AL, Huxley AF (1952) A quantitative description of membrane current and its application to conduction and excitation in nerve. *J Physiol (Lond)* 117:500–544
6. Izhikevich EM (2007) *Dynamical systems in neuroscience: the geometry of excitability and bursting*. MIT Press, Cambridge
7. King MR, Mody NA (2011) *Numerical and statistical methods for bioengineering: applications in Matlab*. Cambridge University Press, Cambridge
8. McSharry PE, Clifford GD, Tarassenko L, Smith LA (2003) A dynamical model for generating synthetic electrocardiogram signals. *IEEE Trans Biomed Eng* 50:289–294
9. Ottesen JT, Olufsen MS, Larsen JK (2004) *Applied mathematical models in human physiology*. SIAM, Philadelphia
10. Rabenstein AL (1972) *Introduction to ordinary differential equations*. Academic Press, New York
11. van der Pol B, van der Mark J (1928) The heartbeat considered as a relaxation oscillation, and an electrical model of the heart. *Philos Mag Ser 7(6)*:763–775

Chapter 3

Numerical Integration of Ordinary Differential Equations

Almost all lumped-parameter ODE models encountered in physics, biology or medicine cannot be solved in closed-loop form, but require numerical integration to find their solution. As we have seen from the previous chapter, Matlab provides an extensive suite of ODE solvers including `ode15s` and `ode45`, and COMSOL also employs its own ODE solvers based on the backward differential formula (BDF) and generalized- α methods. This chapter provides an overview of numerical methods commonly used for integrating ODE systems, including those algorithms used by Matlab and COMSOL.

3.1 Taylor's Theorem

The basis of all numerical integration algorithms is *Taylor's theorem*, which can be stated for a function of a single variable as:

Theorem 3.1 *Let f be a real-valued function of a single variable with $n + 1$ continuous derivatives in the closed interval $[t, t + h]$. Then there exists some $\xi \in [t, t + h]$ such that*

$$f(t + h) = f(t) + hf'(t) + \frac{h^2}{2!} f''(t) + \dots + \frac{h^n}{n!} f^{(n)}(t) + \frac{h^{n+1}}{(n + 1)!} f^{(n+1)}(\xi) \quad (3.1)$$

Proof We begin with the fundamental theorem of calculus:

$$\int_a^b f'(\tau) \, d\tau = f(b) - f(a)$$

Using $a = t$, $b = t + h$ and re-arranging:

$$f(t + h) = f(t) + \overbrace{\int_t^{t+h} f'(\tau) \, d\tau}^{R_1}$$

The integral on the right (remainder term R_1) can be expanded using the integration by parts formula:

$$\int_a^b u \cdot v \, d\tau = \left[u \int v \, d\tau \right]_a^b - \int_a^b \left(\frac{du}{d\tau} \int v \, d\tau \right) \, d\tau \quad (3.2)$$

Using $u = t + h - \tau$ and $v = f''(\tau)$, we have:

$$\begin{aligned} \int_t^{t+h} (t + h - \tau) f''(\tau) \, d\tau &= \left[(t + h - \tau) f'(\tau) \right]_t^{t+h} - \int_t^{t+h} -f'(\tau) \, d\tau \\ &= -hf'(t) + \int_t^{t+h} f'(\tau) \, d\tau \end{aligned}$$

Hence,

$$R_1 = \int_t^{t+h} f'(\tau) \, d\tau = hf'(t) + \overbrace{\int_t^{t+h} (t + h - \tau) f''(\tau) \, d\tau}^{R_2}$$

We can continue expanding R_2 using integration by parts (Eq. 3.2), now with $u = \frac{(t+h-\tau)^2}{2!}$ and $v = f'''(\tau)$:

$$\begin{aligned} \int_t^{t+h} \frac{(t + h - \tau)^2}{2!} f'''(\tau) \, d\tau &= \left[\frac{(t + h - \tau)^2}{2!} f''(\tau) \right]_t^{t+h} - \int_t^{t+h} -(t + h - \tau) f''(\tau) \, d\tau \\ &= -\frac{h^2}{2!} f''(t) + \int_t^{t+h} (t + h - \tau) f''(\tau) \, d\tau \end{aligned}$$

Hence,

$$R_2 = \int_t^{t+h} (t + h - \tau) f''(\tau) \, d\tau = \frac{h^2}{2!} f''(t) + \overbrace{\int_t^{t+h} \frac{(t + h - \tau)^2}{2!} f'''(\tau) \, d\tau}^{R_3}$$

We can continue indefinitely expand the remainder terms integration by parts to obtain

$$f(t+h) = f(t) + \sum_{i=1}^n \frac{h^i}{i!} f^{(i)}(t) + \overbrace{\int_t^{t+h} \frac{(t+h-\tau)^n}{n!} f^{(n+1)}(\tau) d\tau}^{R_{n+1}} \quad (3.3)$$

To verify that Eq. 3.3 is true, we make use of the principle of mathematical induction. Assume it is true for some value $n = k$. That is,

$$f(t+h) = f(t) + \sum_{i=1}^k \frac{h^i}{i!} f^{(i)}(t) + R_{k+1}$$

with $R_{k+1} = \int_t^{t+h} \frac{(t+h-\tau)^k}{k!} f^{(k+1)}(\tau) d\tau$

We then show that Eq. 3.3 also holds for $k+1$. This is equivalent to showing that $R_{k+1} = \frac{h^{k+1}}{(k+1)!} f^{(k+1)}(t) + R_{k+2}$. To do this, we use integration by parts (Eq. 3.2) with $u = \frac{(t+h-\tau)^{k+1}}{(k+1)!}$ and $v = f^{(k+1)}(\tau)$, to obtain:

$$\begin{aligned} \overbrace{\int_t^{t+h} \frac{(t+h-\tau)^{k+1}}{(k+1)!} f^{(k+1)}(\tau) d\tau}^{R_{k+2}} &= \left[\frac{(t+h-\tau)^{k+1}}{(k+1)!} f^{(k+1)}(\tau) \right]_t^{t+h} \\ &\quad - \int_t^{t+h} -\frac{(t+h-\tau)^k}{k!} f^{(k+1)}(\tau) d\tau \\ &= -\frac{h^{k+1}}{(k+1)!} f^{(k+1)}(t) \\ &\quad + \overbrace{\int_t^{t+h} \frac{(t+h-\tau)^k}{k!} f^{(k+1)}(\tau) d\tau}^{R_{k+1}} \end{aligned}$$

Hence, $R_{k+1} = \frac{h^{k+1}}{(k+1)!} f^{(k+1)}(t) + R_{k+2}$ and therefore Eq. 3.3 is true for $n = k+1$. Since we have already shown it is true for $n = 0$ (from the fundamental theorem of calculus), it must be true for all n .

The final part of our proof is to show that the remainder term R_{n+1} in Eq. 3.3 is equivalent to the final term in Eq. 3.1. Denote the maximum value of $f^{(n+1)}(\tau)$ ($\tau \in [t, t+h]$) by M . Then,

$$\begin{aligned}
R_{n+1} &= \int_t^{t+h} \frac{(t+h-\tau)^n}{n!} f^{(n+1)}(\tau) \, d\tau \\
&\leq \int_t^{t+h} \frac{(t+h-\tau)^n}{n!} M \, d\tau \\
&= \left[-\frac{(t+h-\tau)^{n+1}}{(n+1)!} \right]_t^{t+h} M \\
&= \frac{h^{n+1}}{(n+1)!} M
\end{aligned}$$

represents an upper bound on R_{n+1} . Using a similar analysis, we can show that if L is the minimum value of $f^{(n+1)}(\tau)$ over the same interval, then $\frac{h^{n+1}}{(n+1)!} L$ represents the lower bound for R_{n+1} . That is,

$$\frac{h^{n+1}}{(n+1)!} L \leq R_{n+1} \leq \frac{h^{n+1}}{(n+1)!} M$$

Since $f^{(n+1)}(\tau)$ is continuous over $[t, t+h]$, with values between L and M , then there must exist some $\xi \in [t, t+h]$ for which

$$R_{n+1} = \frac{h^{n+1}}{(n+1)!} f^{(n+1)}(\xi)$$

This completes the proof of Taylor's theorem. □

The polynomial

$$f(t) + hf'(t) + \frac{h^2}{2!} f''(t) + \cdots + \frac{h^n}{n!} f^{(n)}(t)$$

is known as the *Taylor polynomial* of $f(\tau)$ centred at t , and the remainder term

$$\frac{h^{n+1}}{(n+1)!} f^{(n+1)}(\xi)$$

is known as the *truncation error*. Since this error consists of a quantity proportional to h^{n+1} , we say that the error is of *order* h^{n+1} , written using the notation $\mathcal{O}(h^{n+1})$.

For a real-valued function of many variables, the multivariate form of Taylor's theorem is:

Theorem 3.2 *Let f be a real-valued function of m variables with continuous partial derivatives up to order $n + 1$. Then there exists some $\xi \in [0, 1]$ such that*

$$\begin{aligned}
 f(\mathbf{x} + \mathbf{h}) &= f(\mathbf{x}) + \sum_{i_1=1}^m h_{i_1} \frac{\partial f}{\partial x_{i_1}}(\mathbf{x}) + \frac{1}{2!} \sum_{i_1, i_2=1}^m h_{i_1} h_{i_2} \frac{\partial^2 f}{\partial x_{i_1} \partial x_{i_2}} + \cdots \\
 &\quad + \frac{1}{n!} \sum_{i_1, i_2, \dots, i_n=1}^m h_{i_1} h_{i_2} \cdots h_{i_n} \frac{\partial^n f}{\partial x_{i_1} \partial x_{i_2} \cdots \partial x_{i_n}}(\mathbf{x}) + \\
 &\quad + \frac{1}{(n+1)!} \sum_{i_1, i_2, \dots, i_{n+1}=1}^m h_{i_1} h_{i_2} \cdots h_{i_{n+1}} \frac{\partial^{n+1} f}{\partial x_{i_1} \partial x_{i_2} \cdots \partial x_{i_{n+1}}}(\mathbf{x} + \xi \mathbf{h})
 \end{aligned} \tag{3.4}$$

with $\mathbf{x}, \mathbf{h} \in \mathbb{R}^m$

This multivariate form of Taylor's theorem can be shown to follow directly from the single-variable version [15].

For two variables, the second-order form of Eq. 3.4 may be written as:

$$\begin{aligned}
 f(x_1 + h_1, x_2 + h_2) &= f(x_1, x_2) + h_1 \frac{\partial f}{\partial x_1}(x_1, x_2) + h_2 \frac{\partial f}{\partial x_2}(x_1, x_2) + \\
 &\quad \frac{1}{2} \left[h_1^2 \frac{\partial^2 f}{\partial x_1^2}(x_1, x_2) + 2h_1 h_2 \frac{\partial^2 f}{\partial x_1 \partial x_2}(x_1, x_2) + h_2^2 \frac{\partial^2 f}{\partial x_2^2}(x_1, x_2) \right] + \\
 &\quad + \frac{1}{6} \left[h_1^3 \frac{\partial^3 f}{\partial x_1^3}(x_1 + \xi h_1, x_2 + \xi h_2) + \right. \\
 &\quad 3h_1^2 h_2 \frac{\partial^3 f}{\partial x_1^2 \partial x_2}(x_1 + \xi h_1, x_2 + \xi h_2) + \\
 &\quad 3h_1 h_2^2 \frac{\partial^3 f}{\partial x_1 \partial x_2^2}(x_1 + \xi h_1, x_2 + \xi h_2) + \\
 &\quad \left. h_2^3 \frac{\partial^3 f}{\partial x_2^3}(x_1 + \xi h_1, x_2 + \xi h_2) \right], \quad \xi \in [0, 1]
 \end{aligned} \tag{3.5}$$

3.2 One-Step Methods

One-step ODE numerical integration methods use information at the current integration step to determine the dependent variable values at the next step.

Forward-Euler Method

The simplest numerical ODE integration algorithm is the forward-Euler¹ method, which utilises the Taylor series expansion with $n = 1$. For the ODE

$$\frac{dy}{dt} = f(t, y), \quad y(0) = y_0$$

we can use the Taylor series expansion to form

$$\begin{aligned} y(t+h) &= y(t) + hy'(t) + \frac{h^2}{2}y''(\xi) \\ &= y(t) + hf(t, y) + \mathcal{O}(h^2) \end{aligned}$$

which yields the forward-Euler algorithm for stepsize h :

$$y(t+h) = y(t) + hf(t, y)$$

The *local truncation error* for this method is $\mathcal{O}(h^2)$, which represents the error at each step. The *global truncation error* is the accumulated error after many steps, and is one-order less than the local truncation error, or $\mathcal{O}(h)$.

Example 3.1 Solve the following ‘test’ ODE using the forward-Euler method:

$$\frac{dy}{dt} = \lambda y, \quad y(0) = y_0$$

which has the exact solution $y_0 e^{\lambda t}$.

Answer: Assuming a fixed step-size of h , we can use the forward-Euler method to form the following sequence of approximations to y at each time step:

$$\begin{aligned} y(t+h) &= y(t) + hy'(t) \\ &= y(t) + h\lambda y(t) \\ &= y(t)[1 + h\lambda] \end{aligned}$$

¹Pronounced forward-*Oiler*.

The following Matlab function (`forward_Euler_test.m`) implements this test problem up to $t = 3$ using $\lambda = -2$ and $y_0 = 1$:

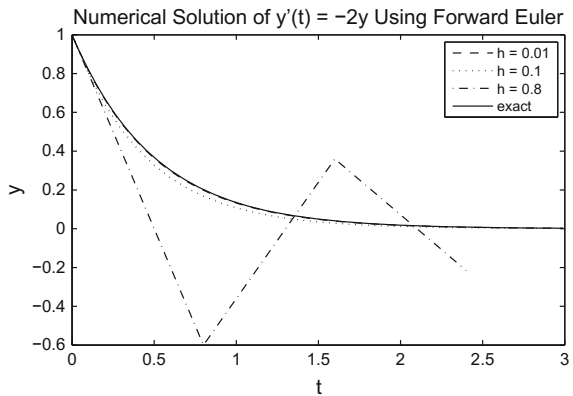
```
function [time,y_out] = forward_Euler_test(h)
lambda = -2;
y0 = 1;           % initial value
tf = 3;          % final time
% initialize arrays
time = zeros(1,round(tf/h));
y_out = zeros(1,round(tf/h));
time(1) = 0;
y_out(1) = y0;
% implement forward-Euler iterations
for i = 1:round(tf/h)-1
time(i+1) = time(i)+h;
y_out(i+1) = y_out(i)*(1+lambda*h);
end;
```

with the following script used to plot numerical solutions for three step-sizes of $h = 0.8, h = 0.1,$ and $h = 0.01,$ as shown in Fig. 3.1.

```
[time_1, y_1] = forward_Euler_test(0.01);
[time_2, y_2] = forward_Euler_test(0.1);
[time_3, y_3] = forward_Euler_test(0.8);
y_exact = exp(-2*time_1);
plot(time_1, y_1, 'k--', time_2, y_2, 'k:', ...
      time_3, y_3, 'k-.', time_1, y_exact, 'k-'), ...
legend('h = 0.01', 'h = 0.1', 'h = 0.8', 'exact'), ...
title('Numerical Solution of y''(t) = -2y Using Forward Euler'), ...
xlabel('t'), ylabel('y');
```

For this test problem, the forward-Euler solution at each time-step multiplies the previous step solution by $(1 + h\lambda)$. When $\lambda < 0$, the exact solution will decay towards 0. For numerical stability, the forward-Euler method must therefore yield

Fig. 3.1 Forward-Euler solutions to $y'(t) = -2y$, $y(0) = 1$, for three step-sizes (h) of 0.01, 0.1, and 0.8. Also shown is the exact solution $y = e^{-2t}$



solutions at each step whose magnitudes are decreasing. This leads to the following absolute stability restriction on the step-size:

$$|1 + h\lambda| \leq 1$$

$$\text{or } h \leq \frac{2}{-\lambda} \quad (\text{for } \lambda < 0)$$

□

Ideally, the step-size of the forward-Euler method should be governed by the local truncation error, given by the Taylor series remainder term $\frac{h^2}{2} f''(\xi)$. When the solution is slowly-varying, $f''(\xi)$ (and therefore the local error), will be less. This indicates that larger step-sizes can be chosen to maintain a given accuracy. However, the absolute stability requirement of the method still holds, restricting the step size from increasing beyond a given limit. When the step size of the forward-Euler method is constrained by stability considerations rather than accuracy, the ODE is said to be *stiff* [2].

For a system of ODEs written in the general form

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(t, \mathbf{y}), \quad \mathbf{y}(0) = \mathbf{y}_0 \quad (3.6)$$

the forward-Euler method becomes:

$$\mathbf{y}(t + h) = \mathbf{y}(t) + h\mathbf{f}(t, \mathbf{y}(t))$$

or more simply

$$\mathbf{y}_n = \mathbf{y}_{n-1} + h\mathbf{f}(t_{n-1}, \mathbf{y}_{n-1}) \quad (3.7)$$

Since the unknown term \mathbf{y}_n appears on only one side of Eq. 3.7, and can be directly determined from $(t_{n-1}, \mathbf{y}_{n-1})$, this method is said to be *explicit*.

Linearizing Eq. 3.6 about the current time (t_c) and \mathbf{y} -point (\mathbf{y}_c), we can approximate the ODE system as:

$$\frac{d\mathbf{y}}{dt} = \mathbf{J} \cdot (\mathbf{y} - \mathbf{y}_c) + \mathbf{f}_c, \quad \mathbf{y}(t_c) = \mathbf{y}_c \quad (3.8)$$

where $\mathbf{f}_c = \mathbf{f}(t_c, \mathbf{y}_c)$, and matrix $\mathbf{J} = \frac{\partial \mathbf{f}}{\partial \mathbf{y}}$ is referred to as the *Jacobian* of \mathbf{f} . Defining an *eigenvalue* (λ) and associated *eigenvector* (\mathbf{v}) of \mathbf{J} such that

$$\mathbf{J} \cdot \mathbf{v} = \lambda \mathbf{v}, \quad (3.9)$$

there will be N such eigenvalue-eigenvector pairs of the $N \times N$ matrix \mathbf{J} , where N is the number of elements of \mathbf{y} [1]. The solution to Eq. 3.8 is given by:

$$\mathbf{y} = \mathbf{y}_c - \mathbf{J}^{-1}\mathbf{f}_c + \sum_{i=1}^N c_i \mathbf{v}_i e^{\lambda_i(t-t_c)}$$

where the c_i are constants chosen to satisfy the initial condition $\mathbf{y}(t_c) = \mathbf{y}_c$. The above solution can be verified by substituting directly into the right-hand side of Eq. 3.8 to obtain:

$$\begin{aligned} \mathbf{J} \cdot (\mathbf{y} - \mathbf{y}_c) + \mathbf{f}_c &= \mathbf{J} \cdot \left(\mathbf{y}_c - \mathbf{J}^{-1}\mathbf{f}_c + \left[\sum_{i=1}^N c_i \mathbf{v}_i e^{\lambda_i(t-t_c)} \right] - \mathbf{y}_c \right) + \mathbf{f}_c \\ &= \mathbf{J} \cdot \left(\sum_{i=1}^N c_i \mathbf{v}_i e^{\lambda_i(t-t_c)} \right) \\ &= \sum_{i=1}^N c_i \mathbf{J} \cdot \mathbf{v}_i e^{\lambda_i(t-t_c)} \\ &= \sum_{i=1}^N c_i \lambda_i \cdot \mathbf{v}_i e^{\lambda_i(t-t_c)} \quad (\text{since the } \mathbf{v}_i \text{ are eigenvectors of } J) \\ &= \frac{d\mathbf{y}}{dt} \end{aligned}$$

Hence, the solution of this system involves a sum of exponential terms, and for forward-Euler stability, the step-size must therefore be restricted to

$$h \leq \frac{2}{|\lambda|_{\max}} \quad (3.10)$$

where $|\lambda|_{\max}$ is the maximum negative eigenvalue magnitude of the Jacobian.

3.2.1 Backward-Euler Method

In order to overcome the stability issues associated with the forward-Euler method, we can centre the Taylor series around the future step at $t + h$, instead of at t , to obtain:

$$\begin{aligned} y(t) &= y(t+h) - hy'(t+h) + \frac{h^2}{2}y''(\xi) \\ &= y(t+h) - hf(t+h, y(t+h)) + \mathcal{O}(h^2) \end{aligned}$$

which yields the backward-Euler method:

$$y(t+h) = y(t) + hf(t+h, y(t+h))$$

In terms of our familiar ODE system (Eq. 3.6), this method can be written as:

$$\mathbf{y}_n = \mathbf{y}_{n-1} + h\mathbf{f}(t_n, \mathbf{y}_n) \quad (3.11)$$

Since the unknown term \mathbf{y}_n appears on both sides of Eq. 3.11, this method is referred to as *implicit*. If $\mathbf{f}(t, \mathbf{y})$ is non-linear, then Eq. 3.11 must be solved using iterative routines to converge to the solution at each time step. Thus represents an increased computational overhead, but as we shall see, the method is stable, even for large step-sizes.

Example 3.2 Solve the test ODE

$$\frac{dy}{dt} = \lambda y, \quad y(0) = y_0$$

using the backward-Euler method.

Answer: We use Eq. 3.11 to form the following approximations to y at each step:

$$\begin{aligned} y(t+h) &= y(t) + hy'(t+h) \\ &= y(t) + h\lambda y(t+h) \\ y(t+h)[1-h\lambda] &= y(t) \\ \therefore y(t+h) &= \frac{y(t)}{[1-h\lambda]} \end{aligned}$$

Hence, the solution at each step is formed by multiplying the previous step by a factor of $[1-h\lambda]^{-1}$. For positive step size h and $\lambda < 0$, this factor will be positive and less than unity. The numerical solution will therefore decay towards 0, and the backward-Euler method will be unconditionally stable for all positive step-sizes h . \square

When the ODE system is non-linear, the backward-Euler method involves solving a set of non-linear equations at each step. Rewriting Eq. 3.11 as

$$\mathbf{g}(\mathbf{y}_n) = \mathbf{y}_n - \mathbf{y}_{n-1} - h\mathbf{f}(t_n, \mathbf{y}_n) = \mathbf{0}$$

we can use *Newton's method* to define a set of iterations to converge on a solution to this system. Linearizing $\mathbf{g}(\mathbf{y}_n)$ around the current iterate \mathbf{y}_n^v , we obtain:

$$\frac{\partial \mathbf{g}}{\partial \mathbf{y}}(\mathbf{y}_n^v) [\mathbf{y}_n - \mathbf{y}_n^v] + \mathbf{g}(\mathbf{y}_n^v) = \mathbf{0}$$

Re-arranging to solve for \mathbf{y}_n , we obtain an expression for the next iterate:

$$\mathbf{y}_n^{v+1} = \mathbf{y}_n^v - \left[\frac{\partial \mathbf{g}}{\partial \mathbf{y}}(\mathbf{y}_n^v) \right]^{-1} \mathbf{g}(\mathbf{y}_n^v) \quad (3.12)$$

where $\frac{\partial \mathbf{g}}{\partial \mathbf{y}}(\mathbf{y}_n^v)$ is the Jacobian of \mathbf{g} at the current iteration. A variant of Eq. 3.12 is to take a fractional step γ towards the estimated solution using

$$\mathbf{y}_n^{v+1} = \mathbf{y}_n^v - \gamma \left[\frac{\partial \mathbf{g}}{\partial \mathbf{y}}(\mathbf{y}_n^v) \right]^{-1} \mathbf{g}(\mathbf{y}_n^v) \quad (3.13)$$

where γ ($0 \leq \gamma \leq 1$) is referred to as the *damping factor*. The damped Newton method of Eq. 3.13 often leads to improved convergence.

3.2.2 Trapezoidal Method

The forward and backward-Euler methods yield local truncation errors of $\mathcal{O}(h^2)$. Higher accuracy can be obtained by centring the Taylor series around $t + h/2$, as follows:

$$\begin{aligned} \mathbf{y}(t+h) &= \mathbf{y}(t+h/2) + \frac{h}{2}\mathbf{y}'(t+h/2) + \frac{h^2}{8}\mathbf{y}''(t+h/2) + \frac{h^3}{48}\mathbf{y}'''(\xi_1) \\ \mathbf{y}(t) &= \mathbf{y}(t+h/2) - \frac{h}{2}\mathbf{y}'(t+h/2) + \frac{h^2}{8}\mathbf{y}''(t+h/2) - \frac{h^3}{48}\mathbf{y}'''(\xi_2) \end{aligned}$$

with $\xi_1 \in [t+h/2, t+h]$ and $\xi_2 \in [t, t+h/2]$. Subtracting these expressions, we obtain:

$$\mathbf{y}(t+h) - \mathbf{y}(t) = h\mathbf{y}'(t+h/2) + \mathcal{O}(h^3) \quad (3.14)$$

To evaluate $\mathbf{y}'(t+h/2)$, we take another two Taylor expansions, this time of \mathbf{y}' , to form:

$$\begin{aligned} \mathbf{y}'(t+h) &= \mathbf{y}'(t+h/2) + \frac{h}{2}\mathbf{y}''(t+h/2) + \frac{h^2}{8}\mathbf{y}'''(\xi_3) \\ \mathbf{y}'(t) &= \mathbf{y}'(t+h/2) - \frac{h}{2}\mathbf{y}''(t+h/2) + \frac{h^2}{8}\mathbf{y}'''(\xi_4) \end{aligned}$$

with $\xi_3 \in [t+h/2, t+h]$ and $\xi_4 \in [t, t+h/2]$. Adding these expressions and re-arranging yields:

$$\mathbf{y}'(t+h/2) = \frac{1}{2}(\mathbf{y}'(t+h) + \mathbf{y}'(t)) + \mathcal{O}(h^2)$$

Substituting this back into Eq. 3.14 we obtain:

$$\mathbf{y}(t+h) - \mathbf{y}(t) = \frac{h}{2} (\mathbf{y}'(t+h) + \mathbf{y}'(t)) + \mathcal{O}(h^3)$$

Which yields the trapezoidal numerical integration algorithm, also known as the *modified-Euler method*:

$$\mathbf{y}_n = \mathbf{y}_{n-1} + \frac{h}{2} [\mathbf{f}(t_n, \mathbf{y}_n) + \mathbf{f}(t_{n-1}, \mathbf{y}_{n-1})] \quad (3.15)$$

The method is implicit, with local and global truncation errors of $\mathcal{O}(h^3)$ and $\mathcal{O}(h^2)$ respectively.

Example 3.3 Solve the test ODE

$$\frac{dy}{dt} = \lambda y, \quad y(0) = y_0$$

using the trapezoidal method.

Answer: Using Eq. 3.15, we can determine y at each step as follows:

$$\begin{aligned} y(t+h) &= y(t) + \frac{h}{2} [y'(t) + y'(t+h)] \\ &= y(t) + \frac{h}{2} [\lambda y(t) + \lambda y(t+h)] \\ y(t+h) \left[1 - \frac{h\lambda}{2} \right] &= y(t) \left[1 + \frac{h\lambda}{2} \right] \\ \therefore y(t+h) &= y(t) \left[\frac{2+h\lambda}{2-h\lambda} \right] \end{aligned}$$

For positive step-sizes h and $\lambda < 0$, the magnitude of factor $[2+h\lambda]/[2-h\lambda]$ will always be less than unity, rendering the method unconditionally stable. \square

3.2.3 Runge–Kutta Methods

For the ODE

$$\frac{dy}{dt} = f(t, y), \quad y(0) = y_0$$

Runge–Kutta² methods effectively integrate both sides at each time-step to obtain:

²Pronounced *Run-ghee Cut-ah*.

$$y(t + h) = y(t) + \int_t^{t+h} f(\tau, y) \, d\tau \tag{3.16}$$

and then approximate the integral using numerical *quadrature*. In brief, quadrature methods sub-divide the interval $[t, t + h]$ at intermediate points $t_i = t + c_i h$, $i = 1 \dots s$, where the $c_i \in [0, 1]$ denote fixed nodal positions along the interval. The integral is then approximated using a weighted sum of function evaluations at these intermediate points. For a one-variable function $f(\tau)$, the general quadrature approach may be written as:

$$\int_t^{t+h} f(\tau) \, d\tau \approx h \sum_{i=1}^s b_i f(t + c_i h)$$

where the b_i are constant coefficients that typically depend on s and the c_i . Two popular quadrature formulae for one-variable functions are the *midpoint rule* and *Simpson's rule*:

$$\int_t^{t+h} f(\tau) \, d\tau \approx hf \left(t + \frac{h}{2} \right) \tag{midpoint rule}$$

$$\int_t^{t+h} f(\tau) \, d\tau \approx \frac{h}{6} \left[f(t) + 4f \left(t + \frac{h}{2} \right) + f(t + h) \right] \tag{Simpson's rule}$$

However for the function of Eq. 3.16 which involves an additional variable y , we must estimate the function values at the intermediate points t_i . Denoting these intermediate function values by K_i , Runge–Kutta methods approximate these from a weighted sum of other intermediate values on the interval, then use a weighted sum of the K_i to approximate the integral in Eq. 3.16, and hence determine $y(t + h)$. The general Runge–Kutta algorithm can be expressed in the form:

$$K_i = f \left(t + c_i h, y(t) + h \sum_{j=1}^s a_{ij} K_j \right)$$

$$y(t + h) = y(t) + h \sum_{i=1}^s b_i K_i \tag{3.17}$$

where s is the number of stages. Runge–Kutta algorithms are typically represented using the shorthand notation:

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array}$$

For computational efficiency, we always choose $c_1 = 0$: this conveniently provides one function evaluation at the current step. To find, for example, the remaining coefficients for an explicit two-stage (i.e. $s = 2$) Runge–Kutta method, we rewrite Eq. 3.17 as

$$\begin{aligned} K_1 &= f(t + c_1 h, y(t) + h[a_{11}K_1 + a_{12}K_2]) \\ K_2 &= f(t + c_2 h, y(t) + h[a_{21}K_1 + a_{22}K_2]) \\ y(t + h) &= y(t) + h[b_1K_1 + b_2K_2] \end{aligned}$$

Since we require the method to be explicit, we can immediately see that $a_{11} = a_{12} = a_{22} = 0$, and along with $c_1 = 0$, we are left with four unknown coefficients to be determined for the algorithm:

$$\begin{aligned} K_1 &= f(t, y(t)) \\ K_2 &= f(t + c_2 h, y(t) + h a_{21} K_1) \\ y(t + h) &= y(t) + h[b_1 K_1 + b_2 K_2] \end{aligned} \quad (3.18)$$

We can express $y(t + h)$ in terms of our familiar Taylor series expansion (Eq. 3.1) centered on t as:

$$y(t + h) = y(t) + h y'(t) + \frac{h^2}{2} y''(t) + \mathcal{O}(h^3) \quad (3.19)$$

Since $y'(t) = f(t, y)$ from the original ODE, we have:

$$\begin{aligned} y''(t) &= f'(t, y) \\ &= \frac{\partial f(t, y)}{\partial t} + \frac{\partial f(t, y)}{\partial y} \frac{dy(t)}{dt} \\ &= \frac{\partial f(t, y)}{\partial t} + \frac{\partial f(t, y)}{\partial y} f(t, y) \end{aligned}$$

and Eq. 3.19 becomes:

$$y(t + h) = y(t) + h f(t, y) + \frac{h^2}{2} \left[\frac{\partial f}{\partial t}(t, y) + \frac{\partial f}{\partial y}(t, y) f(t, y) \right] + \mathcal{O}(h^3) \quad (3.20)$$

We then use the multivariate form of Taylor's theorem (Eq. 3.4) to expand K_2 in Eq. 3.18:

$$\begin{aligned} K_2 &= f(t + c_2 h, y(t) + h a_{21} K_1) \\ &= f(t, y) + c_2 h \frac{\partial f}{\partial t}(t, y) + h a_{21} K_1 \frac{\partial f}{\partial y}(t, y) + \mathcal{O}(h^2) \\ &= f(t, y) + c_2 h \frac{\partial f}{\partial t}(t, y) + h a_{21} f(t, y) \frac{\partial f}{\partial y}(t, y) + \mathcal{O}(h^2) \end{aligned} \quad (3.21)$$

Substituting Eq. 3.21 into 3.18, we obtain:

$$\begin{aligned}
 y(t+h) &= y(t) + h [b_1 K_1 + b_2 K_2] \\
 &= y(t) + h \left[b_1 f(t, y) + b_2 \left(f(t, y) + c_2 h \frac{\partial f}{\partial t}(t, y) \right. \right. \\
 &\quad \left. \left. + h a_{21} f(t, y) \frac{\partial f}{\partial y}(t, y) + \mathcal{O}(h^2) \right) \right] + \mathcal{O}(h^3) \\
 &= y(t) + h(b_1 + b_2) f(t, y) + h^2 c_2 b_2 \frac{\partial f}{\partial t}(t, y) \\
 &\quad + h^2 a_{21} b_2 f(t, y) \frac{\partial f}{\partial y}(t, y) + \mathcal{O}(h^3) \tag{3.22}
 \end{aligned}$$

Comparing the coefficients of Eqs. 3.20 and 3.22, we have:

$$\begin{aligned}
 b_1 + b_2 &= 1 \\
 c_2 b_2 &= 1/2 \\
 a_{21} b_2 &= 1/2
 \end{aligned}$$

representing three equations in four unknowns. Setting $a_{21} = \alpha$, where α is a parameter, we obtain $b_2 = 1/2\alpha$, $b_1 = 1 - 1/2\alpha$, $c_2 = \alpha$, with the algorithm represented as:

$$\begin{array}{c|cc}
 0 & 0 & 0 \\
 \alpha & \alpha & 0 \\
 \hline
 1 & -\frac{1}{2\alpha} & \frac{1}{2\alpha}
 \end{array}$$

Since the local truncation error is $\mathcal{O}(h^3)$, the global truncations error will be $\mathcal{O}(h^2)$. This one-parameter family of Runge–Kutta algorithms is therefore second-order. For $\alpha = 1/2$, we have the *explicit midpoint method*:

$$y(t+h) = y(t) + h \left[f\left(t + \frac{h}{2}, y(t) + \frac{h}{2} f(t, y)\right) \right]$$

which in terms of our familiar ODE system (Eq. 3.6), may be written as:

$$\begin{aligned}
 \mathbf{K}_1 &= \mathbf{f}(t_{n-1}, \mathbf{y}_{n-1}) \\
 \mathbf{K}_2 &= \mathbf{f}\left(t_{n-1} + \frac{h}{2}, \mathbf{y}_{n-1} + \frac{h}{2} \mathbf{K}_1\right) \\
 \mathbf{y}_n &= \mathbf{y}_{n-1} + h \mathbf{K}_2
 \end{aligned}$$

Example 3.4 Solve the following test ODE using the second-order explicit midpoint Runge–Kutta method:

$$\frac{dy}{dt} = \lambda y, \quad y(0) = y_0$$

Answer: For each step, we perform the following stage evaluations:

$$\begin{aligned}
 K_1 &= f(t, y) = \lambda y \\
 K_2 &= f\left(t + \frac{h}{2}, y + \frac{h}{2}K_1\right) \\
 &= f\left(t + \frac{h}{2}, y + \frac{h}{2}\lambda y\right) \\
 &= \lambda\left(1 + \frac{h}{2}\lambda\right)y \\
 y(t+h) &= y(t) + hK_2 \\
 &= y(t) + h\lambda\left(1 + \frac{h}{2}\lambda\right)y(t) \\
 &= y(t)\left[1 + h\lambda + \frac{1}{2}h^2\lambda^2\right]
 \end{aligned}$$

For absolute stability, we require $-1 \leq 1 + h\lambda + \frac{1}{2}h^2\lambda^2 \leq 1$, which is satisfied when $-2 \leq h\lambda \leq 0$, or $h \leq \frac{2}{|\lambda|}$. \square

A similar derivation to the above second-order methods, although more tedious, can be used to obtain the well-known classical fourth-order Runge–Kutta algorithm:

$$\begin{aligned}
 \mathbf{K}_1 &= \mathbf{f}(t_{n-1}, \mathbf{y}_{n-1}) \\
 \mathbf{K}_2 &= \mathbf{f}\left(t_{n-1} + \frac{h}{2}, \mathbf{y}_{n-1} + \frac{h}{2}\mathbf{K}_1\right) \\
 \mathbf{K}_3 &= \mathbf{f}\left(t_{n-1} + \frac{h}{2}, \mathbf{y}_{n-1} + \frac{h}{2}\mathbf{K}_2\right) \\
 \mathbf{K}_4 &= \mathbf{f}(t_{n-1} + h, \mathbf{y}_{n-1} + h\mathbf{K}_3) \\
 \mathbf{y}_n &= \mathbf{y}_{n-1} + \frac{h}{6}(\mathbf{K}_1 + 2\mathbf{K}_2 + 2\mathbf{K}_3 + \mathbf{K}_4)
 \end{aligned}$$

represented in shorthand as:

$$\begin{array}{c|cccc}
 0 & 0 & 0 & 0 & 0 \\
 \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\
 \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\
 1 & 0 & 0 & 1 & 0 \\
 \hline
 & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6}
 \end{array}$$

Runge–Kutta algorithms are employed by many ODE numerical solvers, together with adaptive step-size routines that monitor the local error and adjust the step-size accordingly. Matlab’s ode23, ode23tb and ode45 solvers use a pair of Runge–Kutta algorithms of orders p and $p + 1$ to determine the local error.³ Denoting the current

³For ode23/ode23tb, $p = 2$, whilst for ode45, $p = 4$.

step solutions for these higher and lower order methods by $\hat{\mathbf{y}}_n$ and \mathbf{y}_n respectively, these codes estimate the local error, l_n , using

$$l_n = \|\hat{\mathbf{y}}_n - \mathbf{y}_n\|_\infty \quad (3.23)$$

where $\|\cdot\|_\infty$ denotes the infinity or maximum norm, corresponding to the maximum absolute value of all vector elements.⁴ From each element i of the lower-order estimates \mathbf{y}_n and \mathbf{y}_{n-1} , the maximum absolute values from each are taken to form $y_i = \max(|(y_i)_n|, |(y_i)_{n-1}|)$. Denoting $y = \min_i(y_i)$, the step is rejected if

$$l_n > \max(R_{tol}y, A_{tol}) \quad (3.24)$$

where R_{tol} and A_{tol} are user-specified parameters denoting the relative and absolute tolerances respectively. Matlab uses default values of $R_{tol} = 10^{-3}$ and $A_{tol} = 10^{-6}$. If a step is rejected, it is performed again using the smaller step-size

$$h_{new} = h_{old} \max\left(0.5, 0.8 \left[\frac{\max(R_{tol}y, A_{tol})}{l_n} \right]^{\frac{1}{p+1}}\right) \quad (3.25)$$

The rationale behind this step-size choice is based on the fact that the local truncation error for a p -order method is equal to ch^{p+1} , where c is a constant. To reduce the step-size such that the local error satisfies the specified tolerance levels, the codes require the new step to satisfy $l_n \approx \gamma \max(R_{tol}y, A_{tol})$, with γ being around 0.9. In other words, Eq. 3.24 no longer holds, with a safety factor of γ employed to ensure this. The new step-size can be determined from

$$\begin{aligned} ch_{new}^{p+1} &= \gamma \max(R_{tol}y, A_{tol}) \\ ch_{old}^{p+1} &= l_n \\ \therefore \left(\frac{h_{new}}{h_{old}}\right)^{p+1} &= \gamma \left[\frac{\max(R_{tol}y, A_{tol})}{l_n} \right] \\ \text{or } h_{new} &= h_{old} \left(\gamma^{\frac{1}{p+1}}\right) \left[\frac{\max(R_{tol}y, A_{tol})}{l_n} \right]^{\frac{1}{p+1}} \end{aligned}$$

To speed-up the computations, the factor $\left(\gamma^{\frac{1}{p+1}}\right)$ is simply replaced by 0.8, which is adequate for most p used. Equation 3.25 also employs a minimum reduction factor of 0.5 to avoid excessive small steps. The Matlab ODE solvers also employ additional restrictions on the step-size reduction, including that h_{new} is no smaller than the computational precision. If the step is successful, Eq. 3.25 can be also used to

⁴Matlab ODE solver options also allow the local error to be determined using the 2-norm. By default, however, the maximum norm is used.

increase the step-size to ensure the local error is not too small, but still satisfies $l_n \approx \gamma \max(R_{tol}y, A_{tol})$.

As we have seen, using a pair of Runge–Kutta algorithms allows for an adaptive-step strategy. However to reduce the computational burden of employing two algorithms per step, it is advantageous to choose a pair of methods that share the same stage computations (i.e. the a_{ij} and c_i in Eq. 3.17), known as *embedded methods*. These embedded pairs typically have orders that differ by 1 and are termed a $p(q)$ pair, where p is the order of the method used to advance to the next step, and q is the order used to determine the local error. The most well-known of these is the Fehlberg 4(5) pair. Leaving out the zero coefficients above the diagonal, this pair can be represented as

| | | | | | | |
|-----------------|---------------------|----------------------|----------------------|-----------------------|-----------------|----------------|
| 0 | | | | | | |
| $\frac{1}{4}$ | $\frac{1}{4}$ | | | | | |
| $\frac{3}{8}$ | $\frac{3}{32}$ | $\frac{9}{32}$ | | | | |
| $\frac{12}{13}$ | $\frac{1932}{2197}$ | $-\frac{7200}{2197}$ | $\frac{7296}{2197}$ | | | |
| 1 | $\frac{439}{216}$ | -8 | $\frac{3680}{513}$ | $-\frac{845}{4104}$ | | |
| $\frac{1}{2}$ | $-\frac{8}{27}$ | 2 | $-\frac{3544}{2565}$ | $\frac{1859}{4104}$ | $-\frac{1}{5}$ | |
| | $\frac{25}{216}$ | 0 | $\frac{1408}{2565}$ | $\frac{2197}{4104}$ | $-\frac{1}{5}$ | 0 |
| | $\frac{16}{135}$ | 0 | $\frac{6656}{12825}$ | $\frac{28561}{56430}$ | $-\frac{9}{50}$ | $\frac{2}{55}$ |

where the rows below the horizontal lines denote two sets of b_i coefficients (cf. Eq. 3.17): the first row pertains to the fourth-order method and the second row to the fifth. This pair has six stage computations.

In many implementations of embedded Runge–Kutta pairs, the higher-order method is instead used to provide the solution for the next step, a process referred to as *local extrapolation*. When this occurs, the local error estimate given by Eq. 3.23 is no longer accurate, but the adaptive-step strategy of Eq. 3.25 is nonetheless used. An example of such a local extrapolation method is the 3(2) pair implemented by the Matlab ode23 solver [4]:

| | | | |
|---------------|----------------|---------------|---------------|
| 0 | | | |
| $\frac{1}{2}$ | $\frac{1}{2}$ | | |
| $\frac{3}{4}$ | 0 | $\frac{3}{4}$ | |
| | $\frac{2}{9}$ | $\frac{1}{3}$ | $\frac{4}{9}$ |
| | $\frac{7}{24}$ | $\frac{1}{4}$ | $\frac{1}{3}$ |
| | | $\frac{1}{3}$ | $\frac{1}{8}$ |

Here, the first row of b_i coefficients represents the third-order method, whilst the second row represents the second-order method. Another-example is the 5(4) Dormand–Prince pair [7] implemented by Matlab’s ode45:

| | | | | | | | |
|----------------|----------------------|-----------------------|----------------------|--------------------|-------------------------|--------------------|----------------|
| 0 | | | | | | | |
| $\frac{1}{5}$ | $\frac{1}{5}$ | | | | | | |
| $\frac{3}{10}$ | $\frac{3}{40}$ | $\frac{9}{40}$ | | | | | |
| $\frac{4}{5}$ | $\frac{44}{45}$ | $-\frac{56}{15}$ | $\frac{32}{9}$ | | | | |
| $\frac{8}{9}$ | $\frac{19372}{6561}$ | $-\frac{25360}{2187}$ | $\frac{64448}{6561}$ | $-\frac{212}{729}$ | | | |
| 1 | $\frac{9017}{3168}$ | $-\frac{355}{33}$ | $\frac{46732}{5247}$ | $\frac{49}{176}$ | $-\frac{5103}{18656}$ | | |
| 1 | $\frac{35}{384}$ | 0 | $\frac{500}{1113}$ | $\frac{125}{192}$ | $-\frac{2187}{6784}$ | $\frac{11}{84}$ | |
| | $\frac{35}{384}$ | 0 | $\frac{500}{1113}$ | $\frac{125}{192}$ | $-\frac{2187}{6784}$ | $\frac{11}{84}$ | 0 |
| | $\frac{5179}{57600}$ | 0 | $\frac{7571}{16695}$ | $\frac{393}{640}$ | $-\frac{92097}{339200}$ | $\frac{187}{2100}$ | $\frac{1}{40}$ |

Here, the first b_i row yields the fifth order estimate whilst the second row yields the fourth order one. This pair has a total of seven stages, corresponding to seven function evaluations per step. However, as can be seen from the above table, the last row of the a_{ij} has the same coefficients as the first b_i row. This means that the final function evaluation in a step corresponds to the first function evaluation of the next step. Hence, after each successful step, only six function evaluations are required for the next step, effectively reducing the computational cost to a six-stage method.

3.2.4 The Generalized- α Method

For many ODE systems commonly encountered in structural or fluid mechanics applications, it may be desirable (or even necessary) to incorporate some degree of high-frequency suppression in the numerical integration procedure, known as *algorithmic damping*. When present, these high-frequency components typically represent numerical artefacts due to an excessively large step-size, compounded by poor spatial discretization when formulating the underlying ODE system. One side-effect is that algorithmic damping can also introduce undesired damping in the important low-frequency ranges as well. To this end, the *generalized- α method* [5, 14] implements user-controlled high-frequency damping whilst minimizing the damping over lower frequencies. An extension of the generalized trapezoidal family [12], the generalized- α method takes on slightly different algorithm parameters depending on whether the ODE for a given variable is first or second order. We will describe in detail the first-order ODE implementation below, and simply give the result for the second-order system.

For the first-order ODE

$$\dot{y} = f(t, y)$$

the generalized- α method is written as:

$$\begin{aligned}\dot{y}_{n+\alpha_m} &= f(t_{n+\alpha_f}, y_{n+\alpha_f}) \\ y_{n+1} &= y_n + h\dot{y}_n + h\gamma(\dot{y}_{n+1} - \dot{y}_n)\end{aligned}\quad (3.26)$$

where h is the step-size and

$$\begin{aligned}\dot{y}_{n+\alpha_m} &= \dot{y}_n + \alpha_m (\dot{y}_{n+1} - \dot{y}_n) \\ y_{n+\alpha_f} &= y_n + \alpha_f (y_{n+1} - y_n) \\ t_{n+\alpha_f} &= t_n + h\alpha_f\end{aligned}$$

α_m , α_f and γ are parameters of the algorithm whose values are determined from considerations of accuracy, stability and user-specified damping. The method is implicit, and Newton's method (Eq. 3.12) must be used to solve for the y_{n+1} , \dot{y}_{n+1} at each step. To improve convergence, a predictor is commonly used to provide initial estimates for these values.

To determine the values of the algorithm parameters, we apply the method to the simple first-order ODE:

$$\dot{y} = \lambda y$$

The generalized- α method (Eq. 3.26) can then be written as:

$$\begin{aligned}\dot{y}_n + \alpha_m (\dot{y}_{n+1} - \dot{y}_n) &= \lambda [y_n + \alpha_f (y_{n+1} - y_n)] \\ y_{n+1} &= y_n + h\dot{y}_n + h\gamma(\dot{y}_{n+1} - \dot{y}_n)\end{aligned}$$

Grouping together terms at steps n and $n + 1$, the above can be re-arranged into the matrix system

$$\begin{bmatrix} -\lambda\alpha_f & \frac{\alpha_m}{h} \\ 1 & -\gamma \end{bmatrix} \begin{bmatrix} y_{n+1} \\ h\dot{y}_{n+1} \end{bmatrix} = \begin{bmatrix} \lambda(1 - \alpha_f) & \frac{\alpha_m - 1}{h} \\ 1 & (1 - \gamma) \end{bmatrix} \begin{bmatrix} y_n \\ h\dot{y}_n \end{bmatrix}\quad (3.27)$$

Noting that

$$\begin{aligned}\begin{bmatrix} -\lambda\alpha_f & \frac{\alpha_m}{h} \\ 1 & -\gamma \end{bmatrix}^{-1} &= \frac{1}{\gamma\lambda\alpha_f - \frac{\alpha_m}{h}} \begin{bmatrix} -\gamma & -\frac{\alpha_m}{h} \\ -1 & -\lambda\alpha_f \end{bmatrix} \\ &= \frac{1}{\gamma\lambda h\alpha_f - \alpha_m} \begin{bmatrix} -\gamma h & -\alpha_m \\ -h & -\lambda h\alpha_f \end{bmatrix} \\ &= \frac{1}{\alpha_m - \gamma\Omega\alpha_f} \begin{bmatrix} \gamma h & \alpha_m \\ h & \Omega\alpha_f \end{bmatrix}\end{aligned}$$

where $\Omega = \lambda h$ represents a normalized frequency term. Multiplying both sides of Eq. 3.27 by this inverse matrix, we obtain:

$$\begin{aligned}
\begin{bmatrix} y_{n+1} \\ h\dot{y}_{n+1} \end{bmatrix} &= \frac{1}{\alpha_m - \gamma\Omega\alpha_f} \begin{bmatrix} \gamma h & \alpha_m \\ h & \Omega\alpha_f \end{bmatrix} \begin{bmatrix} \lambda(1 - \alpha_f) & \frac{\alpha_m - 1}{h} \\ 1 & (1 - \gamma) \end{bmatrix} \begin{bmatrix} y_n \\ h\dot{y}_n \end{bmatrix} \\
&= \frac{1}{\alpha_m - \gamma\Omega\alpha_f} \begin{bmatrix} \gamma\lambda h(1 - \alpha_f) + \alpha_m & \gamma(\alpha_m - 1) + \alpha_m(1 - \gamma) \\ \Omega(1 - \alpha_f) + \Omega\alpha_f & \alpha_m - 1 + \Omega\alpha_f(1 - \gamma) \end{bmatrix} \begin{bmatrix} y_n \\ h\dot{y}_n \end{bmatrix} \\
&= \frac{1}{\alpha_m - \gamma\Omega\alpha_f} \begin{bmatrix} \alpha_m + \gamma\Omega(1 - \alpha_f) & \alpha_m - \gamma \\ \Omega & \alpha_m - 1 + \Omega\alpha_f(1 - \gamma) \end{bmatrix} \begin{bmatrix} y_n \\ h\dot{y}_n \end{bmatrix}
\end{aligned}$$

Denoting the solution vector by $\mathbf{y}_n = [y_n, h\dot{y}_n]^T$, the above matrix system can be written simply as:

$$\mathbf{y}_{n+1} = \mathbf{A}\mathbf{y}_n \quad (3.28)$$

where \mathbf{A} is the *amplification matrix* given by

$$\mathbf{A} = \frac{1}{\alpha_m - \gamma\Omega\alpha_f} \begin{bmatrix} \alpha_m + \gamma\Omega(1 - \alpha_f) & \alpha_m - \gamma \\ \Omega & \alpha_m - 1 + \Omega\alpha_f(1 - \gamma) \end{bmatrix} \quad (3.29)$$

From Eq. 3.28, we can see that

$$\mathbf{y}_{n+1} = \mathbf{A}^n \mathbf{y}_0$$

where \mathbf{y}_0 is the initial solution vector at $t = 0$. For the generalized- α method to be stable, we require that for real $\lambda \leq 0$, $\mathbf{y}_n \rightarrow \mathbf{0}$ as $n \rightarrow \infty$. This is equivalent to stating that the modulus of the eigenvalues of \mathbf{A} are all less than or equal to 1. Each eigenvalue of \mathbf{A} , ρ , must satisfy

$$\det(\mathbf{A} - \rho\mathbf{I}) = 0$$

Denoting the elements of \mathbf{A} by A_{ij} ($i, j = 1, 2$), we can expand the above as:

$$\begin{aligned}
\det \begin{bmatrix} A_{11} - \rho & A_{12} \\ A_{21} & A_{22} - \rho \end{bmatrix} &= (A_{11} - \rho)(A_{22} - \rho) - A_{12}A_{21} \\
&= A_{11}A_{22} - (A_{11} + A_{22})\rho + \rho^2 - A_{12}A_{21} \\
&= A_{11}A_{22} - A_{12}A_{21} - (A_{11} + A_{22})\rho + \rho^2 \\
&= \det(\mathbf{A}) - \text{trace}(\mathbf{A})\rho + \rho^2 \\
&= 0
\end{aligned}$$

where $\text{trace}(\mathbf{A})$ is the sum of diagonal elements of \mathbf{A} . Hence, the eigenvalues of \mathbf{A} satisfy the characteristic equation

$$\rho^2 - \text{trace}(\mathbf{A})\rho + \det(\mathbf{A}) = 0 \quad (3.30)$$

the solution of which leads to two eigenvalues ρ_1, ρ_2 given by

$$\rho_{1,2} = \frac{\text{trace}(\mathbf{A}) \pm \sqrt{\text{trace}(\mathbf{A})^2 - 4 \det(\mathbf{A})}}{2} \quad (3.31)$$

Furthermore, using the Caley–Hamilton theorem, a square matrix must also satisfy its own characteristic equation. Thus, \mathbf{A} also satisfies Eq. 3.30:

$$\mathbf{A}^2 - \text{trace}(\mathbf{A})\mathbf{A} + \det(\mathbf{A}) = \mathbf{0}$$

Right-multiplying all terms by the previous solution vector, \mathbf{y}_{n-1} , we obtain:

$$\mathbf{A}^2 \mathbf{y}_{n-1} - \text{trace}(\mathbf{A})\mathbf{A} \mathbf{y}_{n-1} + \det(\mathbf{A})\mathbf{y}_{n-1} = \mathbf{0}$$

Using $\mathbf{y}_{n+1} = \mathbf{A}^2 \mathbf{y}_{n-1}$ and $\mathbf{y}_n = \mathbf{A} \mathbf{y}_{n-1}$, this simplifies to:

$$\mathbf{y}_{n+1} - \text{trace}(\mathbf{A})\mathbf{y}_n + \det(\mathbf{A})\mathbf{y}_{n-1} = \mathbf{0}$$

Taking the first element of each solution vector in the above leads to the following recurrence relation:

$$y_{n+1} - \text{trace}(\mathbf{A})y_n + \det(\mathbf{A})y_{n-1} = 0 \quad (3.32)$$

If we substitute the exact values of the solution in Eq. 3.32, using $y(t_{n+1})$ for y_{n+1} , $y(t_n)$ for y_n , and $y(t_{n-1})$ for y_{n-1} , we can define the local truncation error $\tau(t_n)$ according to [12]:

$$y(t_{n+1}) - \text{trace}(\mathbf{A})y(t_n) + \det(\mathbf{A})y(t_{n-1}) = h\tau(t_n) \quad (3.33)$$

Using Taylor series expansions about $y(t_n)$, we can rewrite Eq. 3.33 as:

$$y(t_n) + h\dot{y}(t_n) + \frac{h^2}{2}\ddot{y}(t_n) - \text{trace}(\mathbf{A})y(t_n) + \det(\mathbf{A}) \left[y(t_n) - h\dot{y}(t_n) + \frac{h^2}{2}\ddot{y}(t_n) \right] + \mathcal{O}(h^3) = h\tau(t_n)$$

Evaluating the above using exact derivatives, we have:

$$y(t_n) + h\lambda y(t_n) + \frac{h^2\lambda^2 y(t_n)}{2} - \text{trace}(\mathbf{A})y(t_n) + \det(\mathbf{A}) \left[y(t_n) - h\lambda y(t_n) + \frac{h^2\lambda^2 y(t_n)}{2} \right] + \mathcal{O}(h^3) = h\tau(t_n)$$

Substituting $\Omega = h\lambda$:

$$y(t_n) \left[1 + \Omega + \frac{\Omega^2}{2} - \text{trace}(\mathbf{A}) + \det(\mathbf{A})(1 - \Omega + \frac{\Omega^2}{2}) \right] + \mathcal{O}(h^3) = h\tau(t_n)$$

The method will be second-order accurate (i.e. local truncation error will be $\mathcal{O}(h^2)$) if the term in square brackets above is zero. That is

$$1 + \Omega + \frac{\Omega^2}{2} - \text{trace}(\mathbf{A}) + \det(\mathbf{A}) \left(1 - \Omega + \frac{\Omega^2}{2} \right) = 0 \quad (3.34)$$

From Eq. 3.29, we can readily determine that

$$\begin{aligned} \text{trace}(\mathbf{A}) &= \frac{2\alpha_m - 2\alpha_f\gamma\Omega + \gamma\Omega + \alpha_f\Omega - 1}{\alpha_m - \alpha_f\gamma\Omega} \\ &= 2 + \frac{\Omega\alpha_f + \gamma\Omega - 1}{\alpha_m - \alpha_f\gamma\Omega} \end{aligned} \quad (3.35)$$

$$\begin{aligned} \det(\mathbf{A}) &= \frac{[\alpha_m - \alpha_f\gamma\Omega + \gamma\Omega][\alpha_m - \alpha_f\gamma\Omega + \alpha_f\Omega - 1] - \alpha_m\Omega + \gamma\Omega}{[\alpha_m - \alpha_f\gamma\Omega]^2} \\ &= 1 - \frac{\Omega - \Omega\alpha_f - \gamma\Omega + 1}{\alpha_m - \alpha_f\gamma\Omega} \end{aligned} \quad (3.36)$$

Substituting these into Eq. 3.34 and after much re-arranging, we obtain:

$$\frac{\Omega^2 \left(\frac{\Omega}{2} + \alpha_f - \alpha_m + \gamma - \frac{\alpha_f\Omega}{2} - \frac{\gamma\Omega}{2} - \alpha_f\gamma\Omega - \frac{1}{2} \right)}{\alpha_m - \alpha_f\gamma\Omega} = 0$$

Which leads to the solution

$$\gamma = \frac{2\alpha_m - 2\alpha_f - \Omega(1 - \alpha_f) + 1}{2\Omega\alpha_f - \Omega + 2}$$

For very small step sizes, $\Omega \rightarrow 0$, and the method will be second-order accurate when

$$\gamma = \frac{1}{2} + \alpha_m - \alpha_f \quad (3.37)$$

The choice of remaining algorithm parameters α_m and β_m will determine the stability of the generalized- α method, as well as its high-frequency damping. Recall that the method will be stable if for all real $\lambda \leq 0$, the magnitudes of the eigenvalues of \mathbf{A} are less than or equal to 1. From Eq. 3.31, these eigenvalues are given by

$$\rho_{1,2} = \frac{\text{trace}(\mathbf{A}) \pm \sqrt{\text{trace}(\mathbf{A})^2 - 4 \det(\mathbf{A})}}{2}$$

where $\text{trace}(\mathbf{A})$ and $\det(\mathbf{A})$ are given by Eqs. 3.35 and 3.36. In the limit as $\Omega \rightarrow 0$:

$$\begin{aligned}\text{trace}(\mathbf{A}) &= 2 - \frac{1}{\alpha_m} \\ \det(\mathbf{A}) &= 1 - \frac{1}{\alpha_m}\end{aligned}$$

and therefore

$$\rho_1 = 1, \quad \rho_2 = 1 - \frac{1}{\alpha_m}$$

From ρ_2 , we see that stability requires $\alpha_m \geq \frac{1}{2}$. Furthermore, for large step-sizes with $\lambda < 0$, $\Omega \rightarrow -\infty$ and

$$\begin{aligned}\lim_{|\Omega| \rightarrow \infty} \text{trace}(\mathbf{A}) &= 2 - \frac{\alpha_f + \gamma}{\alpha_f \gamma} \\ \lim_{|\Omega| \rightarrow \infty} \det(\mathbf{A}) &= 1 - \frac{\alpha_f + \gamma - 1}{\alpha_f \gamma}\end{aligned}$$

for which we obtain

$$\rho_1 = 1 - \frac{1}{\alpha_f}, \quad \rho_2 = 1 - \frac{1}{\gamma}$$

Hence, for stability, we require $\alpha_f \geq \frac{1}{2}$, $\gamma \geq \frac{1}{2}$. Combining all the above stability constraints with Eq. 3.37, we obtain the overall stability requirement $\alpha_m \geq \alpha_f \geq \frac{1}{2}$.

To choose specific values for α_f and α_m , we can specify a required level of high-frequency damping, ρ_∞ defined as

$$\rho_\infty = \lim_{|\Omega| \rightarrow \infty} \max(|\rho_1|, |\rho_2|)$$

where $0 \leq \rho_\infty \leq 1$. From Eq. 3.31, the sum of ρ_1 and ρ_2 will be $\text{trace}(\mathbf{A})$: if both eigenvalues are real, then one will necessarily grow with Ω whilst the other is reduced. To keep the magnitude of both eigenvalues ≤ 1 , the optimal constraint is to force both to be complex for all values of Ω . Then, from Eq. 3.31, they will be complex conjugate and have the same magnitude. Only in the limit as $|\Omega| \rightarrow \infty$ will both eigenvalues become real and equal to ρ_∞ . Therefore, we have:

$$\begin{aligned}\rho_\infty &= 1 - \frac{1}{\alpha_f} \\ \therefore \alpha_f &= \frac{1}{1 + \rho_\infty} \\ \text{and} \quad \rho_\infty &= 1 - \frac{1}{\gamma}\end{aligned} \tag{3.38}$$

$$\begin{aligned} \therefore \gamma &= \frac{1}{1 + \rho_\infty} = \alpha_f = \frac{1}{2} + \alpha_m - \alpha_f && \text{(from Eq. 3.37)} \\ \therefore \alpha_m &= 2\alpha_f - \frac{1}{2} \\ &= \frac{1}{2} \left(\frac{3 - \rho_\infty}{1 + \rho_\infty} \right) && (3.39) \end{aligned}$$

Equations 3.37–3.39 define the algorithmic parameters that specify given high-frequency damping and optimize stability over all frequencies. When $\rho_\infty = 1$ (i.e. no damping) the method is equivalent to the trapezoidal method (Sect. 3.2.2). When $\rho_\infty = 0$, the method provides maximal high-frequency damping.

COMSOL implements the generalized- α method as an option in its time dependent solver settings. In particular, the high-frequency damping parameter ρ_∞ is set by specifying the “amplification for high frequency” setting, as shown in Fig. 3.2. For first-order ODEs, COMSOL does not allow the user to specify no-damping (i.e. $\rho_\infty = 1$), but allows any other positive damping value < 1 .

Example 3.5 Use the generalized- α method in COMSOL to solve the logistic-growth ODE (see Eq. 2.1)

$$\frac{du}{dt} = 1000u(1 - u), \quad u(0) = 0.1$$

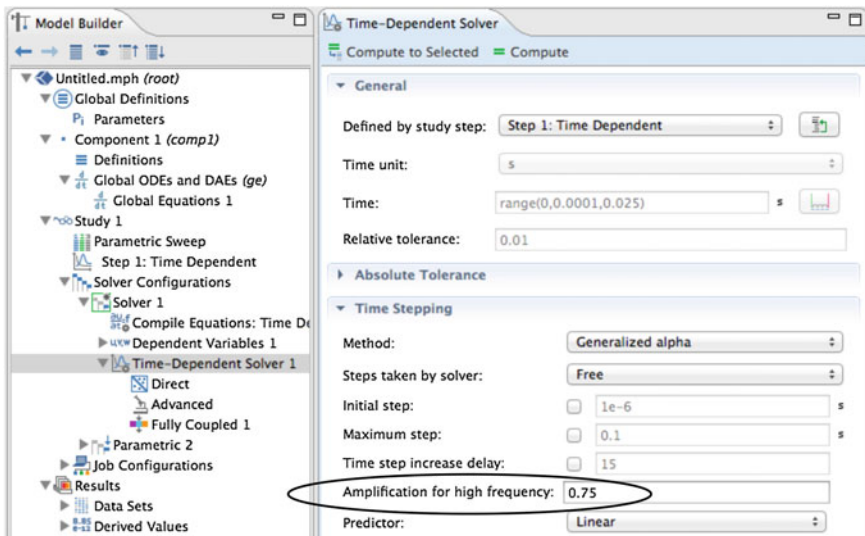


Fig. 3.2 Generalized- α method interface in COMSOL. The amplification for high frequency setting (shown circled) corresponds to ρ_∞ in the text, and is set to a default value of 0.75

for $\rho_\infty = 0, 0.5, 0.75, 0.99$, and compare with the exact solution:

$$u(t) = \frac{e^{1000t}}{[9 + e^{1000t}]}$$

Answer: To solve this ODE in COMSOL, we implement a 0D model with a single Global ODE and DAE interface, specifying the Generalized- α method in the time-stepping settings. The full implementation of the model is given in the steps below. For readers not familiar with COMSOL, a brief overview is provided in Appendix B.

Model Wizard

1. Open the Model Wizard and select the 0D spatial dimension.
2. In the Select Physics panel, choose Mathematics|ODE and DAE Interfaces|Global ODEs and DAEs. Click “Add”.
3. Click the Study arrow to open the Select Study panel. Select Time Dependent, and click “Done” to exit the Model Wizard.

Global Definitions

1. Right-click Global Definitions in the model tree and select Parameters. Enter a single parameter in the Parameters table of the Settings window with name `rho_inf` and expression `0.75`.

Global ODEs and DAEs


1. Select the Global Equations 1 subnode of the Global ODEs and DAEs node. In the settings table, enter a single row specifying the variable name as `u` and in the `f(u, ut, utt, t)` column (column 2), enter the expression

$$ut - (1000 [1/s]) * u * (1 - u)$$

This corresponds to the global equation $f(u, ut, utt, t) = 0$, where `ut`, `utt` denotes the first- and second-derivatives of `u` with respect to time. Note that `utt` is not used here, since our ODE is only first-order. In the third column, enter the initial value as `0.1`.

2. In the units section, just below the settings table, leave the units of the dependent variable quantity as dimensionless, but specify the source term quantity units as ‘frequency factor (1/s)’.

Study

1. Select the Step1: Time Dependent sub-node of the Study 1 node. In the Settings window, Click the Range button () adjacent to the Times field. Leave the entry method as ‘Step’ and enter Start and Stop values of `0`, `0.0001` and `0.025` respectively. Click Replace. This will create a range of output time values from `0` to `0.025` s in time steps of `0.0001` s.
2. Right-click the Study 1 node and select Show Default Solver. Select the Study 1| Solver Configurations|Solver 1|Time-Dependent Solver 1 node. In the Settings window, expand the Time Stepping tab, and select Generalized alpha for the

method. In the Amplification for high frequency setting, enter `rho_inf`. This will set the user-specified high-frequency damping to the global parameter defined earlier.

3. Right-click the Study 1 node and select the Parametric Sweep option. In the settings table, click the Add button (+) to add the parameter as `rho_inf`. In the parameter value list, type 0, 0.5, 0.75, 0.99. These comma-separated entries represent values the parameter will be successively assigned to during the sweep.
4. To solve the model, right-click Study 1 and select Compute (=).

Results

1. When the model has completed solving, the Graphics window will display a default plot of the solutions for each `rho_inf` parameter value. Right-click the 1D Plot Group 1 sub-node of the Results node and select Global. This will create a Global 2 sub-node of 1D Plot Group 1. In the settings table, specify
$$\exp((1000[1/s])*t)/(9+\exp((1000[1/s])*t))$$
 as the expression to plot, overwriting the default expression `comp1.u`. Note that units have also been entered in this expression to maintain unit consistency. Specify the Data set as Solution 2 and the Parameter selection as First.
2. Go to the Coloring and Style section below the settings table. Under Line style, specify the line as Dashed, the colour as Black, and the line width to be 2.
3. Next, go to the Legends tab below the Coloring and Style section. In the Legends setting, select Manual. In the legends table, type `Exact Solution`, overwriting the default entry `rho_inf=0`.
4. Clicking the plot button (🖨) will display the plot shown in Fig. 3.3.

It can be seen from the COMSOL solution that higher damping factors of 0 and 0.5 suppress high-frequency numerical oscillations, yielding results closer to the exact solution. □

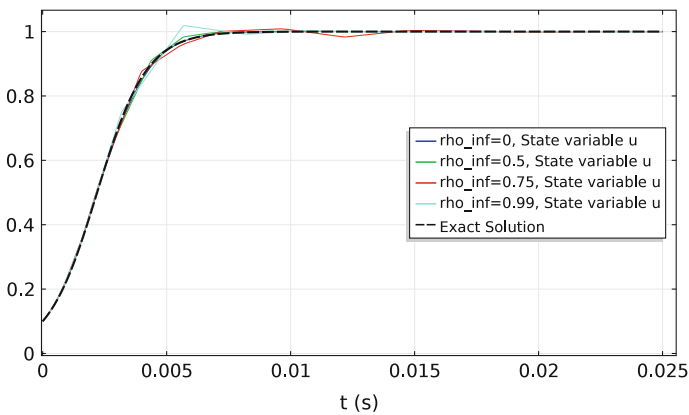


Fig. 3.3 COMSOL generalized- α solution to $u'(t) = 1000u(1 - u)$, $u(0) = 0.1$, using four high-frequency damping factors (ρ_∞) of 0, 0.5, 0.75 and 0.99. Also shown is the exact solution

For second-order systems, the generalized- α algorithm of Eq. 3.26 is slightly modified. For the second-order ODE

$$\ddot{y} + g(t, y)\dot{y} = f(t, y)$$

the method may be written as [5]:

$$\begin{aligned}\ddot{y}_{n+1-\alpha_m} &= f(t_{n+1-\alpha_f}, y_{n+1-\alpha_f}) - g(t_{n+1-\alpha_f}, y_{n+1-\alpha_f})\dot{y}_{n+1-\alpha_f} \\ \dot{y}_{n+1} &= \dot{y}_n + h \left[(1 - \gamma)\ddot{y}_n + \gamma\ddot{y}_{n+1} \right] \\ y_{n+1} &= y_n + h\dot{y}_n + h^2 \left[\left(\frac{1}{2} - \beta \right) \ddot{y}_n + \beta\ddot{y}_{n+1} \right]\end{aligned}\tag{3.40}$$

where h is the step-size and

$$\begin{aligned}\ddot{y}_{n+1-\alpha_m} &= (1 - \alpha_m)\ddot{y}_{n+1} + \alpha_m\ddot{y}_n \\ \dot{y}_{n+1-\alpha_f} &= (1 - \alpha_f)\dot{y}_{n+1} + \alpha_f\dot{y}_n \\ y_{n+1-\alpha_f} &= (1 - \alpha_f)y_{n+1} + \alpha_f y_n \\ t_{n+1-\alpha_f} &= t_n + h(1 - \alpha_f)\end{aligned}$$

Note that α_m and α_f in Eq. 3.40 have a slightly different definitions from the first-order case, and that a new algorithm parameter β has now been introduced. As in the first-order case, considerations of second-order accuracy, stability, and specified high-frequency damping lead to the new algorithm parameter values as [5]:

$$\begin{aligned}\alpha_m &= \frac{2\rho_\infty - 1}{\rho_\infty + 1} \\ \alpha_f &= \frac{\rho_\infty}{\rho_\infty + 1} \\ \beta &= \frac{1}{4}(1 - \alpha_m + \alpha_f)^2 \\ \gamma &= \frac{1}{2} - \alpha_m + \alpha_f\end{aligned}$$

where as before, ρ_∞ represents the user-specified high-frequency damping.

3.3 Multistep Methods

Multistep methods for numerically-integrating ODEs use information stored in previous time steps to advance the solution to the next step. For the general first-order ODE given by

$$\frac{dy}{dt} = f(t, y)$$

these methods store the values of y and/or f from the previous k steps. Denoting these stored values as $y_{n-1} \dots y_{n-k}$ and $f_{n-1} \dots f_{n-k}$, *linear multistep* methods take the form

$$\sum_{i=0}^k \alpha_i y_{n-i} = h \sum_{i=0}^k \beta_i f_{n-i} \quad (3.41)$$

where h is the step-size. The α_i, β_i coefficients are determined using an interpolating polynomial passing through the previous k step solutions. Consider the general set of k equi-spaced points $(t_1, y_1), (t_2, y_2) \dots (t_k, y_k)$ such that $t_i = t_1 + (i - 1)h$ ($i = 1 \dots k$), then the unique interpolating polynomial of degree $< k$ passing through these points is given by

$$\begin{aligned} \phi(t) = & \nabla^0 y_k + \frac{(t - t_k)}{h} \nabla^1 y_k + \frac{(t - t_k)(t - t_{k-1})}{2! h^2} \nabla^2 y_k + \dots \\ & + \frac{(t - t_k)(t - t_{k-1}) \dots (t - t_2)}{(k - 1)! h^{k-1}} \nabla^{k-1} y_k \end{aligned} \quad (3.42)$$

where ∇^i denotes the *backward difference operator* of order i , defined by

$$\begin{aligned} \nabla^0 y_j &= y_j \\ \nabla^i y_j &= \nabla^{i-1} y_j - \nabla^{i-1} y_{j-1} \end{aligned} \quad (3.43)$$

Equation 3.42 is referred to as *Newton's backward difference formula*. To show that it represents the required interpolating polynomial, we immediately see that when $t = t_k$,

$$\phi(t_k) = \nabla^0 y_k = y_k$$

as required. Furthermore, substituting $t = t_{k-1}$ yields

$$\begin{aligned} \phi(t_{k-1}) &= \nabla^0 y_k + \frac{(t_{k-1} - t_k)}{h} \nabla^1 y_k \\ &= y_k - \nabla^1 y_k \quad (\text{since } t_{k-1} - t_k = -h) \\ &= y_k - (\nabla^0 y_k - \nabla^0 y_{k-1}) \quad (\text{from (3.43)}) \\ &= y_{k-1} \end{aligned}$$

also as required. In general, substituting t_i into Eq. 3.42, we obtain:

$$\begin{aligned} \phi(t_i) &= \nabla^0 y_k + \frac{(t_i - t_k)}{h} \nabla^1 y_k + \dots + \frac{(t_i - t_k) \dots (t_i - t_{i+1})}{(k - i)! h^{k-i}} \nabla^{k-i} y_k \\ &= \nabla^0 y_k - (k - i) \nabla^1 y_k + \frac{(k - i)(k - i - 1)}{2!} \nabla^2 y_k \dots + (-1)^{k-i} \frac{(k - i)!}{(k - i)!} \nabla^{k-i} y_k \\ &\quad (\text{using } t_i - t_j = (i - j)h) \end{aligned}$$

$$\begin{aligned}
&= \nabla^0 y_k - (k-i)\nabla^1 y_k + \frac{(k-i)!}{2!(k-i-2)!} \nabla^2 y_k \cdots + (-1)^{k-i} \frac{(k-i)!}{(k-i)!} \nabla^{k-i} y_k \\
&= \sum_{r=0}^{k-i} (-1)^r C_r^{k-i} \nabla^r y_k, \quad \text{where } C_r^{k-i} = \frac{(k-i)!}{r!(k-i-r)!} \tag{3.44}
\end{aligned}$$

To show that Eq. 3.42 is satisfied at all the (t_i, y_i) points ($i = 1 \dots k$), we proceed by mathematical induction. Assume that for some i ($2 \leq i \leq k$), we have $\phi(t_i) = y_i$. Our task is to show that $\phi(t_{i-1}) = y_{i-1}$. We begin with the backward difference relationship:

$$\begin{aligned}
\nabla^1 y_i &= y_i - y_{i-1} \\
\therefore y_{i-1} &= y_i - \nabla^1 y_i
\end{aligned}$$

Substituting $\phi(t_i)$ from Eq. 3.44 in place of y_i on the right-hand side of the above (since we have assumed that $y_i = \phi(t_i)$), we obtain:

$$\begin{aligned}
y_{i-1} &= \sum_{r=0}^{k-i} (-1)^r C_r^{k-i} \nabla^r y_k - \nabla^1 \left[\sum_{r=0}^{k-i} (-1)^r C_r^{k-i} \nabla^r y_k \right] \\
&= \sum_{r=0}^{k-i} (-1)^r C_r^{k-i} \nabla^r y_k - \sum_{r=0}^{k-i} (-1)^r C_r^{k-i} \nabla^{r+1} y_k \\
&= \sum_{r=0}^{k-i} (-1)^r C_r^{k-i} \nabla^r y_k - \sum_{s=1}^{k-i+1} (-1)^{s-1} C_{s-1}^{k-i} \nabla^s y_k \\
&\quad \text{(where we have used } s = r + 1 \text{ in the second summation)} \\
&= \sum_{r=0}^{k-i} (-1)^r C_r^{k-i} \nabla^r y_k + \sum_{s=1}^{k-i+1} (-1)^s C_{s-1}^{k-i} \nabla^s y_k \\
&= \sum_{r=0}^{k-i} (-1)^r C_r^{k-i} \nabla^r y_k + \sum_{r=1}^{k-i+1} (-1)^r C_{r-1}^{k-i} \nabla^r y_k \\
&\quad \text{(where dummy index } s \text{ has been replaced with } r) \\
&= C_0^{k-i} \nabla^0 y_k + \sum_{r=1}^{k-i} (-1)^r [C_r^{k-i} + C_{r-1}^{k-i}] \nabla^r y_k \\
&\quad + (-1)^{k-i+1} C_{k-i}^{k-i} \nabla^{k-i+1} y_k
\end{aligned}$$

Now,

$$\begin{aligned}
C_r^{k-i} + C_{r-1}^{k-i} &= \frac{(k-i)!}{r!(k-i-r)!} + \frac{(k-i)!}{(r-1)!(k-i-r+1)!} \\
&= \frac{(k-i)!(k-i-r+1)}{r!(k-i-r+1)!} + \frac{(k-i)!r}{r!(k-i-r+1)!}
\end{aligned}$$

$$\begin{aligned}
&= \frac{(k-i)(k-i+1)}{r!(k-i-r+1)!} \\
&= \frac{(k-i+1)!}{r!(k-i+1-r)!} \\
&= C_{k-i+1}^r
\end{aligned}$$

Hence,

$$\begin{aligned}
y_{i-1} &= C_0^{k-i} \nabla^0 y_k + \left[\sum_{r=1}^{k-i} (-1)^r C_r^{k-i+1} \nabla^r y_k \right] + (-1)^{k-i+1} C_{k-i}^{k-i} \nabla^{k-i+1} y_k \\
&= \sum_{r=0}^{k-i+1} (-1)^r C_r^{k-i+1} \nabla^r y_k \\
&\quad (\text{since } C_0^{k-i} = C_0^{k-i+1} = 1 \text{ and } C_{k-i}^{k-i} = C_{k-i+1}^{k-i+1} = 1) \\
&= \phi(t_{i-1}) \quad \text{from Eq. 3.44, with } i \text{ replaced by } i-1.
\end{aligned}$$

and since we know that Eq. 3.42 satisfies $\phi(t_k) = y_k$, it must satisfy $\phi(t_i) = y_i$ for all $i : 1 \leq i \leq k$. Thus, Newton's backward difference polynomial passes through all the k (t_i, y_i) points, and is the required interpolating polynomial.

To determine the error when using Newton's backward difference formula, we assume that the y_i values above represent discrete samples of a continuous function $y(t)$ that is k -times differentiable over some closed interval $t \in [a, b]$ that includes the interpolation points t_i . We then define the error $E(t) = y(t) - \phi(t)$: where $E(t)$ will also be continuous and differentiable up to the same order as $y(t)$. It can be readily shown that for any value of $t \in [a, b]$, $E(t)$ satisfies

$$E(t) = \lambda(t - t_1)(t - t_2) \cdots (t - t_k) \quad (3.45)$$

where λ is some constant. To see that this expression represents the error, clearly $E(t)$ in Eq. 3.45 will be exactly zero at the interpolation points t_1, t_2, \dots, t_k as required. Furthermore, for a given value $t = \tau$ between the interpolation points, we have:

$$E(\tau) = \lambda(\tau - t_1)(\tau - t_2) \cdots (\tau - t_k) = y(\tau) - \phi(\tau)$$

and therefore λ can be assigned to

$$\lambda = \frac{y(\tau) - \phi(\tau)}{(\tau - t_1)(\tau - t_2) \cdots (\tau - t_k)}, \quad \tau \neq t_1, t_2, \dots, t_k$$

to precisely yield the error at $t = \tau$. With this choice of λ , we can form the new expression:

$$G(t) = y(t) - \phi(t) - \lambda(t - t_1)(t - t_2) \cdots (t - t_k) \quad (3.46)$$

$G(t)$ will have at least $k + 1$ zeros in $[a, b]$: k of these at $t = t_1, t_2, \dots, t_k$, and one at $t = \tau$. Since $G(t)$ is continuous, it will have one local maximum or minimum between two adjacent zeros. Its first-derivative, $G'(t)$, will therefore have at least k zeros. Similarly, its second-derivative $G''(t)$ will have at least $k - 1$ zeros, and so on. Continuing in this manner, and since $G(t)$ is also k -times differentiable, its k -th derivative, $G^{(k)}(t)$, must have at least one zero. From Eq. 3.46, we have:

$$G^{(k)}(t) = y^{(k)}(t) - \phi^{(k)}(t) - \lambda k!$$

and since $\phi(t)$ is a polynomial of degree $k - 1$, $\phi^{(k)}(t) = 0$. Hence,

$$G^{(k)}(t) = y^{(k)}(t) - \lambda k!$$

Now, denoting $\xi \in [a, b]$ as the zero of $G^{(k)}(t)$, we see that

$$\lambda = \frac{1}{k!} y^{(k)}(\xi)$$

And hence,

$$E(t) = \frac{y^{(k)}(\xi)}{k!} (t - t_1)(t - t_2) \cdots (t - t_k) \quad (3.47)$$

for some $\xi \in [a, b]$. This represents the error in Newton's backward difference formula, and depending on the choice of interval $[a, b]$, can be used to bound the error for either interpolation or extrapolation.

Since multistep methods make use of previous stored values of the solution, they are not self-starting and require other methods such as Runge–Kutta algorithms to generate the first few values. Furthermore, since the previous solutions are stored at equi-spaced intervals, if the step-size is subsequently changed, new stored values must be generated by interpolation/extrapolation based on the new step-size.

3.3.1 Predictor-Corrector Methods

The idea behind predictor-corrector methods is to utilise two linear multistep algorithms of the form of Eq. 3.41, one corresponding to the *predictor*, the other to the *corrector*. The predictor and corrector employ polynomial extrapolation/interpolation to estimate the subsequent step value, with the difference in values between the two multistep algorithms providing an estimate of the local truncation error, which can be used to adjust the step-size. We illustrate the approach in detail using the *Adams–Bashforth–Moulton* predictor-corrector scheme, which employs a cubic polynomial extrapolation for the predictor, followed by another cubic polynomial interpolation for the corrector.

Consider the general ODE

$$\frac{dy}{dt} = f(t, y) \quad (3.48)$$

we can use the fundamental theorem of calculus to determine the value of y at the next step, t_n , according to:

$$y_n = y_{n-1} + \int_{t_{n-1}}^{t_n} f(t, y) dt \quad (3.49)$$

To evaluate the integral, we use Newton's backward difference polynomial to determine a cubic polynomial that interpolates the previous four step-values of f . Denoting these values by f_{n-1} , f_{n-2} , f_{n-3} and f_{n-4} , the required predictor cubic interpolating polynomial, $F_p(t)$, approximating $f(t, y)$ from Eq. 3.42 is:

$$\begin{aligned} F_p(t) = \nabla^0 f_{n-1} + \frac{(t - t_{n-1})}{h} \nabla^1 f_{n-1} + \frac{(t - t_{n-1})(t - t_{n-2})}{2! h^2} \nabla^2 f_{n-1} \\ + \frac{(t - t_{n-1})(t - t_{n-2})(t - t_{n-3})}{3! h^3} \nabla^3 f_{n-1} \end{aligned} \quad (3.50)$$

We can evaluate the backward differences about f_{n-1} as follows:

$$\begin{aligned} \nabla^0 f_{n-1} &= f_{n-1} \\ \nabla^1 f_{n-1} &= f_{n-1}^0 - f_{n-2}^0 = f_{n-1} - f_{n-2} \\ \nabla^2 f_{n-1} &= f_{n-1}^1 - f_{n-2}^1 = (f_{n-1} - f_{n-2}) - (f_{n-2} - f_{n-3}) \\ &= f_{n-1} - 2f_{n-2} + f_{n-3} \\ \nabla^3 f_{n-1} &= f_{n-1}^2 - f_{n-2}^2 = (f_{n-1} - 2f_{n-2} + f_{n-3}) - (f_{n-2} - 2f_{n-3} + f_{n-4}) \\ &= f_{n-1} - 3f_{n-2} + 3f_{n-3} - f_{n-4} \end{aligned}$$

Substituting these into Eq. 3.50, we obtain

$$\begin{aligned} F_p(t) = f_{n-1} + \left[\frac{f_{n-1} - f_{n-2}}{h} \right] (t - t_{n-1}) \\ + \left[\frac{f_{n-1} - 2f_{n-2} + f_{n-3}}{2h^2} \right] (t - t_{n-1})(t - t_{n-2}) \\ + \left[\frac{f_{n-1} - 3f_{n-2} + 3f_{n-3} - f_{n-4}}{6h^3} \right] (t - t_{n-1})(t - t_{n-2})(t - t_{n-3}) \end{aligned}$$

We now substitute this cubic polynomial approximation to $f(t, y)$ in Eq. 3.49, to obtain:

$$\begin{aligned}
 y_n &= y_{n-1} + \int_{t_{n-1}}^{t_n} F_p(t) dt \\
 &= y_{n-1} + f_{n-1} I_0 + \left[\frac{f_{n-1} - f_{n-2}}{h} \right] I_1 + \left[\frac{f_{n-1} - 2f_{n-2} + f_{n-3}}{2h^2} \right] I_2 \\
 &\quad + \left[\frac{f_{n-1} - 3f_{n-2} + 3f_{n-3} - f_{n-4}}{6h^3} \right] I_3
 \end{aligned} \tag{3.51}$$

where

$$I_0 = \int_{t_{n-1}}^{t_n} dt = h$$

$$I_1 = \int_{t_{n-1}}^{t_n} (t - t_{n-1}) dt = \left[\frac{(t - t_{n-1})^2}{2} \right]_{t_{n-1}}^{t_n} = \frac{h^2}{2}$$

$$\begin{aligned}
 I_2 &= \int_{t_{n-1}}^{t_n} (t - t_{n-1})(t - t_{n-2}) dt \\
 &= \left[(t - t_{n-2}) \int (t - t_{n-1}) dt \right]_{t_{n-1}}^{t_n} - \int_{t_{n-1}}^{t_n} 1 \cdot \int (t - t_{n-1}) dt dt
 \end{aligned}$$

(using integration by parts)

$$= \left[\frac{(t - t_{n-2})(t - t_{n-1})^2}{2} \right]_{t_{n-1}}^{t_n} - \left[\frac{(t - t_{n-1})^3}{6} \right]_{t_{n-1}}^{t_n}$$

$$= \frac{5h^3}{6}$$

$$\begin{aligned}
 I_3 &= \int_{t_{n-1}}^{t_n} (t - t_{n-1})(t - t_{n-2})(t - t_{n-3}) dt \\
 &= \left[(t - t_{n-3}) \int (t - t_{n-1})(t - t_{n-2}) dt \right]_{t_{n-1}}^{t_n} - \int_{t_{n-1}}^{t_n} 1 \cdot \int (t - t_{n-1})(t - t_{n-2}) dt dt
 \end{aligned}$$

$$= \left[\frac{(t - t_{n-3})(t - t_{n-2})(t - t_{n-1})^2}{2} - \frac{(t - t_{n-3})(t - t_{n-1})^3}{6} \right]_{t_{n-1}}^{t_n}$$

$$- \int_{t_{n-1}}^{t_n} \left(\frac{(t - t_{n-2})(t - t_{n-1})^2}{2} - \frac{(t - t_{n-1})^3}{6} \right) dt$$

$$= \left(\frac{3h \cdot 2h \cdot h^2}{2} - \frac{3h \cdot h^3}{6} \right) - \left[(t - t_{n-2}) \int \frac{(t - t_{n-1})^2}{2} \right]_{t_{n-1}}^{t_n}$$

$$+ \int_{t_{n-1}}^{t_n} 1 \cdot \int \frac{(t - t_{n-1})^2}{2} dt dt + \int_{t_{n-1}}^{t_n} \frac{(t - t_{n-1})^3}{6} dt$$

$$\begin{aligned}
&= \frac{5h^4}{2} - \left[\frac{(t-t_{n-2})(t-t_{n-1})^3}{6} \right]_{t_{n-1}}^{t_n} + \left[\frac{(t-t_{n-1})^4}{24} \right]_{t_{n-1}}^{t_n} + \left[\frac{(t-t_{n-1})^4}{24} \right]_{t_{n-1}}^{t_n} \\
&= \frac{5h^4}{2} - \frac{h^4}{3} + \frac{h^4}{24} + \frac{h^4}{24} \\
&= \frac{9h^4}{4}
\end{aligned}$$

Substituting these integral evaluations into Eq. 3.51, we obtain:

$$\begin{aligned}
y_n &= y_{n-1} + hf_{n-1} + h \left[\frac{f_{n-1} - f_{n-2}}{2} \right] + 5h \left[\frac{f_{n-1} - 2f_{n-2} + f_{n-3}}{12} \right] \\
&\quad + 9h \left[\frac{f_{n-1} - 3f_{n-2} + 3f_{n-3} - f_{n-4}}{24} \right] \\
&= y_{n-1} + \frac{h}{24} [55f_{n-1} - 59f_{n-2} + 37f_{n-3} - 9f_{n-4}] \tag{3.52}
\end{aligned}$$

Equation 3.52 is known as the fourth-order *Adams–Bashforth* predictor. To determine its truncation error, ε_p , we note that the error in the polynomial approximation $F_p(t)$ at each point $t \in [t_{n-4}, t_n]$ can be determined from Eq. 3.47 as

$$E(t) = \frac{f^{(4)}(\xi)}{4!} (t-t_{n-1})(t-t_{n-2})(t-t_{n-3})(t-t_{n-4}) \tag{3.53}$$

for some $\xi \in [t_{n-4}, t_n]$. This error can then be incorporated into the integral of $F_p(t)$ in Eq. 3.51 to obtain:

$$\begin{aligned}
\varepsilon_p &= \int_{t_{n-1}}^{t_n} E(t) dt \\
&= \int_{t_{n-1}}^{t_n} \frac{f^{(4)}(\xi)}{4!} (t-t_{n-1})(t-t_{n-2})(t-t_{n-3})(t-t_{n-4}) dt
\end{aligned}$$

where ξ is a function of t . Denoting the maximum and minimum values of $\frac{1}{4!} f^{(4)}(\xi)$ in the interval $[t_{n-4}, t_n]$ by M and m respectively, we have:

$$m \int_{t_{n-1}}^{t_n} p(t) dt \leq \varepsilon_p \leq M \int_{t_{n-1}}^{t_n} p(t) dt$$

where $p(t) = (t-t_{n-1})(t-t_{n-2})(t-t_{n-3})(t-t_{n-4})$. Since we assume $\frac{1}{4!} f^{(4)}(t)$ to be continuous, taking all values between m and M , then there must be a value of $t \in [t_{n-4}, t_n]$, say ξ_1 , for which

$$\varepsilon_p = \int_{t_{n-1}}^{t_n} \frac{f^{(4)}(\xi)}{4!} p(t) dt = \frac{f^{(4)}(\xi_1)}{4!} \int_{t_{n-1}}^{t_n} p(t) dt$$

Evaluating the definite integral of $p(t)$ above is a tedious calculation which can be carried out by hand, or through the use of symbolic computation software,⁵ to obtain:

$$\int_{t_{n-1}}^{t_n} p(t) dt = \int_{t_{n-1}}^{t_n} (t - t_{n-1})(t - t_{n-2})(t - t_{n-3})(t - t_{n-4}) dt = \frac{251h^5}{30}$$

Hence,

$$\begin{aligned} \varepsilon_p &= \frac{f^{(4)}(\xi_1)}{4!} \frac{251h^5}{30} = \frac{251}{720} f^{(4)}(\xi_1)h^5 \\ &= \frac{251}{720} y^{(5)}(\xi_1)h^5, \quad \xi_1 \in [t_{n-4}, t_n] \end{aligned} \tag{3.54}$$

represents the local truncation error for the Adams–Bashforth predictor.

A similar method to the above can be used to derive an alternative cubic polynomial that will be used as the corrector. In this case, we shift the interpolation points one-step forwards to the four points $t_{n-3}, t_{n-2}, t_{n-1}$ and t_n . Referring to the corresponding values of $f(t, y)$ as $f_{n-3}, f_{n-2}, f_{n-1}$ and f_n , the required polynomial from Eq. 3.42 is

$$\begin{aligned} F_c(t) &= \nabla^0 f_n + \frac{(t - t_n)}{h} \nabla^1 f_n + \frac{(t - t_n)(t - t_{n-1})}{2! h^2} \nabla^2 f_n \\ &\quad + \frac{(t - t_n)(t - t_{n-1})(t - t_{n-2})}{3! h^3} \nabla^3 f_n \\ &= f_n + \left[\frac{f_n - f_{n-1}}{h} \right] (t - t_n) + \left[\frac{f_n - 2f_{n-1} + f_{n-2}}{2h^2} \right] (t - t_n)(t - t_{n-1}) \\ &\quad + \left[\frac{f_n - 3f_{n-1} + 3f_{n-2} - f_{n-3}}{6h^3} \right] (t - t_n)(t - t_{n-1})(t - t_{n-2}) \end{aligned}$$

Substituting this polynomial as an approximation to $f(t, y)$ in the integral of Eq. 3.49, we obtain:

⁵If you have installed Matlab’s Symbolic Math Toolbox, the following commands can be entered from the command prompt to readily evaluate this integral:

```
syms t1 t h
P = (t-t1) * (t - (t1-h)) * (t - (t1-2*h)) * (t - (t1-3*h)) ;
I4 = int(P, t, t1, t1+h) ;
simplify(I4)
```

where `syms` declares the symbolic variables `t1`, `t` and `h`, where `t1` denotes t_{n-1} . The command `int` symbolically evaluates the definite integral, and `simplify` simplifies the resulting expression to obtain the required answer.

$$\begin{aligned}
y_n &= y_{n-1} + \int_{t_{n-1}}^{t_n} F_c(t) dt \\
&= y_{n-1} + f_n J_0 + \left[\frac{f_n - f_{n-1}}{h} \right] J_1 + \left[\frac{f_n - 2f_{n-1} + f_{n-2}}{2h^2} \right] J_2 \\
&\quad + \left[\frac{f_n - 3f_{n-1} + 3f_{n-2} - f_{n-3}}{6h^3} \right] J_3
\end{aligned}$$

where

$$\begin{aligned}
J_0 &= \int_{t_{n-1}}^{t_n} dt = h \\
J_1 &= \int_{t_{n-1}}^{t_n} (t - t_n) dt = -\frac{h^2}{2} \\
J_2 &= \int_{t_{n-1}}^{t_n} (t - t_n)(t - t_{n-1}) dt = -\frac{h^3}{6} \\
J_3 &= \int_{t_{n-1}}^{t_n} (t - t_n)(t - t_{n-1})(t - t_{n-2}) dt = -\frac{h^4}{4}
\end{aligned}$$

and hence,

$$\begin{aligned}
y_n &= y_{n-1} + hf_n - \frac{h^2}{2} \left[\frac{f_n - f_{n-1}}{h} \right] - \frac{h^3}{6} \left[\frac{f_n - 2f_{n-1} + f_{n-2}}{2h^2} \right] \\
&\quad - \frac{h^4}{4} \left[\frac{f_n - 3f_{n-1} + 3f_{n-2} - f_{n-3}}{6h^3} \right] \\
&= y_{n-1} + \frac{h}{24} [9f_n + 19f_{n-1} - 5f_{n-2} + f_{n-3}] \tag{3.55}
\end{aligned}$$

which is known as the fourth-order *Adams–Moulton* corrector. Note that unlike the corresponding explicit Adams–Bashforth predictor, the corrector is implicit. Its truncation error, ε_c , can be determined using similar principles to those used to derive the predictor error (Eq. 3.54): namely by integrating the local error of the interpolating polynomial. This interpolation error is given from Eq. 3.47 as

$$E(t) = \frac{f^{(4)}(\xi)}{4!} (t - t_n)(t - t_{n-1})(t - t_{n-2})(t - t_{n-3})$$

for some $\xi \in [t_{n-3}, t_n]$. We can then integrate this error term to obtain:

$$\begin{aligned}
\varepsilon_c &= \int_{t_{n-1}}^{t_n} \frac{f^{(4)}(\xi)}{4!} (t - t_n)(t - t_{n-1})(t - t_{n-2})(t - t_{n-3}) dt \\
&= \frac{f^{(4)}(\xi_2)}{4!} \int_{t_{n-1}}^{t_n} (t - t_n)(t - t_{n-1})(t - t_{n-2})(t - t_{n-3}) dt \quad (\xi_2 \in [t_{n-3}, t_n])
\end{aligned}$$

$$\begin{aligned}
&= \frac{f^{(4)}(\xi_2)}{4!} \left(-\frac{19h^5}{30} \right) \\
&= -\frac{19}{720} y^{(5)}(\xi_2) h^5 \tag{3.56}
\end{aligned}$$

We can now combine the predictor-corrector pair (Eqs. 3.52 and 3.55) to form *Milne's method* as follows:

$$\begin{aligned}
y_p &= y_{n-1} + \frac{h}{24} [55f_{n-1} - 59f_{n-2} + 37f_{n-3} - 9f_{n-4}] + \frac{251}{720} y^{(5)}(\xi_1) h^5 \\
y_c &= y_{n-1} + \frac{h}{24} [9f(t_n, y_p) + 19f_{n-1} - 5f_{n-2} + f_{n-3}] - \frac{19}{720} y^{(5)}(\xi_2) h^5
\end{aligned}$$

where y_p and y_c are the predictor and corrector estimates of y at the next step t_n . Note that the f_n term in the corrector (Eq. 3.55) has been estimated from the predicted value y_p as $f(t_n, y_p)$. For small time-steps h , we can assume that $\xi_1 \approx \xi_2$, and denoting the exact value of y_n by y , we obtain the following error estimates for the predictor and corrector:

$$\begin{aligned}
y_p - y &= \frac{251}{720} y^{(5)}(\xi) h^5 \\
y_c - y &= -\frac{19}{720} y^{(5)}(\xi) h^5
\end{aligned}$$

where $\xi \in [t_{n-4}, t_n]$. Subtracting these error terms, we obtain:

$$\begin{aligned}
y_p - y_c &= \frac{270}{720} y^{(5)}(\xi) h^5 \\
&= -\frac{270}{19} (y_c - y)
\end{aligned}$$

And hence,

$$l_n = |y_c - y| = \frac{19}{270} |y_p - y_c| \tag{3.57}$$

is an estimate of the local error, which can be used to adjust the step-size similar to that of Eq. 3.25.

Tables 3.1 and 3.2 display the coefficients and truncation errors for the Adams–Bashforth and Adams–Moulton families of predictors and correctors up to order 6.

Table 3.1 Coefficients (Eq. 3.41) of Adams–Bashforth predictors up to order 6. Note that for the error term, $\xi \in [t_{n-k}, t_n]$, where k is the order

| Order | β_1 | β_2 | β_3 | β_4 | β_5 | β_6 | Error term |
|-------|---------------------|----------------------|---------------------|----------------------|---------------------|---------------------|--------------------------------------|
| 1 | 1 | | | | | | $\frac{1}{2}y''(\xi)h^2$ |
| 2 | $\frac{3}{2}$ | $-\frac{1}{2}$ | | | | | $\frac{5}{12}y'''(\xi)h^3$ |
| 3 | $\frac{23}{12}$ | $-\frac{16}{12}$ | $\frac{5}{12}$ | | | | $\frac{3}{8}y^{(4)}(\xi)h^4$ |
| 4 | $\frac{55}{24}$ | $-\frac{59}{24}$ | $\frac{37}{24}$ | $-\frac{9}{24}$ | | | $\frac{251}{720}y^{(5)}(\xi)h^5$ |
| 5 | $\frac{1901}{720}$ | $-\frac{2774}{720}$ | $\frac{2616}{720}$ | $-\frac{1274}{720}$ | $\frac{251}{720}$ | | $\frac{95}{288}y^{(6)}(\xi)h^6$ |
| 6 | $\frac{4277}{1440}$ | $-\frac{7923}{1440}$ | $\frac{9982}{1440}$ | $-\frac{7298}{1440}$ | $\frac{2877}{1440}$ | $-\frac{475}{1440}$ | $\frac{19087}{60480}y^{(7)}(\xi)h^7$ |

Table 3.2 Coefficients (Eq. 3.41) of Adams–Moulton correctors up to order 6. For the error, $\xi \in [t_{n-k}, t_n]$, where k is the order

| Order | β_1 | β_2 | β_3 | β_4 | β_5 | β_6 | Error term |
|-------|--------------------|---------------------|---------------------|--------------------|---------------------|-------------------|-------------------------------------|
| 1 | 1 | | | | | | $-\frac{1}{2}y''(\xi)h^2$ |
| 2 | $\frac{1}{2}$ | $\frac{1}{2}$ | | | | | $-\frac{1}{12}y'''(\xi)h^3$ |
| 3 | $\frac{5}{12}$ | $\frac{8}{12}$ | $-\frac{1}{12}$ | | | | $-\frac{1}{24}y^{(4)}(\xi)h^4$ |
| 4 | $\frac{9}{24}$ | $\frac{19}{24}$ | $-\frac{5}{24}$ | $\frac{1}{24}$ | | | $-\frac{19}{720}y^{(5)}(\xi)h^5$ |
| 5 | $\frac{251}{720}$ | $\frac{646}{720}$ | $-\frac{264}{720}$ | $\frac{106}{720}$ | $-\frac{19}{720}$ | | $-\frac{3}{160}y^{(6)}(\xi)h^6$ |
| 6 | $\frac{475}{1440}$ | $\frac{1427}{1440}$ | $-\frac{798}{1440}$ | $\frac{482}{1440}$ | $-\frac{173}{1440}$ | $\frac{27}{1440}$ | $-\frac{863}{60480}y^{(7)}(\xi)h^7$ |

3.3.2 Backward Differentiation Formulas

Another family of linear multistep methods, first made popular by C.W. Gear [8], are those based on *Backward Differentiation Formula* (BDF) methods. For the ODE

$$\frac{dy}{dt} = f(t, y)$$

BDF methods utilise Newton’s backward difference polynomial, Eq. 3.42, to obtain an estimate for the next step y_n from the interpolating polynomial $\phi(t)$ of degree k passing through y_n and the previous k step solutions y_{n-1}, \dots, y_{n-k} , then differentiating this polynomial to obtain:

$$\begin{aligned} \left. \frac{d\phi(t)}{dt} \right|_{t=t_n} &= \frac{d}{dt} \left[\nabla^0 y_n + \frac{(t-t_n)}{h} \nabla^1 y_n + \frac{(t-t_n)(t-t_{n-1})}{2! h^2} \nabla^2 y_n + \dots \right. \\ &\quad \left. + \frac{(t-t_n)(t-t_{n-1}) \dots (t-t_{n-k+1})}{k! h^k} \nabla^k y_n \right]_{t=t_n} \end{aligned}$$

$$\begin{aligned}
 &= \left[\frac{1}{h} \nabla^1 y_n + \frac{(t - t_{n-1})}{2! h^2} \nabla^2 y_n + \dots \right. \\
 &\quad \left. + \frac{(t - t_{n-1}) \dots (t - t_{n-k+1})}{k! h^k} \nabla^k y_n \right]_{t=t_n} \\
 &= \frac{1}{h} \nabla^1 y_n + \frac{h}{2! h^2} \nabla^2 y_n + \dots + \frac{h \dots (k-1)h}{k! h^k} \nabla^k y_n \\
 &= \frac{1}{h} \left[\nabla^1 y_n + \frac{\nabla^2 y_n}{2} + \dots + \frac{\nabla^k y_n}{k} \right]
 \end{aligned}$$

And since

$$\left. \frac{d\phi(t)}{dt} \right|_{t=t_n} \approx \left. \frac{dy}{dt} \right|_{t=t_n} = f(t_n, y_n)$$

the k -step BDF is obtained:

$$\frac{1}{h} \left[\nabla^1 y_n + \frac{\nabla^2 y_n}{2} + \dots + \frac{\nabla^k y_n}{k} \right] = f(t_n, y_n)$$

or

$$\sum_{i=1}^k \frac{1}{i} \nabla^i y_n = hf(t_n, y_n) \tag{3.58}$$

Defining the truncation error ε_n by

$$\varepsilon_n = \sum_{i=1}^k \frac{1}{i} \nabla^i y_n - hf(t_n, y_n)$$

the leading term may be conveniently written as

$$\varepsilon_n = \frac{1}{k+1} \nabla^{k+1} y_n \tag{3.59}$$

which represents the next term that would be added to the sum after $i = k$. When the backward differences in Eq.3.58 are evaluated for $k = 1 \dots 5$, the resulting BDF multistep coefficients (Eq.3.41), normalized such that $\alpha_0 = 1$, are given in Table3.3, where the number of steps k is equal to the order of the method. Since Eq.3.58 contains y_n on both sides, BDF methods are implicit and have excellent stability properties. They are therefore popular for stiff ODE systems.

To solve Eq.3.58, Newton’s method is used with an initial guess determined from the predicted value

$$y_n^p = \sum_{i=0}^k \nabla^i y_{n-1} \tag{3.60}$$

Table 3.3 Multistep coefficients of BDF methods up to order 5 (see Eq. 3.41)

| Order | β_0 | α_0 | α_1 | α_2 | α_3 | α_4 | α_5 |
|-------|------------------|------------|--------------------|-------------------|--------------------|------------------|-------------------|
| 1 | 1 | 1 | -1 | | | | |
| 2 | $\frac{2}{3}$ | 1 | $-\frac{4}{3}$ | $\frac{1}{3}$ | | | |
| 3 | $\frac{6}{11}$ | 1 | $-\frac{18}{11}$ | $\frac{9}{11}$ | $-\frac{2}{11}$ | | |
| 4 | $\frac{12}{25}$ | 1 | $-\frac{48}{25}$ | $\frac{36}{25}$ | $-\frac{16}{25}$ | $\frac{3}{25}$ | |
| 5 | $\frac{60}{137}$ | 1 | $-\frac{300}{137}$ | $\frac{300}{137}$ | $-\frac{200}{137}$ | $\frac{75}{137}$ | $-\frac{12}{137}$ |

This predictor leads to a useful estimate of the truncation error. Writing

$$y_n - y_n^p = y_n - \sum_{i=0}^k \nabla^i y_{n-1}$$

and noting from Eq. 3.43 that

$$\begin{aligned} \nabla^{i+1} y_n &= \nabla^i y_n - \nabla^i y_{n-1} \\ \therefore \nabla^i y_{n-1} &= \nabla^i y_n - \nabla^{i+1} y_n \end{aligned}$$

we have

$$\begin{aligned} y_n - y_n^p &= y_n - \sum_{i=0}^k (\nabla^i y_n - \nabla^{i+1} y_n) \\ &= y_n - \sum_{i=0}^k \nabla^i y_n + \sum_{i=0}^k \nabla^{i+1} y_n \\ &= -\sum_{i=1}^k \nabla^i y_n + \sum_{i=0}^k \nabla^{i+1} y_n \quad (\text{since } \nabla^0 y_n = y_n) \\ &= \nabla^{k+1} y_n \end{aligned} \tag{3.61}$$

And using Eq. 3.59, we obtain

$$\varepsilon_n = \frac{1}{k+1} (y_n - y_n^p) \tag{3.62}$$

This estimate of the error can be used to adjust the step-size in a manner similar to Eq. 3.25. Furthermore, typical BDF codes also adjust the order in addition to the step-size. One order-selection strategy is to simply maintain, increment or decrement the order by one, depending on which order yields the maximum step-size. BDF

order-varying methods begin with an order of 1 (which is self-starting), and then increase the order as required. COMSOL implements the BDF method by default in its time-dependent solver, with orders varying between 1–5. If desired, a maximum order less than 5 can also be specified.

3.3.3 Numerical Differentiation Formulas

A variant of BDF methods is the family of *numerical differentiation formulas*, or NDFs, whereby Eq. 3.58 is modified to:

$$\left(\sum_{i=1}^k \frac{1}{i} \nabla^i y_n \right) - \kappa \gamma_k (y_n - y_n^p) = hf(t_n, y_n) \tag{3.63}$$

where κ is a parameter, $\gamma_k = \sum_{i=1}^k (1/i)$ and y_n^p is the predicted value given by Eq. 3.60. Substituting this predicted value into Eq. 3.63, and making use of Eq. 3.61, we obtain the equivalent NDF form:

$$\left(\sum_{i=1}^k \frac{1}{i} \nabla^i y_n \right) - \kappa \gamma_k \nabla^{k+1} y_n = hf(t_n, y_n) \tag{3.64}$$

The truncation error of Eq. 3.64 is given by

$$\varepsilon_n = \left(\sum_{i=1}^k \frac{1}{i} \nabla^i y_n \right) - \kappa \gamma_k \nabla^{k+1} y_n - hf(t_n, y_n)$$

The leading term of this error may be estimated from the increment in its value when i is increased from k to $k + 1$:

$$\begin{aligned} \varepsilon_n &= \frac{1}{k+1} \nabla^{k+1} y_n - \kappa (\gamma_{k+1} \nabla^{k+2} y_n - \gamma_k \nabla^{k+1} y_n) \\ &= \left(\kappa \gamma_k + \frac{1}{k+1} \right) \nabla^{k+1} y_n - \kappa \gamma_{k+1} \nabla^{k+2} y_n \\ &\approx \left(\kappa \gamma_k + \frac{1}{k+1} \right) \nabla^{k+1} y_n \end{aligned} \tag{3.65}$$

Hence, it is possible to choose a negative value of parameter κ to reduce the truncation error below that of the corresponding BDF (Eq. 3.62). Unfortunately, such a reduction in truncation error will compromise the stability of the method. Shampine and Reichelt [18] report values of κ for NDF orders from 1 to 5 numerically-determined to achieve a balance between good reduction in the truncation error with only slight

Table 3.4 Multistep coefficients of NDF methods up to order 5 (see Eq. 3.41), scaled such that $\alpha_0 = 1$. The κ coefficients in the final column have been incorporated into the multistep coefficients of previous columns by expanding Eq. 3.64

| Order | β_0 | α_0 | α_1 | α_2 | α_3 | α_4 | α_5 | κ |
|-------|-----------------------|------------|--------------------------|-------------------------|-------------------------|---------------------|--------------------|----------------|
| 1 | $\frac{200}{237}$ | 1 | $-\frac{274}{237}$ | $\frac{37}{237}$ | | | | -0.185 |
| 2 | $\frac{3}{5}$ | 1 | $-\frac{3}{2}$ | $\frac{3}{5}$ | $-\frac{1}{10}$ | | | $-\frac{1}{9}$ |
| 3 | $\frac{6000}{119053}$ | 1 | $-\frac{216212}{119053}$ | $\frac{144318}{119053}$ | $-\frac{56212}{119053}$ | $\frac{823}{10823}$ | | -0.0823 |
| 4 | $\frac{960}{2083}$ | 1 | $-\frac{4255}{2083}$ | $\frac{3710}{2083}$ | $-\frac{2110}{2083}$ | $\frac{655}{2083}$ | $-\frac{83}{2083}$ | -0.0415 |
| 5 | $\frac{60}{137}$ | 1 | $-\frac{300}{137}$ | $\frac{300}{137}$ | $-\frac{200}{137}$ | $\frac{75}{137}$ | $-\frac{12}{137}$ | 0 |

change in stability. These values are given in Table 3.4, along with the corresponding multistep coefficients determined from the expansion Eq. 3.64. NDF methods form the basis of Matlab’s `ode15s` ODE solver.

3.4 ODE Solver Implementations in Matlab and COMSOL

Matlab and COMSOL provide an extensive range of ODE solvers based on several of the numerical integration algorithms described in this chapter. A summary of ODE solvers used in both Matlab and COMSOL is given in Table 3.5.

For models involving a large number of variables, it is common to express the ODE system in the general form:

$$\mathbf{M}(t, \mathbf{y}) \frac{d\mathbf{y}}{dt} = \mathbf{f}(t, \mathbf{y}), \quad \mathbf{y}(0) = \mathbf{y}_0 \tag{3.66}$$

where \mathbf{M} is referred to as the *mass matrix*,⁶ which may depend on t and \mathbf{y} . Both COMSOL and Matlab allow ODE systems to be expressed in this form. In Matlab, for example, to solve the ODE system:

$$\begin{bmatrix} 1 & -1 \\ t & uv \end{bmatrix} \begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} u - v^2 \\ u + v \end{bmatrix}$$

with $u(0) = v(0) = 1$, we could use the following code to solve the system from $t = 0$ to 1, which uses `odeset` to specify the mass matrix:⁷

⁶For second-order ODE systems of the form $\mathbf{E}\ddot{\mathbf{y}} + \mathbf{M}\dot{\mathbf{y}} + \mathbf{K}\mathbf{y} = \mathbf{F}$, \mathbf{E} is instead referred to as the mass matrix, \mathbf{M} is the damping matrix, and \mathbf{K} is the stiffness matrix.

⁷In recent releases of Matlab, setting the mass matrix via `odeset` requires that function handles are used to specify the user-defined functions, rather than string expressions. Hence for this example, using `options = odeset('Mass', 'my_mass_fun');` will not work.

Table 3.5 Summary of ODE solvers in Matlab and COMSOL. An overview of the Matlab ODE suite may be found in Shampine and Reichelt [18]

| Solver | Description |
|-----------------------|---|
| ode45 | Matlab ODE solver based on the 5(4) Runge–Kutta Dormand–Prince pair [7] (see p. 72) |
| ode23 | Matlab ODE solver based on the 3(2) Runge–Kutta Bogacki–Shampine pair [4] |
| ode113 | Matlab ODE solver based on the Adams–Bashforth–Milne predictor-corrector family (see Sect. 3.3.1) of orders 1–13 |
| ode15s | Matlab ODE solver based of the NDF family of orders 1–5 (see Sect. 3.3.3), suitable for solving stiff systems. It represents the default solver of choice for most ODE systems encountered in bioengineering applications |
| ode23s | Matlab ODE solver based on a modified Rosenbrock second-order algorithm [20], suitable for stiff systems |
| ode23t | Matlab ODE solver based on the trapezoidal method (see Sect. 3.2.2). Error estimation for step-size control is based on the difference between the trapezoidal evaluation and a third-order polynomial interpolant at each step |
| ode23tb | Matlab ODE solver based on a trapezoidal-BDF2 pair [11] |
| BDF | COMSOL time-dependent solver based on the BDF family of orders 1–5 (see Sect. 3.3.2). |
| Generalized- α | COMSOL time-dependent solver based on the generalized- α method (see Sect. 3.2.4). The solver detects which variables have ODE orders of 1 or 2, and applies the appropriate method parameters and algorithm based on this order |
| ode15i | Matlab DAE solver for implicit DAEs in the form $\mathbf{F}(t, \mathbf{y}, \dot{\mathbf{y}}) = \mathbf{0}$ |

```

Y_init = [1, 1];
t_span = [0, 1];
options = odeset('Mass', @my_mass_fun);
[time, Y_out] = ode15s(@my_fun, t_span, Y_init, options);

```

where the user-defined functions `my_mass_fun` and `my_fun`, which define the mass matrix and the right-hand side of the above system respectively, are:

```

function M = my_mass_fun(t,Y)
M = [1, -1; t, Y(1)*Y(2)];

```

and

```

function Y_prime = my_fun(t,Y)
Y_prime = zeros(2,1);
Y_prime(1) = Y(1) - Y(2)^2;
Y_prime(2) = Y(1) + Y(2);

```

If the mass matrix \mathbf{M} is non-singular, left-multiplying both sides of Eq. 3.66 by \mathbf{M}^{-1} will transform Eq. 3.66 to the standard form of Eq. 2.6, which we have used

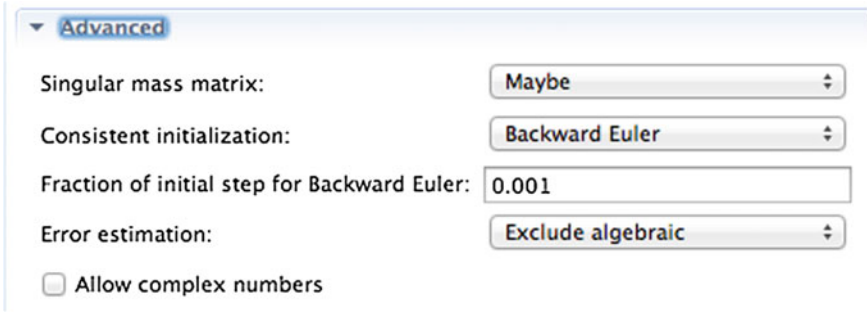


Fig. 3.4 Advanced settings tab of COMSOL’s time-dependent solver. The user can specify a singular mass matrix (default setting is ‘Maybe’), as well as the method of consistent initialization of variables. For the default backward-Euler method of consistent initialization, the user can specify the fraction of the initial solver step to be taken. Finally, the error estimation method for adjusting the solver time-step can be set to include or exclude any algebraic variables present

throughout this chapter. However, if \mathbf{M} is singular, then the ODE system is equivalent to a differential-algebraic equation (DAE) system, which may be expressed as:

$$\frac{dy}{dt} = \mathbf{f}(t, \mathbf{y}, \mathbf{z}) \quad (3.67)$$

$$\mathbf{g}(t, \mathbf{y}, \mathbf{z}) = \mathbf{0} \quad (3.68)$$

where the \mathbf{z} represent purely algebraic variables, that is, variables with no derivatives appearing in the equations. For such DAE systems, care must be taken to specify initial values which are consistent with the solution. For example, the DAE

$$\begin{aligned} \frac{dy}{dt} &= y + z \\ y + z - t &= 0 \end{aligned}$$

must satisfy the initial condition $y(0) + z(0) = 0$ (substituting $t = 0$ into the second equation), so any initial values not satisfying this relationship are not consistent with the DAE. Matlab and COMSOL provide options to allow the user to specify if the mass matrix is singular, and if so, to adjust initial values to ensure they are consistent.⁸ Figure 3.4 shows the advanced settings tab of the time-dependent solver in COMSOL, where the user can specify a singular mass matrix, as well as the method for consistent initialization of variables. By default, COMSOL uses the backward-Euler method

⁸Matlab and COMSOL can readily solve DAEs for which $\frac{\partial \mathbf{g}}{\partial \mathbf{z}}$ in Eq. 3.68 is non-singular, also known as index-1 DAEs [2]. The DAE index is defined as the maximum number of system differentiations required such that for all variables \mathbf{y} , $d\mathbf{y}/dt$ can be uniquely determined from \mathbf{y} and t . For more information on solving index-1 DAEs using the standard Matlab ODE suite, see Shampine et al. [19].

to take a small time-step of 0.1% of the initial solver step (this fraction can also be adjusted), in order to generate a consistent solution using Newton's method. This solution is then taken as the initial value.

3.5 Further Reading

Comprehensive texts covering numerical methods for ODEs are those of Ascher and Petzold [2], Shampine [17], Deuffhard and Bornemann [6] and the two-volume work of Hairer et al. [9, 10]. The finite element method text of Hughes [12] provides an introduction to the generalized trapezoidal family on which COMSOL's generalized- α method is based. Finally, the text of Press et al. [16] provides a general overview of numerical methods for a range of scientific computation tasks, including the numerical integration of ODEs.

Problems

3.1 The Beeler–Reuter [3] model of cardiac ventricular myocyte electrical activity is given by the following system of 8 ODEs:

$$\begin{aligned}\frac{dV_m}{dt} &= -\frac{1}{C_m} [i_{K_1} + i_{x_1} + i_{Na} + i_s - i_{stim}] \\ \frac{d[Ca]_i}{dt} &= -r_{Ca}i_s + k_{up} ([Ca]_{SR} - [Ca]_i) \\ \frac{dx_1}{dt} &= \alpha_{x_1} (1 - x_1) - \beta_{x_1}x_1 \\ \frac{dm}{dt} &= \alpha_m (1 - m) - \beta_m m \\ \frac{dh}{dt} &= \alpha_h (1 - h) - \beta_h h \\ \frac{dj}{dt} &= \alpha_j (1 - j) - \beta_j j \\ \frac{dd}{dt} &= \alpha_d (1 - d) - \beta_d d \\ \frac{df}{dt} &= \alpha_f (1 - f) - \beta_f f\end{aligned}$$

with four membrane ionic currents: a time-independent outward K^+ current (i_{K_1}), a voltage-dependent outward K^+ current (i_{x_1}), an inward Na^+ current (i_{Na}), and a slow inward Ca^{2+} current (i_s), given by the expressions:

$$\begin{aligned}
i_{K_1} &= A_{K_1} \left\{ \frac{4(e^{0.04(V_m+85)} - 1)}{e^{0.08(V_m+53)} + e^{0.04(V_m+53)}} + \frac{0.2(V_m + 23)}{1 - e^{-0.04(V_m+23)}} \right\} \\
i_{x_1} &= A_{x_1} x_1 \frac{e^{0.04(V_m+77)} - 1}{e^{0.04(V_m+35)}} \\
i_{Na} &= (g_{Na} m^3 h j + g_{Na} C) (V_m - V_{Na}) \\
i_s &= g_s d f (V_m - V_{Ca})
\end{aligned}$$

with

$$V_{Ca} = -82.3 - 13.0287 \ln [\text{Ca}]_i$$

where V_m is the transmembrane potential (in mV) and $[\text{Ca}]_i$ denotes the free intracellular Ca^{2+} ion concentration (in M). The remaining variables x_1 , m , h , j , d , f are dimensionless kinetic gating terms whose voltage-dependent forward (α) and reverse (β) rates (in ms^{-1}) are given by:

$$\begin{aligned}
\alpha_{x_1} &= \frac{0.0005e^{0.083(V_m+50)}}{e^{0.057(V_m+50)} + 1} & \beta_{x_1} &= \frac{0.0013e^{-0.06(V_m+20)}}{e^{-0.04(V_m+20)} + 1} \\
\alpha_m &= \frac{-(V_m+47)}{e^{-0.1(V_m+47)} - 1} & \beta_m &= 40e^{-0.056(V_m+72)} \\
\alpha_h &= 0.126e^{-0.25(V_m+77)} & \beta_h &= \frac{1.7}{e^{-0.082(V_m+22.5)} + 1} \\
\alpha_j &= \frac{0.055e^{-0.25(V_m+78)}}{e^{-0.2(V_m+78)} + 1} & \beta_j &= \frac{0.3}{e^{-0.1(V_m+32)} + 1} \\
\alpha_d &= \frac{0.095e^{-0.01(V_m-5)}}{e^{-0.072(V_m-5)} + 1} & \beta_d &= \frac{0.07e^{-0.017(V_m+44)}}{e^{0.05(V_m+44)} + 1} \\
\alpha_f &= \frac{0.012e^{-0.008(V_m+28)}}{e^{0.15(V_m+28)} + 1} & \beta_f &= \frac{0.0065e^{-0.02(V_m+30)}}{e^{-0.2(V_m+30)} + 1}
\end{aligned}$$

Finally, i_{stim} is the applied stimulus current given by

$$i_{stim} = \begin{cases} A_s & t_{on} \leq t < t_{on} + t_{dur} \\ 0 & \text{otherwise} \end{cases}$$

where A_s , t_{on} and t_{dur} represent the stimulus current amplitude, onset time and duration respectively.

Initial variable values at $t = 0$ are $V_m = -83.3$ mV, $[\text{Ca}]_i = 1.87 \times 10^{-7}$ M, $x_1 = 0.1644$, $m = 0.01$, $h = 0.9814$, $j = 0.9673$, $d = 0.0033$ and $f = 0.9884$, with remaining model parameters given below:

| Parameter | Value | Parameter | Value |
|--------------------|---|------------|----------------------------|
| C_m | $1 \mu\text{F cm}^{-2}$ | g_{Na} | 4 ms cm^{-2} |
| r_{Ca} | $1 \times 10^{-7} \text{ M cm}^2 \text{ nC}^{-1}$ | $g_{Na} C$ | 0.003 ms cm^{-2} |
| $[\text{Ca}]_{SR}$ | $1 \times 10^{-7} \text{ M}$ | g_s | 0.09 ms cm^{-2} |
| k_{up} | 0.07 ms^{-1} | A_s | $40 \mu\text{A cm}^{-2}$ |
| A_{K_1} | $0.35 \mu\text{A cm}^{-2}$ | t_{on} | 50 ms |
| A_{x_1} | $0.8 \mu\text{A cm}^{-2}$ | t_{dur} | 1 ms |
| V_{Na} | 50 mV | | |

- (a) Use Matlab's `ode15s` solver to solve and plot for membrane potential V_m against time from $t = 0$ to 500 ms.
- (b) Write your own Matlab code to solve the same model using the forward-Euler method with a fixed step-size of 0.01 ms, plotting on the same graph V_m obtained with both the forward-Euler and `ode15s` methods.
- (c) Verify that when the forward-Euler step-size is increased to 0.03 ms, the method becomes unstable.
- (d) Now write Matlab code to solve this model using the backward-Euler method with fixed step-sizes of 0.01 and 0.1 ms, plotting these solutions with the V_m obtained using `ode15s`. Note that because the backward-Euler is implicit, you will need to implement Newton's method (Eq. 3.12) to iteratively obtain the solution at each step.
- (e) Finally, solve the model using Matlab's in-built ODE solvers `ode15s`, `ode23s`, `ode23t`, `ode23tb`, `ode45`, `ode23` and `ode113`, plotting the V_m obtained for each method on the same plot. Based on the time taken for each of these to solve the model, which solvers would you recommend for this system? HINT: use Matlab's `tic` and `toc` timing commands to determine the computational time taken for each solver.

3.2 A minimal model of neural spiking, known as the $I_{Na,p} + I_K$ model [13] (pronounced persistent sodium plus potassium), is defined by the ODE pair

$$\frac{dV}{dt} = -\frac{1}{C} \left[g_{Na} m_{\infty} (V - E_{Na}) + g_K n (V - E_K) + g_L (V - E_L) - I \right]$$

$$\frac{dn}{dt} = (n_{\infty} - n) / \tau_n$$

with

$$m_{\infty} = \frac{1}{1 + \exp\left[-\frac{V+20}{15}\right]}, \quad n_{\infty} = \frac{1}{1 + \exp\left[-\frac{V+25}{5}\right]}$$

where V represents the neuronal membrane potential, n governs the kinetics of the outward K^+ current, and I is the applied stimulus. Initial values for V and n are -72.9 mV and 0.36 respectively, with all remaining model parameters given below:

| Parameter | Value | Parameter | Value |
|-----------|-------------------------|-----------|--------------------------|
| C | $1 \mu\text{F cm}^{-2}$ | τ_n | 1 ms |
| g_{Na} | 20 ms cm^{-2} | g_L | 8 ms cm^{-2} |
| E_{Na} | 60 mV | E_L | -80 mV |
| g_K | 10 ms cm^{-2} | I | $40 \mu\text{A cm}^{-2}$ |
| E_K | -90 mV | | |

Write custom Matlab code to solve the model and plot V from $t = 0$ to 30 ms for the following two fixed-step methods:

- (a) A third-order Runge–Kutta algorithm with coefficients

$$\begin{array}{c|c} 0 & \\ \frac{1}{2} & \frac{1}{2} \\ \frac{3}{4} & 0 \frac{3}{4} \\ \hline & \frac{2}{9} \frac{1}{3} \frac{4}{9} \end{array}$$

using step-sizes of 0.01 and 0.1 ms. Plot both solutions for V on the same plot, along with the solution obtained from `ode15s`.

(b) The generalized- α method with step size of 0.1 ms and high-frequency damping factors of 1 (no-damping), 0.5 and 0 (maximum damping). As above, plot your solutions on the same plot with the solution obtained from `ode15s`.

3.3 For the BDF family of implicit solvers, Newton’s method is used with an initial estimate for the solution at the next step y_n determined from the predictor of Eq. 3.60:

$$y_n^p = \sum_{i=0}^k \nabla^i y_{n-1}$$

Verify that this predictor is equivalent to extrapolation of a polynomial of degree k passing through the previous $k + 1$ step solutions $y_{n-1}, y_{n-2}, \dots, y_{n-k-1}$.

3.4 Determine the formula for the 7-step BDF method. For the test ODE

$$\frac{dy}{dt} = \lambda y$$

with λ real and negative, numerically estimate the absolute stability constraint on the step-size, namely step-sizes ensuring $|y_n| < |y_{n-1}|$. Note that for this test ODE, BDF methods of orders 1-6 are absolutely stable for all step-sizes, but orders 7 and above have only limited stability and are therefore not used in practice.

References

1. Anton H, Rorres C (1987) Elementary linear algebra with applications. Wiley, New York
2. Ascher UM, Petzold LR (1998) Computer methods for ordinary differential equations and differential-algebraic equations. SIAM, Philadelphia
3. Beeler GW, Reuter H (1977) Reconstruction of the action potential of ventricular myocardial fibres. J Physiol 268:177–210
4. Bogacki P, Shampine LF (1989) A 3(2) pair of Runge-Kutta formulas. Appl Math Lett 2:321–325
5. Chung J, Hulbert GM (1993) A time integration algorithm for structural dynamics with improved numerical dissipation: the generalized-alpha method. Appl Mech 60:371–375
6. Deuffhard P, Bornemann F (2002) Scientific computing with ordinary differential equations. Springer, New York
7. Dormand JR, Prince PJ (1980) A family of embedded Runge-Kutta formulae. J Comp Appl Math 6:19–26

8. Gear CW (1973) Numerical initial value problems in ordinary differential equations. Prentice-Hall, Englewood Cliffs
9. Hairer E, Wanner G (1991) Solving ordinary differential equations II: stiff and differential-algebraic problems. Springer, Berlin
10. Hairer E, Nørsett SP, Wanner G (1987) Solving ordinary differential equations I: nonstiff problems. Springer, Berlin
11. Hosea ME, Shampine LF (1996) Analysis and implementation of TR-BDF2. *Appl Num Math* 20:21–37
12. Hughes TJR (1987) The finite element method: linear static and dynamic finite element analysis. Dover, New York
13. Izhikevich EM (2007) Dynamical systems in neuroscience: the geometry of excitability and bursting. MIT Press, Cambridge
14. Jansen KE, Whiting CH, Hulbert GM (2000) A generalized- α method for integrating the filtered Navier-Stokes equations with a stabilized finite-element method. *Comp Meth Appl Eng* 190:305–319
15. Marsden JE, Tromba AJ (2003) Vector calculus, 5th edn. W H Freeman, New York
16. Press WH, Teukolsky SA, Vetterling WT, Flannery BP (2007) Numerical recipes: the art of scientific computing, 3rd edn. Cambridge University Press, Cambridge
17. Shampine LF (1994) Numerical solution of ordinary differential equations. Chapman and Hall, New York
18. Shampine LF, Reichelt MW (1997) The Matlab ODE suite. *SIAM J Sci Comput* 18:1–22
19. Shampine LF, Reichelt MW, Kierzenka JA (1999) Solving index-1 DAEs in Matlab and Simulink. *SIAM Rev* 41:538–552
20. Zedan H (1987) An AN-stable Rosenbrock-type method for solving stiff differential equations. *Comput Math Appl* 13:611–615

Chapter 4

Distributed Systems Modelling with Partial Differential Equations

Modelling physiological systems and medical devices often requires the formulation of quantities that are distributed over space. Such quantities include, for example, pressure, electric field, drug concentration, mechanical stress, or fluid velocity. Distributed systems models are formulated through the use of *partial differential equations* or PDEs. In this chapter, we will provide an overview of the fundamentals of modelling distributed systems with PDEs, focussing on applications in bioengineering.

4.1 Modelling with PDEs

Fundamental laws of physics, including those which specify conservation of quantities such as mass and charge, as well as macroscopic, continuum approximations of biological tissue behaviour, can all be expressed mathematically through PDEs. This section provides an overview of the building blocks necessary in understanding model formulation with PDEs.

4.1.1 The Gradient

Consider a real-valued scalar function of two spatial variables

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

where x and y represent the 2D coordinates of a point with respect to fixed Cartesian axes and w represents a spatially-varying scalar quantity such as the concentration of a substance, the temperature, or the pressure in a fluid. We wish to describe how this

function changes when the spatial position is varied from (x, y) to $(x + \Delta x, y + \Delta y)$. From the two-variable form of Taylor's theorem (Eq. 3.5), we have:

$$f(x + \Delta x, y + \Delta y) \approx f(x, y) + \Delta x \frac{\partial f}{\partial x} + \Delta y \frac{\partial f}{\partial y}$$

Hence,

$$\begin{aligned} \Delta f &= f(x + \Delta x, y + \Delta y) - f(x, y) \\ &\approx \Delta x \frac{\partial f}{\partial x} + \Delta y \frac{\partial f}{\partial y} \end{aligned}$$

This equation is similar to the expression for the scalar (i.e. dot) product of two vectors, $\mathbf{a} \cdot \mathbf{b} = a_x b_x + a_y b_y$.¹ Hence, we can write

$$\Delta f = \Delta \mathbf{r} \cdot \nabla f \quad (4.1)$$

where

$$\Delta \mathbf{r} \equiv \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}, \quad \nabla f = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix} \quad \text{and} \quad \nabla \equiv \begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{pmatrix},$$

the latter known as the 'nabla'² or more commonly the 'del' operator. A similar treatment can be undertaken for 3D and 1D functions, with the del operator defined analogously as

$$\nabla \equiv \begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \end{pmatrix} \quad (3\text{D}), \quad \text{and} \quad \nabla \equiv \frac{\partial}{\partial x} \quad (1\text{D})$$

where x, y, z are the relevant spatial coordinates. When this operator is applied to a scalar function f , the result will be a vector field ∇f , and is known as the *gradient* of f .

The physical interpretation of the gradient can be seen from Eq. 4.1: for an infinitesimal displacement $\Delta \mathbf{r}$ of fixed-length dr , the change in function value Δf will be a maximum when $\Delta \mathbf{r}$ is parallel to ∇f . Thus, the direction of the gradient of f represents the direction of steepest-slope, and its magnitude represents the change in function value per unit length along its steepest-slope, df/dr . Conversely, when $\Delta \mathbf{r}$ is perpendicular to Δf , there will be no change in the value of f . Hence, this perpendicular direction is tangential to a *contour* of f (in 2D), or tangential to a *level surface* (in 3D) in which the value of f remains constant.

¹Here, the x and y components of \mathbf{a} are referred to as a_x, a_y , and similarly for \mathbf{b} .

² ∇ is pronounced 'del', meaning harp.

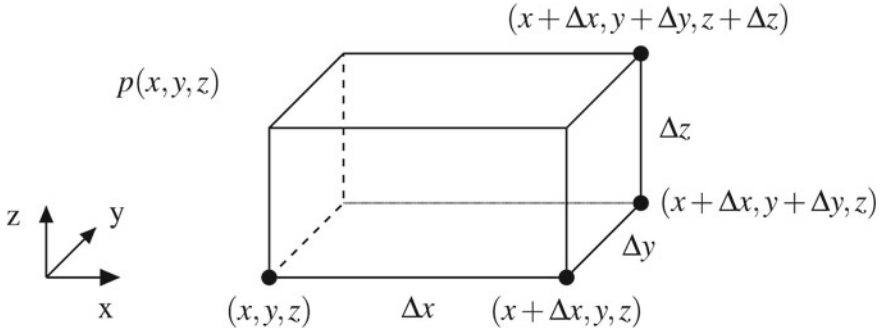


Fig. 4.1 Determining the total force acting on an infinitesimal block in a 3D pressure field $p(x, y, z)$

Example 4.1 Consider an infinitesimal 3D block of sides Δx , Δy , Δz , immersed in a fluid with 3D pressure field given by $p(x, y, z)$, as shown in Fig. 4.1. Find the total force \mathbf{f} acting on the block due to this pressure.

Answer: Along the x -direction, the x -component of the force, f_x , acts normal to the two faces perpendicular to the x -axis. For positive pressure p , the force will be directed inward to the block. Since pressure is equal to the force per unit area, we multiply the pressure by the area of each face to obtain the required component of force:

$$f_x = - [p(x + \Delta x, y, z) - p(x, y, z)] \Delta y \Delta z \approx - \frac{\partial p}{\partial x} \Delta x \Delta y \Delta z$$

Similarly for the y - and z -directions, we obtain:

$$f_y = - [p(x, y + \Delta y, z) - p(x, y, z)] \Delta x \Delta z \approx - \frac{\partial p}{\partial y} \Delta x \Delta y \Delta z$$

$$f_z = - [p(x, y, z + \Delta z) - p(x, y, z)] \Delta x \Delta y \approx - \frac{\partial p}{\partial z} \Delta x \Delta y \Delta z$$

Combining all these components of force, we obtain the total force, \mathbf{f} :

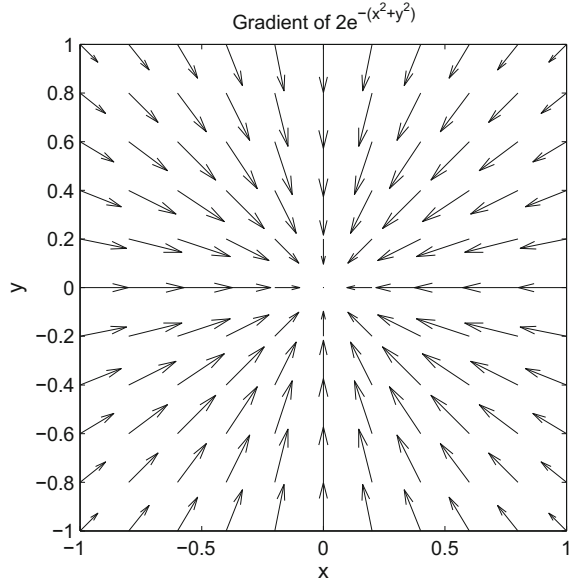
$$\mathbf{f} = -\nabla p \Delta V$$

where $\Delta V = \Delta x \Delta y \Delta z$ is the volume of the block and ∇p is the gradient of the pressure. □

Example 4.2 Find the gradient of the function

$$f(x, y) = 2e^{-(x^2+y^2)}$$

Fig. 4.2 Arrow plot of ∇f , where $f = 2e^{-(x^2+y^2)}$. Each arrow points along the direction of the gradient, with length proportional to the gradient magnitude



Answer: Taking partial derivatives in x and y , we obtain:

$$\begin{aligned}\frac{\partial f}{\partial x} &= -4xe^{-(x^2+y^2)} \\ \frac{\partial f}{\partial y} &= -4ye^{-(x^2+y^2)}\end{aligned}$$

Hence,

$$\nabla f = -2f \begin{pmatrix} x \\ y \end{pmatrix}$$

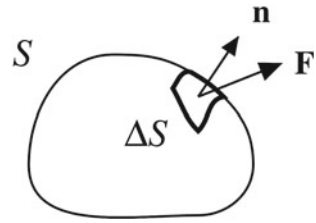
which is shown as an arrow plot in Fig.4.2. □

4.1.2 The Divergence

Consider a vector function $\mathbf{F}(x, y, z)$ which describes the variation of some vector quantity in space. Such a vector could represent for example, fluid velocity, electric field or heat flux. $\mathbf{F} = (F_x, F_y, F_z)^T$ has components in the x -, y -, and z -directions, and is referred to as a *vector field*. Using our previous del operator, we can define the *divergence* of \mathbf{F} as:

$$\nabla \cdot \mathbf{F} = \frac{\partial F_x}{\partial x} + \frac{\partial F_y}{\partial y} + \frac{\partial F_z}{\partial z} \quad (4.2)$$

Fig. 4.3 Vector field \mathbf{F} through an arbitrary closed surface S having infinitesimal element ΔS with outward normal \mathbf{n}



from which it can be seen that the divergence of a vector field will be a scalar quantity.

Consider an arbitrary infinitesimal surface element of area ΔS having unit normal \mathbf{n} . We can define the *flux* of \mathbf{F} through this surface element as

$$\Delta\phi = (\mathbf{F} \cdot \mathbf{n}) \Delta S \tag{4.3}$$

If we now assemble a large number of such infinitesimal elements to form a closed surface, S , as shown in Fig. 4.3, and specifying each \mathbf{n} to be the outward unit normal to the surface, we can define the total outward flux of \mathbf{F} through S as:

$$\phi = \sum_S (\mathbf{v} \cdot \mathbf{F}) \Delta S$$

Taking the limit as $\Delta S \rightarrow 0$, this summation becomes a surface integral:

$$\phi = \int_S (\mathbf{F} \cdot \mathbf{n}) \, dS = \int_S \mathbf{F} \cdot d\mathbf{S}$$

where $d\mathbf{S}$ denotes the infinitesimal surface element vector of magnitude dS and direction \mathbf{n} . It can be shown that the total outward flux of a vector field \mathbf{F} through a closed surface is related to its divergence, as illustrated in the example below.

Example 4.3 Consider a vector field $\mathbf{v}(x, y, z)$ representing the velocity of a fluid at point (x, y, z) . Determine the total volume of fluid per unit time flowing out of an infinitesimal block of sides Δx , Δy , and Δz , as shown in Fig. 4.4.

Answer: For this vector field, the flux given by Eq. 4.3 corresponds to the volume of fluid passing through the surface element ΔS per unit time.³ We denote the x -, y -, and z -components of \mathbf{v} as v_x , v_y and v_z respectively. In the x -direction, the volume of fluid per unit time leaving through each end of the block is $v_x A$, where $A = \Delta x \Delta y$ is the area of each face perpendicular to the x -axis. Hence, the total outward flow (as volume per per unit time) through both ends is

$$v_x(x + \Delta x, y, z) \Delta y \Delta z - v_x(x, y, z) \Delta y \Delta z \approx \left(\frac{\partial v_x}{\partial x} \right) \Delta x \Delta y \Delta z$$

³In this case, flux corresponds to volumetric flow, with SI units of $\text{m}^3 \text{s}^{-1}$.

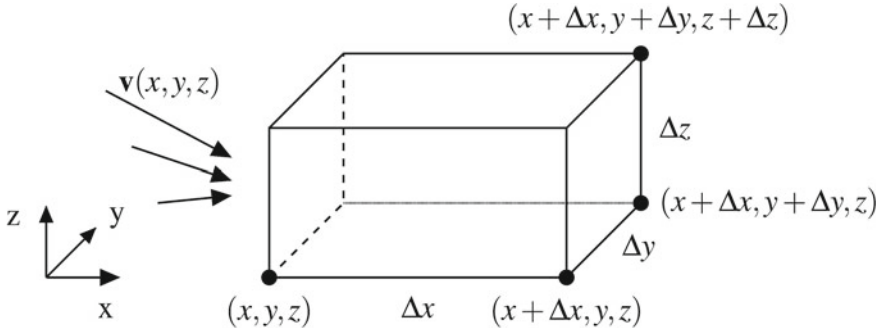


Fig. 4.4 Determining the total volume of fluid per unit time flowing out of an infinitesimal block of sidelengths Δx , Δy , and Δz . The fixed block is immersed in a fluid with velocity field $\mathbf{v}(x, y, z)$

We can obtain similar expressions for the y - and z -directions to obtain the total volume of fluid per unit time, Q , leaving the block as:

$$\begin{aligned}
 Q &= \left(\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z} \right) \Delta x \Delta y \Delta z \\
 &= (\nabla \cdot \mathbf{v}) \Delta V
 \end{aligned}$$

where $\Delta V = \Delta x \Delta y \Delta z$ is the volume of the block. From this expression, we see that the divergence of a vector field corresponds to the total outward flux per unit volume through an infinitesimal block located at that point. We will return to this concept in Sect. 4.1.4 when we introduce the divergence theorem. \square

Another physical interpretation of the divergence can be obtained from considering a fluid velocity field \mathbf{v} , similar to the above Example 4.3. Again we consider an infinitesimal block located at some point, however the vertices of the block are now free to move with the fluid. In a given time interval Δt , each vertex is displaced to a new position by an amount $\mathbf{v} \Delta t$, where \mathbf{v} is the fluid velocity at that point. If we denote the edges of the block by the vectors $\Delta \mathbf{x}$, $\Delta \mathbf{y}$, $\Delta \mathbf{z}$, these edges are re-oriented into the vectors $\Delta \mathbf{x}'$, $\Delta \mathbf{y}'$, $\Delta \mathbf{z}'$ by the action of the fluid, as shown in Fig. 4.5. Denoting the sidelengths of the original edges by Δx , Δy , and Δz , we have:

$$\mathbf{x} = \begin{pmatrix} \Delta x \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} 0 \\ \Delta y \\ 0 \end{pmatrix}, \quad \mathbf{z} = \begin{pmatrix} 0 \\ 0 \\ \Delta z \end{pmatrix}$$

To find \mathbf{x}' , the displacement of each of the two vertices of \mathbf{x} are given by $\mathbf{v}(x, y, z) \Delta t$ and $\mathbf{v}(x + \Delta x, y, z) \Delta t$. \mathbf{x}' can be found from the difference of these displacements, namely:

$$\mathbf{x}' = \mathbf{x} + [\mathbf{v}(x + \Delta x, y, z) - \mathbf{v}(x, y, z)] \Delta t$$

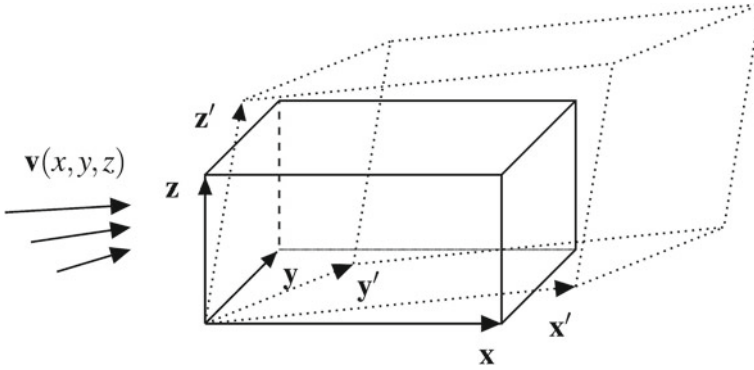


Fig. 4.5 Deformation of an infinitesimal block of fluid due to velocity field $\mathbf{v}(x, y, z)$

Using Taylor’s theorem (3.1) to approximate the components of the velocity field at vertex $(x + \Delta x, y, z)$, we have:

$$v_x(x + \Delta x, y, z) = v_x(x, y, z) + \frac{\partial v_x}{\partial x} \Delta x$$

$$v_y(x + \Delta x, y, z) = v_y(x, y, z) + \frac{\partial v_y}{\partial x} \Delta x$$

$$v_z(x + \Delta x, y, z) = v_z(x, y, z) + \frac{\partial v_z}{\partial x} \Delta x$$

from which we obtain the new edge vector

$$\mathbf{x}' = \begin{pmatrix} \Delta x \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} \frac{\partial v_x}{\partial x} \Delta x \\ \frac{\partial v_y}{\partial x} \Delta x \\ \frac{\partial v_z}{\partial x} \Delta x \end{pmatrix} \Delta t = \begin{pmatrix} 1 + \frac{\partial v_x}{\partial x} \Delta t \\ \frac{\partial v_y}{\partial x} \Delta t \\ \frac{\partial v_z}{\partial x} \Delta t \end{pmatrix} \Delta x$$

A similar derivation leads to the following expressions for \mathbf{y}' and \mathbf{z}' :

$$\mathbf{y}' = \begin{pmatrix} \frac{\partial v_x}{\partial y} \Delta t \\ 1 + \frac{\partial v_y}{\partial y} \Delta t \\ \frac{\partial v_z}{\partial y} \Delta t \end{pmatrix} \Delta y, \quad \mathbf{z}' = \begin{pmatrix} \frac{\partial v_x}{\partial z} \Delta t \\ \frac{\partial v_y}{\partial z} \Delta t \\ 1 + \frac{\partial v_z}{\partial z} \Delta t \end{pmatrix} \Delta z$$

The volume V' of the fluid-deformed block is given by

$$V' = \mathbf{z}' \cdot (\mathbf{x}' \times \mathbf{y}')$$

$$\begin{aligned}
&= \begin{pmatrix} \frac{\partial v_x}{\partial z} \Delta t \\ \frac{\partial v_y}{\partial z} \Delta t \\ 1 + \frac{\partial v_z}{\partial z} \Delta t \end{pmatrix} \Delta z \cdot \begin{pmatrix} \left\{ \frac{\partial v_y}{\partial x} \frac{\partial v_z}{\partial y} \Delta t - \left[1 + \frac{\partial v_x}{\partial y} \Delta t \right] \frac{\partial v_z}{\partial x} \right\} \Delta t \\ \left\{ \frac{\partial v_z}{\partial x} \frac{\partial v_x}{\partial y} \Delta t - \left[1 + \frac{\partial v_x}{\partial x} \Delta t \right] \frac{\partial v_z}{\partial y} \right\} \Delta t \\ \left[1 + \frac{\partial v_x}{\partial x} \Delta t \right] \left[1 + \frac{\partial v_y}{\partial y} \Delta t \right] - \frac{\partial v_y}{\partial x} \frac{\partial v_x}{\partial y} \Delta^2 t \end{pmatrix} \Delta x \Delta y \\
&\approx \left[1 + \left(\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z} \right) \Delta t \right] \Delta x \Delta y \Delta z
\end{aligned}$$

ignoring higher-order terms in Δt . Noting the original volume of the block is $V = \Delta x \Delta y \Delta z$, we obtain the fractional volume expansion

$$\frac{V' - V}{V} = \left(\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z} \right) \Delta t = (\nabla \cdot \mathbf{v}) \Delta t$$

From this expression, we see that the divergence corresponds to the rate of fractional volume expansion of a fluid at a point.

4.1.3 The Curl

The curl of vector field \mathbf{F} is defined by taking the vector cross product of the del operator with $\mathbf{F} = (F_x, F_y, F_z)^T$ to obtain:

$$\nabla \times \mathbf{F} = \begin{pmatrix} \frac{\partial F_z}{\partial y} - \frac{\partial F_y}{\partial z} \\ \frac{\partial F_x}{\partial z} - \frac{\partial F_z}{\partial x} \\ \frac{\partial F_y}{\partial x} - \frac{\partial F_x}{\partial y} \end{pmatrix} \quad (4.4)$$

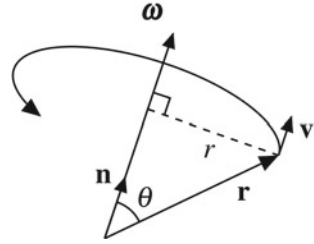
forming a new vector field. Similar to the divergence, we can obtain a physical interpretation of the curl by considering a fluid velocity field $\mathbf{v}(x, y, z)$ such that the fluid is rotating about a fixed axis \mathbf{n} of unit magnitude with angular velocity ω , as shown in Fig. 4.6. Defining $\boldsymbol{\omega} = \omega \mathbf{n}$,⁴ the fluid velocity at any point $\mathbf{r} = (x, y, z)^T$ can be expressed by the vector cross-product:

$$\mathbf{v} = \boldsymbol{\omega} \times \mathbf{r}$$

This can be seen from the fact that the cross-product of two vectors is perpendicular to both, as is the case for the velocity \mathbf{v} which is orthogonal to both \mathbf{r} and $\boldsymbol{\omega}$. Furthermore, if r is the radius of rotation of each point in the fluid (see Fig. 4.6), and θ is the angle between \mathbf{r} and $\boldsymbol{\omega}$, then $r = \|\mathbf{r}\| \cos(\theta)$ and the magnitude of the velocity is given by $v = r\omega = \|\mathbf{r}\| \cos(\theta) \|\boldsymbol{\omega}\| = \|\boldsymbol{\omega} \times \mathbf{r}\|$. Defining the components of $\boldsymbol{\omega}$ as $\boldsymbol{\omega} = (\omega_1, \omega_2, \omega_3)^T$, we therefore have

⁴ $\boldsymbol{\omega}$ is also referred to as the angular velocity vector.

Fig. 4.6 Rotating fluid about around unit axis \mathbf{n} , aligned with vector $\boldsymbol{\omega}$, such that the angular velocity of rotation is $\omega = \|\boldsymbol{\omega}\|$. For each point \mathbf{r} , r is the radius of rotation, and θ is the angle between \mathbf{r} and $\boldsymbol{\omega}$. \mathbf{v} is the velocity of point \mathbf{r}



$$\mathbf{v} = \boldsymbol{\omega} \times \mathbf{r} = \begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{pmatrix} \times \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \omega_2 z - \omega_3 y \\ \omega_3 x - \omega_1 z \\ \omega_1 y - \omega_2 x \end{pmatrix}$$

and taking the curl of \mathbf{v} using Eq. 4.4, we obtain:

$$\nabla \times \mathbf{v} = \begin{pmatrix} \omega_1 - -\omega_1 \\ \omega_2 - -\omega_2 \\ \omega_3 - -\omega_3 \end{pmatrix} = \begin{pmatrix} 2\omega_1 \\ 2\omega_2 \\ 2\omega_3 \end{pmatrix} = 2\boldsymbol{\omega}$$

Hence, the curl of a rotating velocity field is equal to twice the angular velocity vector. The direction of the curl denotes the axis of rotation, and its magnitude is equal to twice the angular velocity.

Example 4.4 Show that the vector field $\mathbf{F} = \nabla\phi$, where $\phi(x, y, z)$ is some scalar function, has a curl which is zero everywhere.⁵

Answer:

$$\nabla \times (\nabla\phi) = \begin{pmatrix} \frac{\partial[\nabla\phi]_z}{\partial y} - \frac{\partial[\nabla\phi]_y}{\partial z} \\ \frac{\partial[\nabla\phi]_x}{\partial z} - \frac{\partial[\nabla\phi]_z}{\partial x} \\ \frac{\partial[\nabla\phi]_y}{\partial x} - \frac{\partial[\nabla\phi]_x}{\partial y} \end{pmatrix} = \begin{pmatrix} \frac{\partial}{\partial y} \left[\frac{\partial\phi}{\partial z} \right] - \frac{\partial}{\partial z} \left[\frac{\partial\phi}{\partial y} \right] \\ \frac{\partial}{\partial z} \left[\frac{\partial\phi}{\partial y} \right] - \frac{\partial}{\partial y} \left[\frac{\partial\phi}{\partial z} \right] \\ \frac{\partial}{\partial x} \left[\frac{\partial\phi}{\partial y} \right] - \frac{\partial}{\partial y} \left[\frac{\partial\phi}{\partial x} \right] \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Hence, the curl of the gradient of a scalar function is equal to zero. □

4.1.4 The Divergence Theorem

An important theorem in formulating partial differential equations characterising physical systems is the *divergence theorem*, also known as Gauss' divergence theorem. The theorem relates the volume integral of the divergence of a vector field to the surface integral of its flux:

⁵Vector fields having a zero curl everywhere are referred to as “curl-free”.

Theorem 4.1 *Let \mathbf{F} be a continuous vector field defined over a closed region in space V with boundary S . Then*

$$\int_V (\nabla \cdot \mathbf{F}) dV = \int_S \mathbf{F} \cdot d\mathbf{S} \quad (4.5)$$

Proof Define \mathbf{F} by $(F_x, F_y, F_z)^T$. Then the left-hand side of Eq. 4.5 expands to

$$\int_V (\nabla \cdot \mathbf{F}) dV = \int_V \frac{\partial F_x}{\partial x} dV + \int_V \frac{\partial F_y}{\partial y} dV + \int_V \frac{\partial F_z}{\partial z} dV$$

Furthermore, by defining the components of the outward surface normal by $\mathbf{n} = (n_x, n_y, n_z)^T$, we can also expand the right-hand side of Eq. 4.5 using

$$\int_S \mathbf{F} \cdot d\mathbf{S} = \int_S (\mathbf{F} \cdot \mathbf{n}) dS = \int_S F_x n_x dS + \int_S F_y n_y dS + \int_S F_z n_z dS$$

Equating these left and right-hand side expansions, we have:

$$\int_V \frac{\partial F_x}{\partial x} dV + \int_V \frac{\partial F_y}{\partial y} dV + \int_V \frac{\partial F_z}{\partial z} dV = \int_S F_x n_x dS + \int_S F_y n_y dS + \int_S F_z n_z dS$$

The theorem will be proven if we can show that

$$\int_V \frac{\partial F_x}{\partial x} dV = \int_S F_x n_x dS \quad (4.6)$$

$$\int_V \frac{\partial F_y}{\partial y} dV = \int_S F_y n_y dS \quad (4.7)$$

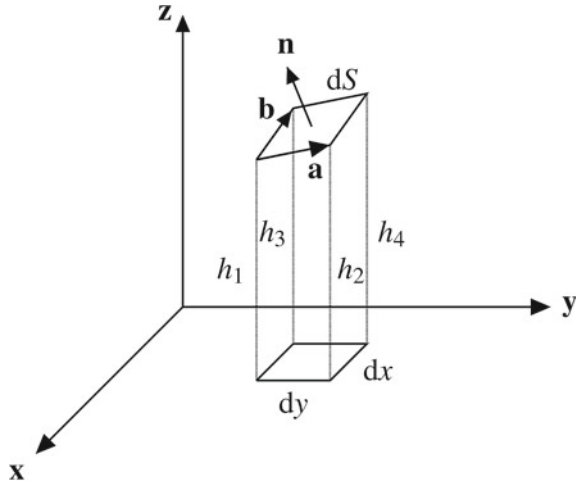
$$\int_V \frac{\partial F_z}{\partial z} dV = \int_S F_z n_z dS \quad (4.8)$$

To do so, we must first establish a useful expression for the area of an infinitesimal surface element having outward normal \mathbf{n} , relative to its projection along a particular coordinate-axis. Consider an infinitesimal area element dS , as shown in Fig. 4.7, with projected area $A = dx dy$ on the xy -plane. The area of the the quadrilateral element dS may be found from the cross-product of vectors \mathbf{a} and \mathbf{b} , denoting two of its adjacent edges (see Fig. 4.7).⁶

$$dS = \|\mathbf{a} \times \mathbf{b}\|$$

⁶The area of a quadrilateral having two adjacent sides given by vectors \mathbf{a} and \mathbf{b} subtending an angle θ can readily be shown to be $\|\mathbf{a}\| \cdot \|\mathbf{b}\| \cos \theta$. This is simply the magnitude of the cross-product $\mathbf{a} \times \mathbf{b}$.

Fig. 4.7 Infinitesimal area element dS with unit normal \mathbf{n} and projected area ($dx dy$) on the xy -plane



Denoting the z -coordinate values of each vertex of dS by h_1 - h_4 (see Fig. 4.7), we have

$$\mathbf{a} \times \mathbf{b} = \begin{pmatrix} 0 \\ dy \\ h_2 - h_1 \end{pmatrix} \times \begin{pmatrix} -dx \\ 0 \\ h_3 - h_1 \end{pmatrix} = \begin{pmatrix} dy(h_3 - h_1) \\ -dx(h_2 - h_1) \\ dx dy \end{pmatrix}$$

The magnitude of this vector equals dS , and since it is perpendicular to both \mathbf{a} and \mathbf{b} , it must be parallel to the surface normal \mathbf{n} . Hence,

$$dS = \sqrt{d^2y(h_3 - h_1)^2 + d^2x(h_2 - h_1)^2 + (dx dy)^2}$$

$$\mathbf{n} = \frac{1}{dS} \begin{pmatrix} dy(h_3 - h_1) \\ -dx(h_2 - h_1) \\ dx dy \end{pmatrix}$$

From the latter expression for the normal, we note from the z -component:

$$n_z = \frac{dx dy}{dS}$$

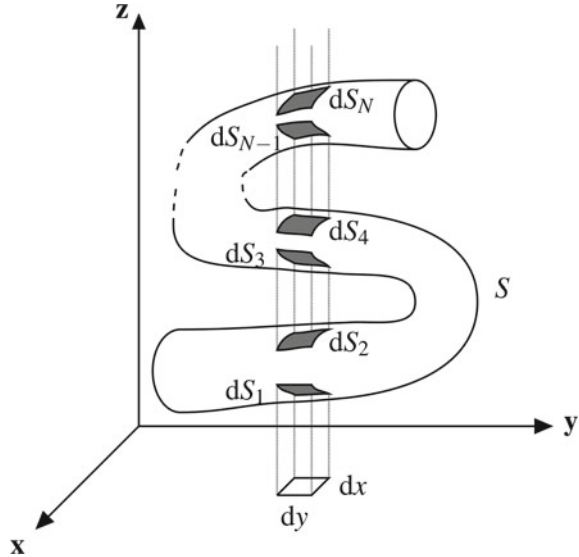
$$\therefore n_z dS = dx dy \tag{4.9}$$

In a similar way, we can show that for the other two projections of dS on the yz - and zx -planes, we have

$$n_x dS = dy dz \quad \text{and} \quad n_y dS = dx dz \tag{4.10}$$

We are now in a position to prove Eqs. 4.6–4.8. Beginning with Eq. 4.8, consider the arbitrary closed surface S shown in Fig. 4.8. An infinitesimal area element in the xy -plane is projected along the z -axis onto the surface, projecting a total of N

Fig. 4.8 Infinitesimal area element, $dx \times dy$, projected along the z-axis onto an arbitrary closed surface S , resulting in N infinitesimal surface elements $dS_1 \cdots dS_N$, where N is even



infinitesimal area elements $dS_1 \cdots dS_N$ onto the surface, with N being even. For such a surface, the left-hand side of Eq. 4.8 may be written as:

$$\begin{aligned} \int_V \frac{\partial F_z}{\partial z} dV &= \iiint_V \left(\frac{\partial F_z}{\partial z} dz \right) dx dy \\ &= \iint_S [F_z(dS_2) - F_z(dS_1)] dx dy + \cdots + \iint_S [F_z(dS_N) - F_z(dS_{N-1})] dx dy \end{aligned}$$

where $F_z(dS_1), F_z(dS_2), \dots, F_z(dS_N)$ denote F_z evaluated at surfaces dS_1, dS_2, \dots, dS_N respectively. Using Eq. 4.9, we have $dx dy = n_z dS$ for each integral above. Furthermore, since n_z alternates in sign from dS_i to $dS_{i+1}, i = 1 \cdots N - 1$, the entire right-hand side of the above simplifies to $\int_S F_z n_z dS$. Hence,

$$\int_V \frac{\partial F_z}{\partial z} dV = \int_S F_z n_z dS$$

and we have proven Eq. 4.8. Using a similar analysis, we can also verify Eqs. 4.6–4.7 using infinitesimal areas $dy dz$ and $dx dz$ projected onto S along the x - and y -axes respectively. Hence, we have proved the divergence theorem. \square

Example 4.5 Consider an incompressible fluid with velocity field $\mathbf{u}(x, y, z)$. Using only the divergence theorem and the principle of mass conservation, derive a PDE for the components of its velocity.

Answer: Consider a hypothetical, fixed closed surface S enclosing a volume V placed anywhere within the fluid. Using the divergence theorem, we have:

$$\int_S \mathbf{u} \cdot d\mathbf{S} = \int_V (\nabla \cdot \mathbf{u}) dV$$

where the left-hand side represents the total flux of fluid (in volume per unit time) flowing out of the surface. Since the fluid is incompressible, its density must remain constant. Therefore, the total mass of fluid within the fixed volume is conserved. As a consequence, the total flux of fluid passing through the closed surface, and therefore the left-hand side of the above equation, must therefore be zero. Hence,

$$\int_V (\nabla \cdot \mathbf{u}) dV = 0$$

And since this is true for any arbitrary volume, the above integrand must identically be equal to zero everywhere. Therefore,

$$\nabla \cdot \mathbf{u} = 0 \tag{4.11}$$

is the required PDE. □

4.1.5 Conservation Law Formulation

The divergence theorem is highly useful in formulating PDEs for a range of physical applications. For conserved quantities such as mass, charge, heat or energy, we can use it to derive a general PDE *conservation law* as follows:

Consider a conserved scalar quantity $u(\mathbf{x}, t)$, where $\mathbf{x} \equiv (x, y, z)^T$. We assume that u is transported through space with a flux density $\mathbf{F}(\mathbf{x}, t)$, expressed in units of amount of u per unit time per unit area perpendicular to the direction of transport. Furthermore, we assume that u can also be produced or annihilated at a rate of $f(\mathbf{x}, t)$ in units of amount of u per unit volume per unit time. Now, the amount of u in a fixed volume V , say $U(t)$ is given by

$$U(t) = \int_V u(\mathbf{x}, t) dV$$

The rate at which U increases in this fixed volume is

$$\frac{dU}{dt} = \int_V \frac{\partial u}{\partial t} dV$$

From the definition of flux density and rate of production, this rate of increase must also be equal to

$$\frac{dU}{dt} = - \int_S \mathbf{F} \cdot d\mathbf{S} + \int_V f dV$$

Equating both of these expressions, we have

$$\int_V \frac{\partial u}{\partial t} dV = - \int_S \mathbf{F} \cdot d\mathbf{S} + \int_V f dV$$

and using the divergence theorem, we obtain:

$$\int_V \frac{\partial u}{\partial t} dV = - \int_V (\nabla \cdot \mathbf{F}) dV + \int_V f dV$$

$$\int_V \left[\frac{\partial u}{\partial t} + \nabla \cdot \mathbf{F} - f \right] dV = 0$$

Since this integral holds for any arbitrary volume V , it follows that the integrand must be equal to zero:

$$\frac{\partial u}{\partial t} + \nabla \cdot \mathbf{F} - f = 0$$

or simply

$$\frac{\partial u}{\partial t} + \nabla \cdot \mathbf{F} = f \quad (4.12)$$

where u is the conserved quantity, \mathbf{F} is the flux density (or simply flux), and f is the source term. In simple terms Eq.4.12 states that the rate of increase of a conserved quantity (i.e. $\frac{\partial u}{\partial t}$) plus the rate of outward flow ($\nabla \cdot \mathbf{F}$) is equal to the rate of production (f).

Example 4.6 A substance with concentration $c(\mathbf{x}, t)$ in a 3D region is subjected to Fick's Law of diffusion:

$$\mathbf{F} = -D\nabla c$$

which states that the substance diffuses down its concentration gradient at a rate equal to the negative of the gradient (i.e. the substance diffuses from regions of high to low concentration), multiplied by a diffusion coefficient D (with SI units of $\text{m}^2 \text{s}^{-1}$). Assuming there are no sources or sinks for the substance, derive the governing PDE for the concentration c .

Answer: Since there are no sources or sinks, $f(\mathbf{x}, t) = 0$. From Eq.4.12, we therefore have:

$$\frac{\partial c}{\partial t} + \nabla \cdot \mathbf{F} = 0$$

which becomes

$$\frac{\partial c}{\partial t} = \nabla \cdot (D\nabla c) \quad (4.13)$$

□

Example 4.7 Ohm's Law governing isotropic electric current flow in a 3D volume is given by

$$\mathbf{J} = \sigma \mathbf{E}$$

where \mathbf{J} is the current density (SI units of A m^{-2}), σ is the scalar conductivity (SI units of Siemens, or S), and \mathbf{E} is the electric field (V m^{-1}), given by the negative gradient of the electric potential V :

$$\mathbf{E} = -\nabla V$$

Derive the governing PDE for the electric potential V .

Answer: Assuming there are no sources or sinks of electric charge, we have $f(\mathbf{x}, t) = 0$. Defining u in Eq. 4.12 as the electric charge, then \mathbf{F} corresponds to current density \mathbf{J} . From the principle of conservation of charge, the total amount of electric charge must remain fixed in any fixed region V . Therefore $\frac{\partial u}{\partial t} = 0$, and from Eq. 4.12 and Ohm's Law, we obtain:

$$\nabla \cdot (\sigma \mathbf{E}) = 0$$

or simply

$$\nabla \cdot (-\sigma \nabla V) = 0 \quad (4.14)$$

□

4.1.6 The Laplacian

A variety of PDEs describing physical systems, such as the diffusion PDE of Eq. 4.13 and the electric potential PDE of Eq. 4.14, include terms such as $\nabla \cdot (D\nabla c)$. From the definition of the del operator, this term can be expanded to

$$\begin{aligned} \nabla \cdot (D\nabla c) &= \begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \end{pmatrix} \cdot \begin{pmatrix} D \frac{\partial c}{\partial x} \\ D \frac{\partial c}{\partial y} \\ D \frac{\partial c}{\partial z} \end{pmatrix} \\ &= \frac{\partial}{\partial x} \left[D \frac{\partial c}{\partial x} \right] + \frac{\partial}{\partial y} \left[D \frac{\partial c}{\partial y} \right] + \frac{\partial}{\partial z} \left[D \frac{\partial c}{\partial z} \right] \end{aligned}$$

If D is constant, independent of spatial position, then the above reduces to

$$\begin{aligned}\nabla \cdot (D\nabla c) &= D \left(\frac{\partial}{\partial x} \left[\frac{\partial c}{\partial x} \right] + \frac{\partial}{\partial y} \left[\frac{\partial c}{\partial y} \right] + \frac{\partial}{\partial z} \left[\frac{\partial c}{\partial z} \right] \right) \\ &= D \left(\frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2} + \frac{\partial^2 c}{\partial z^2} \right) \\ &= D\nabla^2 c\end{aligned}$$

where we have introduced the *Laplacian* operator ∇^2 , defined by

$$\nabla^2 \equiv \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$$

Using this Laplacian notation, the diffusion PDE given by Eq. 4.13 can be expressed as

$$\frac{\partial c}{\partial t} = D\nabla^2 c$$

provided D is independent of spatial position. Furthermore, under steady-state conditions, $\frac{\partial c}{\partial t} = 0$, and the diffusion equation can be reduced to its static formulation:

$$\nabla^2 c = 0$$

This PDE is known as *Poisson's equation*. If we further assume a non-zero source or sink term f throughout space, the static PDE diffusion formulation becomes

$$\nabla^2 c = g$$

where $g = f/D$. This type of PDE is known as *Laplace's equation*.

4.1.7 PDE Boundary Conditions

In order to uniquely solve a PDE, it is necessary to specify appropriate boundary conditions. These boundary conditions represent constraints on the dependent variable at the boundaries of the domain of interest. The simplest type of boundary condition is the *Dirichlet*⁷ condition, where the value of the dependent variable itself is specified at the boundary.

Example 4.8 Solve the 1D steady-state diffusion equation

$$\frac{\partial^2 c}{\partial x^2} = 0$$

⁷Pronounced 'Diri-klay'.

over the domain $x \in [0, 1]$, subject to the Dirichlet boundary conditions $c|_{x=0} = 0$, $c|_{x=1} = 2$

Answer: Successively integrating both sides of the PDE with respect to x , we obtain

$$\begin{aligned}\frac{\partial c}{\partial x} &= C_1 \\ c &= C_1x + C_2\end{aligned}$$

where C_1, C_2 are constants of integration. Substituting the boundary values of c at $x = 0$ and $x = 1$, we obtain $C_2 = 0$ and $C_1 = 2$. Hence, the PDE solution is

$$c = 2x$$

□

A second type of boundary condition is to specify the value of flux at the boundary, known as a *Neumann*⁸ boundary condition. For the diffusion PDE given by Eq. 4.13, recall that the flux was given by Fick's Law:

$$\mathbf{\Gamma} = -D\nabla c$$

which represents a vector quantity. On the boundaries of the PDE domain, instead of specifying all three-components of this flux vector, only the inward component normal to the boundary is typically specified. For an outward unit normal \mathbf{n} , the inward diffusion flux normal to the boundary is a scalar quantity given by

$$\Gamma = (D\nabla c) \cdot \mathbf{n}$$

If the inward flux is specified to be zero, this type of boundary condition is also termed a *zero-flux* boundary condition.

Example 4.9 Solve the 1D steady-state diffusion equation

$$\frac{\partial^2 c}{\partial x^2} = 0$$

over the domain $x \in [0, 1]$, subject to the Dirichlet boundary condition $c|_{x=0} = 0$, and the Neumann boundary condition $\frac{\partial c}{\partial x}|_{x=1} = 3$

Answer: As in the previous example, successively integrating both sides of the PDE with respect to x , we obtain

$$\begin{aligned}\frac{\partial c}{\partial x} &= C_1 \\ c &= C_1x + C_2\end{aligned}$$

⁸Pronounced 'Noy-mahn'.

where C_1, C_2 are constants of integration. Substituting the boundary value of c at $x = 0$, we obtain $C_2 = 0$. Substituting the Neumann boundary condition, $\frac{\partial c}{\partial x}|_{x=1} = 3$, we obtain $C_1 = 3$. Hence, the PDE solution is

$$c = 3x$$

□

A third type of boundary condition, known as a *mixed* or *Robin* boundary condition, is to specify the weighted sum of the variable and its inward flux at the boundary. For example, for our 1D diffusion example above, a mixed boundary condition would be $\left[c + \frac{\partial c}{\partial x} = 1 \right]_{x=0}$.

Example 4.10 Solve the 1D steady-state diffusion equation

$$\frac{\partial^2 c}{\partial x^2} = 0$$

over the domain $x \in [0, 1]$, subject to the Dirichlet boundary condition $c|_{x=0} = 0$, and the mixed boundary condition $\left[x + \frac{\partial c}{\partial x} \right]_{x=1} = 2$

Answer: As in the previous examples, successively integrating both sides of the PDE with respect to x , we obtain

$$\begin{aligned} \frac{\partial c}{\partial x} &= C_1 \\ c &= C_1 x + C_2 \end{aligned}$$

where C_1, C_2 are constants of integration. Substituting the boundary value of c at $x = 0$, we obtain $C_2 = 0$. Substituting the mixed boundary condition, $\left[x + \frac{\partial c}{\partial x} \right]_{x=1} = 2$, we obtain

$$\begin{aligned} \left[x + \frac{\partial c}{\partial x} \right]_{x=1} &= (C_1 + C_2) + C_1 \\ &= 2C_1 + C_2 \\ &= 2 \end{aligned}$$

Since $C_2 = 0$, then $C_1 = 1$. Hence, the PDE solution is

$$c = x$$

□

It is important to note that not all boundary conditions will result in a unique solution to a given PDE. For example, specifying only Neumann boundary conditions for the diffusion PDE of Eq.4.13 will not result in a unique solution. This can readily be seen from the following: if $c = \phi(\mathbf{x}, t)$ satisfies the PDE with associated Neumann boundary conditions, then so does $c = c_0 + \phi(\mathbf{x}, t)$, where c_0 is an arbitrary constant,

since any value of c_0 will satisfy the PDE as well as the boundary conditions. To uniquely solve a given PDE, it is necessary to specify at least one *essential* boundary condition. For our diffusion PDE example, an example of an essential boundary condition would be a Dirichlet condition on part of the boundary.

For time-dependent PDEs, i.e. PDEs containing time-derivatives, in addition to specifying appropriate boundary conditions, it is also necessary to specify an initial value for the dependent variable at every point in the spatial domain.

4.2 Basic Analytical and Numerical Solution Techniques

4.2.1 Separation of Variables

Analytical solutions to many linear PDEs can be obtained using the method of *separation of variables*. The method assumes that the solution to the dependent PDE variable can be split into a product of separate time- and spatial-dependent functions, as in

$$u(\mathbf{x}, t) = V(\mathbf{x})S(t) = P(x)Q(y)R(z)S(t)$$

where $u(\mathbf{x}, t)$ is the dependent variable and $P(x), Q(y), R(z), S(t)$ are scalar functions of the separate spatial and time components. We will use this method to obtain an analytical solution of the 1D time-dependent diffusion PDE of Eq. 4.13.

Example 4.11 Use the method of separation of variables to solve the 1D time-dependent diffusion equation

$$\frac{\partial c}{\partial t} = \frac{\partial^2 c}{\partial x^2}$$

for $x \in [0, 1]$ subject to zero-flux boundary conditions at $x = 0$ and $x = 1$, with initial value of $c(x, t)$ at $t = 0$ given by the square-wave distribution:

$$c(x, 0) = \Phi(x) = \begin{cases} 1 & 0.4 \leq x \leq 0.6 \\ 0 & \text{otherwise} \end{cases} \tag{4.15}$$

Answer: Using the method of separation of variables, we can write $c(x, t)$ as

$$c(x, t) = V(x)S(t)$$

Substituting this expression into the PDE, we have

$$\begin{aligned} \frac{\partial c}{\partial t} &= \frac{\partial^2 c}{\partial x^2} \\ V(x)S'(t) &= V''(x)S(t) \\ \frac{S'(t)}{S(t)} &= \frac{V''(x)}{V(x)} \end{aligned}$$

Since the left-hand side of the above expression only involves terms in t , and the right-hand side only terms in x , it follows that for both sides to be equal, they must be constant. Denoting this constant by λ , we have

$$\begin{aligned}\frac{S'(t)}{S(t)} &= \lambda \\ S'(t) &= \lambda S(t) \\ \therefore S(t) &= \mu e^{\lambda t}\end{aligned}$$

where μ, λ are constants. Similarly for $V(x)$, we have

$$\begin{aligned}\frac{V''(x)}{V(x)} &= \lambda \\ V''(x) &= \lambda V(x)\end{aligned}$$

This ODE has the characteristic equation

$$m^2 - \lambda = 0$$

with roots $m = \pm\sqrt{\lambda}$. Hence its general solution is given by

$$V(x) = f e^{\sqrt{\lambda}x} + g e^{-\sqrt{\lambda}x}$$

where f and g are additional constants of integration. To enforce the zero-flux boundary conditions, we require that $\frac{\partial c}{\partial x} = 0$ at $x = 0$ and $x = 1$, or more simply, $V'(0) = V'(1) = 0$. Differentiating the $V(x)$ expression above, we obtain:

$$V'(x) = f\sqrt{\lambda}e^{\sqrt{\lambda}x} - g\sqrt{\lambda}e^{-\sqrt{\lambda}x}$$

Substituting $V'(0) = V'(1) = 0$ leads to the following pair of equations for f and g :

$$\begin{aligned}f - g &= 0 \\ f e^{\sqrt{\lambda}} - g e^{-\sqrt{\lambda}} &= 0\end{aligned}$$

For $\lambda > 0$, the only solution to this pair of equations is $f = g = 0$ leading to $V(x) = 0$, and therefore $c(x, t) = 0$. This clearly cannot be the solution to the PDE, as it does not even satisfy the given initial condition for $t = 0$. If however we allow $\lambda < 0$, say $\lambda = -k$, where $k > 0$, then the second equation in the above pair becomes

$$\begin{aligned}
 f e^{i\sqrt{k}} - g e^{-i\sqrt{k}} &= 0 \\
 f \left[\cos(\sqrt{k}) + i \sin(\sqrt{k}) \right] - g \left[\cos(\sqrt{k}) - i \sin(\sqrt{k}) \right] &= 0 \\
 (f - g) \cos(\sqrt{k}) + i(f + g) \sin(\sqrt{k}) &= 0
 \end{aligned}$$

Substituting $f - g = 0$ from the first equation (i.e. $f = g$), the above becomes

$$2gi \sin(\sqrt{k}) = 0$$

which, for general values of g , can only be satisfied when $\sin(\sqrt{k}) = 0$. This will be true only for certain values of k satisfying

$$\sqrt{k} = n\pi, \quad n = 0, 1, 2, \dots$$

or

$$k = \pi^2 n^2$$

for integer values of $n \geq 0$. Hence, the general solution for $V(x)$ is

$$\begin{aligned}
 V(x) &= g e^{in\pi x} + g e^{-in\pi x} \\
 &= g [\cos(n\pi x) + i \sin(n\pi x)] + g [\cos(n\pi x) - i \sin(n\pi x)] \\
 &= 2g \cos(n\pi x)
 \end{aligned}$$

where g is a constant of integration. Furthermore, we have $S(t) = \mu e^{-\pi^2 n^2 t}$. Multiplying these solutions together leads to

$$V(x)S(t) = C_n e^{-\pi^2 n^2 t} \cos(n\pi x) \quad (4.16)$$

where $C_n = 2g\mu$ is the sole remaining constant of integration, lumped together from constants μ and g . Since the original diffusion PDE is linear, its general solution will consist of the sum of all solutions of the form of Eq. 4.16, namely

$$c(x, t) = \sum_{n=0}^{\infty} C_n e^{-\pi^2 n^2 t} \cos(n\pi x) \quad (4.17)$$

To satisfy the initial condition for $c(x, t)$ given by Eq. 4.15, we substitute $t = 0$ into Eq. 4.17 to obtain

$$\Phi(x) = \sum_{n=0}^{\infty} C_n \cos(n\pi x)$$

Multiplying both sides by $\cos(m\pi x)$, where m is some integer ≥ 0 , and integrating over the PDE domain, we obtain

$$\begin{aligned}
\int_0^1 \Phi(x) \cos(m\pi x) dx &= \int_0^1 \left[\sum_{n=0}^{\infty} C_n \cos(n\pi x) \right] \cos(m\pi x) dx \\
&= \sum_{n=0}^{\infty} \int_0^1 C_n \cos(n\pi x) \cos(m\pi x) dx \\
&= \int_0^1 C_m \cos(m\pi x) \cos(m\pi x) dx \quad (4.18)
\end{aligned}$$

since each integral term in the summation will be zero for $m \neq n$. Evaluating the left-hand side of Eq. 4.18, we have

$$\begin{aligned}
\int_0^1 \Phi(x) \cos(m\pi x) dx &= \int_{0.4}^{0.6} \cos(m\pi x) dx \\
&= \begin{cases} \int_{0.4}^{0.6} \cos(0) dx & m = 0 \\ \left[\frac{1}{m\pi} \sin(m\pi x) \right]_{0.4}^{0.6} & m = 1, 2, 3, \dots \end{cases} \\
&= \begin{cases} [x]_{0.4}^{0.6} & m = 0 \\ \frac{1}{m\pi} \{ \sin(0.6m\pi) - \sin(0.4m\pi) \} & m = 1, 2, 3, \dots \end{cases}
\end{aligned}$$

Using the trigonometric relation

$$\begin{aligned}
\sin(A+B) - \sin(A-B) &= \sin(A)\cos(B) + \cos(A)\sin(B) - [\sin(A)\cos(B) - \cos(A)\sin(B)] \\
&= 2\cos(A)\sin(B)
\end{aligned}$$

with $A = 0.5m\pi$ and $B = 0.1m\pi$, the above integral evaluates to

$$\int_0^1 \Phi(x) \cos(m\pi x) dx = \begin{cases} 0.2 & m = 0 \\ \frac{2}{m\pi} \cos(0.5m\pi) \sin(0.1m\pi) & m = 1, 2, 3, \dots \end{cases}$$

Evaluating the right-hand side of Eq. 4.18, we have

$$\begin{aligned}
\int_0^1 C_m \cos^2(m\pi x) dx &= C_m \int_0^1 \frac{1}{2} [1 + \cos(2m\pi x)] dx \\
&= \begin{cases} C_m \int_0^1 \frac{1}{2} [1 + \cos(0)] dx & m = 0 \\ \frac{C_m}{2} \left[x + \frac{1}{2m\pi} \sin(2m\pi x) \right]_0^1 & m = 1, 2, 3, \dots \end{cases} \\
&= \begin{cases} C_m & m = 0 \\ \frac{C_m}{2} & m = 1, 2, 3, \dots \end{cases}
\end{aligned}$$

Equating both the left and right-hand side evaluations of Eq. 4.18, we obtain

$$C_m = \begin{cases} 0.2 & m = 0 \\ \frac{4}{m\pi} \cos(0.5m\pi) \sin(0.1m\pi) & m = 1, 2, 3, \dots \end{cases}$$

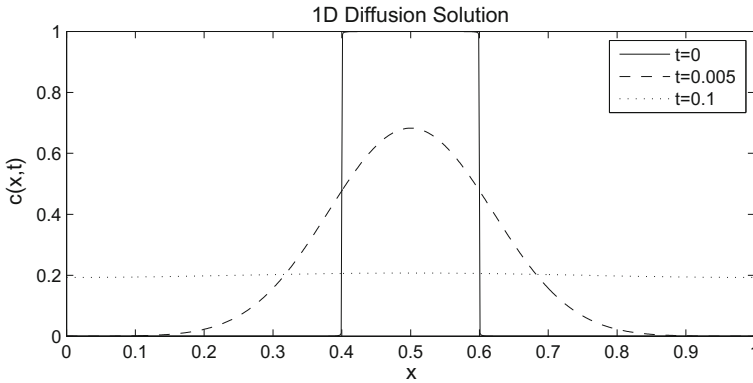


Fig. 4.9 Solution of 1D diffusion example at times $t = 0, 0.005$ and 0.1 , as generated by the Matlab code given

Finally, substituting these values of the integration constants into Eq. 4.17, we obtain the PDE solution

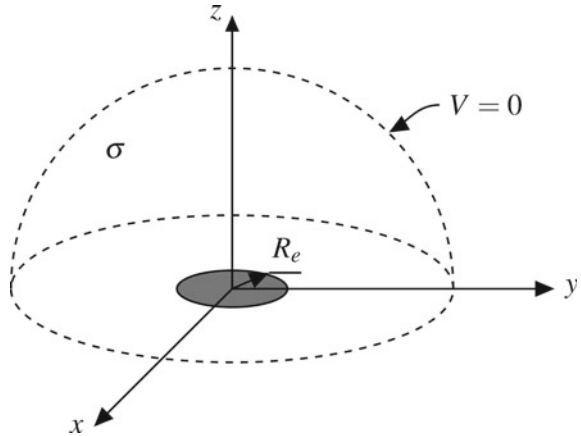
$$c(x, t) = 0.2 + \sum_{n=1}^{\infty} \left\{ \frac{4}{n\pi} \cos(0.5n\pi) \sin(0.1n\pi) e^{-\pi^2 n^2 t} \cos(n\pi x) \right\} \quad (4.19)$$

The following Matlab code plots this solution for values of n up to 10000 for $t = 0, 0.005$, and 0.1 . The resulting plot shown in Fig. 4.9.

```
x = 0:0.001:1;
c_0 = 0.2*ones(size(x));
c_1 = 0.2*ones(size(x));
c_2 = 0.2*ones(size(x));
for n = 1:10000
    t = 0;
    c_0 = c_0 + (4/(pi*n))*cos(0.5*n*pi)*...
        exp(-pi^2*n^2*t)*sin(0.1*n*pi)*cos(n*pi*x);
    t = 0.005;
    c_1 = c_1 + (4/(pi*n))*cos(0.5*n*pi)*...
        exp(-pi^2*n^2*t)*sin(0.1*n*pi)*cos(n*pi*x);
    t = 0.1;
    c_2 = c_2 + (4/(pi*n))*cos(0.5*n*pi)*...
        exp(-pi^2*n^2*t)*sin(0.1*n*pi)*cos(n*pi*x);
end;
plot(x,c_0,'k',x,c_1,'k--',x,c_2,'k:')...
    legend('t=0','t=0.005','t=0.1'), xlabel('x'), ylabel('c(x,t)'),...
    title('1D Diffusion Solution');
```

□

Fig. 4.10 Circular-disc electrode of radius R_e stimulating a hemispherical infinite domain of conductivity σ . The boundary at infinity is taken to be at electrical ground (i.e. $V = 0$)



Example 4.12 A far more complex example relates to 3D volumetric current flow arising from a circular disc electrode embedded in a hemispherical infinite medium of conductivity σ , as shown in Fig. 4.10. The PDE governing the voltage distribution is given by Eq. 4.14:

$$\nabla \cdot (-\sigma \nabla V) = 0$$

with current density

$$\mathbf{J} = -\sigma \nabla V$$

The radius of the disc electrode is R_e , and the potential at the infinite boundary of the hemisphere is taken to be ground (i.e. $V = 0$). The plane of the electrode disc outside the electrode is assumed to be electrically-insulating (i.e. a zero-flux boundary condition). Assuming the surface of the disc electrode is at an isopotential value V_s , and that a total stimulus current of I_s is injected into the electrode, use the method of separation of variables to solve for the voltage distribution, and find the access resistance Z of the electrode, defined as the ratio V_s/I_s .

Answer: Using the method of separation of variables, and the fact that the voltage distribution $V(x, y, z)$ must be axisymmetric about the z -axis due to the radial symmetry of the problem, the voltage distribution can be written as:

$$V(r, z) = P(r)Q(z)$$

where $r = \sqrt{x^2 + y^2}$. Furthermore, since the conductivity σ is constant throughout the domain, the relevant PDE reduces to

$$\nabla \cdot (\nabla V) = \frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} + \frac{\partial^2 V}{\partial z^2} = 0$$

Evaluating the first term, we have

$$\begin{aligned}
\frac{\partial^2 V}{\partial x^2} &= \frac{\partial^2 P(r)Q(z)}{\partial x^2} \\
&= Q(z) \frac{\partial^2 P(r)}{\partial x^2} \\
&= Q(z) \frac{\partial}{\partial x} \left[\frac{\partial P(r)}{\partial x} \right] \\
&= Q(z) \frac{\partial}{\partial x} \left[\frac{dP(r)}{dr} \frac{\partial r}{\partial x} \right] \\
&= Q(z) \frac{\partial}{\partial x} \left[P'(r) \frac{\partial \sqrt{x^2 + y^2}}{\partial x} \right] \\
&= Q(z) \frac{\partial}{\partial x} \left[P'(r) \frac{x}{\sqrt{x^2 + y^2}} \right] \\
&= Q(z) \left[\frac{\partial P'(r)}{\partial x} \frac{x}{\sqrt{x^2 + y^2}} + P'(r) \left(\frac{1}{\sqrt{x^2 + y^2}} + x \frac{-\frac{1}{2}}{(x^2 + y^2)^{\frac{3}{2}}} 2x \right) \right] \\
&= Q(z) \left[\frac{dP'(r)}{dr} \frac{\partial x}{\partial r} \frac{x}{r} + P'(r) \left(\frac{1}{r} - \frac{x^2}{r^3} \right) \right] \\
&= Q(z) \left[P''(r) \frac{x^2}{r^2} + \frac{1}{r} P'(r) \left(1 - \frac{x^2}{r^2} \right) \right]
\end{aligned}$$

In a similar fashion, the second term evaluates to

$$\frac{\partial^2 V}{\partial y^2} = Q(z) \left[P''(r) \frac{y^2}{r^2} + \frac{1}{r} P'(r) \left(1 - \frac{y^2}{r^2} \right) \right]$$

Finally, the third term is simply

$$\frac{\partial^2 V}{\partial z^2} = P(r)Q''(z)$$

Substituting all three terms into the PDE, we obtain

$$Q(z) \left[P''(r) \frac{x^2}{r^2} + \frac{1}{r} P'(r) \left(1 - \frac{x^2}{r^2} \right) + P''(r) \frac{y^2}{r^2} + \frac{1}{r} P'(r) \left(1 - \frac{y^2}{r^2} \right) \right] + P(r)Q''(z) = 0$$

Gathering all terms together, this simplifies to

$$\begin{aligned}
Q(z) \left[P''(r) \frac{x^2 + y^2}{r^2} + \frac{1}{r} P'(r) \left(2 - \frac{x^2}{r^2} - \frac{y^2}{r^2} \right) \right] + P(r)Q''(z) &= 0 \\
Q(z) \left[P''(r) + \frac{1}{r} P'(r) (2 - 1) \right] + P(r)Q''(z) &= 0
\end{aligned}$$

$$Q(z) \left[P''(r) + \frac{1}{r} P'(r) \right] + P(r) Q''(z) = 0$$

Dividing throughout by $P(r)Q(z)$, the above expression may be written as

$$\frac{1}{P(r)} \left[P''(r) + \frac{1}{r} P'(r) \right] + \frac{Q''(z)}{Q(z)} = 0$$

or

$$\frac{1}{P(r)} \left[P''(r) + \frac{1}{r} P'(r) \right] = -\frac{Q''(z)}{Q(z)}$$

Since each side of this expression is a function of r or z alone, it follows that each must equal a constant. That is

$$\frac{1}{P(r)} \left[P''(r) + \frac{1}{r} P'(r) \right] = \lambda \quad \text{and} \quad \frac{Q''(z)}{Q(z)} = -\lambda \quad (4.20)$$

where λ is a constant. Examining the first of these equations, we have

$$P''(r) + \frac{1}{r} P'(r) - \lambda P(r) = 0$$

This ODE can be solved by assuming that $P(r)$ may be expressed as a power series of the form

$$P(r) = \sum_{k=0}^{\infty} a_k r^k$$

where a_k are coefficients to be determined. Substituting this series expression into the ODE, we obtain:

$$\begin{aligned} \sum_{k=2}^{\infty} a_k k(k-1) r^{k-2} + \frac{1}{r} \sum_{k=1}^{\infty} a_k k r^{k-1} - \lambda \sum_{k=0}^{\infty} a_k r^k &= 0 \\ \sum_{k=2}^{\infty} a_k k(k-1) r^{k-2} + \sum_{k=1}^{\infty} a_k k r^{k-2} - \sum_{k=0}^{\infty} \lambda a_k r^k &= 0 \\ \sum_{k=2}^{\infty} a_k k(k-1) r^{k-2} + \left(\frac{a_1}{r} + \sum_{k=2}^{\infty} a_k k r^{k-2} \right) - \sum_{k=0}^{\infty} \lambda a_k r^k &= 0 \\ \frac{a_1}{r} + \sum_{k=2}^{\infty} [a_k k(k-1) + a_k k] r^{k-2} - \sum_{k=0}^{\infty} \lambda a_k r^k &= 0 \\ \frac{a_1}{r} + \sum_{k=2}^{\infty} a_k k^2 r^{k-2} - \sum_{k=0}^{\infty} \lambda a_k r^k &= 0 \end{aligned}$$

Using the substitution $j = k - 2$ in the first summation, the above expression transforms to

$$\frac{a_1}{r} + \sum_{j=0}^{\infty} a_{j+2}(j+2)^2 r^j - \sum_{k=0}^{\infty} \lambda a_k r^k = 0$$

and since j is a dummy index, it can be replaced with k again to combine both sums into a single series:

$$\frac{a_1}{r} + \sum_{k=0}^{\infty} [a_{k+2}(k+2)^2 - \lambda a_k] r^k = 0$$

Since the left-hand side must be identically equal to zero for all values of r , the coefficients of all the powers of r in the above expression must be zero. Hence,

$$a_1 = 0 \quad \text{and} \quad a_{k+2}(k+2)^2 - \lambda a_k = 0$$

which may be re-written as

$$a_1 = 0$$

$$a_{k+2} = \left[\frac{\lambda}{(k+2)^2} \right] a_k$$

from which we determine

$$a_k = \begin{cases} 0 & \text{if } k \text{ is odd} \\ \left(\frac{\lambda^{\frac{k}{2}}}{k^2(k-2)^2 \dots 2^2} \right) a_0 & \text{if } k \text{ is even} \end{cases}$$

where a_0 is the first coefficient of the series. Expanding the above coefficient a_k for the case when k is even, we obtain:

$$a_k = \left(\frac{\lambda^{\frac{k}{2}}}{k^2(k-2)^2 \dots 2^2} \right) a_0$$

$$= \left(\frac{\lambda^{\frac{k}{2}}}{\left\{ \left(\frac{k}{2}\right)^2 \left(\frac{k}{2}-1\right)^2 \dots 1^2 \right\} 2^{\binom{k}{2}} 2^{\binom{k}{2}}} \right) a_0$$

$$= \left(\frac{\lambda^{\frac{k}{2}}}{2^k \left(\frac{k!}{2}\right)^2} \right) a_0$$

Hence, by summing these even terms, $P(r)$ may be expressed as the power series

$$P(r) = a_0 \left[\sum_{n=0}^{\infty} \left(\frac{\lambda^n}{2^{2n}(n!)^2} \right) r^{2n} \right]$$

from which it is evident that for $P(r)$ to be real-valued, both a_0 and λ must be real. Furthermore, since the voltage V will approach ground as $r \rightarrow \infty$, we require that $\lim_{r \rightarrow \infty} P(r) = 0$. This can only happen if the coefficients of the individual r^{2n} terms of the power series alternate in sign, which implies that $\lambda < 0$. Denoting $\lambda = -\mu^2$, where μ is real, we can express $P(r)$ as

$$P(r) = a_0 \left[\sum_{n=0}^{\infty} (-1)^n \left(\frac{\mu^{2n}}{2^{2n}(n!)^2} \right) r^{2n} \right]$$

This expression is related to the Bessel function of the first kind of order zero, $J_0(x)$, defined as [1]:

$$\begin{aligned} J_0(x) &= 1 - \frac{x^2}{2^2} + \frac{x^4}{2^2 \cdot 4^2} - \frac{x^6}{2^2 \cdot 4^2 \cdot 6^2} + \dots \\ &= \sum_{n=0}^{\infty} (-1)^n \left(\frac{x^{2n}}{(2^n n!)^2} \right) \\ &= \sum_{n=0}^{\infty} (-1)^n \left(\frac{x^{2n}}{2^{2n} (n!)^2} \right) \end{aligned}$$

from which we deduce that

$$P(r) = a_0 J_0(\mu r) \quad (4.21)$$

Now that we have determined $P(r)$, we can now turn to $Q(z)$ in Eq. 4.20. In this case, the relevant ODE is

$$Q''(z) - \mu^2 Q(z) = 0$$

whose characteristic equation has the real roots $\pm\mu$. Its general solution is therefore

$$Q(z) = C_1 e^{\mu z} + C_2 e^{-\mu z}$$

where C_1, C_2 are constants. Furthermore, since V will be zero at infinite distance from the electrode, we must have $Q(z) \rightarrow 0$ as $z \rightarrow \infty$. Therefore $C_1 = 0$, and $Q(z)$ will have the form

$$Q(z) = C_2 e^{-\mu z} \quad (\mu > 0)$$

Forming the product $P(r)Q(z)$ we have

$$\begin{aligned} V(r, z) &= P(r)Q(z) \\ &= KJ_0(\mu r)e^{-\mu z} \quad (\mu > 0) \end{aligned}$$

where K is a new constant such that $K = a_0 C_2$. The general solution to the PDE can be determined by adding together all possible solutions of the above form. Since each term will have different values for K and μ , we can represent this sum using a definite integral, to obtain the following general solution for $V(r, z)$:

$$V(r, z) = \int_0^{\infty} K(\mu) J_0(\mu r) e^{-\mu z} d\mu \quad (4.22)$$

To determine $K(\mu)$, we implement the following mixed boundary conditions on the lower boundary, where $z = 0$:

$$\begin{aligned} V &= V_s & (r \leq R_e: \text{i.e. equipotential state within the electrode disc}) \\ \frac{\partial V}{\partial z} &= 0 & (r > R_e: \text{i.e. electric insulation elsewhere on this boundary}) \end{aligned}$$

From Eq. 4.22, these conditions lead to the following pair of dual integral equations:

$$\begin{aligned} V_s &= \int_0^{\infty} K(\mu) J_0(\mu r) d\mu & (0 \leq r \leq R_e) \\ 0 &= \int_0^{\infty} -\mu K(\mu) J_0(\mu r) d\mu & (r > R_e) \end{aligned}$$

and substituting the variable $\rho = r/R_e$, this pair of integrals becomes

$$\int_0^{\infty} K(\mu) J_0(\mu R_e \rho) d\mu = V_s \quad (0 \leq \rho \leq 1) \quad (4.23)$$

$$\int_0^{\infty} \mu K(\mu) J_0(\mu R_e \rho) d\mu = 0 \quad (\rho > 1) \quad (4.24)$$

To solve this pair of integral equations for $K(\mu)$, we set

$$K(\mu) = \int_0^1 \phi(\xi) \cos(\mu R_e \xi) d\xi \quad (4.25)$$

and substituting into the left-hand side of the second integral equation (Eq. 4.24), we obtain:

$$\begin{aligned} \int_0^{\infty} \mu K(\mu) J_0(\mu R_e \rho) d\mu &= \int_0^{\infty} \mu \left(\int_0^1 \phi(\xi) \cos(\mu R_e \xi) d\xi \right) J_0(\mu R_e \rho) d\mu \\ &= \int_0^{\infty} \mu \left[\phi(\xi) \frac{\sin(\mu R_e \xi)}{\mu R_e} - \int \phi'(\xi) \frac{\sin(\mu R_e \xi)}{\mu R_e} d\xi \right]_0^1 J_0(\mu R_e \rho) d\mu \end{aligned}$$

$$\begin{aligned}
&= \frac{1}{R_e} \int_0^\infty \left[\phi(\xi) \sin(\mu R_e \xi) - \int \phi'(\xi) \sin(\mu R_e \xi) d\xi \right]_0^1 J_0(\mu R_e \rho) d\mu \\
&= \frac{1}{R_e} \int_0^\infty \phi(1) \sin(\mu R_e) J_0(\mu R_e \rho) d\mu \\
&\quad - \frac{1}{R_e} \int_0^\infty \left(\int_0^1 \phi'(\xi) \sin(\mu R_e \xi) d\xi \right) J_0(\mu R_e \rho) d\mu \\
&= \frac{1}{R_e} \int_0^\infty \phi(1) \sin(\mu R_e) J_0(\mu R_e \rho) d\mu \\
&\quad - \frac{1}{R_e} \int_0^1 \phi'(\xi) \left(\int_0^\infty \sin(\mu R_e \xi) J_0(\mu R_e \rho) d\mu \right) d\xi \tag{4.26}
\end{aligned}$$

To evaluate this expression, we make use of Weber's discontinuous integral identities [1]:

$$\int_0^\infty J_0(ax) \sin(bx) dx = \begin{cases} \frac{1}{\sqrt{b^2 - a^2}} & (a < b) \\ 0 & (a > b) \end{cases} \tag{4.27}$$

$$\int_0^\infty J_0(ax) \cos(bx) dx = \begin{cases} 0 & (a < b) \\ \frac{1}{\sqrt{a^2 - b^2}} & (a > b) \end{cases} \tag{4.28}$$

Hence, for the first integral of Eq. 4.26, we have:

$$\begin{aligned}
\frac{1}{R_e} \int_0^\infty \phi(1) \sin(\mu R_e) J_0(\mu R_e \rho) d\mu &= \frac{\phi(1)}{R_e} \int_0^\infty J_0(\mu R_e \rho) \sin(\mu R_e) d\mu \\
&= 0 \quad (\text{since } \rho > 1)
\end{aligned}$$

where we have made use of Eq. 4.27. For the second integral of Eq. 4.26, its integrand similarly evaluates to zero from the same identity:

$$\int_0^\infty \sin(\mu R_e \xi) J_0(\mu R_e \rho) d\mu = 0 \quad (\text{since } \rho > 1 \text{ and } 0 \leq \xi \leq 1)$$

Hence for the second integral,

$$\frac{1}{R_e} \int_0^1 \phi'(\xi) \left(\int_0^\infty \sin(\mu R_e \xi) J_0(\mu R_e \rho) d\mu \right) d\xi = 0$$

and we have verified that Eq. 4.24 holds. Substituting Eq. 4.25 into 4.22, we obtain

$$\begin{aligned}
\int_0^\infty K(\mu) J_0(\mu R_e \rho) d\mu &= \int_0^\infty \left(\int_0^1 \phi(\xi) \cos(\mu R_e \xi) d\xi \right) J_0(\mu R_e \rho) d\mu \\
&= \int_0^1 \phi(\xi) \left(\int_0^\infty J_0(\mu R_e \rho) \cos(\mu R_e \xi) d\mu \right) d\xi
\end{aligned}$$

$$\begin{aligned}
&= \int_0^\rho \phi(\xi) \frac{1}{\sqrt{R_e^2 \rho^2 - R_e^2 \xi^2}} d\xi \quad (\text{from Eq. 4.28 using } 0 \leq \rho \leq 1) \\
&= \frac{1}{R_e} \int_0^\rho \frac{\phi(\xi)}{\sqrt{\rho^2 - \xi^2}} d\xi \\
&= V_s \quad (\text{from Eq. 4.23})
\end{aligned}$$

Hence, $\phi(\xi)$ satisfies the integral equation

$$\frac{1}{R_e} \int_0^\rho \frac{\phi(\xi)}{\sqrt{\rho^2 - \xi^2}} d\xi = V_s \quad (0 \leq \rho \leq 1) \quad (4.29)$$

Using the identity

$$\int_0^\rho \frac{d\xi}{\sqrt{\rho^2 - \xi^2}} = \left[\sin^{-1} \left(\frac{\xi}{\rho} \right) \right]_0^\rho = \frac{\pi}{2}$$

we see that Eq. 4.29 is solved when

$$\phi(\xi) = \frac{2V_s R_e}{\pi}$$

Substituting this value for $\phi(\xi)$ into Eq. 4.25, we obtain the required expression for $K(\mu)$:

$$\begin{aligned}
K(\mu) &= \int_0^1 \phi(\xi) \cos(\mu R_e \xi) d\xi \\
&= \int_0^1 \frac{2V_s R_e}{\pi} \cos(\mu R_e \xi) d\xi \\
&= \frac{2V_s}{\pi} \left[\frac{\sin(\mu R_e \xi)}{\mu} \right]_0^1 \\
&= \frac{2V_s \sin(\mu R_e)}{\mu \pi}
\end{aligned}$$

Finally, we can substitute this expression for $K(\mu)$ into the PDE general solution (Eq. 4.22), to obtain:

$$V(r, z) = \int_0^\infty \frac{2V_s J_0(\mu r) \sin(\mu R_e)}{\mu \pi} e^{-\mu z} d\mu$$

which can be solved using the integral identity [1]:

$$\int_0^{\infty} \frac{J_0(ax) \sin(bx)}{x} e^{-cx} dx = \tan^{-1} \left(\frac{b}{\gamma} \right)$$

$$\gamma = \sqrt{\frac{a^2 + c^2 - b^2 + \sqrt{(a^2 + c^2 - b^2)^2 + 4b^2c^2}}{2}}$$

to yield the PDE solution:

$$V(r, z) = \frac{2V_s}{\pi} \tan^{-1} \left(\frac{R_e \sqrt{2}}{\sqrt{r^2 + z^2 - R_e^2} + \sqrt{(r^2 + z^2 - R_e^2)^2 + 4z^2 R_e^2}} \right) \quad (4.30)$$

When $z = 0$ and $r < R_e$ (i.e. on the lower boundary within the electrode disc), then the denominator term of the arctan argument will be zero, leading to a solution of

$$V(r, 0)|_{\text{inside disc}} = \frac{2V_s}{\pi} \cdot \frac{\pi}{2} = V_s$$

as required. Furthermore, noting that $\frac{d}{dx} \tan^{-1}(x) = 1/(1+x^2)$, we can differentiate Eq. 4.30 with respect to z using the chain rule, substituting $\beta = r^2 + z^2 - R_e^2$ to simplify the analysis, to obtain:

$$\frac{\partial V}{\partial z} = \frac{2V_s}{\pi} \frac{\partial D}{(1+D^2) \partial z}$$

where we have used

$$D = \frac{R_e \sqrt{2}}{\sqrt{\beta + \sqrt{\beta^2 + 4z^2 R_e^2}}}$$

Now,

$$\frac{\partial D}{\partial z} = -\frac{1}{2} R_e \sqrt{2} \left(\beta + \sqrt{\beta^2 + 4z^2 R_e^2} \right)^{-\frac{3}{2}} \frac{\partial E}{\partial z}$$

$$E = \beta + \sqrt{\beta^2 + 4z^2 R_e^2}$$

$$\frac{\partial E}{\partial z} = \frac{\partial B}{\partial z} + \frac{1}{2} (\beta^2 + 4z^2 R_e^2)^{-\frac{1}{2}} \frac{\partial F}{\partial z}$$

$$F = \beta^2 + 4z^2 R_e^2$$

$$\frac{\partial F}{\partial z} = 2\beta \frac{\partial \beta}{\partial z} + 8z R_e^2$$

$$\frac{\partial B}{\partial z} = 2z$$

Hence,

$$\begin{aligned}
 \frac{\partial F}{\partial z} &= 4\beta z + 8zR_e^2 \\
 \frac{\partial E}{\partial z} &= 2z + \frac{1}{2}(\beta^2 + 4z^2R_e^2)^{-\frac{1}{2}}(4\beta z + 8zR_e^2) \\
 &= 2z + \frac{2\beta z + 4zR_e^2}{\sqrt{\beta^2 + 4z^2R_e^2}} \\
 \frac{\partial D}{\partial z} &= -\frac{1}{2}R_e\sqrt{2}\left(\beta + \sqrt{\beta^2 + 4z^2R_e^2}\right)^{-\frac{3}{2}}\left[2z + \frac{2\beta z + 4zR_e^2}{\sqrt{\beta^2 + 4z^2R_e^2}}\right] \\
 &= -R_e\sqrt{2}\left(\beta + \sqrt{\beta^2 + 4z^2R_e^2}\right)^{-\frac{3}{2}}\left[z + \frac{\beta z + 2zR_e^2}{\sqrt{\beta^2 + 4z^2R_e^2}}\right]
 \end{aligned}$$

and therefore

$$\begin{aligned}
 \frac{\partial V}{\partial z} &= \frac{-2\sqrt{2}V_sR_e}{\pi\left(1 + \frac{2R_e^2}{\beta + \sqrt{\beta^2 + 4z^2R_e^2}}\right)}\left(\beta + \sqrt{\beta^2 + 4z^2R_e^2}\right)^{-\frac{3}{2}}\left[z + \frac{\beta z + 2zR_e^2}{\sqrt{\beta^2 + 4z^2R_e^2}}\right] \\
 &= \frac{-2\sqrt{2}V_sR_e}{\pi\left(\beta + \sqrt{\beta^2 + 4z^2R_e^2} + 2R_e^2\right)}\left(\frac{1}{\sqrt{\beta + \sqrt{\beta^2 + 4z^2R_e^2}}}\right)\left[z + \frac{\beta z + 2zR_e^2}{\sqrt{\beta^2 + 4z^2R_e^2}}\right] \\
 &= \frac{-2\sqrt{2}V_sR_e}{\pi\left(\beta + \sqrt{\beta^2 + 4z^2R_e^2} + 2R_e^2\right)}\left(\frac{z}{\sqrt{\beta + \sqrt{\beta^2 + 4z^2R_e^2}}}\right)\left[1 + \frac{\beta + 2R_e^2}{\sqrt{\beta^2 + 4z^2R_e^2}}\right]
 \end{aligned}$$

For small values of z near zero, we can approximate the expression $\sqrt{\beta^2 + 4z^2R_e^2}$ from its Taylor series about $z = 0$ as

$$\sqrt{\beta^2 + 4z^2R_e^2} \approx |\beta| + \frac{4z^2R_e^2}{2|\beta|} = |\beta| + \frac{2z^2R_e^2}{|\beta|}$$

and substituting this into the above expression for $\partial V/\partial z$, we obtain:

$$\lim_{z \rightarrow 0} \frac{\partial V}{\partial z} = \lim_{z \rightarrow 0} \left\{ \frac{-2\sqrt{2}V_sR_e}{\pi\left(\beta + |\beta| + \frac{2z^2R_e^2}{|\beta|} + 2R_e^2\right)} \left(\frac{z}{\sqrt{\beta + |\beta| + \frac{2z^2R_e^2}{|\beta|}}}\right) \left[1 + \frac{\beta + 2R_e^2}{|\beta| + \frac{2z^2R_e^2}{|\beta|}}\right] \right\}$$

When $\beta > 0$ (i.e. in regions outside the electrode disc), this limit will be zero due to the middle term being a multiple of z . Thus $\partial V/\partial z = 0$ on the lower boundary when $r > R_e$, also as required. However within the electrode disc, $\beta \leq 0$ and $\beta + |\beta|$ will equal zero. Hence,

$$\begin{aligned}
 \lim_{z \rightarrow 0} \frac{\partial V}{\partial z} \Big|_{\beta < 0} &= \lim_{z \rightarrow 0} \left\{ \frac{-2\sqrt{2}V_s R_e}{\pi \left(\frac{2z^2 R_e^2}{|\beta|} + 2R_e^2 \right)} \left(\frac{z}{\sqrt{\frac{2z^2 R_e^2}{|\beta|}}} \right) \left[1 + \frac{\beta + 2R_e^2}{|\beta| + \frac{2z^2 R_e^2}{|\beta|}} \right] \right\} \\
 &= \frac{-2\sqrt{2}V_s R_e}{2\pi R_e^2} \left(\frac{1}{\sqrt{\frac{2R_e^2}{|\beta|}}} \right) \left[1 + \frac{\beta + 2R_e^2}{|\beta|} \right] \\
 &= -\frac{V_s}{\pi R_e} \left(\frac{\sqrt{|\beta|}}{R_e} \right) \left[\frac{|\beta| + \beta + 2R_e^2}{|\beta|} \right] \\
 &= -\frac{V_s}{\pi R_e} \left(\frac{\sqrt{|\beta|}}{R_e} \right) \left[\frac{2R_e^2}{|\beta|} \right] \quad (\text{since } |\beta| + \beta = 0) \\
 &= -\frac{2V_s}{\pi \sqrt{|\beta|}} \\
 &= -\frac{2V_s}{\pi \sqrt{R_e^2 - r^2}}
 \end{aligned}$$

Over the electrode disc, the current density is given by

$$\begin{aligned}
 \mathbf{J} &= -\sigma \nabla V \\
 &= -\sigma \begin{pmatrix} \partial V/\partial x \\ \partial V/\partial y \\ \partial V/\partial z \end{pmatrix}
 \end{aligned}$$

and since the disc is at a constant potential of V_s , $\partial V/\partial x = \partial V/\partial y = 0$. Thus, the only non-zero component of current density is the z -component, given by

$$\begin{aligned}
 J_z &= -\sigma (\partial V/\partial z) \\
 &= \frac{2\sigma V_s}{\pi \sqrt{R_e^2 - r^2}}
 \end{aligned}$$

Note that near the edge of the disc where $r \rightarrow R_e$, J_z will approach infinity. To find the total current through the disc electrode, the current flowing through an infinitesimally thin annulus in the disc with inner and outer radii r and $r + dr$ is simply given by $J_z 2\pi r dr$. Hence, the total current through the electrode can be determined from

$$I_s = \int_0^{R_e} J_z 2\pi r dr$$

$$\begin{aligned}
 &= 4\sigma V_s \int_0^{R_e} \frac{r}{\sqrt{R_e^2 - r^2}} dr \\
 &= 4\sigma V_s \left[-\sqrt{R_e^2 - r^2} \right]_0^{R_e} \\
 &= 4\sigma V_s R_e
 \end{aligned}$$

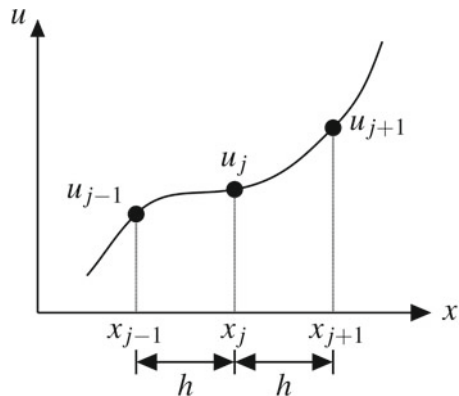
Hence, the access resistance of the electrode is given by $Z = \frac{V_s}{I_s} = \frac{1}{4\sigma R_e}$. □

The above example illustrates that even for moderately simple PDE systems for which closed form solutions exist, analytical solutions may be tedious to obtain. For many PDE systems, such closed form solutions do not even exist. We must therefore resort to computational techniques to obtain approximate solutions. The next two subsections provide an overview of the two most basic numerical techniques used for solving PDEs: the *finite difference method* and for time-dependent PDEs, the *method of lines*. The finite element method will be introduced in Chap. 5.

4.2.2 Finite Difference Method

The finite difference (FD) method is a relatively simple approach for numerically solving PDEs. In this method, the underlying PDE is discretized into a regular grid covering the PDE domain, and partial derivatives are approximated from the discretized grid variables. Consider a quantity $u(x)$, a function of the 1D spatial position x . We can discretize the 1D spatial domain into a regularly-spaced set of nodes in x , given by $x_1, x_2, \dots, x_j, \dots$, each spaced a distance h apart, as shown in Fig. 4.11. If we denote the value of u at node j by u_j , then we can use Taylor’s theorem (Eq. 3.1) to obtain

Fig. 4.11 Discretization of $u(x)$ onto a regular 1D spatial grid



$$\begin{aligned}
 u_{j+1} &= u_j + h \frac{\partial u}{\partial x} + \frac{h^2}{2} \frac{\partial^2 u(\xi)}{\partial x^2}, \quad \xi \in [x_j, x_{j+1}] \\
 &= u_j + h \frac{\partial u}{\partial x} + \mathcal{O}(h^2)
 \end{aligned}$$

where the derivative is evaluated at x_j . Hence, we obtain the *forward finite difference* approximation to the derivative as:

$$\left. \frac{\partial u}{\partial x} \right|_{x=x_j} = \frac{u_{j+1} - u_j}{h} + \mathcal{O}(h) \quad (4.31)$$

Similarly, we can approximate u_{i-1} using

$$\begin{aligned}
 u_{j-1} &= u_j - h \frac{\partial u}{\partial x} + \frac{h^2}{2} \frac{\partial^2 u(\xi)}{\partial x^2}, \quad \xi \in [x_{j-1}, x_j] \\
 &= u_j - h \frac{\partial u}{\partial x} + \mathcal{O}(h^2)
 \end{aligned}$$

to obtain the *backward finite difference* approximation to the derivative at x_j :

$$\left. \frac{\partial u}{\partial x} \right|_{x=x_j} = \frac{u_j - u_{j-1}}{h} + \mathcal{O}(h) \quad (4.32)$$

We can also add a further terms to each Taylor series approximation above to obtain:

$$u_{j+1} = u_j + h \frac{\partial u}{\partial x} + \frac{h^2}{2} \frac{\partial^2 u}{\partial x^2} + \frac{h^3}{6} \frac{\partial^3 u}{\partial x^3} + \mathcal{O}(h^4) \quad (4.33)$$

$$u_{j-1} = u_j - h \frac{\partial u}{\partial x} + \frac{h^2}{2} \frac{\partial^2 u}{\partial x^2} - \frac{h^3}{6} \frac{\partial^3 u}{\partial x^3} + \mathcal{O}(h^4) \quad (4.34)$$

where all partial derivatives are evaluated at $x = x_j$. Adding Eqs. 4.33 and 4.34, we obtain:

$$u_{j+1} + u_{j-1} = 2u_j + h^2 \frac{\partial^2 u}{\partial x^2} + \mathcal{O}(h^4)$$

which can be re-arranged to obtain the FD approximation of the second-order derivative:

$$\left. \frac{\partial^2 u}{\partial x^2} \right|_{x=x_j} = \frac{u_{j+1} - 2u_j + u_{j-1}}{h^2} + \mathcal{O}(h^2) \quad (4.35)$$

Such FD approximations to derivatives can be substituted into a given PDE to obtain a system of equations in the nodal variables u_j . For example, to solve the 1D diffusion equation

$$\frac{\partial c}{\partial t} = \frac{\partial^2 c}{\partial x^2}$$

we can approximate the first-order time derivative from the forward finite difference approximation in the time domain, similar to Eq. 4.31, as:

$$\frac{\partial c(t)}{\partial t} = \frac{c(t + \Delta t) - c(t)}{\Delta t} + \mathcal{O}(\Delta t) \quad (4.36)$$

where Δt is the time-step size and all c -terms are evaluated at the same spatial position. Similarly, the second-order spatial derivative can be approximated from Eq. 4.35 as:

$$\left. \frac{\partial^2 c(t)}{\partial x^2} \right|_{x=x_j} = \frac{c_{j+1}(t) - 2c_j(t) + c_{j-1}(t)}{h^2} + \mathcal{O}(h^2) \quad (4.37)$$

where h is the spatial step-size and all values of u are determined at the current time t . Equating the right-hand sides of Eqs. 4.36 and 4.37 we obtain:

$$\frac{c_j(t + \Delta t) - c_j(t)}{\Delta t} = \frac{c_{j+1}(t) - 2c_j(t) + c_{j-1}(t)}{h^2} + \mathcal{O}(\Delta t) + \mathcal{O}(h^2)$$

where the error term $\mathcal{O}(\Delta t) + \mathcal{O}(h^2)$ represents the local truncation error of the method. Neglecting this error, and re-arranging, we obtain the following *explicit* FD scheme:

$$c_j(t + \Delta t) = c_j(t) + \mu \left\{ c_{j+1}(t) - 2c_j(t) + c_{j-1}(t) \right\} \quad (4.38)$$

where $\mu = \Delta t/h^2$. In other words, the future value of c at the next time step $t + \Delta t$ for a given spatial position j can be approximated from its values at the current time step t for the same spatial position as well as its neighbouring spatial positions $j + 1$ and $j - 1$.

To explore the stability properties of Eq. 4.38 for various spatial and time step sizes, we can use *Fourier analysis* of the error, to insert the following proposed solution for $c(x, t)$ into the explicit algorithm:

$$c(jh, n\Delta t) = \lambda^n e^{ik(jh)} \quad (4.39)$$

where $x_j = jh$ and $t_n = n\Delta t$ are the solution grid points, $i = \sqrt{-1}$ and λ, k are parameters associated with the given solution. In particular, λ represents an amplification factor that multiplies the solution at the previous time step. For stability, we require $|\lambda| \leq 1$. Substituting Eq. 4.39 into the explicit scheme Eq. 4.38, we obtain:

$$\begin{aligned} c(jh, (n+1)\Delta t) &= c(jh, n\Delta t) + \mu \left\{ c((j+1)h, n\Delta t) - 2c(jh, n\Delta t) + c((j-1)h, n\Delta t) \right\} \\ \lambda^{n+1} e^{ik(jh)} &= \lambda^n e^{ik(jh)} + \mu \left\{ \lambda^n e^{ik(j+1)h} - 2\lambda^n e^{ik(jh)} + \lambda^n e^{ik(j-1)h} \right\} \end{aligned}$$

and dividing through by $\lambda^n e^{ik(jh)}$, the above simplifies to:

$$\begin{aligned}\lambda &= 1 + \mu \{e^{ikh} - 2 + e^{-ikh}\} \\ &= 1 + \mu \{2 \cos(kh) - 2\} \\ &= 1 - 2\mu \{1 - \cos(kh)\}\end{aligned}$$

and using the trigonometric identity $\sin^2(\alpha) = \frac{1}{2} \{1 - \cos(2\alpha)\}$, we obtain:

$$\lambda = 1 - 4\mu \sin^2\left(\frac{1}{2}kh\right)$$

For stability, we require $|\lambda| \leq 1$, which is satisfied when $0 \leq \mu \leq \frac{1}{2}$. This corresponds to $\Delta t \leq \frac{1}{2}h^2$.

Example 4.13 Use the explicit FD scheme of Eq. 4.38 to numerically solve the 1D diffusion equation

$$\frac{\partial c}{\partial t} = \frac{\partial^2 c}{\partial x^2}$$

over the interval $x \in [0, 1]$ from $t = 0$ to 0.005, subject to the boundary conditions:

$$\left. \frac{\partial c}{\partial x} \right|_{x=0} = \left. \frac{\partial c}{\partial x} \right|_{x=1} = 0 \quad (\text{i.e. zero-flux boundary conditions})$$

with initial value at $t = 0$ given by:

$$c = \begin{cases} 1 & 0.4 \leq x \leq 0.6 \\ 0 & \text{otherwise} \end{cases}$$

Implement the explicit FD scheme using a spatial grid FD resolution h of 0.01 and a time step Δt of 0.00001.

Answer: Let c_j^n denote the value of c at $x = (j - 1)h$ and $t = (n - 1)\Delta t$, where j and n are integer values given by

$$j = 1, 2, \dots, \frac{1}{h} + 1, \quad n = 1, 2, \dots, \frac{0.005}{\Delta t} + 1$$

and h , Δt are the regular grid-spacing in x and t respectively. Implementing the explicit scheme of Eq. 4.38, we obtain:

$$c_j^{n+1} = c_j^n + \frac{\Delta t}{h^2} (c_{j+1}^n - 2c_j^n + c_{j-1}^n)$$

To implement the zero-flux boundary conditions, we use the forward and backward FD approximations of the spatial derivative (Eqs. 4.31 and 4.32 respectively)

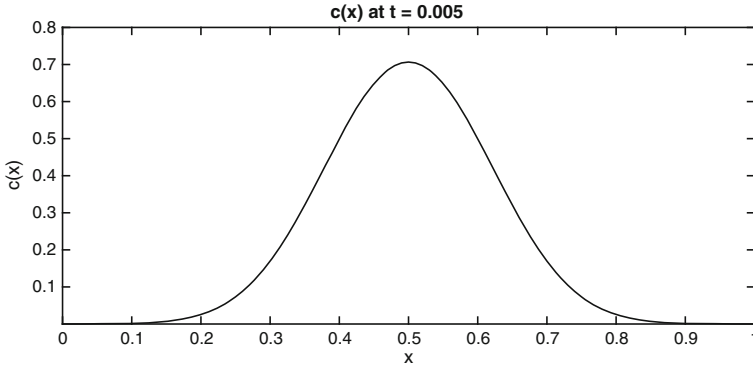


Fig. 4.12 Solution of 1D diffusion example at time $t = 0.005$ using the FD explicit scheme of Eq.4.38 with $h = 0.01$ and $\Delta t = 0.00001$

to obtain:

$$\frac{c_2^n - c_1^n}{h} = 0, \quad \frac{c_M^n - c_{M-1}^n}{h} = 0$$

or more simply

$$c_2^n - c_1^n = 0, \quad c_M^n - c_{M-1}^n = 0$$

for all time steps n , where $M = 1/h + 1$.

The following Matlab code implements this explicit scheme, plotting the solution at $t = 0.005$ in Fig.4.12. This solution compares well with the analytical result obtained earlier in Example 4.11 (see Fig.4.9 for $t = 0.005$).

```

h = 0.01;           % x-increment
dt = 0.00001;      % time-increment
T = 0.005;         % total time
M = round(1/h)+1;  % total x points
N = round(T/dt)+1; % total time points
c = zeros(M,N);

% set initial waveform
for j = 1:M
    if ((j-1)*h >= 0.4)&&((j-1)*h <= 0.6)
        c(j,1) = 1;
    else
        c(j,1) = 0;
    end;
end;

% solve for subsequent times
for n = 1:N-1
    for j = 2:M-1
        c(j,n+1) = c(j,n)+(dt/h^2)*(c(j+1,n)-2*c(j,n)+c(j-1,n));
    end;
end;
    
```

```

c(1,n+1) = c(2,n+1); % zero-flux b.c.
c(M,n+1) = c(M-1,n+1); % zero-flux b.c.
end;

% plot solution
plot(0:h:1,c(:,end),'k'), xlabel('x'), ylabel('c(x)'),...
title('c(x) at t = 0.005');

```

□

The stability of the explicit FD scheme given by Eq. 4.38 can be improved by replacing the FD approximation of the second-order spatial derivative in Eq. 4.37, evaluated at the current time step, with a similar approximation evaluated at the next time step. For the diffusion PDE example, this leads to the following scheme:

$$c_j(t + \Delta t) = c_j(t) + \mu \left\{ c_{j+1}(t + \Delta t) - 2c_j(t + \Delta t) + c_{j-1}(t + \Delta t) \right\} \quad (4.40)$$

where $\mu = \Delta t/h^2$. This finite difference algorithm is known as an *implicit* scheme, whereby the c terms on the right-hand side are determined from the next time step. This scheme represents a coupled system of equations that must be solved for at each step. Although such implicit schemes are more difficult to implement than their explicit counterparts, they are more stable. To see this, we can perform a Fourier analysis of the error by substituting the general solution form given by Eq. 4.39, $c(jh, n\Delta t) = \lambda^n e^{ik(jh)}$, into Eq. 4.40, to obtain:

$$\begin{aligned} c(jh, (n+1)\Delta t) &= c(jh, n\Delta t) + \mu \left\{ c((j+1)h, (n+1)\Delta t) \right. \\ &\quad \left. - 2c(jh, (n+1)\Delta t) + c((j-1)h, (n+1)\Delta t) \right\} \\ \lambda^{n+1} e^{ik(jh)} &= \lambda^n e^{ik(jh)} + \mu \left\{ \lambda^{n+1} e^{ik(j+1)h} - 2\lambda^{n+1} e^{ik(jh)} + \lambda^{n+1} e^{ik(j-1)h} \right\} \end{aligned}$$

Dividing through by $\lambda^n e^{ik(jh)}$, the above simplifies to:

$$\begin{aligned} \lambda &= 1 + \mu \left\{ \lambda e^{ikh} - 2\lambda + \lambda e^{-ikh} \right\} \\ &= 1 + \mu \lambda \left\{ 2 \cos(kh) - 2 \right\} \\ &= 1 - 4\mu \lambda \sin^2\left(\frac{1}{2}kh\right) \end{aligned}$$

On re-arranging, we obtain

$$\lambda = \frac{1}{1 + 4\mu \sin^2\left(\frac{1}{2}kh\right)}$$

Hence, for any choice of positive $\mu (= \Delta t/h^2)$, we have $0 \leq \lambda \leq 1$. This method is therefore unconditionally stable.

Example 4.14 Use the implicit FD algorithm of Eq. 4.40 to solve the diffusion PDE with boundary/initial conditions as given in Example 4.13, using a spatial grid size of $h = 0.01$ and $\Delta t = 0.0001$.

Answer: As in Example 4.13, we let c_j^n denote the value of c at $x = (j - 1)h$ and $t = (n - 1)\Delta t$, where j and n are integer values given by

$$j = 1, 2, \dots, \frac{1}{h} + 1, \quad n = 1, 2, \dots, \frac{0.005}{\Delta t} + 1$$

Implementing the implicit scheme of Eq. 4.40, we have:

$$c_j^{n+1} = c_j^n + \frac{\Delta t}{h^2} (c_{j+1}^{n+1} - 2c_j^{n+1} + c_{j-1}^{n+1})$$

which can be re-arranged to

$$-\left(\frac{\Delta t}{h^2}\right) c_{j+1}^{n+1} + \left(1 + \frac{2\Delta t}{h^2}\right) c_j^{n+1} - \left(\frac{\Delta t}{h^2}\right) c_{j-1}^{n+1} = c_j^n \tag{4.41}$$

To implement the zero-flux boundary conditions, we use the forward and backward FD approximations of the spatial derivative (Eqs. 4.31 and 4.32 respectively), but this time evaluated at time step $n + 1$ to obtain:

$$\frac{c_2^{n+1} - c_1^{n+1}}{h} = 0, \quad \frac{c_M^{n+1} - c_{M-1}^{n+1}}{h} = 0$$

or more simply

$$c_1^{n+1} - c_2^{n+1} = 0, \quad c_M^{n+1} - c_{M-1}^{n+1} = 0 \tag{4.42}$$

for all time steps $n + 1$, where $M = 1/h + 1$. Equations 4.41 and 4.42 can be written as the following matrix system of equations:

$$\begin{bmatrix} 1 & -1 & 0 & \dots & 0 & 0 & 0 \\ -\frac{\Delta t}{h^2} & 1 + \frac{2\Delta t}{h^2} & -\frac{\Delta t}{h^2} & \dots & 0 & 0 & 0 \\ \vdots & & \ddots & & & & \vdots \\ 0 & \dots & -\frac{\Delta t}{h^2} & 1 + \frac{2\Delta t}{h^2} & -\frac{\Delta t}{h^2} & \dots & 0 \\ \vdots & & & & \ddots & & \vdots \\ 0 & 0 & 0 & \dots & -\frac{\Delta t}{h^2} & 1 + \frac{2\Delta t}{h^2} & -\frac{\Delta t}{h^2} \\ 0 & 0 & 0 & \dots & 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} c_1^{n+1} \\ c_2^{n+1} \\ \vdots \\ c_j^{n+1} \\ \vdots \\ c_{M-1}^{n+1} \\ c_M^{n+1} \end{bmatrix} = \begin{bmatrix} 0 \\ c_2^n \\ \vdots \\ c_j^n \\ \vdots \\ c_{M-1}^n \\ 0 \end{bmatrix}$$

The following Matlab code implements this implicit scheme, solving the above matrix system of equations at each time step. The resulting concentration profile at $t = 0.005$ is plotted in Fig. 4.13, which compares well with the explicit scheme solution in

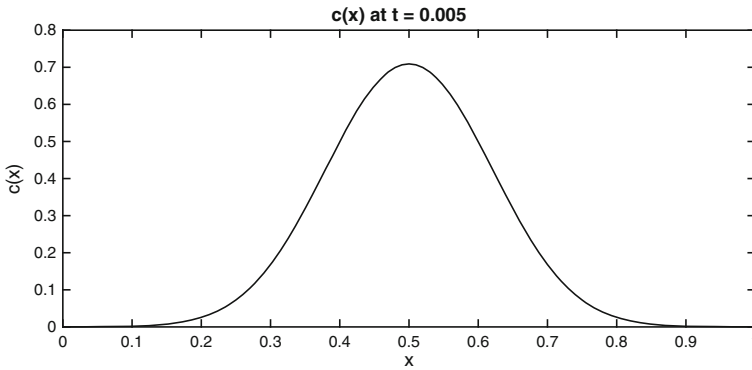


Fig. 4.13 Solution of 1D diffusion example at time $t = 0.005$ using the FD implicit scheme of Eq. 4.41 with $h = 0.01$ and $\Delta t = 0.0001$

Example 4.13. This time, however, a larger time step was used due to the superior stability of the implicit method.

```

h = 0.01;           % x-increment
dt = 0.0001;       % time-increment
T = 0.005;         % total time
M = round(1/h)+1;  % total x points
N = round(T/dt)+1; % total time points
c = zeros(M,N);    % concentration array

% initialise system matrix
A = diag((1+2*dt/h^2)*ones(M,1),0) + ...
    diag(-(dt/h^2)*ones(M-1,1),1) + ...
    diag(-(dt/h^2)*ones(M-1,1),-1);
A(1,1) = 1;
A(1,2) = -1;
A(M,M-1) = -1;
A(M,M) = 1;

% set initial waveform
for j = 1:M
    if ((j-1)*h >= 0.4)&&((j-1)*h <= 0.6)
        c(j,1) = 1;
    else
        c(j,1) = 0;
    end;
end;
end;
```

```

% solve for subsequent times
for n = 2:N
    b = c(:,n-1);
    b(1) = 0;
    b(M) = 0;
    c(:,n) = A\b;
end;

% plot solution
plot(0:h:1,c(:,end),'k'), xlabel('x'), ylabel('c(x)'),...
    title('c(x) at t = 0.005');
    
```

□

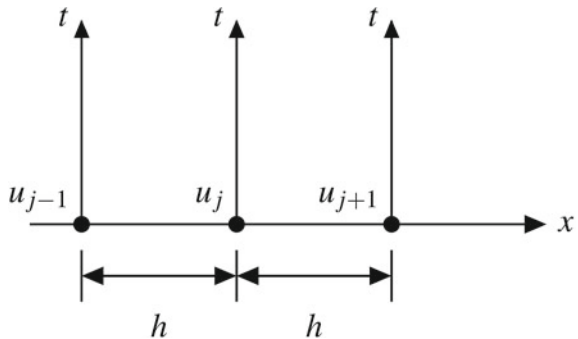
4.2.3 Method of Lines

Another common numerical technique used to solve time-dependent PDEs is the *method of lines*, in which the spatial domain is discretized into a regular grid to form a system of ODEs in the grid variables, as illustrated in Fig. 4.14. For example, to solve our familiar diffusion PDE:

$$\frac{\partial c}{\partial t} = \frac{\partial^2 c}{\partial x^2}$$

over the interval $x \in [0, 1]$ subject to zero-flux boundary conditions, we let c_j be the value of c at the nodes $x_j = (j - 1)h, j = 1 \dots M$, where $M = (1/h) + 1$. The above PDE can then be replaced by a coupled ODE system in $M - 2$ variables, by using a FD difference approximation to the second-order spatial derivative as follows:

Fig. 4.14 Method of lines for numerically solving a PDE in the dependent variable $u(x, t)$. $u(x, t)$ is discretized in the spatial domain using a regular grid-spacing of h , to obtain a series of time-dependent state variables u_j at each node. The PDE is thereby transformed into a regular ODE system in the nodal variables u_j



$$\frac{dc_j}{dt} = \frac{c_{j+1} - 2c_j + c_{j-1}}{h^2}, \quad j = 2 \dots M - 1 \quad (4.43)$$

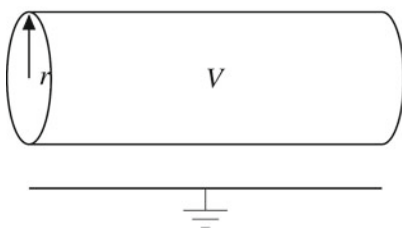
The zero-flux boundary conditions can be implemented by expressing the boundary variables c_1 and c_M in terms of interior c variables using a similar concept to Eq. 4.42:

$$\begin{aligned} c_1 &= c_2 \\ c_M &= c_{M-1} \end{aligned}$$

Example 4.15 Consider a cylindrical axonal nerve membrane of radius r , as shown in Fig. 4.15. The intracellular potential at each point is denoted by V , which is a function of both the 1D spatial position x and time t . Assuming the extracellular potential is everywhere set to ground, then V will also represent the transmembrane potential. The axon is assumed to have an axoplasmic (i.e. intracellular) resistivity of ρ_i ($\Omega \text{ m}$), a membrane capacitance of C_m (F m^{-2}), and a membrane ionic current of i_{ion} (A m^{-2}), the latter governed by the Hodgkin-Huxley formulations, as described in Sect. 2.3.2 on p. 42. In that earlier example, the axon was assumed to be space-clamped, in which V was a function of time alone with its value shared across all spatial positions of the axon. In the present example however, V is allowed to vary in both space and time. i_{ion} in the Hodgkin-Huxley formulations is given by:

$$\begin{aligned} i_{ion} &= \bar{g}_{Na} m^3 h (V - V_{Na}) + \bar{g}_K n^4 (V - V_K) + \bar{g}_L (V - V_L) \\ \frac{dn}{dt} &= \alpha_n(V) (1 - n) - \beta_n(V) n \\ \frac{dm}{dt} &= \alpha_m(V) (1 - m) - \beta_m(V) m \\ \frac{dh}{dt} &= \alpha_h(V) (1 - h) - \beta_h(V) h \end{aligned}$$

where \bar{g}_{Na} , \bar{g}_K , \bar{g}_L , V_{Na} , V_K and V_L are parameters, and n , m , h are time-dependent gating variables with rate coefficients (in units of s^{-1}) given by:



ρ_i : axoplasmic resistivity ($\Omega \text{ m}$)

C_m : membrane capacitance (F m^{-2})

i_{ion} : membrane ionic current (A m^{-2})

Fig. 4.15 Nerve cable model

$$\begin{aligned} \alpha_n &= \frac{10(V+50)}{1-\exp\left[\frac{-(V+50)}{10}\right]} & \beta_n &= 125 \exp\left[\frac{-(V+60)}{80}\right] \\ \alpha_m &= \frac{100(V+35)}{1-\exp\left[\frac{-(V+35)}{10}\right]} & \beta_m &= 4000 \exp\left[\frac{-(V+60)}{18}\right] \\ \alpha_h &= 70 \exp\left[\frac{-(V+60)}{20}\right] & \beta_h &= \frac{1000}{1+\exp\left[\frac{-(V+30)}{10}\right]} \end{aligned}$$

where V is in units of mV. Model parameters are $r = 0.0025$ cm, $\rho_i = 600 \Omega$ cm, with remaining parameters as given earlier in Table 2.2.

(a) By discretizing the x domain into spatial units of width Δx , derive the underlying PDE governing V in the limit as $\Delta x \rightarrow 0$.

(b) Solve the PDE using the method of lines for a fixed cable length of 1 cm, assuming zero-flux boundary conditions for V at each end of the cable, and initial values of variables given by

$$\begin{aligned} V(x, 0) &= \begin{cases} 20 \text{ mV} & 0 \leq x \leq 0.01 \text{ cm} \\ -60 \text{ mV} & \text{otherwise} \end{cases} \\ n(x, 0) &= 0.3177 \\ m(x, 0) &= 0.0529 \\ h(x, 0) &= 0.5961 \end{aligned}$$

Answer: (a) To derive the underlying PDE of this system, we discretize the nerve into segments of length Δx . The electrical equivalent circuit of three adjacent nodes is shown in Fig. 4.16.

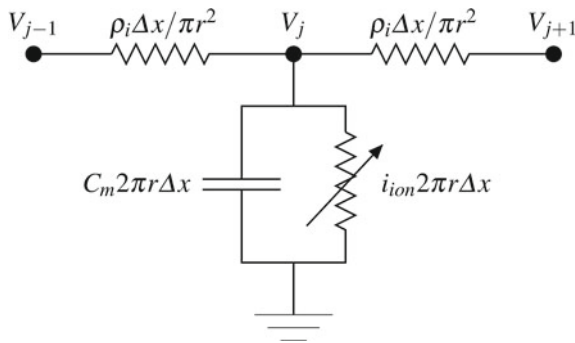


Fig. 4.16 Electrical equivalent circuit representation of three adjacent nodes, V_{j-1} , V_j , and V_{j+1} in the discretized nerve cable model. The internodal separation is given by Δx , r denotes the radius of the axon, and ρ_i is the axoplasmic resistivity. Membrane capacitance C_m and ionic current i_{ion} are given per unit membrane area

To determine the axoplasmic resistance between two nodes, we make use of *Pouillet's law*, which states that the resistance R of a material specimen of length L , resistivity ρ , and cross-sectional area A is given by:

$$R = \frac{\rho L}{A} \quad (4.44)$$

Substituting $\rho = \rho_i$, $L = \Delta x$ and $A = \pi r^2$ into Eq. 4.44, we obtain an intracellular resistance of $\rho_i \Delta x / \pi r^2$ between two nodes, as shown in Fig. 4.16. Furthermore, the total membrane capacitance of a cylindrical nerve segment of length Δx is given by C_m multiplied by the area of its curved surface, or $C_m 2\pi r \Delta x$. A similar argument follows for the total membrane ionic current of the segment, namely $i_{ion} 2\pi r \Delta x$. These values are also shown in Fig. 4.16. From Kirchhoff's current law, the total current entering node j must equal the current leaving through the parallel membrane capacitance and ionic pathways. This can be expressed by the equation

$$\overbrace{\frac{V_{j+1} - V_j}{\rho_i \Delta x / \pi r^2} + \frac{V_{j-1} - V_j}{\rho_i \Delta x / \pi r^2}}^{\text{current entering}} = \overbrace{C_m 2\pi r \Delta x \frac{dV_j}{dt} + i_{ion} 2\pi r \Delta x}_{\text{current leaving}}$$

On re-arranging, this becomes:

$$\frac{V_{j+1} - 2V_j + V_{j-1}}{\Delta^2 x} = \left(\frac{2\rho_i}{r} \right) \left[C_m \frac{dV_j}{dt} + i_{ion} \right] \quad (4.45)$$

The left-hand side of this equation is simply the FD approximation of the second-order derivative of V (see Eq. 4.35). Hence, in the limit as $\Delta x \rightarrow 0$, Eq. 4.45 becomes

$$\frac{\partial^2 V}{\partial x^2} = \left(\frac{2\rho_i}{r} \right) \left[C_m \frac{\partial V}{\partial t} + i_{ion} \right] \quad (4.46)$$

where C_m , ρ_i and r are assumed to be fixed along the nerve.

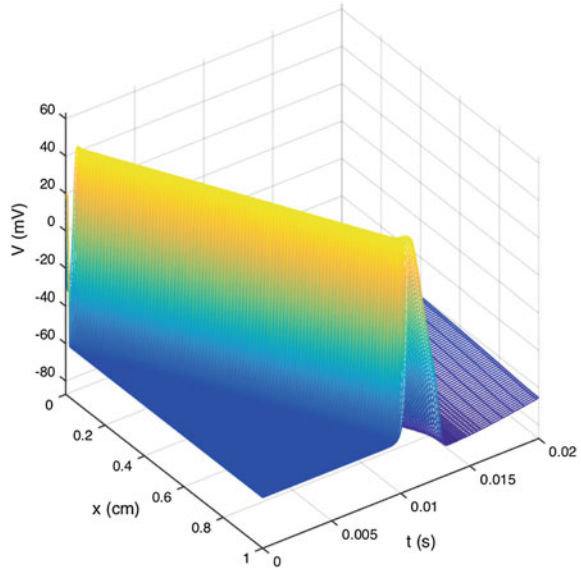
(b) To solve Eq. 4.46 using the method of lines, we utilise the FD approximation of Eq. 4.45, re-arranged as:

$$\frac{dV_j}{dt} = \frac{1}{C_m} \left[\frac{r}{2\rho_i} \left(\frac{V_{j+1} - 2V_j + V_{j-1}}{\Delta^2 x} \right) - i_{ion} \right] \quad j = 2 \dots M - 1$$

representing an ODE system in $M - 2$ variables, where $M = 1/\Delta x + 1$. Zero-flux boundary conditions are implemented by specifying the boundary values V_1 and V_M as

$$\begin{aligned} V_1 &= V_2 \\ V_M &= V_{M-1} \end{aligned}$$

Fig. 4.17 Solution of nerve cable PDE (Eq. 4.46) using the method of lines, as implemented by the Matlab code HH_cable_solve.m



These can be substituted into the expressions for dV_2/dt and dV_{M-1}/dt to obtain the following set of ODEs:

$$\frac{dV_j}{dt} = \begin{cases} \frac{1}{C_m} \left[\frac{r}{2\rho_i} \left(\frac{V_3 - 2V_2 + V_1}{\Delta^2 x} \right) - i_{ion} \right] & j = 2 \\ \frac{1}{C_m} \left[\frac{r}{2\rho_i} \left(\frac{V_{j+1} - 2V_j + V_{j-1}}{\Delta^2 x} \right) - i_{ion} \right] & j = 3 \cdots M - 2 \\ \frac{1}{C_m} \left[\frac{r}{2\rho_i} \left(\frac{V_{M-1} - 2V_{M-2} + V_{M-3}}{\Delta^2 x} \right) - i_{ion} \right] & j = M - 1 \end{cases} \quad (4.47)$$

The above ODE system is implemented and solved in the following Matlab code using a spatial resolution of $\Delta x = 0.01$ cm. The code is divided into two Matlab files: HH_cable_prime.m (which outputs the right-hand side derivatives of the ODE system in Eq. 4.47, and HH_cable_solve.m which calls the in-built Matlab solver ode15s to solve the ODE system, plotting the solution for V as a function of x and t as shown in Fig. 4.17. The action potential is seen to travel along the axon with constant propagation velocity, taking around 10 ms to travel a distance of 1 cm, corresponding to a velocity of approximately 1 m s^{-1} .

HH_cable_prime.m

```
function y_prime = HH_cable_prime(t,y)
% calculates derivatives for solving the HH nerve cable.
global dx Cm g_Na g_K g_L rho_i r;
y_prime = zeros(size(y));
M2 = length(y)/4;
for j = 1:M2
```

```

V = y(4*j-3);
n = y(4*j-2);
m = y(4*j-1);
h = y(4*j);
alpha_n = 10*(V+50)/(1-exp(-(V+50)/10));
beta_n = 125*exp(-(V+60)/80);
alpha_m = 100*(V+35)/(1-exp(-(V+35)/10));
beta_m = 4000*exp(-(V+60)/18);
alpha_h = 70*exp(-(V+60)/20);
beta_h = 1000/(1+exp(-(V+30)/10));
i_K = g_K*n^4*(V+72);
i_Na = g_Na*m^3*h*(V-55);
i_L = g_L*(V+49.387);
i_ion = i_K+i_Na+i_L;
if (j == 1) % zero-flux b.c.
    y_prime(1) = (r/(2*rho_i*dx^2)*(y(5)-2*y(1)+y(1))-i_ion)/Cm;
elseif (j == M2) % zero-flux b.c.
    y_prime(4*M2-3) = (r/(2*rho_i*dx^2)*...
        (y(4*M2-7)-2*y(4*M2-3)+y(4*M2-3))-i_ion)/Cm;
else
    y_prime(4*j-3) = (r/(2*rho_i*dx^2)*...
        (y(4*j-7)-2*y(4*j-3)+y(4*j+1))-i_ion)/Cm;
end;
y_prime(4*j-2) = alpha_n*(1-n)-beta_n*n;
y_prime(4*j-1) = alpha_m*(1-m)-beta_m*m;
y_prime(4*j) = alpha_h*(1-h)-beta_h*h;
end;

```

HH_cable_solve.m

```

% solves the HH cable equations using the method of lines
global L dx Cm g_Na g_K g_L rho_i r;
L = 1; % axon length (cm)
dx = 0.01; % x_increment (cm)
M = round(L/dx)+1; % total nodal points
Cm = 1; % uF/cm^2
g_Na = 120000; % uS/cm^2
g_K = 36000; % uS/cm^2
g_L = 300; % uS/cm^2
rho_i = 0.0006; % Mohm.cm
r = 0.0025; % cm
Y_init = [];
for j = 1:M-2
    Y_init = [Y_init, -60, 0.3177, 0.0529, 0.5961];
end;
Y_init(1) = 20; % initial conditions at x(0) (== V_2)
[time, Y] = ode15s('HH_cable_prime', [0 0.02], Y_init);
% plot
VV = [Y(:,1), Y(:,1:4:end), Y(:,end-3)]; % V (incl. boundary values)
[TT, XX] = meshgrid(time,0:dx:L);
mesh(XX,TT,VV'), xlabel('x (cm)'), ylabel('t (s)'), zlabel('V (mV)');

```

indexMatlab!example code!Hodgkin-Huxley nerve cable—



4.3 Further Reading

A useful reference on vector calculus, including the divergence theorem, is the text of Marsden and Tromba [5]. Excellent treatment of finite difference methods for PDEs is given in the texts of Morton and Mayers [7] and Eriksson et al. [3]. Interesting applications of PDE modelling in physical systems in general may be found in the texts of van Groesen and Molenaar [9], Lin and Segel [4] and Cumberbatch and Fitt [2]. Older texts which cover analytical techniques for solving PDEs in a wide range of physical systems include those of Menzel [6] and Webster [10].

Problems

4.1 Determine the gradient of the following scalar functions:

(a) $f(x, y, z) = 2x - y + 3z$

(b) $g(x, y, z) = x^2 + y^2 + z^2$

(c) $h(x, y, z) = \frac{x}{y}$

4.2 Find the divergence and the curl of the following vector functions:

(a) $\mathbf{u}(x, y, z) = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$

(b) $\mathbf{v}(x, y, z) = \begin{pmatrix} -y \\ x \\ 1 \end{pmatrix}$

(c) $\mathbf{w}(x, y, z) = \begin{pmatrix} 2x + 3y - z \\ x - y + z \\ x + 2y + 3z \end{pmatrix}$

4.3 Verify the following identities for arbitrary vector fields \mathbf{u} and \mathbf{v} :

(a) $\nabla \cdot (\nabla \times \mathbf{u}) = 0$

(b) $\nabla \cdot (\mathbf{u} \times \mathbf{v}) = \mathbf{v} \cdot (\nabla \times \mathbf{u}) - \mathbf{u} \cdot (\nabla \times \mathbf{v})$

(c) $\nabla \times (\nabla \times \mathbf{u}) = \nabla(\nabla \cdot \mathbf{u}) - \nabla^2 \mathbf{u}$

4.4 Show that the volume of an arbitrary closed region bounded by surface S is given by

$$V = \int_S n_x x \, dS = \int_S n_y y \, dS = \int_S n_z z \, dS$$

where n_x, n_y, n_z are the x -, y -, and z -components of the outward normal to the surface respectively.

4.5 A drug-eluting microsphere of radius R is implanted into a tumour, having a fixed concentration c_0 of drug within its volume as well as on its outer surface. The drug diffuses into the surrounding interstitial space with diffusion coefficient D , and is taken up into tumour cells at a rate of $k_{up}c$, where c is the local drug concentration

and k_{up} is the cell uptake rate.

(a) Assuming the tumour tissue and cells can be represented by a continuum, and that the model is spherically symmetric, derive the underlying PDE for the drug concentration, $c(r, t)$, as a function of radial distance r from the centre of the microsphere and time t .

(b) Using the substitution $c = u/r$, transform this PDE in terms of u , and hence determine the analytical solution for the steady-state concentration profile $c_\infty(r)$.

4.6 A large room is full of people arranged in a square grid, each person surrounded by four neighbours a distance of h away. Each person has a varying amount of coins in their possession c . At successive time frames spaced Δt apart, the distribution of coins changes according to given rules. In the limit as $h \rightarrow 0$ and $\Delta t \rightarrow 0$, derive the governing PDE for each of the three rules below:

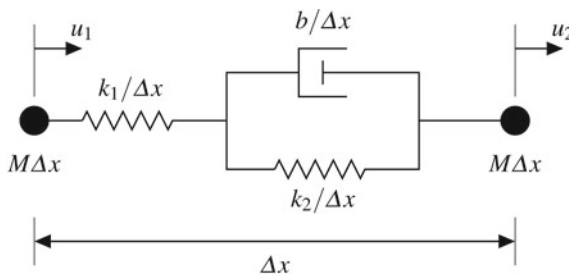
(a) Each person gives away a fixed proportion of the coins they currently have to each of their four neighbours. This proportion is the same for each person.

(b) As above, each person gives away a fixed proportion of the coins they have to each of their four neighbours. However, they simultaneously receive a fixed income of coins per unit time. The income and the proportion of coins given away represent system parameters, which are the same for every person in the room.

(c) Each person receives a fixed proportion of coins per unit time from each of their neighbours. They also give away a proportion of coins they currently have per unit time to charity. The amount of coins received per unit time and the proportion given to charity per unit time are parameters identical for every person.

4.7 For the 1D axonal nerve cable of Example 4.15, derive the PDE for the transmembrane potential V , this time for the case when axon radius r is assumed to vary along the length of the nerve.

4.8 A strand of cardiac papillary muscle of resting length L and mass M per unit length may be discretized into multiple 1D sub-units of length Δx as shown in the figure below. The sub-units are characterised by three parameters: a series linear spring of stiffness k_1 , a parallel linear spring of stiffness k_2 , and a viscous dashpot component of value b . These parameters scale with Δx as shown in the figure, but otherwise are assumed to remain constant during deformation of the muscle. The sub-unit connects two lumped point masses, each of mass $M\Delta x$.



Displacements of u_1 and u_2 are applied to each end of each sub-unit, as shown in the figure. If the change in length of the series element is denoted by l_1 and the change in length of the parallel dashpot/stiffness combination is given by l_2 , then the magnitude of forces produced by each of these components opposing the length changes are given by:

$$\begin{aligned} \text{Series spring:} & \quad \left(\frac{k_1}{\Delta x}\right) l_1 \\ \text{Parallel spring:} & \quad \left(\frac{k_2}{\Delta x}\right) l_2 \\ \text{Parallel dashpot:} & \quad \left(\frac{b}{\Delta x}\right) \frac{dl_2}{dt} \end{aligned}$$

In the limit as $\Delta x \rightarrow 0$, derive the PDE for the displacement u at each point along the muscle.

4.9 Initiation of the mammalian heartbeat normally occurs in the specialised pacemaker cells of the sinoatrial node (SAN), located in the right atrial wall of the heart. From there, the electrical impulse spreads across the surface of the atria before reaching the ventricles. Disruptions to atrial excitation and/or propagation represent the most common forms of arrhythmias encountered clinically. One such arrhythmia, atrial flutter, arises from self-sustained periodic excitation of the atria. In this problem, we will simulate the onset of atrial flutter using a modified form of the Rogers-McCulloch PDE formulations [8] for the cardiac action potential:

$$\begin{aligned} \beta C_m \left(\frac{\partial V_m}{\partial t} \right) &= \nabla \cdot (\sigma \nabla V_m) - \beta i_{ion} + i_{stim} \\ \frac{\partial u}{\partial t} &= e (V_m - du - b) \end{aligned}$$

with

$$i_{ion} = c_1 (V_m - a) (V_m - A) (V_m - B) + c_2 u (V_m - B)$$

where u is an auxiliary ‘recovery’ variable, σ is the electrical conductivity within the atrial tissue, β is the tissue surface to volume ratio, C_m is cell membrane capacitance per unit area, i_{ion} is the ionic current per unit cell membrane area, i_{stim} is the applied stimulus current per unit tissue volume, and A, B, a, b, d, e, c_1 and c_2 are parameters describing the active electrical activity of the atria, as given in the table below. Initial values at $t = 0$ throughout the tissue are $V_m = -85$ mV and $u = 0$.

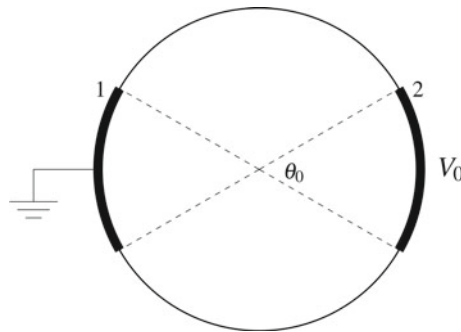
The atrial tissue domain is a 2D square of side-length 10 cm, with the left-hand corner located at $x = 0, y = 0$, where x, y represent the 2D spatial coordinates. Zero-flux boundary conditions for V_m apply on all four external boundaries of the domain. The stimulus current is given by:

$$i_{stim}(x, y, t) = \begin{cases} 50 \text{ A m}^{-3} & y \leq 1 \text{ cm}, 10 \text{ ms} \leq t \leq 11 \text{ ms} \\ 50 \text{ A m}^{-3} & x \leq 1 \text{ cm}, 150 \text{ ms} \leq t \leq 151 \text{ ms} \\ 0 & \text{otherwise} \end{cases}$$

- (a) Using the method of lines, solve this PDE system for a spatial discretization of 51×51 , plotting V_m at times $t = 0.3, 0.35, 0.4,$ and 0.45 s.
- (b) Solve the PDE using an explicit finite difference scheme for the same spatial discretization, with a fixed time step Δt of 1×10^{-5} s. As above, plot the solution for V_m for $t = 0.3, 0.35, 0.4,$ and 0.45 s.

| Parameter | Value | Parameter | Value |
|-----------|---------------------------------------|-----------|---------------------------------------|
| A | 55 mV | σ | 0.001 S m^{-1} |
| B | -85 mV | c_1 | $530 \text{ S V}^{-2} \text{ m}^{-2}$ |
| a | -66.8 mV | c_2 | $400 \mu \text{ S cm}^{-2}$ |
| b | -85 mV | C_m | $1 \mu \text{ F cm}^{-2}$ |
| d | 140 mV | β | 100 m^{-1} |
| e | $285.7 \text{ V}^{-1} \text{ s}^{-1}$ | | |

4.10 An experiment is to be conducted to determine the effect of electric fields on cells in a Petri dish. The electric field is delivered using an identical pair of wire electrodes formed from circular arcs placed at opposite ends of the dish against its walls, as shown in the figure below. Each electrode subtends an angle of θ_0 at the centre of the dish, whose radius is R . The electrical conductivity of the medium in which the cells are placed is constant at all points in the dish, and electrode 1 is grounded whilst electrode 2 has a voltage of V_0 applied to it. All other boundaries are electrically-insulating: that is, the voltage gradient in the normal direction of the boundary is zero.



- (a) Determine the PDE governing the voltage distribution in the Petri dish in terms of polar coordinates (r, θ) , such that $r = 0$ corresponds to the centre of the dish, and $\theta = 0$ corresponds to the Cartesian x-axis bisecting both electrodes on the dish boundary.
- (b) Using the method of finite differences, solve the above PDE to obtain the voltage distribution for $R = 45$ mm, $\theta_0 = 60^\circ$ and $V_0 = 2$ V.
- (c) Using the same R and V_0 values as in (b), estimate the optimal value for the electrode angle θ_0 that yields the largest area in which the electric field magnitude is uniformly within $\pm 10\%$ of its value at the centre of the dish.

References

1. Bowman F (1958) Introduction to Bessel functions. Dover, New York
2. Cumberbatch E, Fitt A (2001) Mathematical modeling: case studies from industry. Cambridge University Press, Cambridge
3. Eriksson E, Estep D, Hansbo P, Johnson C (1996) Computational differential equations. Cambridge University Press, Cambridge
4. Lin CC, Segel LA (1988) Mathematics applied to deterministic problems in the natural sciences. SIAM, Philadelphia
5. Marsden JE, Tromba AJ (2003) Vector calculus, 5th edn. W H Freeman, New York
6. Menzel DA (1961) Mathematical physics, Dover edn. Dover, New York
7. Morton KW, Mayers DF (2005) Numerical solution of partial differential equations. Cambridge University Press, Cambridge
8. Rogers JM, McCulloch AD (1994) A collocation-Galerkin finite element model of cardiac action potential propagation. *IEEE Trans Biomed Eng* 41:743–757
9. van Groesen E, Molenaar J (2007) Continuum modeling in the physical sciences. SIAM, Philadelphia
10. Webster AG (1955) Partial differential equations of mathematical physics, Dover edn. Dover, New York

Chapter 5

The Finite Element Method

The *finite element method* (FEM) represents a well-established and widespread numerical approach for solving partial differential equations in physics and engineering. The basics of the method were first published in 1943 in seminal paper by Richard Courant [4], who solved 2D plate and plane torsion problems by discretizing the spatial domain into triangular “elements”, resulting in piecewise approximations of the displacements to be solved for. In the early years, the main application of FEM was in the area of structural mechanics [12], but its use quickly spread to other engineering applications, including electromagnetics [5, 10], and fluid dynamics. Today, the method is used in a numerous range of physics applications, including bioengineering [8]. A major advantage of FEM over the finite difference method is that irregular geometries can be readily accommodated. This feature is particularly useful for bioengineering, in which organs, tissues and other structures of the body typically exhibit complex anatomies.

5.1 Finite Elements for 1D Systems

The finite element method is based on two important concepts:

1. Formulation of the underlying PDE in terms of its so-called *weak* form.
2. Approximating the dependent variable in the weak form using a finite sum of *basis functions*.

Both of these concepts will be illustrated using our familiar example of 1D diffusion, represented by the PDE

$$\frac{\partial c}{\partial t} = D \frac{\partial^2 c}{\partial x^2}$$

defined over the domain $x \in [0, 1]$, where D is a fixed parameter denoting the diffusion coefficient. We assume an initial value of $c_0(x)$ at $t = 0$, where c_0 is a given function. Furthermore, we impose a Dirichlet boundary condition at $x = 0$

and a Neumann boundary condition at $x = 1$, according to

$$\begin{aligned} c|_{x=0} &= p(t) \\ D \frac{\partial c}{\partial x} \Big|_{x=1} &= q(t) \end{aligned}$$

where $p(t), q(t)$ are given functions of time. Note that $q(t)$ represents the inward diffusional flux at $x = 1$.¹ The PDE with its associated boundary and initial conditions comprise what is referred to as the *strong form* representation, given by:

Find $c(x, t), x \in [0, 1]$, such that

$$\begin{aligned} \frac{\partial c}{\partial t} &= D \frac{\partial^2 c}{\partial x^2} \\ c(x, 0) &= c_0(x), \quad c(0, t) = p(t), \quad D \frac{\partial c}{\partial x} \Big|_{x=1} = q(t) \end{aligned} \quad (5.1)$$

5.1.1 Weak Form PDE Equivalent

To derive the equivalent weak form of Eq. 5.1, let H^1 be the set of all functions with finite square-derivative integrals.² That is, for all $u \in H^1$:

$$\int_0^1 \left(\frac{\partial u}{\partial x} \right)^2 dx < \infty$$

We now define a two sets of functions: the set of *trial solutions*, S , and the set of *test functions*, V , as follows:

$$\begin{aligned} S &= \{c(x) : c(x, t) \in H^1, c(x, 0) = c_0(x), c(0, t) = p(t)\} \\ V &= \{w(x) : w(x) \in H^1, w(0) = 0\} \end{aligned}$$

Multiplying both sides of the original PDE by any such test function $w(x) \in V$, then integrating over the spatial domain, we obtain:

$$\int_0^1 \frac{\partial c}{\partial t} w dx = \int_0^1 D \frac{\partial^2 c}{\partial x^2} w dx$$

¹From Fick's Law of diffusion (see Exercise 4.6).

²The set H^1 is an example of a *Sobolev space* [1].

$$\begin{aligned} \frac{d}{dt} \int_0^1 cw \, dx &= \left[D \frac{\partial c}{\partial x} w \right]_0^1 - \int_0^1 D \frac{\partial c}{\partial x} \frac{dw}{dx} \, dx \quad (\text{using integration by parts}) \\ &= q(t)w(1) - \int_0^1 D \frac{\partial c}{\partial x} \frac{dw}{dx} \, dx \quad (\text{using Eq. 5.1 and } w(0) = 0) \end{aligned}$$

Re-arranging by bringing all the integrals to one side, we obtain the weak form equivalent of Eq. 5.1, including all boundary and initial conditions, as:

Find $c \in S$ such that

$$\frac{d}{dt} \int_0^1 cw \, dx + \int_0^1 D \frac{\partial c}{\partial x} \frac{dw}{dx} \, dx = q(t)w(1) \tag{5.2}$$

for $\forall w \in V$.

It is evident from the above derivation that any solution of the strong form Eq. 5.1 also satisfies the weak equivalent Eq. 5.2. To show that the converse is also true, we proceed with the following steps:

1. Integrate Eq. 5.2 by parts to obtain

$$\begin{aligned} \frac{d}{dt} \int_0^1 cw \, dx + \left[D \frac{\partial c}{\partial x} w \right]_0^1 - \int_0^1 D \frac{\partial^2 c}{\partial x^2} w \, dx &= q(t)w(1) \\ \frac{d}{dt} \int_0^1 cw \, dx + w(1) \left[D \frac{\partial c}{\partial x} \right]_{x=1} - \int_0^1 D \frac{\partial^2 c}{\partial x^2} w \, dx &= q(t)w(1) \end{aligned}$$

where we have used $w(0) = 0$ (since $w \in V$). We can now integrate all terms in the above equation with respect to time from $t = 0$ to T , where T is arbitrary, to obtain

$$\left[\int_0^1 cw \, dx \right]_0^T + w(1) \int_0^T \left[D \frac{\partial c}{\partial x} \right]_{x=1} dt - \int_0^T \int_0^1 D \frac{\partial^2 c}{\partial x^2} w \, dx \, dt = w(1) \int_0^T q(t) \, dt$$

Re-arranging the above, we have

$$\int_0^1 \left[c(x, T) - c_0(x) - \int_0^T D \frac{\partial^2 c}{\partial x^2} dt \right] w \, dx + w(1) \int_0^T \left(\left[D \frac{\partial c}{\partial x} \right]_{x=1} - q(t) \right) dt = 0 \tag{5.3}$$

where we have used the initial condition $c(x, 0) = c_0(x)$.

2. We note that Eq. 5.3 is satisfied if both $c(x, T) - c_0(x) - \int_0^T D \frac{\partial^2 c}{\partial x^2} dt = 0$ and $\left[D \frac{\partial c}{\partial x} \right]_{x=1} - q(t) = 0$ hold. We now proceed to verify both of these in the steps below.
3. To show that $c(x, T) - c_0(x) - \int_0^T D \frac{\partial^2 c}{\partial x^2} dt = 0$, we choose the following test function w in Eq. 5.3:

$$w(x) = \left[c(x, T) - c_0(x) - \int_0^T D \frac{\partial^2 c}{\partial x^2} dt \right] \phi(x)$$

where $\phi(x) \in H^1$, $\phi(x) \geq 0$ for $x \in [0, 1]$ and $\phi(0) = \phi(1) = 0$. Clearly, for this choice of w , $w(0) = 0$, and hence $w \in V$. Substituting this test function into Eq. 5.3, we obtain:

$$\int_0^1 \left[c(x, T) - c_0(x) - \int_0^T D \frac{\partial^2 c}{\partial x^2} dt \right]^2 \phi(x) dx + \left[c(x, T) - c_0(x) - \int_0^T D \frac{\partial^2 c}{\partial x^2} dt \right]_{x=1} \phi(1) \int_0^T \left(\left[D \frac{\partial c}{\partial x} \right]_{x=1} - q(t) \right) dt = 0$$

and since $\phi(1) = 0$, the above reduces to

$$\int_0^1 \left[c(x, T) - c_0(x) - \int_0^T D \frac{\partial^2 c}{\partial x^2} dt \right]^2 \phi(x) dx = 0$$

Furthermore, since the integrand is always ≥ 0 , it follows that the above integral can only be zero if

$$c(x, T) - c_0(x) - \int_0^T D \frac{\partial^2 c}{\partial x^2} dt = 0$$

as required.

4. To show that $\left[D \frac{\partial c}{\partial x} \right]_{x=1} - q(t) = 0$, we substitute the above result into Eq. 5.3 to obtain

$$w(1) \int_0^T \left(\left[D \frac{\partial c}{\partial x} \right]_{x=1} - q(t) \right) dt = 0$$

The requirement that $w \in V$ provides no restriction whatsoever on its value at $x = 1$, namely $w(1)$. Hence, for the left-hand side of above expression to be zero, we must have

$$\int_0^T \left(\left[D \frac{\partial c}{\partial x} \right]_{x=1} - q(t) \right) dt = 0$$

Since T is arbitrary, the above integral must hold for all $T \geq 0$. This can only be true if the integrand itself is identically equal to zero, namely:

$$\left[D \frac{\partial c}{\partial x} \right]_{x=1} - q(t) = 0$$

as required.

5. From Step 3 above, we have shown that

$$c(x, T) - c_0(x) - \int_0^T D \frac{\partial^2 c}{\partial x^2} dt = 0$$

which is equivalent to

$$\int_0^T \frac{\partial c}{\partial t} dt - \int_0^T D \frac{\partial^2 c}{\partial x^2} dt = 0$$

or

$$\int_0^T \left[\frac{\partial c}{\partial t} - D \frac{\partial^2 c}{\partial x^2} \right] dt = 0$$

Since this integral must hold for any arbitrary choice of $T \geq 0$, then the integrand must identically be equal to zero, namely

$$\frac{\partial c}{\partial t} - D \frac{\partial^2 c}{\partial x^2} = 0$$

or

$$\frac{\partial c}{\partial t} = D \frac{\partial^2 c}{\partial x^2}$$

which satisfies the original PDE. Furthermore, since $\left[D \frac{\partial c}{\partial x} \right]_{x=1} - q(t) = 0$ (from Step 4), then the required Neumann boundary condition is satisfied at $x = 1$. Finally, since the weak form of the PDE requires $c \in S$, we have $c(x, 0) = c_0(x)$ and $c(0, t) = g(t)$. Hence, the required initial and Dirichlet boundary conditions are also satisfied. We have therefore shown that any solution of the PDE weak form Eq. 5.2 also satisfies its strong form equivalent (Eq. 5.1).

In obtaining the weak form equivalent of a PDE system, the following points are important to note:

- The weak form is obtained by first multiplying the PDE by a suitable test function.
- Integration by parts is then used to obtain the weak form equivalent.
- Finally, we note that in the weak form expression of Eq. 5.2, the Dirichlet boundary condition at $x = 0$ is explicitly enforced by making this condition a property of the set of trial solutions S . However, the Neumann boundary condition at $x = 1$ is not explicitly enforced, but is rather embedded in the derivation of the weak form expression itself. Boundary conditions that are not enforced in this way are known as *natural boundary conditions*. In contrast, those that are explicitly specified in the set of trial solutions are referred to as *essential boundary conditions*.

5.1.2 Basis Function Approximation

To solve the PDE equivalent weak form using the method of finite elements, the next step is to approximate the dependent variable using a finite sum of *basis functions*. For our weak form example of Eq. 5.2 the variable $c(x, t)$ is approximated by

$$c(x, t) \approx \sum_{i=1}^N c_i(t) \varphi_i(x) \quad (5.4)$$

where $\varphi_i(x)$ is the i th basis function, $c_i(t)$ is the i th time-dependent coefficient, and N is the total number of basis functions. The basis functions are chosen with the following properties:

- they have small spatial support: that is, they are non-zero only within small regions, referred to as *elements*.
- Within each element, a number of *nodes* are specified at which the solution of the PDE is desired. The basis functions are chosen such that they equal unity at one specific node and zero at all other nodes, namely

$$\varphi_i(x_j) = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

The above properties of the basis functions mean that the coefficients $c_i(t)$ are simply the value of the dependent variable $c(x, t)$ evaluated at the nodes at each time t , since

$$\begin{aligned} c(x_j, t) &= \sum_{i=1}^N c_i(t) \varphi_i(x_j) \\ &= c_j(t) \varphi_j(x_j) \\ &= c_j(t) \end{aligned}$$

In 1D, a simple choice for the $\varphi_i(x)$ are the *linear basis functions*, given by

$$\varphi_i(x) = \begin{cases} \frac{x-x_{i-1}}{x_i-x_{i-1}} & x_{i-1} \leq x < x_i \\ \frac{x_{i+1}-x}{x_{i+1}-x_i} & x_i \leq x < x_{i+1} \\ 0 & x < x_{i-1}, x \geq x_{i+1} \end{cases}$$

where the x_i denote the node positions, as illustrated in Fig. 5.1, with each adjacent pair of nodes spanning a single element. These 1D linear basis functions are also known as “hat” functions, resembling an inverted “V”. The linear combination of these basis functions results in a piecewise linear approximation of $c(x, t)$, as shown in the upper curve of Fig. 5.1. Within element $x_i \leq x < x_{i+1}$, the variable $c(x, t)$ can be determined by linearly interpolating between its values at the nodes, $c_i(t)$ and $c_{i+1}(t)$, using

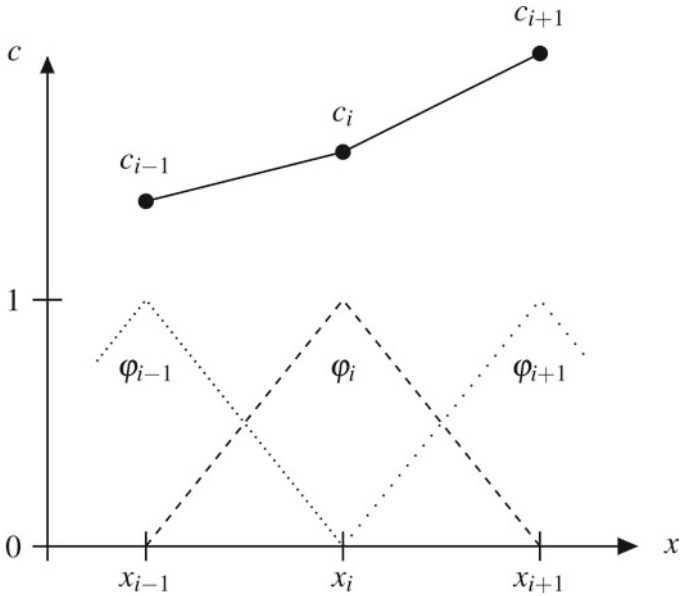


Fig. 5.1 1D linear basis or “witch’s hat” functions. Shown are three nodes, x_{i-1} , x_i and x_{i+1} spanning two adjacent elements. Upper trace shown a linear combination of these basis functions, producing a piecewise linear approximation of the dependent variable c

$$c(x, t) = c_i(t) \left(\frac{x_{i+1} - x}{x_{i+1} - x_i} \right) + c_{i+1}(t) \left(\frac{x - x_i}{x_{i+1} - x_i} \right)$$

The next step in solving the weak form Eq. 5.2 is to utilise a finite number of test functions $w_j(x)$, and require that the weak form holds for each of these test functions individually. The most common choice for the test functions is simply the basis functions themselves, namely

$$w_j(x) = \varphi_j(x) \quad (j = 1 \dots N)$$

This choice of test functions is known as the *Galerkin*³ method. Substituting the basis function approximation for $c(x, t)$ (Eq. 5.4) into the weak form

$$\frac{d}{dt} \int_0^1 c w \, dx + \int_0^1 D \frac{\partial c}{\partial x} \frac{dw}{dx} \, dx = q(t)w(1)$$

we obtain for each test function $\varphi_j(x)$:

³Pronounced ‘Gal-err-kin’. Named after Boris Grigoryevich Galerkin (1871–1945), a Soviet mathematician and engineer.

$$\begin{aligned} \frac{d}{dt} \int_0^1 \left(\sum_{i=1}^N c_i(t) \varphi_i(x) \right) \varphi_j(x) dx + \\ \int_0^1 D \frac{\partial}{\partial x} \left(\sum_{i=1}^N c_i(t) \varphi_i(x) \right) \frac{d\varphi_j(x)}{dx} dx = q(t) \varphi_j(1) \end{aligned}$$

which can be written as

$$\frac{dc_i(t)}{dt} \int_0^1 \left(\sum_{i=1}^N \varphi_i(x) \right) \varphi_j(x) dx + c_i(t) \int_0^1 D \frac{d}{dx} \left(\sum_{i=1}^N \varphi_i(x) \right) \frac{d\varphi_j(x)}{dx} dx = q(t) \varphi_j(1)$$

and taking out the summation operator, we obtain

$$\sum_{i=1}^N \left[\frac{dc_i(t)}{dt} \int_0^1 \varphi_i(x) \varphi_j(x) dx + c_i(t) \int_0^1 D \frac{d\varphi_i(x)}{dx} \frac{d\varphi_j(x)}{dx} dx \right] = q(t) \varphi_j(1)$$

which is equivalent to a matrix system of ODE's:

$$\mathbf{D} \frac{d\mathbf{c}}{dt} + \mathbf{K}\mathbf{c} = \mathbf{f}$$

where \mathbf{c} is the vector of coefficients $c_i(t)$, \mathbf{K} is the *stiffness matrix* with elements

$$K_{ij} = \int_0^1 D \frac{d\varphi_i(x)}{dx} \frac{d\varphi_j(x)}{dx} dx, \quad (5.5)$$

\mathbf{D} is the *damping matrix*, whose i, j th element is

$$D_{ij} = \int_0^1 \varphi_i(x) \varphi_j(x) dx \quad (5.6)$$

and \mathbf{f} is the *source or load vector*, with components

$$f_i = q(t) \varphi_i(1) = \begin{cases} 0 & i < N \\ q(t) & i = N \end{cases} \quad (5.7)$$

since $\varphi_i(1) = 0$ ($i < N$) and $\varphi_N(1) = 1$. Note that the i th diagonal element of the stiffness matrix is given by

$$K_{ii} = \int_0^1 D \left(\frac{d\varphi_i(x)}{dx} \right)^2 dx$$

which is why we required earlier that our test functions had square-integrable derivatives.

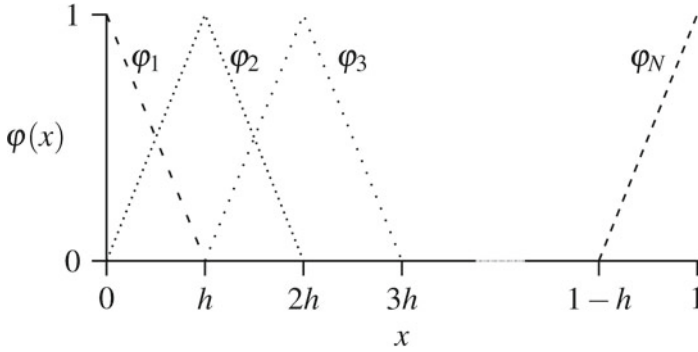


Fig. 5.2 N linear basis functions $\varphi_1(x) \cdots \varphi_N(x)$ defined on $x \in [0, 1]$ for equi-spaced nodes spaced h apart

To solve our PDE weak form (Eq. 5.2), we can choose linear basis functions such that the nodes x_i are regularly spaced, with internodal distance of h . For this choice of basis functions, we have

$$\varphi_i(x) = \begin{cases} \frac{h-x}{h} & i = 1, \quad x < h \\ \frac{x-(1-h)}{h} & i = N, \quad 1-h \leq x \leq 1 \\ \frac{x-(i-2)h}{h} & 1 < i < N, \quad (i-2)h \leq x < (i-1)h \\ \frac{ih-x}{h} & 1 < i < N, \quad (i-1)h \leq x < ih \\ 0 & \text{otherwise} \end{cases} \quad (5.8)$$

A plot of these basis functions is shown in Fig. 5.2, from which it can be seen that these are hat-shaped, with the exception of $\varphi_1(x)$ and $\varphi_N(x)$, which are “half-hats”. We are now in a position to evaluate the stiffness matrix \mathbf{K} , the damping matrix \mathbf{D} , and the load vector \mathbf{f} . For the stiffness matrix, we must evaluate

$$K_{ij} = \int_0^1 D \frac{d\varphi_i(x)}{dx} \frac{d\varphi_j(x)}{dx} dx$$

for all $i, j = 1 \dots N$. From Eq. 5.8, we note that

$$\frac{d\varphi_i(x)}{dx} = \begin{cases} -\frac{1}{h} & i = 1, \quad 0 \leq x < h \\ \frac{1}{h} & i = N, \quad 1-h \leq x \leq 1 \\ -\frac{1}{h} & 1 < i < N, \quad (i-1)h \leq x < ih \\ \frac{1}{h} & 1 < i < N, \quad (i-2)h \leq x < (i-1)h \\ 0 & \text{otherwise} \end{cases}$$

Using this result, we can deduce that

$$\frac{d\varphi_i(x)}{dx} \frac{d\varphi_j(x)}{dx} = \begin{cases} \frac{1}{h^2} & i = j = 1, & 0 \leq x < h \\ \frac{1}{h^2} & i = j = N, & 1 - h \leq x \leq h \\ \frac{1}{h^2} & 1 < i = j < N, & (i - 2)h \leq x < ih \\ -\frac{1}{h^2} & 1 < i = j + 1 \leq N, & (i - 2)h \leq x < (i - 1)h \\ -\frac{1}{h^2} & 1 \leq i = j - 1 < N, & (i - 1)h \leq x < ih \\ 0 & \text{otherwise} \end{cases}$$

and therefore

$$K_{ij} = \int_0^1 D \frac{d\varphi_i(x)}{dx} \frac{d\varphi_j(x)}{dx} dx = \begin{cases} \frac{D}{h} & i = j = 1 \\ \frac{D}{h} & i = j = N \\ \frac{2D}{h} & 1 < i = j < N \\ -\frac{D}{h} & i = j + 1, & 2 \leq i \leq N \\ -\frac{D}{h} & i = j - 1, & 1 \leq i \leq N - 1 \\ 0 & \text{otherwise} \end{cases} \quad (5.9)$$

Similarly for the damping matrix, we must evaluate its components

$$D_{ij} = \int_0^1 \varphi_i(x) \varphi_j(x) dx$$

To do so, we use Eq. 5.8 to determine that

$$\varphi_i(x) \varphi_j(x) = \begin{cases} \frac{[h-x]^2}{h^2} & i = j = 1, & 0 \leq x \leq h \\ \frac{[x-(1-h)]^2}{h^2} & i = j = N, & 1 - h \leq x \leq 1 \\ \frac{[x-(i-2)h]^2}{h^2} & 1 < i = j < N, & (i - 2)h \leq x < (i - 1)h \\ \frac{[ih-x]^2}{h^2} & 1 < i = j < N, & (i - 1)h \leq x < ih \\ \frac{[x-(i-2)h][i-1)h-x]}{h^2} & i = j + 1, & 2 \leq i \leq N, & (i - 2)h \leq x < (i - 1)h \\ \frac{[x-(i-1)h][ih-x]}{h^2} & i = j - 1, & 1 \leq i \leq N - 1, & (i - 1)h \leq x < ih \\ 0 & \text{otherwise} \end{cases}$$

For the case $i = j = 1$, we have

$$\begin{aligned} \int_0^1 \varphi_i(x) \varphi_j(x) dx &= \int_0^h \frac{[h-x]^2}{h^2} dx \\ &= \left[-\frac{[h-x]^3}{3h^2} \right]_0^h \\ &= \frac{h}{3} \end{aligned}$$

For the case $i = j = N$:

$$\begin{aligned} \int_0^1 \varphi_i(x)\varphi_j(x) dx &= \int_{1-h}^1 \frac{[x - (1-h)]^2}{h^2} dx \\ &= \left[\frac{[x - (1-h)]^3}{3h^2} \right]_{1-h}^1 \\ &= \frac{h}{3} \end{aligned}$$

We can combine the next two cases corresponding to $1 < i = j < N$, to obtain:

$$\begin{aligned} \int_0^1 \varphi_i(x)\varphi_j(x) dx &= \int_{(i-2)h}^{(i-1)h} \frac{[x - (i-2)h]^2}{h^2} dx + \int_{(i-1)h}^{ih} \frac{[ih - x]^2}{h^2} dx \\ &= \left[\frac{[x - (i-2)h]^3}{3h^2} \right]_{(i-2)h}^{(i-1)h} + \left[\frac{[ih - x]^3}{3h^2} \right]_{(i-1)h}^{ih} \\ &= \frac{2h}{3} \end{aligned}$$

For the case $i = j + 1$ ($2 \leq i \leq N$), we have

$$\begin{aligned} \int_0^1 \varphi_i(x)\varphi_j(x) dx &= \int_{(i-2)h}^{(i-1)h} \frac{[x - (i-2)h][i-1)h - x]}{h^2} dx \\ &= \int_{(i-2)h}^{(i-1)h} \frac{[(i-1)hx - x^2 - (i-1)(i-2)h^2 + (i-2)hx]}{h^2} dx \\ &= \int_{(i-2)h}^{(i-1)h} \left(-\frac{x^2}{h^2} + \frac{[(i-1) + (i-2)]x}{h} - (i-1)(i-2) \right) dx \\ &= \left[-\frac{x^3}{3h^2} + \frac{[(i-1) + (i-2)]x^2}{2h} - (i-1)(i-2)x \right]_{(i-2)h}^{(i-1)h} \\ &= \frac{h}{3} \left[-(i-1)^3 + (i-2)^3 \right] + \frac{h}{2} [(i-1) + (i-2)] \left[(i-1)^2 - (i-2)^2 \right] \\ &\quad - (i-1)(i-2)h \\ &= \frac{h}{3} (-1) \left[(i-2)^2 + (i-2)(i-1) + (i-1)^2 \right] \\ &\quad + \frac{h}{2} [(i-1) + (i-2)] [(i-1) + (i-2)] (1 - (i-1)(i-2)h) \\ &= -\frac{h}{3} \left[(i-2)^2 + (i-2)(i-1) + (i-1)^2 \right] \\ &\quad + \frac{h}{2} \left[(i-1)^2 + 2(i-1)(i-2) + (i-2)^2 \right] - (i-1)(i-2)h \\ &= \frac{h}{6} \left[(i-1)^2 - \frac{h}{3}(i-1)(i-2) + \frac{h}{6}(i-2)^2 \right] \\ &= \frac{h}{6} [(i-1) - (i-2)]^2 \end{aligned}$$

$$= \frac{h}{6}$$

After a similar calculation, we also find for the case $i = j - 1$ ($1 \leq i \leq N - 1$):

$$\int_0^1 \varphi_i(x)\varphi_j(x) dx = \frac{h}{6}$$

Hence, we have determined the components of the damping matrix \mathbf{D} as

$$D_{ij} = \int_0^1 \varphi_i(x)\varphi_j(x) dx = \begin{cases} \frac{h}{3} & i = j = 1 \\ \frac{h}{3} & i = j = N \\ \frac{2h}{3} & 1 < i = j < N \\ \frac{h}{6} & i = j + 1, \quad 2 \leq i \leq N \\ \frac{h}{6} & i = j - 1, \quad 1 \leq i \leq N - 1 \\ 0 & \text{otherwise} \end{cases} \quad (5.10)$$

From the above evaluations, the resulting ODE matrix system can be written as

$$\overbrace{\begin{bmatrix} \frac{h}{3} & \frac{h}{6} & 0 & \dots & 0 & 0 & 0 \\ \frac{h}{6} & \frac{2h}{3} & \frac{h}{6} & \dots & 0 & 0 & 0 \\ 0 & \frac{h}{6} & \frac{2h}{3} & \dots & 0 & 0 & 0 \\ \vdots & & \ddots & \ddots & \vdots & & \\ 0 & 0 & 0 & \dots & \frac{2h}{3} & \frac{h}{6} & 0 \\ 0 & 0 & 0 & \dots & \frac{h}{6} & \frac{2h}{3} & \frac{h}{6} \\ 0 & 0 & 0 & \dots & 0 & \frac{h}{6} & \frac{h}{3} \end{bmatrix}}^{\mathbf{D}} \begin{bmatrix} \frac{dc_1}{dt} \\ \frac{dc_2}{dt} \\ \vdots \\ \frac{dc_N}{dt} \end{bmatrix} + \overbrace{\begin{bmatrix} \frac{2D}{h} & \frac{-D}{h} & 0 & \dots & 0 & 0 & 0 \\ \frac{-D}{h} & \frac{2D}{h} & \frac{-D}{h} & \dots & 0 & 0 & 0 \\ 0 & \frac{-D}{h} & \frac{2D}{h} & \dots & 0 & 0 & 0 \\ \vdots & & \ddots & \ddots & \vdots & & \\ 0 & 0 & 0 & \dots & \frac{2D}{h} & \frac{-D}{h} & 0 \\ 0 & 0 & 0 & \dots & \frac{-D}{h} & \frac{2D}{h} & \frac{-D}{h} \\ 0 & 0 & 0 & \dots & 0 & \frac{-D}{h} & \frac{D}{h} \end{bmatrix}}^{\mathbf{K}} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{bmatrix} = \overbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ q \end{bmatrix}}^{\mathbf{f}} \quad (5.11)$$

This matrix ODE system has four important features to note:

- Its stiffness (\mathbf{K}) and damping matrices (\mathbf{D}) are tri-diagonal, due to the fact that the product of any two linear basis functions (or their derivatives in the case of the damping matrix) is non-zero only if they are equal or adjacent to each other.
- Since very few basis functions overlap, the matrices will contain mostly zero entries and are therefore *sparse*. Sparse matrices are highly advantageous from a computational point of view because very large matrix systems can be efficiently stored in computer memory and solved for.
- The stiffness and damping matrices are symmetric, as is often the case for many physical PDE systems. Sparse symmetric matrix systems can be efficiently solved for.
- Finally, the above matrix system inherently incorporates the Neumann boundary condition at $x = 1$, through the q term appearing in the last element of the right-

hand side source vector. Hence, this boundary condition is natural to the system. The Dirichlet condition however at $x = 0$ has not yet been implemented, but must be enforced through an additional modification of the above matrices.

To implement the Dirichlet boundary condition at $x = 0$, namely $a_1(t) = p(t)$, we modify the above matrix system by adding an additional row and column to the damping and stiffness matrices in order to directly specify the boundary constraint whilst preserving the symmetry of both matrices. Such a modification however, entails the addition of a dummy variable λ to the system, whose value may be discarded from the solution. Adding an additional row to the \mathbf{D} , \mathbf{K} and \mathbf{f} matrices, with a corresponding symmetric column to \mathbf{D} and \mathbf{K} , we obtain the following DAE matrix system of equations:

$$\begin{bmatrix} \frac{h}{3} & \frac{h}{6} & 0 & \dots & 0 & 0 & 0 \\ \frac{h}{6} & \frac{2h}{3} & \frac{h}{6} & \dots & 0 & 0 & 0 \\ 0 & \frac{h}{6} & \frac{2h}{3} & \dots & 0 & 0 & 0 \\ \vdots & & & \ddots & & & \vdots \\ 0 & 0 & 0 & \dots & \frac{2h}{3} & \frac{h}{6} & 0 \\ 0 & 0 & 0 & \dots & \frac{h}{6} & \frac{h}{3} & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{dc_1}{dt} \\ \vdots \\ \frac{dc_i}{dt} \\ \vdots \\ \frac{dc_N}{dt} \\ \frac{d\lambda}{dt} \end{bmatrix} + \begin{bmatrix} \frac{D}{h} & -\frac{D}{h} & 0 & \dots & 0 & 0 & 1 \\ -\frac{D}{h} & \frac{2D}{h} & -\frac{D}{h} & \dots & 0 & 0 & 0 \\ 0 & -\frac{D}{h} & \frac{2D}{h} & \dots & 0 & 0 & 0 \\ \vdots & & & \ddots & & & \vdots \\ 0 & 0 & 0 & \dots & \frac{2D}{h} & -\frac{D}{h} & 0 \\ 0 & 0 & 0 & \dots & -\frac{D}{h} & \frac{D}{h} & 0 \\ 1 & 0 & 0 & \dots & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} c_1 \\ \vdots \\ c_i \\ \vdots \\ c_N \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \vdots \\ q \\ p \end{bmatrix}$$

with initial values $c_i(0) = c_0(hi)$, $i = 1 \dots N$. The above system can be more compactly written as

$$\begin{bmatrix} \mathbf{D} & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} \begin{bmatrix} \frac{d\mathbf{c}}{dt} \\ \frac{d\lambda}{dt} \end{bmatrix} + \begin{bmatrix} \mathbf{K} & \mathbf{N} \\ \mathbf{N}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{c} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ p \end{bmatrix} \tag{5.12}$$

where \mathbf{D} , \mathbf{K} and \mathbf{f} are as defined in Eq. 5.11, dummy variable λ is known as a *Lagrange multiplier*, and $\mathbf{N} = [1 \ 0 \ \dots \ 0]^T$ is the *constraint matrix*, such that $\mathbf{N}^T \mathbf{c} = p$. For a given PDE with multiple Dirichlet boundary conditions, each of these boundary constraints will be associated with its own Lagrange multiplier. The resulting DAE system (Eq. 5.12) can be solved using any the methods described in Chap. 3, Sect. 3.4.

Example 5.1 Using 1D linear basis functions on a uniformly-spaced grid consisting of 10 elements, determine the finite element \mathbf{D} , \mathbf{K} and \mathbf{f} matrices for the 1D time-dependent diffusion equation

$$\frac{\partial c}{\partial t} = \frac{\partial^2 c}{\partial x^2}$$

for $x \in [0, 1]$ subject to zero-flux boundary conditions at $x = 0$ and $x = 1$, with initial value of $c(x, t)$ at $t = 0$ given by the square-wave distribution:

$$c(x, 0) = \begin{cases} 1 & 0.4 \leq x \leq 0.6 \\ 0 & \text{otherwise} \end{cases}$$

Using the same basis functions and number of elements, solve this PDE in COMSOL, plotting the solution at $t = 0.1$, and confirm that the COMSOL-generated \mathbf{D} and \mathbf{K} matrices are equal to those obtained analytically.

Answer: We can use the matrix ODE system of Eq. 5.11, noting that there are a total of 11 variables to be solved for: one variable per element node. Furthermore, the source vector will contain q flux terms in each of its first and last elements, corresponding to each Neumann boundary condition. Since these are zero-flux conditions, each q term will be zero. Noting that $D = 1$ and $h = 0.1$, we obtain the following system matrices:


$$\mathbf{f} = [0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0]^T$$

$$\mathbf{D} = \begin{bmatrix} \frac{1}{30} & \frac{1}{60} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{60} & \frac{1}{30} & \frac{1}{60} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{60} & \frac{1}{30} & \frac{1}{60} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{60} & \frac{1}{30} & \frac{1}{60} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{60} & \frac{1}{30} & \frac{1}{60} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{60} & \frac{1}{30} & \frac{1}{60} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{60} & \frac{1}{30} & \frac{1}{60} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{60} & \frac{1}{30} & \frac{1}{60} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{60} & \frac{1}{30} & \frac{1}{60} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{60} & \frac{1}{30} & \frac{1}{60} \end{bmatrix} \quad (5.13)$$


$$\mathbf{K} = \begin{bmatrix} 10 & -10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -10 & 20 & -10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -10 & 20 & -10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -10 & 20 & -10 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -10 & 20 & -10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -10 & 20 & -10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -10 & 20 & -10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -10 & 20 & -10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -10 & 20 & -10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -10 & 20 & -10 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -10 & 10 \end{bmatrix} \quad (5.14)$$

To solve this PDE in COMSOL and inspect the resulting damping and stiffness matrices, implement the following steps:


Model Wizard

1. Open the Model Wizard and select the 1D spatial dimension.
2. In the Select Physics panel, choose Mathematics|PDE Interfaces|General Form PDE. Click “Add”.
3. In the Review Physics Interface panel at right, specify c as the Field name and c as the name of the dependent variable in the dependent variables list. For the dependent variable quantity, leave the units and source term quantity units to their default values.
4. Click the Study arrow () to open the Select Study panel. Select Time Dependent, and click “Done”. This will exit the Model Wizard, displaying the main COMSOL interface.

Geometry

1. Right-click Geometry 1 in the model tree and select Interval. Leave the left and right endpoints to their default values of 0 and 1 respectively. Click Build Selected ()


Global Definitions

1. Right-click the Definitions sub-node under the Global node, and select Functions|Rectangle. This will define a square-wave function for implementing the initial value profile. Specify the lower and upper limits as 0.4 and 0.6 respectively.
2. Under the Smoothing tab, specify a value of 0.01 for the size of the transition zone. This defines a smooth transition from 0 to 1, avoiding discontinuities in the rising and falling edges. Leave the function name to its default of `rect1`. Clicking Plot () will display the function, as shown in Fig. 5.3.

General Form PDE

1. Select the General Form PDE 1 sub-node of General Form PDE, and leave the conservative flux expression to its default of $-cx$. Enter a value of 0 for the source term f , and leave the damping and mass coefficients to their default values.
2. Select the Initial Values 1 sub-node of the General Form PDE node. For the initial value of c , enter the initial value expression `rect1(x [1/m])` to implement the initial square-wave concentration profile.

Mesh

1. Right-click Mesh 1 and select Distribution. For the Domain selection, select the line interval in the Graphics window (Domain 1). Specify the number of elements as 10.
2. Right-click Mesh 1 again and select Edge. Leave the default geometric entity level as ‘Remaining’. This will mesh the remaining parts of the model with a free edge mesh.
3. Click Build All () in the Settings window to build and display the mesh.

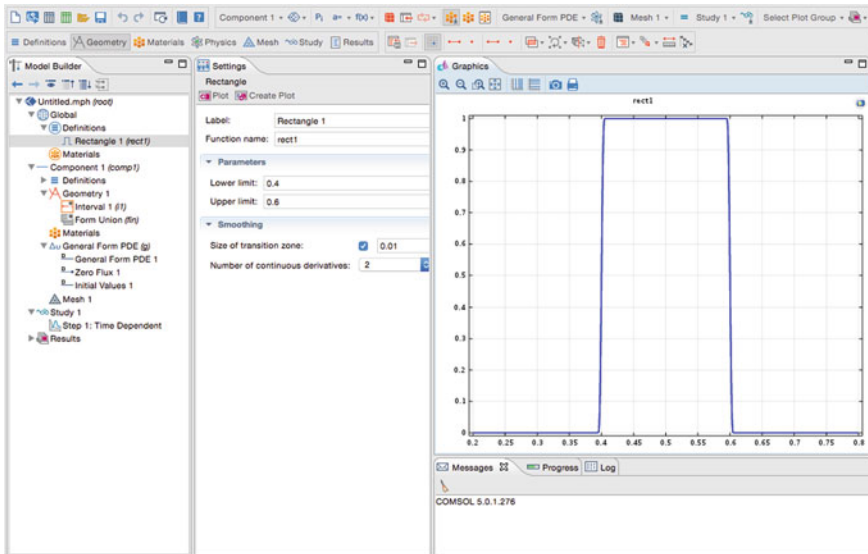




Fig. 5.3 COMSOL user-defined Rect1 function for implementing the initial square-wave concentration profile of Example 5.1


Discretization

1. In the toolbar of the model tree, click the show button () , and select Discretization to view the finite element basis function discretization.
2. Select the General Form PDE node of the model tree. There should now be a Discretization tab visible in the settings panel. Select this tab to view the discretization settings. By default, the element order will show as Quadratic: change this to Linear. This specifies linear basis functions for the elements.

Study

1. To solve the model, right-click Study 1 and select Compute () . This will compute the solution using the default time range of 0 to 1 s in output steps of 0.1 s.

Results

1. Under the Results node of the model tree, select the 1D Plot Group 1 sub-node. In the Settings window, specify the Time selection option as 'From list', and select a time of 0.1 s from the Times dropdown list. Click the Plot button () will display the concentration variable c at this time, as shown in Fig. 5.4.

System Matrices

1. Select Study 1|Solver Configurations|Solution 1. Right-click Solution 1 and select Other|Assemble. In the Settings window, under the Non-Eliminated Output tab,

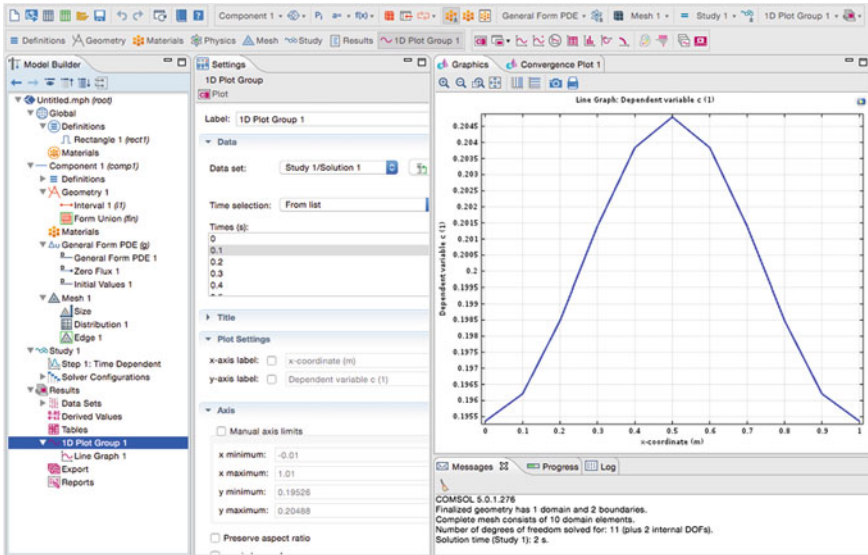


Fig. 5.4 COMSOL result at $t = 0.1$ s for PDE diffusion Example 5.1.

- check the ‘Load vector’, ‘Stiffness matrix’ and ‘Damping matrix’ checkboxes. Click the Compute button (=) to evaluate and store these system matrices.
- Right-click the Results|Derived Values node, and select System Matrix. In the Settings panel under the Output tab, leave the default Matrix as the Stiffness matrix, and select the output Format as ‘Filled’. Click the Evaluate button (=) to display the stiffness matrix below the Graphics window, as shown in Fig. 5.5. It can be readily seen that all entries of this stiffness matrix are identical to that of the analytical stiffness matrix obtained earlier (Eq. 5.13).
 - Select the Results|Derived Values|System Matrix 1 sub-node of the model tree. In the Settings panel, specify the Matrix output as ‘Damping matrix’. Keep the output format as ‘Filled’. Click the Evaluate button (=) to display the damping matrix, as shown in Fig. 5.6. As can be seen from the entries of the COMSOL-generated matrix, its entries are all equal to the analytical damping matrix (Eq. 5.14), correct to several decimal places.

5.1.3 Higher-Order Basis Functions

5.1.3.1 Element Coordinates

In the previous section, we expressed the finite element basis functions φ_i as a function of global spatial coordinates, x i.e. $\varphi_i(x)$ defines the i th global basis function.

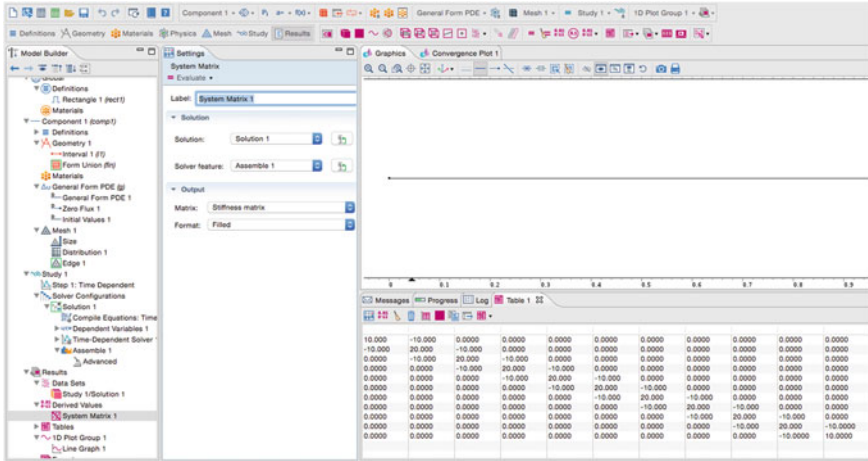


Fig. 5.5 COMSOL-generated stiffness matrix (seen at lower right) for the PDE diffusion Example 5.1

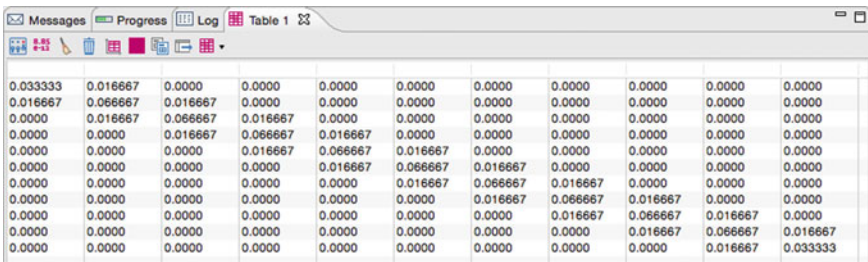


Fig. 5.6 COMSOL-generated damping matrix for the PDE diffusion Example 5.1

However, a more convenient description is to use local element coordinates to define local *shape functions*. For our previous 1D linear basis functions, we can define a local element coordinate ξ , which varies between 0 and 1 within the element, to define two local element shape functions, $\varphi_1(\xi)$ and $\varphi_2(\xi)$, as follows:

$$\xi = \frac{x - x_i}{x_{i+1} - x_i}, \quad \varphi_1(\xi) = 1 - \xi, \quad \varphi_2(\xi) = \xi \tag{5.15}$$

where x_i, x_{i+1} denote the global node positions of element i . These shape functions, together with their local and global descriptions, are plotted in Fig. 5.7. It should be noted that the combination of shape function φ_2 from one element with φ_1 from the next will produce the familiar hat-shaped basis function used previously.

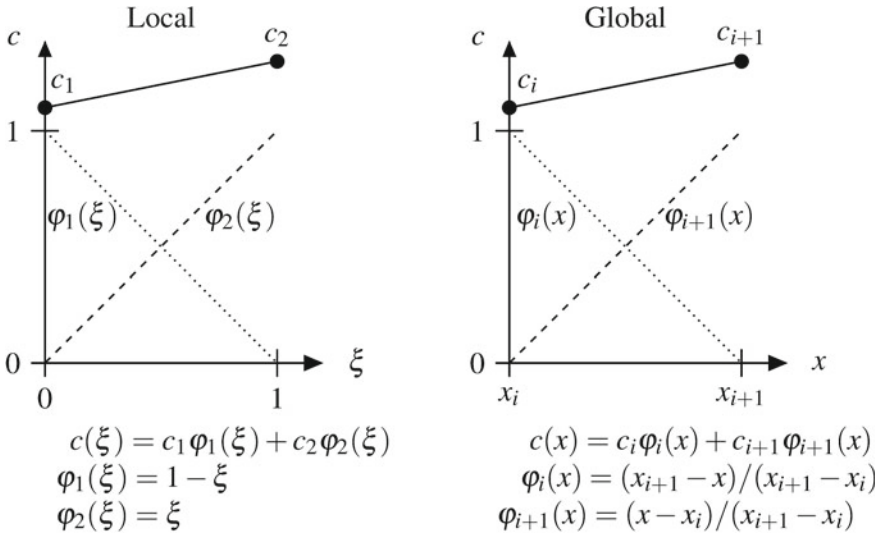


Fig. 5.7 Local (left) versus global (right) descriptions of 1D linear (or Lagrange) element shape functions, resulting in a linear interpolation of the dependent variable c within element i

5.1.3.2 Lagrange Shape Functions

It is possible to choose higher order polynomials for element shape functions that satisfy our FEM requirement that they equal 1 at one node and 0 at all other nodes of the element, namely:

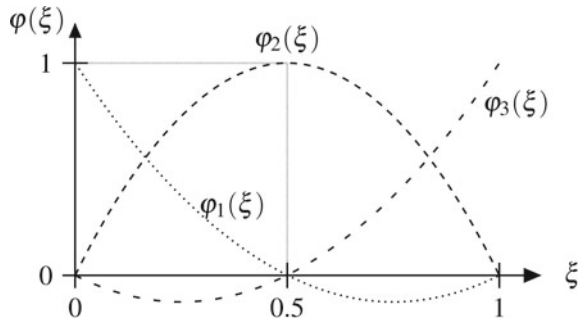
$$\varphi_i(\xi_j) = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

where φ_i denotes the i th element shape function, and ξ_j denotes the j th node of the element. These are known as *Lagrange shape functions*, and those of order higher than 1 will have more than 2 nodes per element, with the number of shape functions equal to the number of nodes. An example are the quadratic Lagrange functions, with three nodes per element located at $\xi = 0$, $\xi = 0.5$, and $\xi = 1$. The three shape functions are given by

$$\begin{aligned} \varphi_1(\xi) &= 2(0.5 - \xi)(1 - \xi) \\ \varphi_2(\xi) &= 4\xi(1 - \xi) \\ \varphi_3(\xi) &= 2\xi(\xi - 0.5) \end{aligned} \tag{5.16}$$

and are plotted in Fig. 5.8. As for the linear shape functions, the value of a dependent variable c within the element is given by the linear combination of quadratic shape

Fig. 5.8 1D quadratic element shape functions



functions, namely:

$$c(\xi) = c_1\varphi_1(\xi) + c_2\varphi_2(\xi) + c_3\varphi_3(\xi)$$

where c_1, c_2, c_3 are the values of c at nodes $\xi = 0, \xi = 0.5$ and $\xi = 1$ respectively.

5.1.3.3 Hermite Shape Functions

Another choice of element shape functions is to use polynomials that interpolate not only the dependent variable, but also its spatial derivative at the nodes. One such example are the *cubic Hermite*⁴ shape functions, used to interpolate within an element which has the dependent variable as well as its spatial derivative specified at each of its two nodes. For these shape functions, there are two nodes per element (located at $\xi = 0$ and $\xi = 1$), and four shape functions, given by:

$$\begin{aligned} \varphi_1(\xi) &= 1 - 3\xi^2 + 2\xi^3 & \varphi_3(\xi) &= \xi^2(3 - 2\xi) \\ \varphi_2(\xi) &= \xi(\xi - 1)^2 & \varphi_4(\xi) &= \xi^2(\xi - 1) \end{aligned}$$

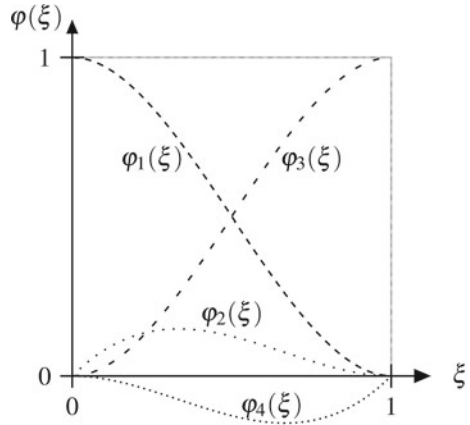
These shape functions satisfy the following relationships:

$$\begin{aligned} \varphi_1(0) &= 1, \varphi_1(1) = 0, \left. \frac{d\varphi_1}{d\xi} \right|_{\xi=0} = 0, \left. \frac{d\varphi_1}{d\xi} \right|_{\xi=1} = 0 \\ \varphi_3(0) &= 0, \varphi_3(1) = 1, \left. \frac{d\varphi_3}{d\xi} \right|_{\xi=0} = 0, \left. \frac{d\varphi_3}{d\xi} \right|_{\xi=1} = 0 \\ \varphi_2(0) &= 0, \varphi_2(1) = 0, \left. \frac{d\varphi_2}{d\xi} \right|_{\xi=0} = 1, \left. \frac{d\varphi_2}{d\xi} \right|_{\xi=1} = 0 \\ \varphi_4(0) &= 0, \varphi_4(1) = 0, \left. \frac{d\varphi_4}{d\xi} \right|_{\xi=0} = 0, \left. \frac{d\varphi_4}{d\xi} \right|_{\xi=1} = 1 \end{aligned}$$

and are plotted in Fig. 5.9. The dependent variable c is interpolated using a linear combination of its nodal values and its nodal derivatives according to the expression

⁴Pronounced Her-meet.

Fig. 5.9 1D cubic Hermite element shape functions



$$c(\xi) = c_1\varphi_1(\xi) + c_2\varphi_3(\xi) + \frac{dc_1}{d\xi}\varphi_2(\xi) + \frac{dc_2}{d\xi}\varphi_4(\xi)$$

where c_1, c_2 are the values of the dependent variable at element nodes 1 ($\xi = 0$) and 2 ($\xi = 1$) respectively, and $\frac{dc_1}{d\xi}, \frac{dc_2}{d\xi}$ are the values of its spatial derivative at each corresponding node. Note that unlike Lagrange elements, in which the dependent variable is only piecewise continuous, cubic Hermite elements are also piecewise continuous in the value of the spatial derivative, resulting in a smoother solution for the dependent variable. The disadvantage is that there are more *degrees of freedom* (i.e. variables to be solved for), since nodal derivatives must also be determined in addition to nodal values.

5.2 Finite Elements for 2D/3D Systems

We can implement the finite element method for higher dimensional systems by using the same concepts we employed for 1D, namely:

1. Formulate the underlying PDE in its weak form.
2. Approximate the dependent variable using a finite sum of basis functions. Substituting this approximation into the weak form will lead to a matrix system of equations for the dependent variable that can be solved for.⁵

For simplicity, we will illustrate these concepts using the following time-independent PDE, written in strong form as

⁵In the case of non-linear PDEs, these system matrices will need to be iteratively updated in order to converge to a solution.

$$\begin{aligned}\nabla \cdot (-\kappa \nabla u) &= f \quad \text{in } \Omega \\ u &= 0 \quad \text{on } \partial\Omega\end{aligned}\tag{5.17}$$

where u is the dependent variable to be solved for, κ , f are given functions of spatial position, and Ω , $\partial\Omega$ denote the PDE closed domain and boundary respectively. For 2D systems, $\partial\Omega$ will consist of 1D bounding edges: for 3D systems, it will be 2D surfaces.⁶

5.2.1 Weak Form Description

Just as in the 1D case, we can express Eq.5.17 in its equivalent weak form by multiplying both sides of the PDE by a suitable test function and integrate over the domain. Before we can do this, however, it will be necessary to establish a multi-dimensional integration by parts formula, also known as *Green's identity*.

5.2.1.1 Green's Identity

Consider two scalar functions $v(\mathbf{x})$ and $w(\mathbf{x})$, each a function of spatial position $\mathbf{x} = (x \ y \ z)^T$. Then,⁷

$$\begin{aligned}\nabla \cdot (v \nabla w) &= \nabla \cdot \begin{pmatrix} v \frac{\partial w}{\partial x} \\ v \frac{\partial w}{\partial y} \\ v \frac{\partial w}{\partial z} \end{pmatrix} \\ &= \frac{\partial}{\partial x} \left[v \frac{\partial w}{\partial x} \right] + \frac{\partial}{\partial y} \left[v \frac{\partial w}{\partial y} \right] + \frac{\partial}{\partial z} \left[v \frac{\partial w}{\partial z} \right] \\ &= \frac{\partial v}{\partial x} \frac{\partial w}{\partial x} + v \frac{\partial^2 w}{\partial x^2} + \frac{\partial v}{\partial y} \frac{\partial w}{\partial y} + v \frac{\partial^2 w}{\partial y^2} + \frac{\partial v}{\partial z} \frac{\partial w}{\partial z} + v \frac{\partial^2 w}{\partial z^2} \\ &= \nabla v \cdot \nabla w + v \nabla^2 w\end{aligned}$$

Integrating both sides of the above identity over the domain Ω , we obtain:

$$\int_{\Omega} \nabla \cdot (v \nabla w) \, dV = \int_{\Omega} (\nabla v \cdot \nabla w) \, dV + \int_{\Omega} (v \nabla^2 w) \, dV \tag{5.18}$$

⁶Needless to say, Eq.5.17 also holds for 1D systems, in which case Ω will be a line interval and $\partial\Omega$ will consist of its two bounding points.

⁷The subsequent analysis also holds for 2D, where $\mathbf{x} = (x \ y)^T$.

where dV in the above integrals denotes an infinitesimal volume element of Ω . Recalling the divergence theorem (see Eq.4.5):

$$\int_V (\nabla \cdot \mathbf{F}) dV = \int_S \mathbf{F} \cdot d\mathbf{S}$$

we can replace the volume integral of the left-hand side of Eq.5.18 with a surface integral, to obtain Green's identity⁸:

$$\int_{\partial\Omega} (v\nabla w) \cdot d\mathbf{S} = \int_{\Omega} (\nabla v \cdot \nabla w) dV + \int_{\Omega} (v\nabla^2 w) dV \quad (5.19)$$

where $d\mathbf{S} = \mathbf{n}dS$ denotes an infinitesimal surface element area vector of boundary $\partial\Omega$, of area dS in the direction of the outward surface unit normal \mathbf{n} .

5.2.1.2 PDE Weak Form

Having derived Green's identity, we are now in position to obtain the weak form equivalent of Eq.5.17. The weak form is obtained by multiplying both sides of the PDE by a suitable test function, $u_{test}(\mathbf{x})$, such that $u_{test} = 0$ on $\partial\Omega$, then integrating over Ω to obtain:

$$\begin{aligned} \int_{\Omega} \nabla \cdot (-\kappa \nabla u) u_{test} dV &= \int_{\Omega} f u_{test} dV \\ \int_{\Omega} -[(\nabla \kappa) \cdot (\nabla u) + \kappa \nabla^2 u] u_{test} dV &= \int_{\Omega} f u_{test} dV \end{aligned}$$

Upon re-arranging, we obtain:

$$\int_{\Omega} [-\kappa u_{test} \nabla^2 u] dV = \int_{\Omega} [(\nabla \kappa) \cdot (\nabla u)] u_{test} dV + \int_{\Omega} f u_{test} dV \quad (5.20)$$

Using Green's identity (Eq.5.19), we can substitute into it $v = -\kappa u_{test}$ and $w = u$ to obtain:

$$\int_{\partial\Omega} (-\kappa u_{test} \nabla u) \cdot d\mathbf{S} = \int_{\Omega} -[\nabla(\kappa u_{test}) \cdot \nabla u] dV + \int_{\Omega} -[\kappa u_{test} \nabla^2 u] dV$$

and noting that $\nabla(\kappa u_{test}) = u_{test} \nabla \kappa + \kappa \nabla u_{test}$, the above becomes:

$$\int_{\partial\Omega} (-\kappa u_{test} \nabla u) \cdot d\mathbf{S} = \int_{\Omega} -[(u_{test} \nabla \kappa + \kappa \nabla u_{test}) \cdot \nabla u] dV + \int_{\Omega} -[\kappa u_{test} \nabla^2 u] dV$$

⁸Also referred to as Green's first identity.

$$= \int_{\Omega} -[(\nabla\kappa) \cdot (\nabla u) u_{test} + \kappa (\nabla u) \cdot (\nabla u_{test})] dV + \int_{\Omega} -[\kappa u_{test} \nabla^2 u] dV$$

Re-arranging, we obtain

$$\begin{aligned} \int_{\Omega} -[\kappa u_{test} \nabla^2 u] dV &= \int_{\partial\Omega} (-\kappa u_{test} \nabla u) \cdot d\mathbf{S} + \int_{\Omega} [(\nabla\kappa) \cdot (\nabla u)] u_{test} dV \\ &\quad + \int_{\Omega} (\kappa \nabla u) \cdot (\nabla u_{test}) dV \end{aligned}$$

Since the left-hand side of this expression is equal to that of Eq. 5.20, we can equate both right-hand sides to obtain:

$$\begin{aligned} \int_{\partial\Omega} (-\kappa u_{test} \nabla u) \cdot d\mathbf{S} + \int_{\Omega} [(\nabla\kappa) \cdot (\nabla u)] u_{test} dV + \int_{\Omega} (\kappa \nabla u) \cdot (\nabla u_{test}) dV \\ = \int_{\Omega} [(\nabla\kappa) \cdot (\nabla u)] u_{test} dV + \int_{\Omega} f u_{test} dV \end{aligned}$$

Cancelling terms, we obtain:

$$\int_{\partial\Omega} (-\kappa u_{test} \nabla u) \cdot d\mathbf{S} + \int_{\Omega} (\kappa \nabla u) \cdot (\nabla u_{test}) dV = \int_{\Omega} f u_{test} dV$$

and since $u_{test} = 0$ on $\partial\Omega$, the first term will be zero, and we obtain the resulting weak form equivalent:

$$\int_{\Omega} (\kappa \nabla u) \cdot (\nabla u_{test}) dV = \int_{\Omega} f u_{test} dV$$

Defining a function set U as the set of all functions u that are:

- piecewise continuous on Ω
- satisfy $\int_{\Omega} (\nabla u) \cdot (\nabla u) dV < \infty$
- satisfy $u = 0$ on $\partial\Omega$,

we can state the weak form of Eq. 5.17 as:

Find $u \in U$ such that

$$\int_{\Omega} (\kappa \nabla u) \cdot (\nabla u_{test}) dV = \int_{\Omega} f u_{test} dV \quad (5.21)$$

for $\forall u_{test} \in U$

5.2.2 Basis Function Approximation

To solve the PDE equivalent weak form, we can discretize the domain Ω into a number of finite elements, each element associated with a set of nodes. In 2D for example, such elements are typically triangles or quadrilaterals, with each vertex corresponding to a node. In 3D, these elements can be tetrahedrons or hexahedrons, again with the vertices corresponding to nodes. It is also possible to define additional interior nodes within each element. The total set of elements over Ω is known as the *finite element mesh*. Associated with a given mesh is a set of basis functions, $\varphi_i(\mathbf{x})$, such that for the j th global node in the mesh located at spatial point \mathbf{x}_j , basis function i , $\varphi_i(\mathbf{x})$, satisfies the following

$$\varphi_i(\mathbf{x}_j) = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

The dependent PDE variable can then be approximated using a finite sum of these basis functions, such that

$$u(\mathbf{x}) \approx \sum_{i=1}^N u_i \varphi_i(\mathbf{x}) \quad (5.22)$$

where $\varphi_i(\mathbf{x})$ is the i th basis function, u_i is the value of the dependent variable at the i th global node, and N is the total number of basis functions.

5.2.2.1 Galerkin Method Revisited

To solve the weak PDE form of Eq. 5.21, we take as our set of candidate test functions, u_{test} the finite element basis functions themselves. Setting $u_{test} = \varphi_j$, and substituting into the weak form expression

$$\int_{\Omega} (\kappa \nabla u) \cdot (\nabla u_{test}) \, dV = \int_{\Omega} f u_{test} \, dV$$

we obtain

$$\begin{aligned} \int_{\Omega} \left(\kappa \nabla \left[\sum_{i=1}^N u_i \varphi_i \right] \right) \cdot (\nabla \varphi_j) \, dV &= \int_{\Omega} f \varphi_j \, dV \\ \sum_{i=1}^N \left[u_i \int_{\Omega} (\kappa \nabla \varphi_i) \cdot (\nabla \varphi_j) \, dV \right] &= \int_{\Omega} f \varphi_j \, dV \end{aligned}$$

This is equivalent to solving the matrix system

$$\mathbf{K} \mathbf{u} = \mathbf{f}$$

where \mathbf{u} is the vector of dependent variables u_i to be solved for, \mathbf{K} is the stiffness matrix whose (i, j) th entry is given by

$$K_{ij} = \int_{\Omega} (\kappa \nabla \varphi_i) \cdot (\nabla \varphi_j) \, dV \quad (5.23)$$

and \mathbf{f} is the load vector, whose j th entry is

$$f_j = \int_{\Omega} f \varphi_j \, dV \quad (5.24)$$

As in the case of 1D elements, the 2D/3D basis functions $\varphi_i(\mathbf{x})$ can also be expressed in terms of local element coordinates $\boldsymbol{\xi} = (\xi_1, \xi_2)^T$ (2D) or $\boldsymbol{\xi} = (\xi_1, \xi_2, \xi_3)^T$ (3D), where each element coordinate ξ_i varies between 0 and 1. Expressed this way within a local element, the $\varphi_i(\mathbf{x})$ are referred to as shape functions.

5.2.2.2 Isoparametric Elements

Within any element, the value of the dependent variable u can be interpolated from its values at the element nodes (u_i) using the element shape functions as follows:

$$u(\boldsymbol{\xi}) = \sum_{i=1}^n u_i \varphi_i(\boldsymbol{\xi}) \quad (5.25)$$

where n is the number of nodes of the element. Denote the value of the global coordinate \mathbf{x} at the i th element node by \mathbf{x}_i . If the mapping between global coordinates \mathbf{x} and local coordinates ($\boldsymbol{\xi}$) anywhere within the element is also given by the same interpolation function as Eq. 5.25, namely:

$$x(\boldsymbol{\xi}) = \sum_{i=1}^n x_i \varphi_i(\boldsymbol{\xi})$$

$$y(\boldsymbol{\xi}) = \sum_{i=1}^n y_i \varphi_i(\boldsymbol{\xi})$$

and for 3D:

$$z(\boldsymbol{\xi}) = \sum_{i=1}^n z_i \varphi_i(\boldsymbol{\xi})$$

which may be compactly expressed as

$$\mathbf{x}(\boldsymbol{\xi}) = \sum_{i=1}^n \mathbf{x}_i \varphi_i(\boldsymbol{\xi})$$

then the element is said to be *isoparametric*. Isoparametric elements represent a powerful feature of the finite element method, allowing curved edges and surfaces of domains to be readily represented. Some examples of 2D/3D isoparametric elements will be given in the following subsections.

5.2.2.3 Bilinear Quadrilateral Elements

We can use the two 1D linear shape functions $\varphi_1(\xi)$, $\varphi_2(\xi)$ defined in Eq. 5.15 to construct a set of four shape functions for the 2D quadrilateral element. Defining 2D local element coordinates by (ξ_1, ξ_2) , we can form four product combinations of the 1D linear shape functions, $\varphi_1(\xi_1)\varphi_1(\xi_2)$, $\varphi_2(\xi_1)\varphi_1(\xi_2)$, $\varphi_2(\xi_1)\varphi_2(\xi_2)$ and $\varphi_1(\xi_1)\varphi_2(\xi_2)$, to define the four *bilinear quadrilateral element* shape functions:

$$\begin{aligned} \varphi_1(\xi_1, \xi_2) &= (1 - \xi_1)(1 - \xi_2) \\ \varphi_2(\xi_1, \xi_2) &= \xi_1(1 - \xi_2) \\ \varphi_3(\xi_1, \xi_2) &= \xi_1\xi_2 \\ \varphi_4(\xi_1, \xi_2) &= (1 - \xi_1)\xi_2 \end{aligned}$$

with four element nodes at the corners of the quadrilateral. These shape functions are plotted in Fig. 5.10. The value of a dependent variable u anywhere within the element can be determined from the interpolation formula of Eq. 5.25, namely:

$$u(\xi_1, \xi_2) = u_1\varphi_1(\xi_1, \xi_2) + u_2\varphi_2(\xi_1, \xi_2) + u_3\varphi_3(\xi_1, \xi_2) + u_4\varphi_4(\xi_1, \xi_2)$$

where u_1-u_4 are the values of u at each of the four nodes.

Using a similar interpolation formula, the spatial (x, y) coordinates within the element can also be determined from the local (ξ_1, ξ_2) coordinate values if these spatial coordinates are known at each of the four nodes. This leads to a bilinear map between the local and global coordinates such that in global (x, y) space, the boundaries of the quadrilateral element remain straight, as shown in the example of Fig. 5.11.

5.2.2.4 Linear Triangular Element

For 2D triangular elements, we can derive simple shape functions from the 2D bilinear shape functions for a quadrilateral. This process can be achieved by merging two nodes of the quadrilateral, a process known as *degeneration*. For example, for the quadrilateral element of Fig. 5.11, we can merge nodes 3 and 4 by assigning their spatial coordinates to the same value, namely $x_3 = x_4$ and $y_3 = y_4$, as shown in Fig. 5.12.

Denote the triangular basis functions by φ_1^* , φ_2^* , and φ_3^* . Starting with the bilinear quadrilateral shape functions, the spatial coordinate of any point \mathbf{x} within the

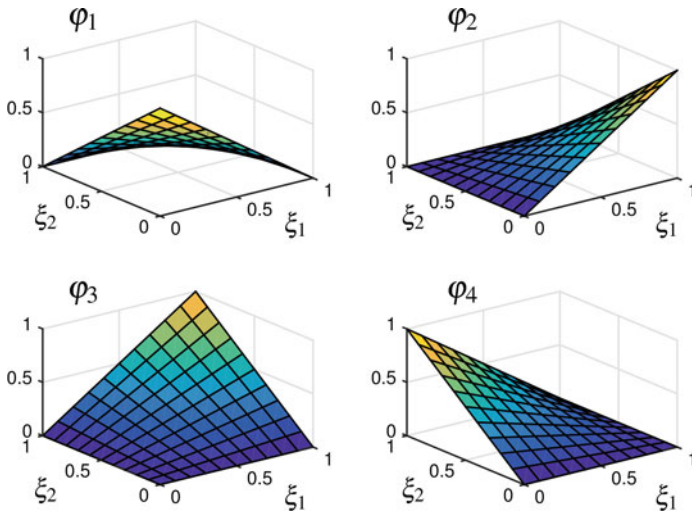


Fig. 5.10 Bilinear quadrilateral shape functions

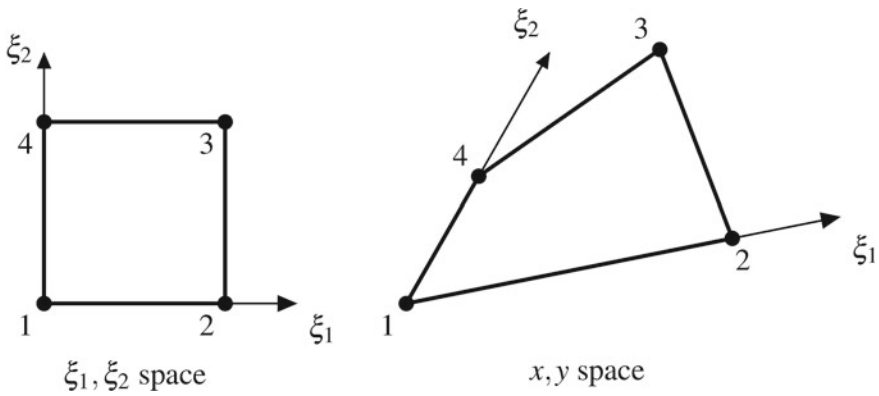


Fig. 5.11 Bilinear quadrilateral element representation in local element (left) and spatial frames (right). Element nodes 1–4 are shown numbered in each case

quadrilateral is given by

$$\mathbf{x} = \sum_{i=1}^4 \mathbf{x}_i \varphi_i(\xi_1, \xi_2)$$

where the \mathbf{x}_i denote the coordinates at the four nodes. Setting $\mathbf{x}_3 = \mathbf{x}_4$, the above becomes

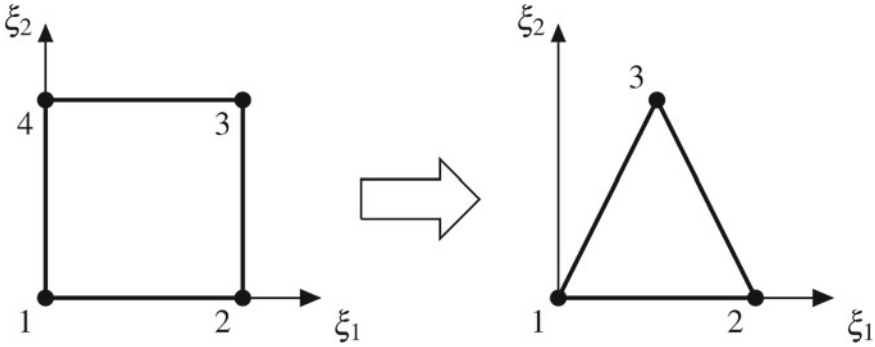


Fig. 5.12 Quadrilateral element degenerated into a triangle by merging nodes 3 and 4

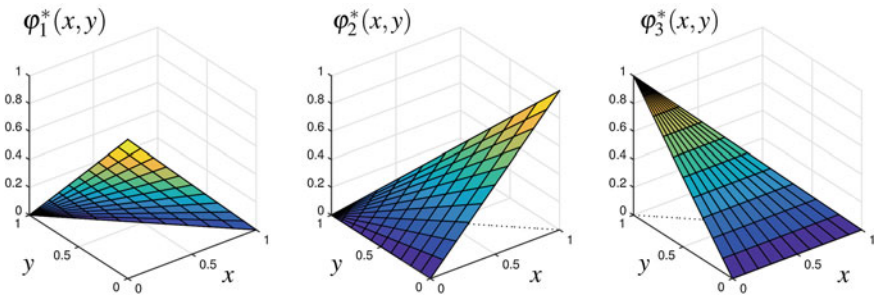


Fig. 5.13 Linear triangular shape functions plotted for the triangular element with spatial (x, y) nodes located at $(0,0), (0,1), (1,0)$

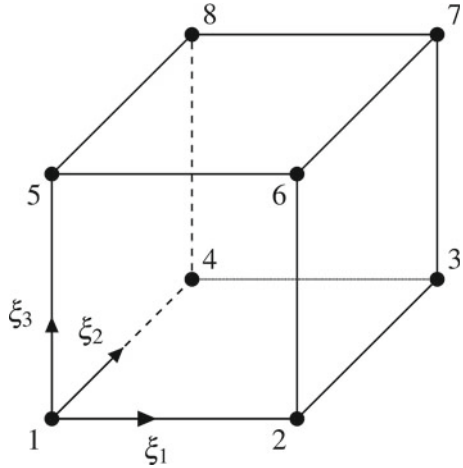
$$\begin{aligned}
 \mathbf{x} &= \mathbf{x}_1\varphi_1(\xi_1, \xi_2) + \mathbf{x}_2\varphi_2(\xi_1, \xi_2) + \mathbf{x}_3\varphi_3(\xi_1, \xi_2) + \mathbf{x}_3\varphi_4(\xi_1, \xi_2) \\
 &= \mathbf{x}_1 \underbrace{\varphi_1^*}_{\varphi_1(\xi_1, \xi_2)} + \mathbf{x}_2 \underbrace{\varphi_2^*}_{\varphi_2(\xi_1, \xi_2)} + \mathbf{x}_3 \underbrace{[\varphi_3(\xi_1, \xi_2) + \varphi_4(\xi_1, \xi_2)]}_{\varphi_3^*} \\
 &= \sum_{i=1}^3 \mathbf{x}_i \varphi_i^*(\xi_1, \xi_2)
 \end{aligned}$$

where

$$\begin{aligned}
 \varphi_1^* &= (1 - \xi_1)(1 - \xi_2) \\
 \varphi_2^* &= \xi_1(1 - \xi_2) \\
 \varphi_3^* &= \xi_1\xi_2 + (1 - \xi_1)\xi_2 \\
 &= \xi_2
 \end{aligned}$$

In the spatial (x, y) frame, these shape functions correspond to planes of height 1 at one node and 0 at the other two nodes. They are shown plotted in Fig. 5.13 for a triangular element with vertices located at $(0,0), (0,1),$ and $(1,0)$ in spatial (x, y)

Fig. 5.14 Trilinear element with local element coordinates ξ_1 , ξ_2 , and ξ_3 and 8 nodes numbered 1–8



coordinates. Within this element, the value of the dependent variable u is determined using a linear interpolation of its value at the nodes, hence these shape functions are known as *linear triangular*.

5.2.2.5 Trilinear Elements

We can extend our application of 1D linear shape functions to 3D hexahedral elements having 8 nodes: one node per vertex. For such a 3D element, there will be three local element coordinates (ξ_1 , ξ_2 , ξ_3), as shown in Fig. 5.14. As for the bilinear quadrilateral element, we can form the product of all combinations of three 1D linear shape functions, one shape function per element coordinate, to obtain a set of eight trilinear shape functions:

$$\begin{aligned}\varphi_1(\xi_1, \xi_2, \xi_3) &= (1 - \xi_1)(1 - \xi_2)(1 - \xi_3) \\ \varphi_2(\xi_1, \xi_2, \xi_3) &= \xi_1(1 - \xi_2)(1 - \xi_3) \\ \varphi_3(\xi_1, \xi_2, \xi_3) &= \xi_1\xi_2(1 - \xi_3) \\ \varphi_4(\xi_1, \xi_2, \xi_3) &= (1 - \xi_1)\xi_2(1 - \xi_3) \\ \varphi_5(\xi_1, \xi_2, \xi_3) &= (1 - \xi_1)(1 - \xi_2)\xi_3 \\ \varphi_6(\xi_1, \xi_2, \xi_3) &= \xi_1(1 - \xi_2)\xi_3 \\ \varphi_7(\xi_1, \xi_2, \xi_3) &= \xi_1\xi_2\xi_3 \\ \varphi_8(\xi_1, \xi_2, \xi_3) &= (1 - \xi_1)\xi_2\xi_3.\end{aligned}$$

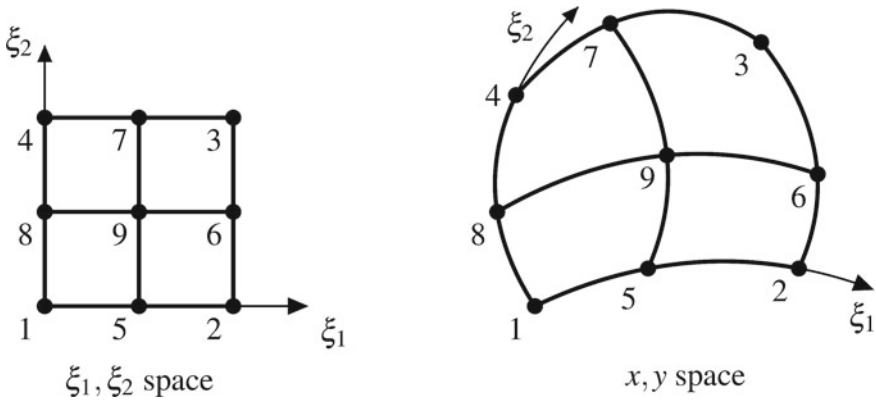


Fig. 5.15 Biquadratic element representation in local element (*left*) and spatial frames (*right*). Element nodes 1–9 are shown numbered in each case

5.2.2.6 Biquadratic Elements

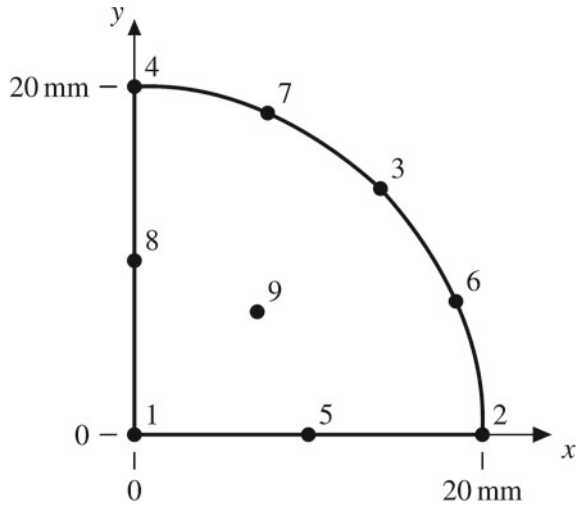
We can form even higher order element shape functions in 2D and 3D by forming various product combinations of 1D shape functions for each element coordinate. For example, using the set of three 1D quadratic shape functions (Eq. 5.16), we can form the following nine *biquadratic* shape functions for a 2D quadrilateral element:

$$\begin{aligned} \varphi_1(\xi_1, \xi_2) &= 4(0.5 - \xi_1)(1 - \xi_1)(0.5 - \xi_2)(1 - \xi_2) \\ \varphi_2(\xi_1, \xi_2) &= 4\xi_1(\xi_1 - 0.5)(0.5 - \xi_2)(1 - \xi_2) \\ \varphi_3(\xi_1, \xi_2) &= 4\xi_1\xi_2(\xi_1 - 0.5)(\xi_2 - 0.5) \\ \varphi_4(\xi_1, \xi_2) &= 4(0.5 - \xi_1)(1 - \xi_1)\xi_2(\xi_2 - 0.5) \\ \varphi_5(\xi_1, \xi_2) &= 8\xi_1(1 - \xi_1)(0.5 - \xi_2)(1 - \xi_2) \\ \varphi_6(\xi_1, \xi_2) &= 8\xi_1\xi_2(\xi_1 - 0.5)(1 - \xi_2) \\ \varphi_7(\xi_1, \xi_2) &= 8\xi_1\xi_2(1 - \xi_1)(\xi_2 - 0.5) \\ \varphi_8(\xi_1, \xi_2) &= 8\xi_2(0.5 - \xi_1)(1 - \xi_1)(1 - \xi_2) \\ \varphi_9(\xi_1, \xi_2) &= 16\xi_1\xi_2(1 - \xi_1)(1 - \xi_2) \end{aligned}$$

where the element has nine nodes, as shown in Fig. 5.15. These set of shape functions define the *biquadratic* element. Such higher order shape functions allow 2D/3D elements to have curved edges and surfaces, as can be seen from Fig. 5.15. Such elements can therefore be used to approximate curved boundaries, representing a major advantage of finite element methods over conventional finite difference schemes.

Example 5.2 Use a single biquadratic element to approximately represent a 2D domain consisting of quarter-circle of radius 20 mm.

Fig. 5.16 Single biquadratic element approximate representation of a quarter-circle domain of radius 20 mm



Answer: Referring to Fig. 5.15, we can place the eight exterior nodes of the element on the boundary of the quarter-circle, with nodes 4, 7, 3, 6, and 2 equi-spaced along the circular arc. The interior node 9 can be placed at half the radial distance from the centre of the arc, at 45° from the radial edge. This leads to the following choice of node positions:

- Node 1: (0, 0) mm
- Node 2: (20, 0) mm
- Node 3: $(20 \cos 45^\circ, 20 \sin 45^\circ)$ mm
- Node 4: (0, 20) mm
- Node 5: (10, 0) mm
- Node 6: $(20 \cos 22.5^\circ, 20 \sin 22.5^\circ)$ mm
- Node 7: $(20 \cos 67.5^\circ, 20 \sin 67.5^\circ)$ mm
- Node 8: (0, 10) mm
- Node 9: $(10 \cos 45^\circ, 10 \sin 45^\circ)$ mm

The biquadratic element with these nodes is shown in Fig. 5.16, where it can be seen that the edges of the element well-approximate the quarter circle. \square

5.3 FEM Numerical Implementation

This section will provide a brief overview of issues related to FEM numerical implementation. Although FEM is far-more difficult to program ‘from scratch’ than the FD approach, fortunately there exists a large range of FEM software, both open-source and commercial, that can be used to implement bioengineering models, including COMSOL Multiphysics®.

5.3.1 Assembly of System Matrices

For our earlier PDE example, Eq. 5.17, namely

$$\nabla \cdot (-\kappa \nabla u) = f$$

subject to given boundary conditions, we have seen that the Galerkin method leads to the matrix system

$$\mathbf{K} \mathbf{u} = \mathbf{f}$$

where the elements of the stiffness matrix \mathbf{K} and load vector \mathbf{f} are given by (see Eqs. 5.23, and 5.24):

$$K_{ij} = \int_{\Omega} (\kappa \nabla \varphi_i) \cdot (\nabla \varphi_j) \, dV$$

$$f_j = \int_{\Omega} f \varphi_j \, dV$$

where φ_i, φ_j denote the i th and j th global basis function, which can be any one of the 1D/2D/3D alternatives covered earlier, depending on the dimension of the PDE domain. In numerical FEM implementations, rather than using global basis functions to construct the system matrices, these are calculated on an element-element basis, then incorporated into the global system matrices, a process known as *assembly*. At the local element level, we can define an analogous element stiffness matrix \mathbf{K}_e and load vector \mathbf{f}_e according to

$$K_{e,ij} = \int_{\Omega_e} (\kappa \nabla \varphi_i(\mathbf{x})) \cdot (\nabla \varphi_j(\mathbf{x})) \, dV$$

$$f_{e,j} = \int_{\Omega_e} f \varphi_j \, dV$$

where Ω_e denotes the element domain, φ_i, φ_j refer to the i th, j th element *shape* function respectively, and \mathbf{x} refers to the spatial coordinate. The element \mathbf{K}_e and \mathbf{f}_e matrices will be of size $n \times n$ and $n \times 1$ respectively, where n is the number of element degrees of freedom. For Lagrange elements, n will simply equal the number of element nodes.

Once the individual element matrices have been determined, the entries are assembled into the global system matrices by numerically adding the element matrix entries to the corresponding global system matrix entries at the appropriate global matrix index positions.

5.3.2 Gaussian Quadrature

We have seen that determining the FEM system matrices requires evaluation of integrals involving the element shape functions. Furthermore, during FEM postprocessing, the integral of a given quantity may be required. For example, in simulations of electric current flow, one of the available outputs for postprocessing is the current density (in SI units of A m^{-2}). Integrating the normal component of current density at a given boundary will yield the total current flowing through that boundary. In FEM codes, integration for determining the system matrices, as well as postprocessing calculations, are performed numerically using the method of *Gaussian quadrature*.

In brief, Gaussian quadrature approximates the integral of a function by the weighted sum of the function evaluated at certain points known as *Gauss points*. For example, to integrate the function $f(\xi)$ over a 1D element, where ξ varies from 0 to 1, the method uses the formula

$$\int_0^1 f(\xi) d\xi \approx \sum_{i=1}^I w_i f(\xi_i) \quad (5.26)$$

where I is the order of the method, ξ_i are the fixed Gauss points, and w_i are weights associated with these points. To exactly integrate a polynomial of order 3, we can use $I = 2$. In this case, we can set $f(\xi) = a\xi^3 + b\xi^2 + c\xi + d$, where a, b, c, d are the polynomial coefficients. The Gauss points and associated weights can then be determined from

$$\begin{aligned} \int_0^1 f(\xi) d\xi &= w_1 f(\xi_1) + w_2 f(\xi_2) \\ \int_0^1 (a\xi^3 + b\xi^2 + c\xi + d) d\xi &= w_1 (a\xi_1^3 + b\xi_1^2 + c\xi_1 + d) + w_2 (a\xi_2^3 + b\xi_2^2 + c\xi_2 + d) \\ \left[\frac{a\xi^4}{4} + \frac{b\xi^3}{3} + \frac{c\xi^2}{2} + d\xi \right]_0^1 &= a (w_1\xi_1^3 + w_2\xi_2^3) + b (w_1\xi_1^2 + w_2\xi_2^2) \\ &\quad + c (w_1\xi_1 + w_2\xi_2) + d (w_1 + w_2) \\ \frac{a}{4} + \frac{b}{3} + \frac{c}{2} + d &= a (w_1\xi_1^3 + w_2\xi_2^3) + b (w_1\xi_1^2 + w_2\xi_2^2) \\ &\quad + c (w_1\xi_1 + w_2\xi_2) + d (w_1 + w_2) \end{aligned}$$

Equating the a, b, c, d coefficients on both sides, we obtain

$$w_1\xi_1^3 + w_2\xi_2^3 = \frac{1}{4} \quad (5.27)$$

$$w_1\xi_1^2 + w_2\xi_2^2 = \frac{1}{3} \quad (5.28)$$

$$w_1\xi_1 + w_2\xi_2 = \frac{1}{2} \quad (5.29)$$

$$w_1 + w_2 = 1 \quad (5.30)$$

which represents four equations in the four unknowns, w_1 , w_2 , ξ_1 , ξ_2 . We can solve this system of equations by assuming symmetry in the weights and the positioning of the Gauss points as follows:

$$w_1 = w_2 \quad (5.31)$$

$$\xi_2 = 1 - \xi_1 \quad (5.32)$$

From Eqs. 5.30 and 5.31, we obtain:

$$w_1 = w_2 = \frac{1}{2} \quad (5.33)$$

Substituting this result along with Eq. 5.32 into Eq. 5.28, yields:

$$\begin{aligned} \frac{\xi_1^2}{2} + \frac{(1 - \xi_1)^2}{2} &= \frac{1}{3} \\ \xi_1^2 + 1 - 2\xi_1 + \xi_1^2 &= \frac{2}{3} \\ 6\xi_1^2 - 6\xi_1 + 1 &= 0 \\ \therefore \xi_1 &= \frac{1}{2} \pm \frac{\sqrt{3}}{6} \end{aligned} \quad (5.34)$$

Either choice of ξ_1 will solve Eq. 5.28. Furthermore, it is straightforward to show these solutions also satisfy Eqs. 5.27 and 5.29. For the left-hand side of Eq. 5.27, using Eqs. 5.32 and 5.33, we have:

$$\begin{aligned} w_1 \xi_1^3 + w_2 \xi_2^3 &= \frac{\xi_1^3}{2} + \frac{(1 - \xi_1)^3}{2} \\ &= \frac{\xi_1^3}{2} + \frac{(1 - 3\xi_1 + 3\xi_1^2 - \xi_1^3)}{2} \\ &= \frac{(1 - 3\xi_1 + 3\xi_1^2)}{2} \\ &= \frac{(6\xi_1^2 - 6\xi_1 + 2)}{4} \\ &= \frac{(6\xi_1^2 - 6\xi_1 + 1)}{4} + \frac{1}{4} \\ &= \frac{1}{4} \quad (\text{from Eq. 5.34}) \end{aligned}$$

as required. Furthermore, we can substitute Eqs. 5.32 and 5.33 into the left-hand side of Eq. 5.29 to obtain:

$$\begin{aligned} w_1 \xi_1 + w_2 \xi_2 &= \frac{\xi_1}{2} + \frac{1 - \xi_1}{2} \\ &= \frac{1}{2} \end{aligned}$$

again, as required. Without loss of generality, we can take the lowest value of the two solutions for ξ_1 to yield our Gauss points and weights as

$$\xi_1 = \frac{1}{2} - \frac{\sqrt{3}}{6}, \quad w_1 = \frac{1}{2}, \quad \xi_2 = \frac{1}{2} + \frac{\sqrt{3}}{6}, \quad w_2 = \frac{1}{2}$$

It is possible to choose even higher order Gaussian quadrature methods that exactly integrate higher-order polynomials. To carry out higher-dimensional integration (i.e. in 2D and 3D), Eq. 5.26 can be used to apply successive summations over each dimension. Hence, to integrate the function $f(\boldsymbol{\xi})$, where $\boldsymbol{\xi}$ are the 2D ($= (\xi_1, \xi_2)$) or 3D ($= (\xi_1, \xi_2, \xi_3)$) element coordinates, we can use

$$\int_0^1 \int_0^1 f(\xi_1, \xi_2) d\xi_1 d\xi_2 \approx \sum_{i=1}^{I_1} \sum_{j=1}^{I_2} w_i w_j f(\xi_{1,i}, \xi_{2,j}) \quad (2D)$$

$$\int_0^1 \int_0^1 \int_0^1 f(\xi_1, \xi_2, \xi_3) d\xi_1 d\xi_2 d\xi_3 \approx \sum_{i=1}^{I_1} \sum_{j=1}^{I_2} \sum_{k=1}^{I_3} w_i w_j w_k f(\xi_{1,i}, \xi_{2,j}, \xi_{3,k}) \quad (3D)$$

where $\xi_{1,i}$, $\xi_{2,j}$, $\xi_{3,k}$ are the Gauss points along each coordinate, w_i , w_k , w_k are the weights, and I_1 , I_2 , I_3 are the Gaussian quadrature orders for each coordinate dimension.

5.3.3 Non-Linear Systems

To solve non-linear PDEs using the of method finite elements, application of Galerkin's method will lead to a system of non-linear equations in the nodal variables u_i , represented by the general vector equation

$$\mathbf{f}(\mathbf{u}) = \mathbf{0} \quad (5.35)$$

To solve Eq. 5.35, we can use the multivariate form of Taylor's theorem (Eq. 3.4) to linearize about some point \mathbf{u}_0 , representing the current estimate of the solution, to obtain:

$$\begin{aligned}\mathbf{u} &= \mathbf{u}_0 + \mathbf{h} \\ \mathbf{f}(\mathbf{u}_0 + \mathbf{h}) &= \mathbf{f}(\mathbf{u}_0) + \mathbf{h}^T \frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{u}_0) \\ &= \mathbf{f}(\mathbf{u}_0) + \frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{u}_0) \mathbf{h}\end{aligned}$$

Hence,

$$\mathbf{f}(\mathbf{u}) = \mathbf{f}(\mathbf{u}_0) + \frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{u}_0) (\mathbf{u} - \mathbf{u}_0)$$

Setting the left-hand side to zero (Eq. 5.35), and using $\mathbf{K}(\mathbf{u}_0) = \frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{u}_0)$, we obtain:

$$\mathbf{f}(\mathbf{u}_0) + \mathbf{K}(\mathbf{u}_0) (\mathbf{u} - \mathbf{u}_0) = \mathbf{0}$$

or

$$\mathbf{u} = \mathbf{u}_0 - \mathbf{K}(\mathbf{u}_0)^{-1} \mathbf{f}(\mathbf{u}_0) \quad (5.36)$$

where $\mathbf{K}(\mathbf{u}_0)$ is the stiffness matrix and $\mathbf{f}(\mathbf{u}_0)$ is the load vector. These system matrices will depend on the current solution \mathbf{u}_0 . To solve Eq. 5.35, we can iteratively apply Eq. 5.36, replacing \mathbf{u} with \mathbf{u}_0 at each iteration, until $\mathbf{f}(\mathbf{u})$ converges to $\mathbf{0}$. This technique is known as Newton's method (see Eq. 3.12), and can be written as

$$\mathbf{u}^{v+1} = \mathbf{u}^v - \mathbf{K}(\mathbf{u}^v)^{-1} \mathbf{f}(\mathbf{u}^v)$$

where v is the iteration number. For highly non-linear problems, convergence of the method can be improved by using the damped Newton's method (see Eq. 3.13), written as:

$$\mathbf{u}^{v+1} = \mathbf{u}^v - \gamma \mathbf{K}(\mathbf{u}^v)^{-1} \mathbf{f}(\mathbf{u}^v)$$

where γ is a damping parameter between 0 and 1.

5.4 Further Reading

Detailed overviews on the finite element method can be found in the texts of Cook et al. [3], Johnson [7], and Chen [2]. More theoretical treatments are provided in the texts of Braess [1] and Larsson and Thomée [9]. A useful FEM text providing practical programming considerations is that of Hughes [6], whilst the text of Saad [11] is an excellent reference on solving sparse linear matrix systems, including the generalized minimal residual method (GMRES) as used in COMSOL. Finally, an overview of FEM models for bioengineering, particularly in the areas of biomechanics and fluid mechanics, is provided in the text of Kojić et al. [8].

Problems

5.1 Solve the following PDE numerically by-hand using FEM, utilising four equi-sized linear Lagrange elements over the interval $x \in [0, 1]$, and compare the solution obtained with the exact solution.

$$\begin{aligned}\nabla \cdot (-\nabla u) &= 2 \\ u(0) &= 1 \\ u(1) &= -1\end{aligned}$$

You may use Matlab to solve the resulting 7×7 system of equations.

5.2 Solve the following PDE numerically by-hand using FEM, utilising four equi-sized linear Lagrange elements over the interval $x \in [0, 1]$, and compare the solution obtained with the exact solution.

$$\begin{aligned}\nabla \cdot (-\nabla u) &= x \\ u(0) &= 1 \\ \frac{\partial u}{\partial x}(1) &= -1\end{aligned}$$

You may use Matlab to solve the resulting 6×6 system of equations.

5.3 For the family of cubic 1D Lagrange functions, we can define four cubic polynomial element shape functions $\varphi_i(\xi)$, $i = 1 \dots 4$, such that

$$\varphi_i(\xi_j) = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

where ξ_j is the local element coordinate of node j , and $i, j = 1 \dots 4$. Assuming these nodes are equi-spaced along the element, determine the four cubic Lagrange functions and plot their shape.

5.4 Using 1D quadratic basis functions on a uniformly-spaced grid consisting of 4 elements, determine the finite element \mathbf{D} , \mathbf{K} and \mathbf{f} matrices for the 1D time-dependent diffusion equation

$$\frac{\partial c}{\partial t} = \frac{\partial^2 c}{\partial x^2}$$

for $x \in [0, 1]$ subject to zero-flux boundary conditions at $x = 0$ and $x = 1$, with initial value of $c(x, t)$ at $t = 0$ given by the square-wave distribution:

$$c(x, 0) = \begin{cases} 1 & 0.4 \leq x \leq 0.6 \\ 0 & \text{otherwise} \end{cases}$$

Using the same basis functions and number of elements, solve this PDE in COMSOL, plotting the solution at $t = 0.1$, and confirm that the COMSOL-generated \mathbf{D} and \mathbf{K} matrices are equal to those you obtained analytically.

References

1. Braess D (2001) *Finite elements: theory, fast solvers, and applications in solid mechanics*, 2nd edn. Cambridge University Press, Cambridge
2. Chen Z (2011) *The finite element method: its fundamentals and applications in engineering*. World Scientific, Singapore
3. Cook RD, Malkus DS, Plesha ME, Witt RJ (2002) *Concepts and applications of finite element analysis*, 4th edn. Wiley, New York
4. Courant R (1943) Variational methods for the solution of problems of equilibrium and vibration. *Bull Am Math Soc* 49:1–23
5. Ferrari R, Silvester PP (2007) The finite-element method, part 2: an innovator in electromagnetic numerical modeling. *IEEE Ant Prop Mag* 49(3):216–234
6. Hughes TJR (2000) *The finite element method: linear static and dynamic finite element analysis*, Dover edn. Dover, Mineola
7. Johnson C (2009) *Numerical solution of partial differential equations by the finite element method*. Dover, Mineola
8. Kojić M, Filipović N, Stojanović B, Kojić N (2008) *Computer modelling in bioengineering*. Wiley, Chichester
9. Larsson S, Thomée V (2003) *Partial differential equations with numerical methods*. Springer, Berlin
10. Pelosi G, Courant RL (2007) The finite-element method, part 1. *IEEE Ant Prop Mag* 49(2):180–182
11. Saad Y (2003) *Iterative methods for sparse linear systems*, 2nd edn. SIAM, Philadelphia
12. Zienkiewicz OC, Cheung YK (1967) *The finite element method in structural and continuum mechanics*. McGraw-Hill, London

Part II

Bioengineering Applications

This section will cover basic theory of bioengineering models and their implementation in COMSOL software, focusing on electrical stimulation, diffusion processes, heat transfer as well as solid and fluid biomechanics.

Chapter 6

Modelling Electrical Stimulation of Tissue

6.1 Electrical Stimulation

Matter is composed of atoms consisting of positively-charged nuclei and negatively-charged electrons. These positive and negative charges produce and respond to *electromagnetic fields*, which are completely described by *Maxwell's equations*.¹

6.1.1 Maxwell's Equations

Maxwell's equations, expressed in so-called "macroscopic" form which characterize electromagnetic fields in materials, are:

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \quad (6.1)$$

$$\nabla \times \mathbf{H} = \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t} \quad (6.2)$$

$$\nabla \cdot \mathbf{D} = \rho_v \quad (6.3)$$

$$\nabla \cdot \mathbf{B} = 0 \quad (6.4)$$

where \mathbf{E} denotes the *electric field* (SI units: V m^{-1}), \mathbf{B} the *magnetic field* (SI units: T), \mathbf{D} the *electric displacement* (SI units: C m^{-2}), \mathbf{H} the *magnetization* (SI units: A m^{-1}), \mathbf{J} the applied *current density* (SI units: A m^{-2}) and ρ_v the *volume charge density* (SI units: C m^{-3}). Equation 6.1 is also known as *Faraday's Law of Induction* and states that a time-varying change in the local magnetic field will produce an electric field. Equation 6.2 represents an extension to *Ampère's Law*, which states that a steady electric current produces a magnetic field. Maxwell's contribution was

¹James Clerk Maxwell (1831–1879), Scottish mathematical physicist whose contributions to physics, along with those of Einstein and Newton, are regarded as greatest in the history of science.

the addition of a “displacement current” term $\partial \mathbf{D} / \partial t$, such that the magnetization field is also produced by a time-varying electric field in addition to a steady current. Equation 6.3 is known as *Gauss’ Law*, and states that an electric field is produced by electric charge. Equation 6.4 is also known as *Gauss’ Law for Magnetism*, and effectively states that there are no analogues of electric charge for magnetic fields (so-called “magnetic charges”).

For homogeneous linear isotropic materials, the electric displacement and magnetization fields satisfy

$$\begin{aligned} \mathbf{D} &= \varepsilon \mathbf{E} \\ \mathbf{B} &= \mu \mathbf{H} \\ \text{with } \mu \varepsilon &= \frac{1}{c^2} \end{aligned}$$

where ε is the *permittivity*, μ is the *permeability*, and c is the speed of light within the material.² Denoting the permittivity and permeability of free space by ε_0 and μ_0 , we can express ε and μ for any such material relative to their free-space values as

$$\begin{aligned} \varepsilon &= \varepsilon_0 \varepsilon_r \\ \mu &= \mu_0 \mu_r \end{aligned}$$

where ε_r, μ_r respectively denote the relative permittivity and permeability. Numerical values for ε_0 and μ_0 are:

$$\begin{aligned} \varepsilon_0 &= 8.854 \times 10^{-12} \text{ F m}^{-1} \\ \mu_0 &= 4\pi \times 10^{-7} \text{ H m}^{-1} \end{aligned}$$

where H denotes the Henry, the unit of electrical inductance. Substituting the above expressions for \mathbf{D} and \mathbf{H} into Maxwell’s equations, we can express these equations in terms of fields \mathbf{E} and \mathbf{B} as follows:

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \quad (6.5)$$

$$\nabla \times \mathbf{B} = \mu_r \mu_0 \mathbf{J} + \frac{1}{c^2} \frac{\partial \mathbf{E}}{\partial t} \quad (6.6)$$

$$\nabla \cdot (\varepsilon_r \varepsilon_0 \mathbf{E}) = \rho_v \quad (6.7)$$

$$\nabla \cdot \mathbf{B} = 0 \quad (6.8)$$

²The relationship between \mathbf{D} and \mathbf{E} is known as the electric *constitutive relation* of the material.

6.1.2 Electrostatic Formulations

If the electric and magnetic fields are steady, then the time derivatives in Eqs. 6.5–6.8 will vanish, resulting in the following formulations:

$$\begin{aligned} \nabla \cdot (\varepsilon_r \varepsilon_0 \mathbf{E}) &= \rho_v & \nabla \times \mathbf{E} &= \mathbf{0} & \text{(Electrostatics)} \\ \nabla \cdot \mathbf{B} &= 0 & \nabla \times \mathbf{B} &= \mu_r \mu_0 \mathbf{J} & \text{(Magnetostatics)} \end{aligned}$$

The above equations indicate that in the static case, electricity and magnetism are distinct phenomena: only in the presence of sufficiently rapid changes will \mathbf{E} and \mathbf{B} depend on each other.

The formulation $\nabla \times \mathbf{E} = \mathbf{0}$ states that the electrostatic electric field is curl-free. Recalling Example 4.4, such curl-free fields can be formed from the gradient of a scalar quantity. Defining this scalar as V , and letting

$$\mathbf{E} = -\nabla V \tag{6.9}$$

we can show that it has zero curl, using:

$$\nabla \times (-\nabla V) = - \begin{pmatrix} \frac{\partial[\nabla V]_z}{\partial y} - \frac{\partial[\nabla V]_y}{\partial z} \\ \frac{\partial[\nabla V]_x}{\partial z} - \frac{\partial[\nabla V]_z}{\partial x} \\ \frac{\partial[\nabla V]_y}{\partial x} - \frac{\partial[\nabla V]_x}{\partial y} \end{pmatrix} = - \begin{pmatrix} \frac{\partial}{\partial y} \left[\frac{\partial V}{\partial z} \right] - \frac{\partial}{\partial z} \left[\frac{\partial V}{\partial y} \right] \\ \frac{\partial}{\partial z} \left[\frac{\partial V}{\partial x} \right] - \frac{\partial}{\partial x} \left[\frac{\partial V}{\partial z} \right] \\ \frac{\partial}{\partial x} \left[\frac{\partial V}{\partial y} \right] - \frac{\partial}{\partial y} \left[\frac{\partial V}{\partial x} \right] \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

where we have used our earlier definition of the curl operator (Eq. 4.4). The scalar V in Eq. 6.9 is known as the *electric potential*, with SI Units of Volts or V. The SI units of electric field $\mathbf{E} = -\nabla V$ are therefore V m^{-1} .

The second equation of electrostatics is $\nabla \cdot (\varepsilon_r \varepsilon_0 \mathbf{E}) = \rho$. Using Eq. 6.9, this formulation is equivalent to:

$$\nabla \cdot (-\varepsilon_r \varepsilon_0 \nabla V) = \rho_v \tag{6.10}$$

Equation 6.10 represents a PDE in the electric potential that can be solved for in any medium given the distributed charge density and appropriate boundary conditions.³

³COMSOL provides the electrostatics module under its AC/DC physics application interface to solve such formulations. Equation 6.10 is specifically implemented using COMSOL's *charge conservation* domain setting in the electrostatics module.

6.1.3 Volume Conductor Theory

In this section, we derive the fundamental equations describing conservation of current in a 3D⁴ conductive medium, also known as a *volume conductor*.

6.1.3.1 Electrical Conductivity

If we wish to determine the voltage distribution in a conducting medium arising from the flow of electric current within it, we can utilise *Ohm's Law*.⁵ In 1D, Ohm's Law relates the voltage drop across a resistor to the current flow through it according to:

$$V = IR \quad (6.11)$$

where V is the potential difference across the resistor, I is the electric current flowing through it (in SI units of Ampères, or A), and R is its resistance (in SI units of Ohms, or Ω). The value of resistance will depend on the resistor material as well as its dimensions. For a resistor of length L and uniform cross-sectional area A , the total resistance across its ends (see Fig. 6.1) will depend on the *resistivity* ρ , a parameter of the material with SI units of $\Omega \text{ m}$, in accordance with *Pouillet's Law*⁶:

$$R = \frac{\rho L}{A} = \frac{L}{\sigma A} \quad (6.12)$$

where $\sigma = 1/\rho$ is the *conductivity* of the material in SI units of Siemens per metre, or S m^{-1} .

For a 3D block of material of infinitesimal length Δx , we can substitute Eq. 6.12 into Ohm's Law (Eq. 6.11), to obtain the voltage drop, ΔV , across it as:

$$\Delta V = IR = I \frac{L}{\sigma A} = I \frac{\Delta x}{\sigma A}$$

On re-arranging, we obtain

$$\frac{I}{A} = \sigma \frac{\Delta V}{\Delta x}$$

and taking the limit as $\Delta x \rightarrow 0$,

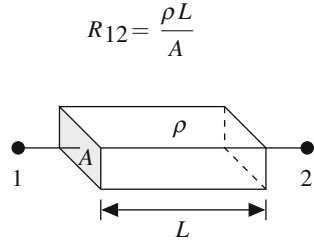
$$\frac{I}{A} = \sigma \frac{\partial V}{\partial x}$$

⁴The subsequent analysis may also be generalised to lower dimensions.

⁵Named after the German physicist Georg Simon Ohm (1789–1854).

⁶French physicist Claude Pouillet (1790–1868).

Fig. 6.1 Pouillet’s Law for the electrical resistance R_{12} between ends 1 and 2 shown of a material of cross-sectional area A , length L , and resistivity ρ



where the derivative on the right-hand side is simply the electric field in the x -direction along the infinitesimal length of the block, E_x . Defining the current density along the x -direction as $J_x = I/A$, the above becomes:

$$J_x = \sigma E_x$$

In a higher dimensional volume conductor, all components of current density and electric field will satisfy the above relationship. The equivalent expression of Ohm’s law therefore becomes

$$\mathbf{J} = \sigma \mathbf{E} \tag{6.13}$$

where current density \mathbf{J} has SI units of A m^{-2} . Using Eq.6.9, this expression is equivalent to

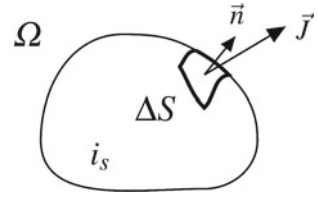
$$\mathbf{J} = -\sigma \nabla V \tag{6.14}$$

For isotropic materials, where the electrical conductivity is independent of direction, σ will be a scalar and \mathbf{J} and \mathbf{E} will be parallel to each other. Many physical structures however, including many biological tissues, exhibit preferential directions of conductivity whereby current flow is not in the same direction as the applied electric field. The white matter of the brain, for example, consists of numerous nerve fibres and tracts, which exhibit lower resistance along the local fibre direction than transverse to it. Such materials are said to possess *anisotropic* conductivity, in which σ in Eq.6.14 is represented by a conductivity *tensor*, equivalent in 3D to a 3×3 matrix:

$$\sigma = \begin{pmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{pmatrix} \tag{6.15}$$

where the σ_{ij} , ($i, j = 1, 2, 3$) denote the individual conductivity components. The eigenvectors of this tensor define the principle directions of conductivity, in which any applied electric field oriented in these directions will result in a current density parallel to it. For most anisotropic materials, it is possible to define three such principle directions which are orthogonal to each other. In such cases, the conductivity tensor will be *symmetric* and given by:

Fig. 6.2 Current flow \mathbf{J} leaving an arbitrary volume Ω through an infinitesimal surface element ΔS with outward normal \mathbf{n} . Ω contains a volumetric current source of i_s A m^{-3}



$$\sigma = \sigma_1 \mathbf{n}_1 \mathbf{n}_1^T + \sigma_2 \mathbf{n}_2 \mathbf{n}_2^T + \sigma_3 \mathbf{n}_3 \mathbf{n}_3^T \quad (6.16)$$

where σ_1 , σ_2 and σ_3 are the scalar conductivities along the three orthogonal principle directions given by the unit vectors \mathbf{n}_1 , \mathbf{n}_2 , \mathbf{n}_3 respectively.

6.1.3.2 Current Conservation

Now assume that an arbitrary volume Ω in a region of space is providing a source of current with volume current density i_s (in SI units of A m^{-3}), as shown in Fig. 6.2. From conservation of current, the total current i_{out} leaving the closed surface S of Ω must be

$$i_{out} = \int_{\Omega} i_s \, d\Omega = \int_S \mathbf{J} \cdot d\mathbf{S} \quad (6.17)$$

where $d\Omega$ denotes an infinitesimal volume element of Ω . Using the divergence theorem (Eq. 4.5), we can transform the right-hand surface integral above into

$$\begin{aligned} \int_S \mathbf{J} \cdot d\mathbf{S} &= \int_{\Omega} \nabla \cdot \mathbf{J} \, d\Omega \\ &= \int_{\Omega} \nabla \cdot (-\sigma \nabla V) \, d\Omega \quad (\text{using Eq. 6.14}) \end{aligned}$$

Substituting this identity into Eq. 6.17, we obtain

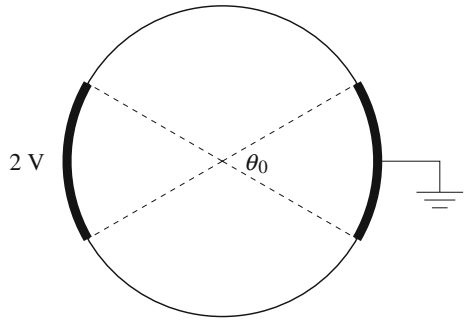
$$\int_{\Omega} i_s \, d\Omega = \int_{\Omega} \nabla \cdot (-\sigma \nabla V) \, d\Omega$$

Since this relationship holds true for any arbitrary volume Ω , the integrands must identically be equal. That is,

$$\nabla \cdot (-\sigma \nabla V) = i_s \quad (6.18)$$

Equation 6.18 represents the governing PDE for volume conductor electric current flow based on the principle of conservation of current. For most conductive media, the volumetric source current on the right-hand side will be zero, since there are no physical sources of current present. That is,

Fig. 6.3 Cell culture electric field stimulator



$$\nabla \cdot (-\sigma \nabla V) = 0 \tag{6.19}$$

However, for electrically active biological tissue, macroscopic approximations of tissue activity incorporate a volumetric current source due to cell membrane currents flowing between the intracellular and extracellular domains. In such cases, as will be seen later in this chapter, Eq. 6.18 is used. For inactive non-biological conductive media such as saline, Eq. 6.19 is used, as in the example of the next section.

6.1.4 Example: Cell Culture Electric Field Stimulator

We wish to design an electric field stimulator to experimentally determine the effect of electric fields on cell cultures in a Petri dish. The electric field is to be delivered using an identical pair of wire electrodes formed from circular arcs placed at opposite ends of the dish against its walls, as shown in Fig. 6.3. Each electrode subtends an angle of θ_0 at the centre of the dish, whose overall radius is 45 mm. The right electrode is grounded whilst the left electrode has a voltage of 2 V applied to it. All other boundaries are electrically-insulating. The electrical conductivity of the saline medium in which the cells are placed is 1 S/m, and the dish is filled to a height of 10 mm.

Using COMSOL, we will determine the optimal value of electrode angle θ_0 that maximises the area in which the electric field magnitude is within $\pm 10\%$ of its value at the dish centre (see also Problem 4.10). To do this, we implement the following sequence of steps in COMSOL:






Model Wizard

1. Open the Model Wizard and select the 2D spatial dimension.
2. In the Select Physics panel, choose AC/DC|Electric Currents. Click “Add”. For the dependent variable in the settings window, leave its name as V .
3. Click the Study arrow (🔍) to open the Select Study panel. Select Stationary, and click “Done”. This will exit the Model Wizard, displaying the main COMSOL interface.

Global Definitions

1. Right-click the Definitions sub-node under the Global node, and select Parameters. Specify the name of the parameter as `theta` and in the expression column enter `60 [deg]`. This will be the default electrode angle θ_0 .

Geometry

1. Select the Geometry 1 node in the model tree. In the settings window, change the default length unit to 'mm'.
2. Right-click Geometry 1 and select Circle. Specify the radius as 45 mm. Click Build Selected ().
3. Right-click Geometry 1 again and select Circle a second time. Specify the radius as 45 mm, the sector angle as `theta` and the rotation angle as `-theta/2`. Click Build Selected ().
4. Repeat the previous step and select Circle for a third time. Specify the the radius as 45 mm, the sector angle as `theta`, and the rotation angle as `180-theta/2`. Click Build Selected ().
5. Right-click Geometry 1 and select Booleans and Partitions|Union. For the input objects, select all three objects in the geometry (one circle and two sectors). Uncheck the checkbox 'Keep interior boundaries', and select Build Selected (). This will remove the interior sector lines, keeping electrode arc regions on the perimeter of the circular domain.
6. Finally, right-click Geometry 1 and select Point. Leave the default x, y values to 0 and select Build Selected (). This creates a geometric point at the centre of the dish.

Component Definitions

1. Right-click the Definitions sub-node under the Component 1 node, and select Component Couplings|Integration. Specify the geometric entity level as 'point', and select point 5 corresponding to the centre of the Petri dish. Leave the default operator name as `intop1`. This integration operator will simply return the value of a given expression evaluated at that point, making it available elsewhere in the model. We will use it to store the electric field magnitude at the centre of the dish.
2. Right-click the Component 1 Definitions sub-node again, and select Component Couplings|Integration. This time, keep the geometric entity level to its default type of 'domain', and select domain 1 corresponding to the entire dish. Leave the default operator name as `intop2`. This integration operator will be used to evaluate the area in which the electric field is relatively uniform.
3. Finally, right-click the Component 1 Definitions sub-node a third time and select 'Variables'. Keep the default geometric entity selection as 'Entire model'. Specify a variable with name `Ec` and expression `intop1 (ec . normE)`. This will equal the electric field magnitude at the centre of the dish. Specify a second variable with name `A` and expression:


```
intop2 ( (ec . normE > 0.9 * Ec) * (ec . normE < 1.1 * Ec) )
```

 Each of the boolean expressions in the argument of `intop2` will return a value of

1 if true or 0 if false.

The product $(ec.normE > 0.9 * Ec) * (ec.normE < 1.1 * Ec)$ is equivalent to a logical AND operation, returning a value of 1 if $(ec.normE > 0.9 * Ec)$ AND $(ec.normE < 1.1 * Ec)$, and 0 otherwise. As a result, the domain integration operator `intop2` will return the area of the dish for which the argument is true. That is, the area in which the electric field magnitude is within $\pm 10\%$ of its magnitude at the centre.

Electric Currents

1. Select the Electric Currents (`ec`) node of the model tree, and specify the out-of-plane thickness as 0.01 m (i.e. 10 mm, corresponding to the height of saline filling the dish).
2. Select the Current Conservation 1 sub-node of Electric Currents. In the settings panel, specify the electrical conductivity as ‘user-defined’ with a value of 1 S m^{-1} . Similarly, specify the relative permeability as user-defined, leaving the default value of 1.
3. Right-click the Current Conservation 1 sub-node and select ‘Electric Potential’. In the settings panel, select boundaries 1 and 2 (i.e. left electrode) and specify the electric potential as 2 V.
4. Right-click the Current Conservation 1 sub-node again and select ‘Ground’. In the settings panel, select boundaries 7 and 8 (i.e. the right electrode) to be the grounded boundaries.

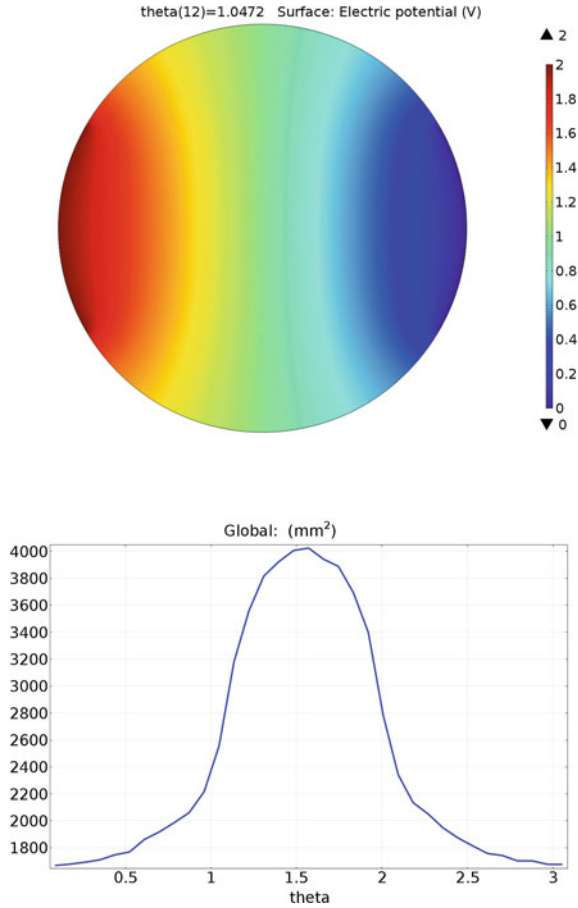
Study

1. Right-click the Study 1 node and select ‘Parametric Sweep’. In the settings panel, click the ‘Add’ button (+) to add parameter `theta`. To specify the parameter value list, click the ‘Range’ button (↕) and specify the start, step and stop values as 5, 5, 175 respectively. Click add to insert this range of parameter values. For the parameter unit, enter `deg`. This will setup a parameter sweep on the electrode angle from 5° to 175° in steps of 5° .
2. To solve the model, right-click Study 1 and select Compute (=). This will compute the solution for the specified range of parameter values of `theta`.

Results

1. Under the Results node of the model tree, select the Electric Potential (`ec`) sub-node. In the Settings window, specify the Parameter value of `theta` as 60° . Click the Plot button (📊) to display the electric potential variable V for this parameter value, as shown in Fig. 6.4.
2. Right-click the Result node and select ‘1D Plot Group’. Right-click the newly-created 1D Plot Group 2 sub-node and select ‘Global’. In the settings panel, click the ‘Add’ button (+) followed by Component 1|Definitions|Variables|A to add expression A to the y-axis data for plotting. Specify the y-axis data unit as mm^2 . For the data set, specify ‘Study 1/Parametric Solutions 1’. For the x-axis data, leave the default entry as ‘Parameter value’. Click the Plot button (📊) to display the plot of uniform electric field area versus electrode angle, as shown in Fig. 6.4.

Fig. 6.4 *Top* COMSOL-generated voltage distribution in Petri-dish for an electrode angle θ (i.e. θ_0) of 60° (≈ 1.0472 rad). *Bottom* COMSOL-generated plot of uniform electric field area (variable A) against electrode angle θ . As can be seen, the maximum uniform electric field area is $\approx 4000 \text{ mm}^2$ for $\theta_0 \approx \pi/2$ rad, or 90°

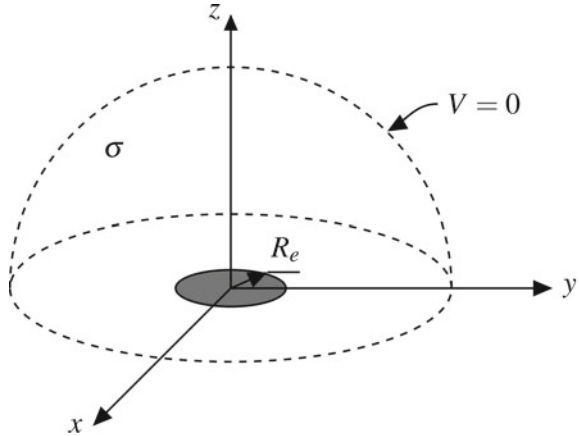


6.1.5 Example: Access Resistance of Electrode Disc

Consider a 2D circular disc electrode embedded in a 3D hemispherical infinite medium of conductivity $\sigma = 1 \text{ S m}^{-1}$ as shown in Fig. 6.5. The radius of the disc electrode is $R_e = 5 \text{ mm}$, and the potential at the infinite boundary of the hemisphere is taken to be ground (i.e. $V = 0$). Outside the electrode, the x-y plane boundary surrounding the electrode is assumed to be electrically-insulating. Assuming the surface of the disc electrode is at an isopotential value of $V_s = 1 \text{ V}$, plot the normal component of inward current density as a function of radial position across the electrode and determine the total electrode current I_s . Compare the normal component of current density in the disc with the theoretical solution (see Example 4.12):

$$J_n = \frac{2\sigma V_s}{\pi \sqrt{R_e^2 - r^2}}$$

Fig. 6.5 Circular-disc electrode of radius R_e stimulating a hemispherical infinite domain of conductivity σ . The boundary at infinity is at ground (i.e. $V = 0$)



where r is the radial distance from the centre of the disc. Find also the access resistance Z of the electrode, defined as the ratio V_s/I_s , comparing this with the theoretical value of

$$Z = \frac{1}{4\sigma R_e}$$

To solve this model in COMSOL, we can use a 2D axisymmetric geometry due to the rotational symmetry about the z -axis (Fig. 6.5). COMSOL also provides the option of adding an *infinite domain* to represent regions with boundaries infinitely far away.⁷ To implement this model in COMSOL, we can utilise the following steps:

Model Wizard

1. Open the Model Wizard and select the 2D Axisymmetric spatial dimension.
2. In the Select Physics panel, choose AC/DC|Electric Currents. Click “Add”. For the dependent variable in the settings window, leave its name as V .
3. Click the Study arrow (🔍) to open the Select Study panel. Select Stationary, and click “Done”. This will exit the Model Wizard, displaying the main COMSOL interface.

Global Definitions

1. Right-click the Definitions sub-node under the Global node, and select Parameters. Specify the name of the parameter as V_s and in the expression column enter 1 [V] . This will be the electrode voltage. Specify a second parameter with name σ and expression 1 [S/m] to define the conductivity. Specify a third parameter R_e with expression 5 mm . This is the electrode disc radius. Finally, specify

⁷Available in the AC/DC module. Infinite domains are implemented by applying a coordinate transformation within the domain such that the transformed coordinate tends to infinity on the appropriate boundary, whilst the spatial coordinate remains finite.

a fourth parameter Z_{th} with expression $1 / (4 * \sigma * Re)$: this is the theoretical access resistance. COMSOL evaluates this expression in terms of the other specified parameters, displaying its value as 50Ω .

Geometry

1. Select the Geometry 1 node in the model tree. In the settings window, change the default length unit to 'mm'.
2. Right-click Geometry 1 and select Circle. Specify the radius as 10 mm and the sector angle as 90° . Click Build Selected ().
3. Right-click Geometry 1 again and select Circle a second time. Specify the radius as 15 mm and the sector angle as 90° . Click Build Selected (). To zoom out to the entire geometry, click the Zoom Extents button ().
4. Right-click Geometry 1 and select Polygon. In the settings window, specify the r coordinate values as 0 Re (insert a space between these two entries), and for the z coordinate, specify 0 0 (again with a space). Click Build Selected () to draw the electrode as a line interval of length 5 mm on the lower boundary. Once complete, the geometry and model tree will look like that shown in Fig. 6.6.

Component Definitions

1. Right-click the Definitions sub-node under the Component 1 node, and select Component Couplings|Integration. Specify the geometric entity level as 'boundary', and select boundary 2 corresponding to the electrode. Leave the default operator name as intop1. This integration operator will return the integral of its

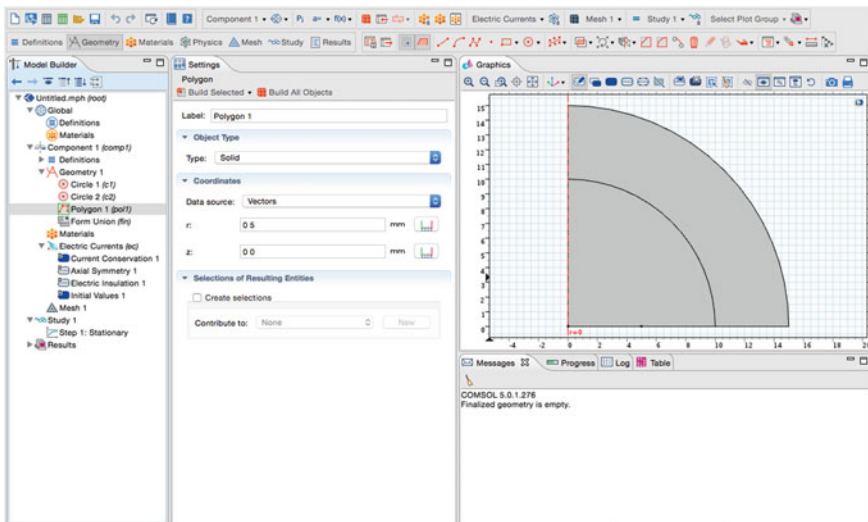


Fig. 6.6 COMSOL interface for axisymmetric electrode disc example, showing geometry and model tree. The edge of the electrode disc can be seen as a point on the *lower boundary* at $r = 5$ mm. The *outer quarter-circle* represents the infinite domain

argument over the electrode. We will use it to determine the total inward current i_s by integrating the normal component of current density as

$$i_s = \int_0^{R_e} 2\pi r J_n dr \quad (6.20)$$

2. Right-click the Component 1 Definitions sub-node again, and select ‘Variables’. Keep the default geometric entity selection as ‘Entire model’. Specify a variable with name I_s and expression $\text{intop1}(2*\pi*r*ec.Jn)$. This will equal return the total electrode current, according to Eq. 6.20. Specify a second variable with name J_{th} and expression $2*\sigma*V_s/(\pi*\text{sqrt}(R_e^2-r^2))$ to specify the theoretical current density. Finally, specify an access resistance variable with name Z and expression V_s/I_s .
3. Right-click the Component 1 Definitions sub-node a third time and select ‘Infinite Element Domain’. Specify domain 2 in the settings panel.

Electric Currents

1. Select the Current Conservation 1 sub-node of Electric Currents. In the settings panel, specify the electrical conductivity as ‘user-defined’ with a value of σ . Similarly, specify the relative permeability as user-defined, leaving the default value of 1.
2. Right-click the Current Conservation 1 sub-node and select ‘Electric Potential’. In the settings panel, select boundary 2 (i.e. electrode disc) and specify the electric potential as V_s .
3. Right-click the Current Conservation 1 sub-node again and select ‘Ground’. In the settings panel, select boundary 7 (i.e. infinite boundary) to be the ground.

Mesh

1. Right-click the Mesh 1 node and select ‘Distribution’. In the settings panel, select boundary 2 and specify 100 as the number of elements.
2. Right-click the Mesh 1 node again and select ‘Free Triangular’, leaving the geometric entity level to its default value of ‘Remaining’. Click the Build all button (🔧) to build and display the mesh. A view of the resulting mesh is shown in Fig. 6.7.

Study

1. To solve the model, right-click Study 1 and select Compute (🔍). This will compute the solution and display the default plot of voltage, as shown in Fig. 6.8.

Results

1. Right-click the Results node of the model tree, and select ‘1D Plot Group’. Right-click the 1D Plot Group 3 sub-node and select ‘Line Graph’. In the Settings window, select boundary 2 in the boundary selection list. Under the y-Axis Data tab, specify $ec.nJ$ as the expression to plot. For the x-Axis Data under Parameter, select ‘Expression’ and specify r . Under the Legends tab, check the ‘Show legends’ checkbox and specify the ‘Manual’ option. In the Legends table, enter the text ‘Computed’.

Fig. 6.7 COMSOL-generated mesh for axisymmetric disc electrode geometry. A higher density of elements can be seen over the electrode disc on the lower left boundary

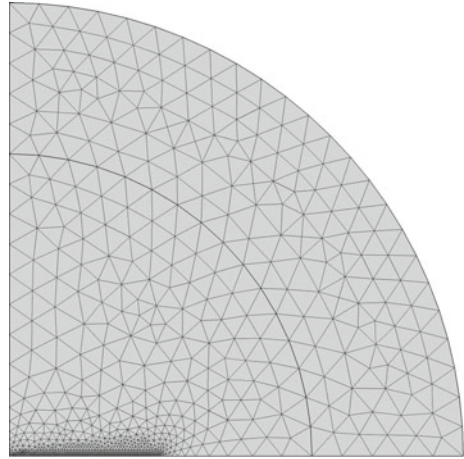
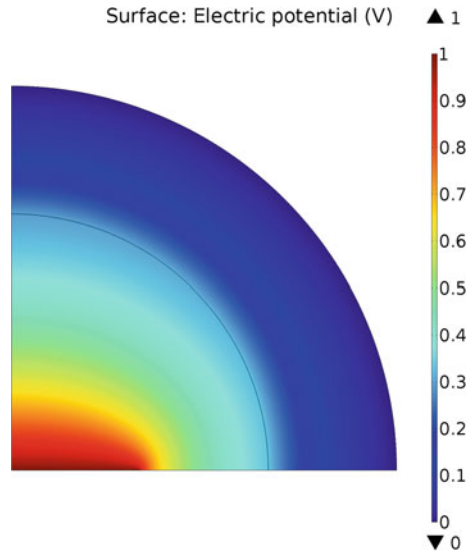


Fig. 6.8 Computed voltage distribution around disc electrode. The electrode at bottom left is at an isopotential level of 1 V, whilst the boundary at infinity is at 0 V



2. Right-click the 1D Plot Group 3 sub-node and select 'Line Graph' again. In the Settings window, select boundary 2 and specify $J_{\tau h}$ as the expression to plot. For the x-Axis Data under Parameter, select 'Expression' and specify r . Under the Legends tab, check the 'Show legends' checkbox and specify the 'Manual' option. In the Legends table, enter the text 'Theoretical'. Click the Plot button (🖨) to display the current density plots, as shown in Fig. 6.9. As can be seen from the plots, the computed and theoretical solutions for current density across the electrode agree closely.
3. Right-click the Derived Values sub-node under Results and select 'Global Evaluation'. In the settings panel, specify Z as the expression to evaluate. Click the

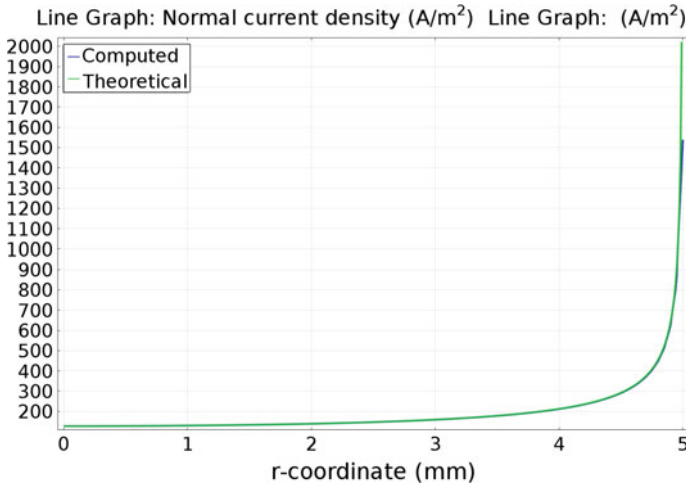


Fig. 6.9 Computed and theoretical normal current density as a function of radial position across the disc electrode

Evaluate button (=) to display the value of the access resistance in a table below the graphics window. COMSOL determines this value to be 52.65Ω which is slightly higher than the 50Ω theoretical value. The discrepancy is due to the mesh resolution not being able to capture the theoretical infinite current density at the edge of the disc. An even finer mesh resolution would improve the estimate of this resistance value.

6.2 Modelling Electrical Activity of Tissues

This section will describe macroscopic approximations of excitable tissue electrical activity, and how these can be used to formulate models of tissue electrical stimulation. An example of such an approach may be found in the model of cardiac defibrillation described in Appendix B.

6.2.1 Continuum Models of Excitable Tissues

As described in Sect. 6.1.3.2, the electrostatic PDE describing voltage distribution in a volume conductor is Eq. 6.18:

$$\nabla \cdot (-\sigma \nabla V) = i_s$$

This equation is the basis of continuum formulations of the electrical activity of excitable tissues and their modulation by extracellular stimulation. We begin with the formulation of nerve axon membrane current proposed by Hodgkin and Huxley (see Sect. 2.3.2), which we will generalise to any excitable tissue:

$$i_m = \overbrace{C_m \frac{dV_m}{dt}}^{\text{capacitive current}} + \overbrace{i_{Na} + i_K + i_L}_{\text{ionic current}} \quad (6.21)$$

where i_m is the total cell membrane current per unit membrane area, C_m , i_{Na} , i_K , i_L are the membrane capacitance, sodium, potassium and leakage currents per unit membrane area respectively, and V_m is the transmembrane voltage, defined as

$$V_m = V_i - V_e$$

where V_i , V_e are the intracellular and extracellular voltages. Equation 6.21 describes total membrane current as consisting of capacitive and ionic components. The equation may be generalised for any excitable cell type (neural, muscular, sensory) by substituting in a generic ionic current, i_{ion} as:

$$i_m = C_m \frac{dV_m}{dt} + i_{ion} \quad (6.22)$$

where i_{ion} is particular to the cell type in question. Rather than model every individual cell in the tissue, we can employ a macroscopic approximation by assuming the tissue consists of two interpenetrating domains: the intracellular and extracellular spaces. That is, at every point in the excitable tissue, we can define coupled variables V_i , V_e corresponding to the intracellular and extracellular voltages. Using Eq. 6.18, we can write the equations for both domains as:

$$\nabla \cdot (-\sigma_e \nabla V_e) = \beta i_m \quad (\text{extracellular domain}) \quad (6.23)$$

$$\nabla \cdot (-\sigma_i \nabla V_i) = -\beta i_m \quad (\text{intracellular domain}) \quad (6.24)$$

where β is the tissue *surface-to-volume ratio* (in SI units of m^{-1}). In these equations, the extracellular volume current source i_s (Eq. 6.18) has been replaced with βi_m , and the corresponding current source for the intracellular domain has been replaced with an equal and opposite magnitude, $-\beta i_m$. The factor β is a parameter of the tissue representing the total cell membrane surface area per unit volume.

Substituting Eq. 6.22 into Eqs. 6.23 and 6.24, we obtain the *bidomain equations*:

$$\nabla \cdot (-\sigma_e \nabla V_e) = \beta \left(C_m \frac{\partial V_m}{\partial t} + i_{ion} \right) \quad (\text{extracellular domain}) \quad (6.25)$$

$$\nabla \cdot (-\sigma_i \nabla V_i) = -\beta \left(C_m \frac{\partial V_m}{\partial t} + i_{ion} \right) \quad (\text{intracellular domain}) \quad (6.26)$$

Furthermore, substituting the expression $V_m = V_i - V_e$, and incorporating extracellular and intracellular volume stimulus current sources i_{se} and i_{si} , we obtain the full-form of the bidomain equations:

$$\nabla \cdot (-\sigma_e \nabla V_e) = \beta \left(C_m \frac{\partial V_i}{\partial t} - C_m \frac{\partial V_e}{\partial t} + i_{ion} \right) + i_{se} \quad (6.27)$$

$$\nabla \cdot (-\sigma_i \nabla V_i) = -\beta \left(C_m \frac{\partial V_e}{\partial t} - \frac{\partial V_i}{\partial t} - i_{ion} \right) + i_{si} \quad (6.28)$$

where i_{si} and i_{se} are external stimulus currents per unit volume for the intracellular and extracellular domains respectively. External intracellular stimulus currents include stimuli delivered directly to the cell interiors through inserted microelectrodes. External extracellular stimulus currents are delivered directly to the extracellular space, but are not typically specified as a volume current source as above, but rather as boundary conditions on the electrode boundaries of the extracellular volume.

By fixing the extracellular potential everywhere to ground (i.e. $V_e = 0$), the bidomain equations reduce to the simpler *monodomain formulation*:

$$\nabla \cdot (\sigma \nabla V_m) = \beta \left(C_m \frac{\partial V_m}{\partial t} + i_{ion} \right) \quad (6.29)$$

where σ denotes the intracellular conductivity of the tissue. Although much simpler than the bidomain equations, the monodomain Eq. 6.29 cannot be used to simulate the effects of an applied extracellular stimulus, or to simulate extracellular voltage signals such as the ECG arising from the electrical activity of tissue.

6.2.2 Example: Modelling Spiral-Wave Reentry in Cardiac Tissue

In this example, we will simulate spiral-wave reentry in cardiac tissue using a modified bidomain formulation of the Rogers-McCulloch PDE formulations [8] for the cardiac action potential:

$$\begin{aligned} \nabla \cdot (-\sigma_e \nabla V_e) &= \beta \left(C_m \frac{\partial V_m}{\partial t} + i_{ion} \right) + i_{stim} \\ \nabla \cdot (-\sigma_i \nabla V_i) &= -\beta \left(C_m \frac{\partial V_m}{\partial t} + i_{ion} \right) \\ \frac{\partial u}{\partial t} &= e (V_m - du - b) \end{aligned}$$

Table 6.1 Model parameters for cardiac spiral wave reentry

| Parameter | Value | Parameter | Value |
|-----------|-------------------------------------|---------------|--------------------------|
| A | 55 mV | c_2 | $400 \mu\text{Scm}^{-2}$ |
| B | -85 mV | C_m | $1 \mu\text{Fcm}^{-2}$ |
| a | -66.8 mV | β | 100m^{-1} |
| b | -85 mV | σ_{ex} | 0.1mScm^{-1} |
| d | 140 mV | σ_{ey} | 0.025mScm^{-1} |
| e | $285.7 \text{V}^{-1} \text{s}^{-1}$ | σ_{ix} | 0.2mScm^{-1} |
| c_1 | $530 \text{S V}^{-2} \text{m}^{-2}$ | σ_{iy} | 0.1mScm^{-1} |

with

$$i_{ion} = c_1(V_m - a)(V_m - A)(V_m - B) + c_2u(V_m - B)$$

$$\sigma_e = \begin{pmatrix} \sigma_{ex} & 0 \\ 0 & \sigma_{ey} \end{pmatrix}$$

$$\sigma_i = \begin{pmatrix} \sigma_{ix} & 0 \\ 0 & \sigma_{iy} \end{pmatrix}$$

where u is an auxiliary ‘recovery’ variable, σ_i and σ_e are the intra- and extracellular conductivity tensors, with principle conductivities along the global x, y axes given by $\sigma_{ix}, \sigma_{ex}, \sigma_{iy}, \sigma_{ey}$, β is the tissue surface to volume ratio, C_m is cell membrane capacitance per unit area, i_{ion} is the ionic current per unit cell membrane area, i_{stim} is the applied extracellular stimulus current per unit tissue volume, and A, B, a, b, d, e, c_1 and c_2 are parameters describing the active electrical activity of the atria, as given in Table 6.1. Initial values at $t = 0$ throughout the tissue are $V_e = 0$ mV, $V_i = -85$ mV and $u = 0$.

The cardiac tissue domain is a 2D square of side-length 10 cm, with the left-hand corner located at $x = 0, y = 0$, where x, y represent the 2D spatial coordinates. Zero-flux boundary conditions for V_e, V_i apply on all four external boundaries of the domain. Furthermore, the extracellular potential at the top right-hand corner at $x = 10$ cm, $y = 10$ cm is held at ground (i.e. $V_e = 0$ V). The stimulus current is given by:

$$i_{stim}(x, y, t) = \begin{cases} 200 \text{ A m}^{-3} & y \leq 1 \text{ cm}, 10 \text{ ms} \leq t \leq 11 \text{ ms} \\ -200 \text{ A m}^{-3} & y \geq 9 \text{ cm}, 10 \text{ ms} \leq t \leq 11 \text{ ms} \\ 200 \text{ A m}^{-3} & x \leq 1 \text{ cm}, 110 \text{ ms} \leq t \leq 111 \text{ ms} \\ -200 \text{ A m}^{-3} & x \geq 9 \text{ cm}, 110 \text{ ms} \leq t \leq 111 \text{ ms} \\ 0 & \text{otherwise} \end{cases}$$

To setup and solve this model in COMSOL, we implement the following steps:

Model Wizard

1. Open the Model Wizard and select the 2D spatial dimension.
2. In the Select Physics panel, choose Mathematics|PDE Interfaces|General Form PDE, and click “Add”. In the Review Physics panel at right, specify V as the Field name and 2 as the number of dependent variables. In the dependent variables list, enter the names of these variables as V_e and V_i . For the dependent variable quantity, specify the units as Electric potential (V), and the source term quantity as Current source (A/m^3).
3. Next, select again Mathematics|PDE Interfaces|General Form PDE, and click “Add”. This will insert a second General Form PDE into the model. In the Review Physics panel, leave the field name as u and the number of dependent variables as 1. Leave the units of the dependent variable as Dimensionless, but enter the source term units manually as $1/s$.
4. Click the Study arrow (🟢) to open the Select Study panel. Select Time Dependent, and click “Done”. This will exit the Model Wizard, displaying the main COMSOL interface.

Global Definitions

1. Right-click Global Definitions in the model tree and select Parameters. Enter the following details in the Parameters table of the Settings window:

| Name | Expression | Description |
|--------|----------------------|------------------------------|
| A | 55 [mV] | Model parameter |
| B | -85 [mV] | Model parameter |
| a | -66.8 [mV] | Model parameter |
| b | -85 [mV] | Model parameter |
| d | 140 [mV] | Model parameter |
| e | 285.7 [$1/(V*s)$] | Model parameter |
| c_1 | 530 [$S/(V*m)^2$] | Model parameter |
| c_2 | 400 [$\mu S/cm^2$] | Model parameter |
| C_m | 1 [$\mu F/cm^2$] | Membrane capacitance |
| beta | 100[1/m] | Surface to volume ratio |
| s_ex | 0.1 [mS/cm] | Extracellular x conductivity |
| s_ey | 0.025 [mS/cm] | Extracellular y conductivity |
| s_ix | 0.2 [mS/cm] | Intracellular x conductivity |
| s_iy | 0.1 [mS/cm] | Intracellular y conductivity |
| I_stim | 200 [A/m^3] | Stimulus amplitude |
| T1_on | 10 [ms] | Stimulus 1 onset |
| T2_on | 110 [ms] | Stimulus 2 onset |
| T_dur | 1 [ms] | Stimulus duration |

2. Right-click Global Definitions and select Functions|Rectangle. Specify the lower limit as $T1_on$ and the upper limit as $T1_on+T_dur$. In the Smoothing tab, specify the size of the transition zone as $T_dur/10$. Leave the function name to its default (rect1).

3. Right-click Global Definitions again and select Functions|Rectangle. Specify the lower limit as $T2_on$ and the upper limit as $T2_on+T_dur$. In the Smoothing tab, specify the size of the transition zone as $T_dur/10$. Leave the function name to its default (rect2).

Geometry

1. Select the Geometry 1 node in the model tree. In the settings window, change the length unit to 'cm'.
2. Right-click Geometry 1 and select Square. Specify the side length as 10 cm and click Build Selected ().
3. Right-click Geometry 1 again and select Rectangle. Specify the width as 1 cm and the height as 10 cm. Click Build Selected ().
4. Right-click Geometry 1 a third time and select Rectangle again. In the settings window, specify the width as 10 cm and the height as 1 cm. Click Build Selected ().
5. Right-click Geometry 1 a fourth time and select Rectangle again. In the settings window, specify the width as 10 cm, the height as 1 cm, and the corner at $x = 0$ cm, $y = 9$ cm. Click Build Selected ().
6. Right-click Geometry 1 a fifth time and select Rectangle again. In the settings window, specify the width as 1 cm, the height as 10 cm, and the corner at $x = 9$ cm, $y = 0$ cm. Click Build Selected (). Once complete, the geometry and model tree will look like that shown in Fig. 6.10.

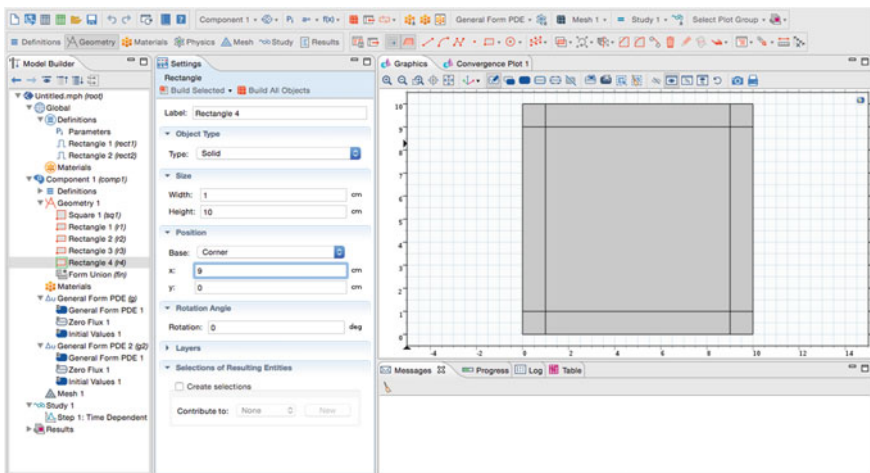


Fig. 6.10 COMSOL interface for cardiac spiral wave reentry example, showing geometry and model tree. The *rectangular* subdomains on the periphery of the tissue square correspond to regions where stimuli are delivered

Component Definitions

1. Right-click the Definitions sub-node of Component 1 and select Variables. Leave the geometric entity level to its default as ‘Entire model’, and enter the following variables in the settings table:

| Name | Expression |
|--------------|---|
| V_m | $V_i - V_e$ |
| i_{ion} | $c_1 * (V_m - a) * (V_m - A) * (V_m - B) + c_2 * u * (V_m - B)$ |
| i_{stim_1} | $I_{stim} * rect1(t \text{ [1/s]})$ |
| i_{stim_2} | $I_{stim} * rect2(t \text{ [1/s]})$ |

2. Right-click the Definitions sub-node of Component 1 a second time and select Variables again. This time, specify the geometric entity level as ‘Domain’, and select domain 1 corresponding to the small square region in the the lower left-hand corner. Define a variable with name `stim` and expression $i_{stim_1} + i_{stim_2}$. This specifies domain 1 as a region in which both stimuli are delivered, albeit at different times.
3. Right-click the Definitions sub-node of Component 1 a third time and select Variables again. As above, specify the geometric entity level as ‘Domain’, and select domain 2 corresponding to the mid-rectangular sub-domain on the left-hand edge of the tissue. Define a variable with the same name as above, `stim`, but this time with expression i_{stim_2} . This specifies domain 2 as the region in which the second stimulus is delivered.
4. Right-click the Definitions sub-node of Component 1 a fourth time and select Variables again. As above, specify the geometric entity level as ‘Domain’, and now select domain 3 corresponding to the small square sub-domain at top-left of the tissue. Define another variable with the name `stim` and expression $-i_{stim_1} + i_{stim_2}$. This specifies domain 3 as the region in which a negative first stimulus is delivered followed by a positive second stimulus.
5. Right-click the Definitions sub-node of Component 1 a fifth time and select Variables again. As above, specify the geometric entity level as ‘Domain’, and now select domain 4 corresponding to the mid-rectangular sub-domain on the bottom edge of the tissue. Define a variable with name `stim` and expression i_{stim_1} . This specifies domain 4 as the region in which the first stimulus is delivered.
6. Right-click the Definitions sub-node of Component 1 a sixth time and select Variables. Again, specify the geometric entity level as ‘Domain’, and select domain 5 corresponding to the larger square sub-domain occupying the central portion of the tissue. Define another variable with the name `stim` and expression 0. This specifies domain 5 as the region in which no stimulus is delivered.
7. Right-click the Definitions sub-node of Component 1 a seventh time and select Variables again. As above, specify the geometric entity level as ‘Domain’, and now select domain 6 corresponding to the mid-rectangular sub-domain on

the top edge of the tissue. Define a variable with name `stim` and expression `-i_stim_1`. This specifies domain 6 as the region in which the negative of the first stimulus is delivered.

8. Right-click the Definitions sub-node of Component 1 an eighth time and select Variables again. As above, specify the geometric entity level as 'Domain', and now select domain 7 corresponding to the small square sub-domain at the bottom-right corner of the tissue. Define another variable with the name `stim` and expression `i_stim_1-i_stim_2`. This specifies domain 7 as the region in which the first stimulus is delivered followed by a negative second stimulus.
9. Right-click the Definitions sub-node of Component 1 a ninth time and select Variables again. As above, specify the geometric entity level as 'Domain', and now select domain 8 corresponding to the mid-rectangular sub-domain on the right edge of the tissue. Define a variable with name `stim` and expression `-i_stim_2`. This specifies domain 8 as the region in which the negative of the second stimulus is delivered.
10. Finally, right-click the Definitions sub-node of Component 1 a tenth time and select Variables again. As above, specify the geometric entity level as 'Domain', and now select domain 9 corresponding to the small square sub-domain at the top-right corner of the tissue. Define another variable with the name `stim` and expression `-i_stim_1-i_stim_2`. This specifies domain 9 as the region in which the negative of the first stimulus is delivered followed by the negative of the second stimulus.

General Form PDE

1. Select the General Form PDE 1 sub-node of General Form PDE, and enter the following expressions for the conservative flux I components for variables V_e and V_i respectively:

| Component | Expression |
|-----------|------------------------|
| x | <code>-s_ex*Vex</code> |
| y | <code>-s_ey*Vey</code> |
| x | <code>-s_ix*Vix</code> |
| y | <code>-s_iy*Viy</code> |

Enter the following expressions for the source term f corresponding to variables V_e and V_i respectively:

```
beta*i_ion + stim
-beta*i_ion
```

Finally, enter the following terms for the damping coefficient matrix d_a :

```
beta*C_m      -beta*C_m
-beta*C_m      beta*C_m
```

Leave the mass coefficient matrix e_a entries to their default value of 0.

2. Right-click General Form PDE again and select Points|Pointwise Constraint. In the settings panel, specify point 16 corresponding to the upper right-hand corner



of the square domain. Under the Constraint expression tab in the settings panel, specify V_e . The constraint expression is held to a value of 0: in this case, this is equivalent to specifying $V_e = 0$ at point 16.

3. Select the Initial Values 1 sub-node of the General Form PDE node. Leave the initial value of V_e to its default value of 0. For V_i however, enter the initial value expression -85 [mV].


General Form PDE 2

1. Select the General Form PDE 1 sub-node of General Form PDE 2, and enter a value of 0 for each of the two components of conservative flux Γ , since there are no spatial derivatives in the equation for u . For the source term f , enter the expression $e * (\nabla m - d * u - b)$ and for the damping and mass coefficients d_a and e_a , leave their value as 1 and 0 respectively.

Study

1. Select the Step1: Time Dependent sub-node of the Study 1 node. In the Settings window, Click the Range button () adjacent to the Times field. Leave the entry method as 'Step' and enter Start, Step and Stop values of 0, 0.001 and 0.6 respectively. Click Replace. This will create a range of output time values from 0 to 0.6 s in time steps of 0.001 s.
2. Right-click the Study 1 node and select Show Default Solver. Select the Study 1|Solver Configurations|Solver 1|Time-Dependent Solver 1 node. In the Settings window, expand the Time Stepping tab and specify 'Strict' for the steps taken by solver option. Expand the Advanced tab, and for the 'Singular mass matrix' option, select 'Yes'.
3. To solve the model, right-click Study 1 and select Compute (). Select the Progress tab in the Information window to see the solver progress.

Results

1. When the model has completed solving,⁸ the Graphics window will display a default plot of the extracellular potential (variable V_e) at $t = 0.6$ s, as shown in Fig. 6.11.
2. To plot the membrane potential at this time, select the Surface 1 sub-node of 2D Plot Group 1. In the settings panel, type the expression to plot as V_m . Specify the unit as 'mV' and click the Plot button (). Selecting the parent 2D Plot Group 1 node, the required time can be selected from the list of times stored in the solution output. Figure 6.12 shows the plot of membrane potential at $t = 0.54$ s. The solution for V_m consists of a spiral wavefront rotating counter-clockwise in the tissue, also referred to as reentrant activation.
3. To plot the V_m and V_e waveforms at a given point in the tissue, right-click the Data Sets sub-node under Results and select 'Cut Point 2D'. In the settings panel, specify the Point Data coordinates as $x = 5$ cm, $y = 5$ cm.

⁸Using a MacBook Air 2013 with 8GB RAM and OS X 10.8.5, it took 47 s to solve this model.

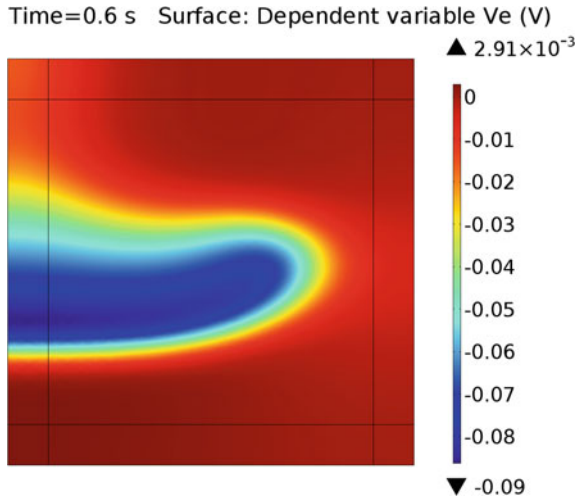


Fig. 6.11 Extracellular potential V_e at $t = 0.6$ s for cardiac reentry simulation

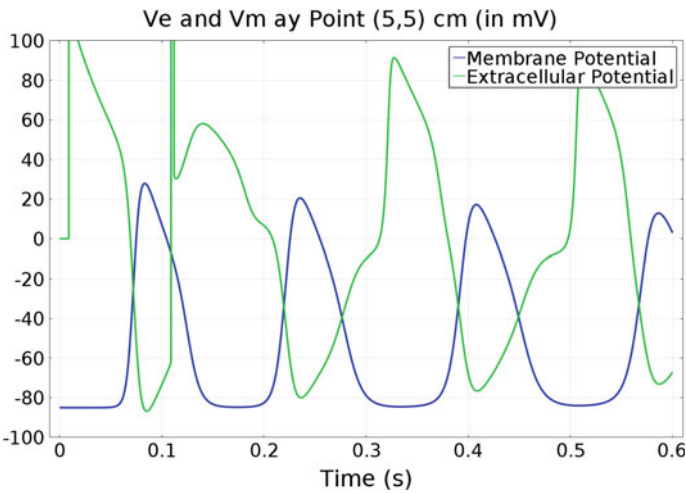


Fig. 6.12 Membrane potential V_m at $t = 0.54$ s for cardiac reentry simulation

4. Right-click the Results node of the model tree, and select '1D Plot Group'. Right-click the 1D Plot Group 3 sub-node and select 'Point Graph'. In the Settings window, select Cut Point 2D 1 for the Data set. Under the y-Axis Data tab, specify V_m as the expression to plot, with units of 'mV'. Under the Legends tab, check the 'Show legends' checkbox and specify the 'Manual' option. In the Legends table, overwrite the default text of '(5,5)' with the text 'Membrane Potential'.
5. Right-click the 1D Plot Group 3 sub-node and select 'Point Graph' again. In the Settings window, select Cut Point 2D 1 for the Data set and specify V_e as the

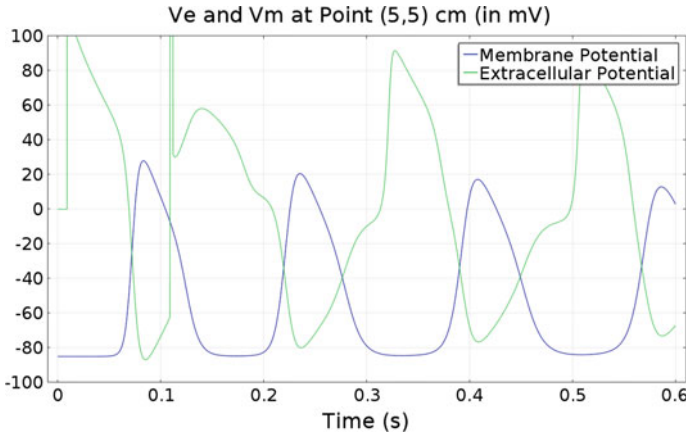



Fig. 6.13 Membrane V_m and extracellular potential V_e waveforms at point (5,5) cm corresponding to the centre of the tissue for the cardiac reentry simulation

expression to plot in units of ‘mV’. Under the Legends tab, check the ‘Show legends’ checkbox and specify the ‘Manual’ option. In the Legends table, enter the text ‘Extracellular Potential’.

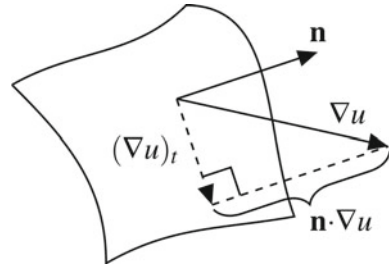
6. Select the 1D Plot Group 3 sub-node and in the settings window, expand the Title tab and select ‘Manual’ for the Title type. Enter the text ‘Ve and Vm at Point (5,5) cm (in mV)’. Under the Axis tab, check the Manual axis limits checkbox and enter the following values: x minimum (−0.01), x maximum (0.61), y minimum (−100), y maximum (100). Click the Plot button () to display the point plots, as shown in Fig. 6.13.

6.2.3 Modelling PDEs/ODEs on Boundaries, Edges and Points

Many models require the formulation of PDEs/ODEs on lower-dimensional boundaries, edges or points, in addition to any PDEs that may be specified in the main spatial domain. For example, to simulate the 3D electrical activity of cardiac atrial tissue, the atria can be modelled as 2D surfaces embedded in 3D space owing to the small thickness of their walls. A 2D PDE such as Eq. 6.29 can then be employed to simulate electrical activity on these surfaces. To implement such equations in COMSOL, we can utilise use the Lower Dimensions PDE application modes (boundary, edge or point) of COMSOL’s Mathematics Physics interface.

For a COMSOL variable u defined only on a lower dimensional boundary, edge or point, the standard COMSOL spatial derivative variables u_x , u_y , u_z are not defined on this boundary, and cannot be used to specify the components of flux Γ in

Fig. 6.14 Tangential gradient $(\nabla u)_t$, defined as the projection of ∇u onto a lower-dimensional boundary with outward unit normal \mathbf{n}



COMSOL's PDE General form:

$$e_a \frac{\partial^2 u}{\partial t^2} + d_a \frac{\partial u}{\partial t} + \nabla \cdot \Gamma = f \quad (6.30)$$

To specify the components of Γ , we instead utilise COMSOL's *tangential derivative* in-built variables u_{Tx} , u_{Ty} , and u_{Tz} , which are available on the boundary. To understand the concept of tangential derivatives, consider a variable u which is defined everywhere in the higher-dimensional domain of a model. If ∇u is the gradient of u , then on a lower-dimensional boundary with outward unit normal \mathbf{n} , the tangential gradient, $(\nabla u)_t$, is defined as the projection of ∇u onto the boundary, as shown in Fig. 6.14. Referring to this figure, we can determine the tangential gradient according to:

$$\begin{aligned} (\nabla u)_t &= \nabla u - \mathbf{n}(\mathbf{n} \cdot \nabla u) \\ &= \nabla u - \mathbf{n}(\mathbf{n}^T \nabla u) \\ &= (\mathbf{I} - \mathbf{n}\mathbf{n}^T) \nabla u \end{aligned} \quad (6.31)$$

For variables u defined only on a lower-dimensional boundary, the tangential derivatives define the components of flux tangential to the surface itself, analogous to Eq. 6.31.

6.2.4 Example: Axonal Stimulation Using Nerve Cuff Electrodes

Consider a 10 mm length nerve bundle of radius 1 mm containing an embedded edge element along its central axis representing a single axon fibre as shown in Fig. 6.15. The two small surface patches represent external cuff electrodes to stimulate the axon, and are squares of sidelength 1 mm 'wrapped around' the bundle, with a centre-centre spacing of 7 mm along the main axis. To stimulate the axon, the active left cuff electrode delivers an outward square-wave current of 1 mA over 1 ms and acts as a cathode, whilst the right electrode is held at ground.



Fig. 6.15 Cylindrical nerve bundle with an internal axonal 1D edge (dashed line) stimulated by external cuff electrodes (shaded grey)

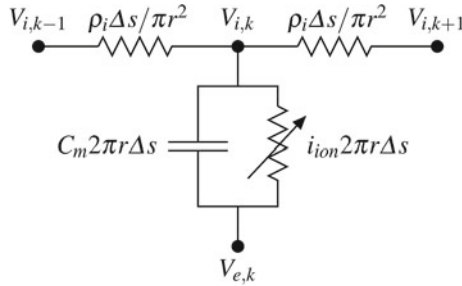


Fig. 6.16 Axon cable discretization showing three adjacent nodes for intracellular potential V_i : $V_{i,k-1}$, $V_{i,k}$, and $V_{i,k+1}$. The extracellular potential at node k is $V_{e,k}$, and the internodal separation is given by Δs , r denotes the radius of the axon, and ρ_i is the axoplasmic resistivity. Membrane capacitance C_m and ionic current i_{ion} are given per unit membrane area

To derive the underlying PDE governing the intra- and extracellular potentials on the 1D neural edge, we can use a process similar to Example 4.15, whereby we discretize the axon into segments of length Δs , where s is the arclength along the axon. The electrical equivalent circuit of three adjacent intracellular nodes is shown in Fig. 6.16.

The axoplasmic resistance between two nodes may be determined from Pouillet’s law, $\rho L/A$ which evaluates to $\rho_i \Delta s / \pi r^2$, as shown in Fig. 4.16 between the nodes. Furthermore, the total membrane capacitance of a cylindrical nerve segment of length Δs is given by C_m multiplied by the area of its curved surface, or $C_m 2\pi r \Delta s$. A similar argument follows for the total membrane ionic current of the segment, namely $i_{ion} 2\pi r \Delta s$. These values are also shown in Fig. 4.16. From Kirchhoff’s current law, the total current entering node k must equal the current leaving through the parallel membrane capacitance and ionic pathways. This can be expressed by the equation

$$\overbrace{\frac{V_{i,k+1} - V_{i,k}}{\rho_i \Delta s / \pi r^2} + \frac{V_{i,k-1} - V_{i,k}}{\rho_i \Delta s / \pi r^2}}^{\text{current entering}} = \overbrace{C_m 2\pi r \Delta s \frac{d(V_{i,k} - V_{e,k})}{dt} + i_{ion} 2\pi r \Delta s}_{\text{current leaving}}$$

On re-arranging, this becomes:

$$\frac{V_{i,k+1} - 2V_{i,k} + V_{i,k-1}}{\rho_i \Delta s^2} = \left(\frac{2}{r}\right) \left[C_m \frac{dV_{m,k}}{dt} + i_{ion} \right] \quad (6.32)$$

where $V_{m,k}$ is the membrane potential at node k , equal to $V_{i,k} - V_{e,k}$. The left-hand side of this equation is simply the FD approximation of the second-order derivative of V_i (see Eq. 4.35). Hence, in the limit as $\Delta s \rightarrow 0$, Eq. 6.32 becomes

$$\frac{1}{\rho_i} \frac{\partial^2 V_i}{\partial s^2} = \left(\frac{2}{r}\right) \left[C_m \frac{\partial V_m}{\partial t} + i_{ion} \right] \quad (6.33)$$

where C_m , ρ_i and r are assumed to be fixed along the axon. Using $V_i = V_m + V_e$, Eq. 6.33 may also be expressed as:

$$\frac{1}{\rho_i} \frac{\partial^2 V_m}{\partial s^2} + \frac{1}{\rho_i} \frac{\partial^2 V_e}{\partial s^2} = \left(\frac{2}{r}\right) \left[C_m \frac{\partial V_m}{\partial t} + i_{ion} \right]$$

Re-arranging all terms of this equation, we can express it in terms of the COMSOL PDE general form Eq. 6.30 in 1D as

$$\left(\frac{2C_m}{r}\right) \frac{\partial V_m}{\partial t} + \frac{\partial}{\partial s} \left[-\frac{1}{\rho_i} \frac{\partial V_m}{\partial s} - \frac{1}{\rho_i} \frac{\partial V_e}{\partial s} \right] = -\left(\frac{2}{r}\right) i_{ion} \quad (6.34)$$

where the flux Γ , source term f and damping coefficient d_a are given by:

$$\begin{aligned} \Gamma &= -\frac{1}{\rho_i} \frac{\partial V_m}{\partial s} - \frac{1}{\rho_i} \frac{\partial V_e}{\partial s} \\ f &= -\left(\frac{2}{r}\right) i_{ion} \\ d_a &= \frac{2C_m}{r} \end{aligned}$$

To express the flux, we make use COMSOL's tangential derivative variables:

$$\Gamma = \begin{pmatrix} -V_m T_x / \rho_{ho_i} - V_e T_x / \rho_{ho_i} \\ -V_m T_y / \rho_{ho_i} - V_e T_y / \rho_{ho_i} \\ -V_m T_z / \rho_{ho_i} - V_e T_z / \rho_{ho_i} \end{pmatrix}$$

where V_m , V_e , ρ_{ho_i} are the COMSOL variables/parameters corresponding to V_m , V_e and ρ_i respectively. For the ionic current i_{ion} in Eq. 6.34, we can use the Hodgkin-Huxley formulation of axonal electrical activity (see Sect. 2.3.2):

$$i_{ion} = \bar{g}_{Na} m^3 h (V_m - V_{Na}) + \bar{g}_K n^4 (V_m - V_K) + \bar{g}_L (V_m - V_L)$$

with

$$\begin{aligned}\frac{dn}{dt} &= \alpha_n (1 - n) - \beta_n n \\ \frac{dm}{dt} &= \alpha_m (1 - m) - \beta_m m \\ \frac{dh}{dt} &= \alpha_h (1 - h) - \beta_h h\end{aligned}$$

$$\begin{aligned}\alpha_n &= \frac{10(V_m + 50)}{1 - \exp\left[\frac{-(V_m + 50)}{10}\right]} & \beta_n &= 125 \exp\left[\frac{-(V_m + 60)}{80}\right] \\ \alpha_m &= \frac{100(V_m + 35)}{1 - \exp\left[\frac{-(V_m + 35)}{10}\right]} & \beta_m &= 4000 \exp\left[\frac{-(V_m + 60)}{18}\right] \\ \alpha_h &= 70 \exp\left[\frac{-(V_m + 60)}{20}\right] & \beta_h &= \frac{1000}{1 + \exp\left[\frac{-(V_m + 30)}{10}\right]}\end{aligned}$$

where V_m in these equations is in units of mV. All Hodgkin-Huxley model parameter values are as given earlier in Table 2.2.

To determine the extracellular potential V_e in Eq. 6.30, we utilise COMSOL's current conservation physics mode for determining electric potential in a volume conductor. To account for the influence of nerve axon excitation on the extracellular potential, we utilise COMSOL's *line current source* to specify outward nerve membrane current per unit length along the axon as


$$Q = 2\pi r \left(C_m \frac{\partial V_m}{\partial t} + i_{ion} \right) \quad (6.35)$$

To implement this model in COMSOL, we implement the following steps:

Model Wizard

1. Select the 3D dimension. Under Select Physics, choose Mathematics|PDE Interfaces|Lower Dimensions|General Form Edge PDE. Click Add to add the physics application. Specify 1 dependent variable with name V_m , units of Electric potential (V), and source term quantity as Current source in units of A/m³.
2. Select the General Form Edge PDE again, and click Add a second time to add another Edge PDE physics node. This time, specify 3 dependent variables with names N, M and H. Note that COMSOL variables are case-sensitive, and that lower-case h is a reserved variable denoting mesh size. Leave the units of these three variables as dimensionless, and specify the source term units as 1/s.
3. Finally, select AC/DC|Electric Currents, and click Add to add the physics. Specify the name of the dependent variable as V_e .
4. For the study type, select 'Time Dependent', then click 'Done'.

Geometry

1. Right click the Geometry 1 node of the model tree and specify a cylinder with radius = 1 mm, height = 10 mm, position (0, 0, 0) and axis type to x-axis in order to make the cylinder align along the x-axis. Click 'Build Selected'.
2. Right click the Geometry 1 node again to define a work plane as a quick yz-plane located at $x = 0$. Select the Plane Geometry sub-node of this work plane and click the Zoom to Extents tool button at the top of the graphics window. Right click the Plane Geometry work plane sub-node node and specify a point at (0, 0). Click 'Build Selected'.
3. Right click the Geometry 1 node a third time and select Extrude. Specify the input object as the point in the work plane array, with the distance to extrude as 10 mm. Click 'Build Selected': this defines a line element in 3D. Click the transparency tool button () to see the edge element within the nerve.
4. Right click the Geometry 1 node again to define a second work plane as a quick-zx plane located at $y = 0$. Select the Plane Geometry sub-node of this work plane and click the Zoom to Extents tool button.
5. Right click the Plane Geometry sub-node and select Polygon. Specify its type as Open curve with x coordinates: -1 mm, -1 mm and y coordinates: 1 mm, 2 mm. Click 'Build Selected'.
6. Right click the Plane Geometry sub-node node again and reselect Polygon. Again specify its type as Open curve with x coordinates: -1 mm, -1 mm and y coordinates: 8 mm, 9 mm. Click 'Build Selected'.
7. Return to the Geometry 1 node and right click to select Revolve. Specify the input objects as 'wp2' (i.e. Work Plane 2), start angle: (π [rad] +1 [rad]) and end angle: π [rad]. Leave the points on the revolution axis and the revolution axis direction to their default values and click 'Build Selected'. This defines the two cuff electrodes. The resulting COMSOL model tree and geometry are shown in Fig. 6.17.

Global Definitions

1. Right click Global Definitions and define the following Parameters:

| Name | Expression |
|---------|------------------------------|
| g_Na | 120000 [uS/cm ²] |
| g_K | 36000 [uS/cm ²] |
| g_L | 300 [uS/cm ²] |
| V_Na | 55 [mV] |
| V_K | -72 [mV] |
| V_L | -49.387 [mV] |
| Cm | 1 [uF/cm ²] |
| rho_i | 60 [ohm*cm] |
| sigma_e | 1 [S/m] |
| r | 0.00025 [cm] |
| I_stim | 1 [mA] |

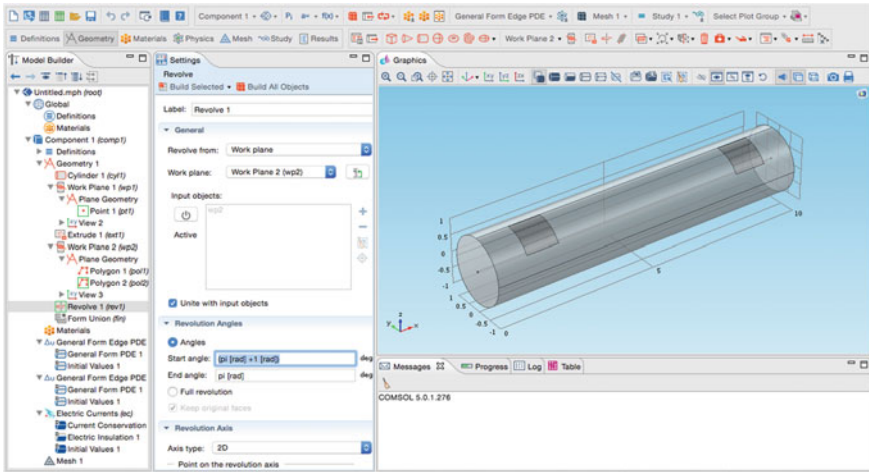


Fig. 6.17 COMSOL interface for nerve bundle cuff electrode example, showing geometry and model tree. The 1D edge corresponding to the axon can be seen in transparent view mode along the central axis of the nerve bundle *cylinder*

2. Right click Global Definitions again and select Functions|Rectangle. Specify the upper and lower limits as 0.001 and 0.002 respectively. In the Smoothing tab, specify the size of the transition zone as $1e - 5$. Leave the name rect1 as the default function name. This defines the stimulus current waveform.

Definitions

1. Right click the Definitions sub-node of Component 1 to specify Variables, and select Edge as the geometric entity level. Specify edge 6 and define variables as shown below. Note that ‘ . . . ’ is used for convenience whenever relevant expressions continue to the next line - it would be omitted when typing in COMSOL:

| Name | Expression |
|------------|--|
| i_{Na} | $g_{Na} * M^3 * H * (V_m - V_{Na})$ |
| i_K | $g_K * N^4 * (V_m - V_K)$ |
| i_L | $g_L * (V_m - V_L)$ |
| i_{ion} | $i_{Na} + i_K + i_L$ |
| α_N | $(10 [1 / (mV * s)]) * (V_m + (50 [mV])) / \dots$ $(1 - \exp(-(V_m + (50 [mV])) / (10 [mV])))$ |
| β_N | $(125 [1 / s]) * \exp(-(V_m + (60 [mV])) / (80 [mV]))$ |
| α_M | $(100 [1 / (mV * s)]) * (V_m + (35 [mV])) / \dots$ $(1 - \exp(-(V_m + (35 [mV])) / (10 [mV])))$ |
| β_M | $(4000 [1 / s]) * \exp(-(V_m + (60 [mV])) / (18 [mV]))$ |
| α_H | $(70 [1 / s]) * \exp(-(V_m + (60 [mV])) / (20 [mV]))$ |
| β_H | $(1000 [1 / s]) / \dots$ $(1 + \exp(-(V_m + (30 [mV])) / (10 [mV])))$ |

General Form Edge PDE

1. Select the General Form Edge PDE node and remove all edges except edge 6 which corresponds to the axon fibre.
2. Select the General Form PDE 1 sub-node and enter the following into the flux (Γ) fields:

$$-V_m T_x / \rho_{oi} - V_e T_x / \rho_{oi}$$

$$-V_m T_y / \rho_{oi} - V_e T_y / \rho_{oi}$$

$$-V_m T_z / \rho_{oi} - V_e T_z / \rho_{oi}$$

Specify the source term (f) as $-(2/r) * i_{ion}$, and the damping coefficient (da) as $(2 * C_m / r)$.

3. Under the Initial Values node, specify -60 [mV].
4. Right click the General Form PDE Edge node and select 'Dirichlet boundary condition'. Add points 3 and 16 to the selection corresponding to the end points of the axon edge. Specify a prescribed value for V_m of -60 [mV].

General Form Edge PDE 2

1. Select the General Form Edge PDE 2 node and remove all edges except edge 6 which corresponds to the axon fibre.
2. Select the General Form PDE 1 sub-node and enter 0 into all of the flux (Γ) fields, since there are no spatial derivatives in the PDEs for variables N, M, H. Enter the following into the source term components (f):

$$N: \alpha_N * (1-N) - \beta_N * N$$

$$M: \alpha_M * (1-M) - \beta_M * M$$

$$H: \alpha_H * (1-H) - \beta_H * H$$

3. Select the Initial Values node and specify initial values for the gating variables as N: 0.3177, M: 0.0529, H: 0.5961

Electric Currents

1. Select the Current Conservation 1 node. Set the conductivity σ to the user-defined value of σ_{e} , and the relative permittivity to the user-defined value of 1.
2. Right click the Electric Currents node and add a Ground boundary condition for boundary 7, which corresponds to the right cuff electrode.
3. Right click the Electric Currents node again and add a Normal Current Density boundary condition. Specify boundary 6 (i.e. the active left cuff electrode) and set the inward current density J_n to $-I_{stim} * \text{rect1}(t [1/s]) / (1 [\text{mm}^2])$. Note that the $(1 [\text{mm}^2])$ term denotes the area of the 1×1 mm cuff electrode, and the negative sign up front specifies cathodic stimulation.

- Right click the Electric Currents node again and select Edges|Line Current Source. Specify edge 6 and enter the expression for Q_j corresponding to Eq. 6.35 as: $2*\pi*r*(Cm*Vmt+i_ion)$.


Mesh

- Right click the Mesh node and select 'Distribution'. Specify the geometric entity level as Edge, and select edge 6. Specify the number of elements as 100 to specify this amount of edge elements along the axon.
- Right click the Mesh node again and add a Free Tetrahedral sub-node to mesh the remaining geometry.

Study 1

- Under the Step 1: Time-Dependent node, specify the time range from 0 to 0.01 s in steps of 0.0001 s.
- Right click the Study 1 node and select 'Show default solver'. Under Solver Configurations|Solver 1|Time-Dependent Solver 1, select the Time Stepping tab. Specify the time steps taken by solver to be 'Strict'.
- Right click the Time-Dependent Solver 1 sub-node, and select 'Fully Coupled'. Right click the 'Direct' solver greyed-out sub-node and select 'enabled'.
- Right click the Study 1 node again and select Compute.

Results

- When the model has completed solving,⁹ the Graphics window will display a default plot of the membrane potential (variable V_m) along the axon at $t = 0.01$ s.
- To plot the membrane potential along the axon at various times, right-click Results and select '1D Plot Group'. Right-click the 1D Plot Group 4 sub-node and select 'Line Graph'. In the Settings window, select edge 6 corresponding to the axon.
- Select the 1D Plot Group 4 sub-node and in the settings window, select 'From list' for the Time selection, and choose times of 0.001, 0.003, 0.005, 0.007 and 0.009 s. Under the Plot Settings Tab, check the x-axis label checkbox and enter 'Axon position (mm)'.
- Click the Line Graph 1 sub-node. Leave V_m as the default expression to plot, but specify the units as 'mV'. Under the Legends tab, click the 'Show legends' check box.
- Finally, click the Plot button () to display the line plots, as shown in Fig. 6.18.

⁹Using a MacBook Air 2013 with 8GB RAM and OS X 10.8.5, it took 4 min, 23 s to solve this model.

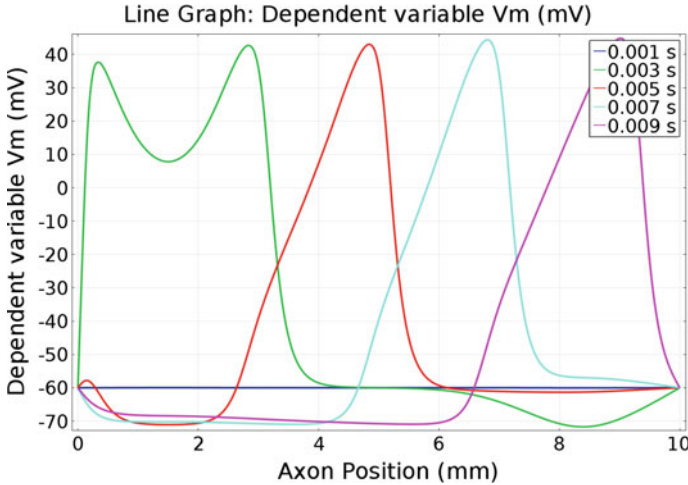


Fig. 6.18 COMSOL-generated membrane potential plots for nerve bundle cuff electrode example along the axon at times of 0.001, 0.003, 0.005, 0.007 and 0.009s. The action potential is seen travelling along the axon from *left to right*

6.3 Further Reading

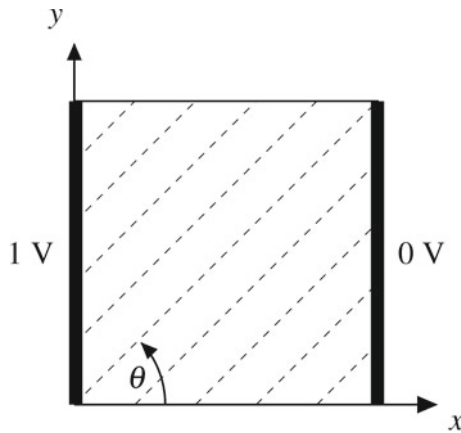
An good overview of the theory of electromagnetism and Maxwell's equations may be found in the very short and readable text of Fleisch [3]. Electromagnetism applied to physiology and biology is covered in the texts of Malmivuo and Plonsey [5], Plonsey and Barr [6], and Barnes and Greenebaum [2]. More comprehensive treatments of the electrophysiology of excitable cells and tissues, including nerve and muscle, is provided in the texts of Hille [4] and Aidley [1]. Further examples of electromagnetics modelling using COMSOL, albeit in a non-biological context, may be found in the text of Pryor [7].

Problems

6.1 Two platinum spherical stimulating electrodes of radius 0.5 mm are placed 2 mm apart centre-centre in an infinite saline solution of conductivity 1 S m^{-1} . Assuming the surface of each electrode lies at an isopotential state, use COMSOL to determine the resistance between the electrodes.

6.2 Consider a 2D square slab of excised cardiac tissue of sidelength 1 cm, as shown below. Electrical stimulating electrodes are placed on its left and right boundaries such that the left boundary is held at a potential of 1 V whilst the right is at ground. All other boundaries are assumed to be electrically insulated. The tissue consists of parallel muscle fibres oriented at an angle θ relative to the x-axis, with longitu-

dinal conductivity (i.e. along the fibre) of 0.2 mS cm^{-1} and transverse conductivity 0.1 mS cm^{-1} . Plot the potential distribution and current streamlines for fibre angles $\theta = 0^\circ$, $\theta = 45^\circ$ and $\theta = 90^\circ$.



6.3 A platinum disc electrode of radius 1 mm is injecting current into a hemispherical infinite saline domain of conductivity 1 S m^{-1} , similar to the geometry shown in Fig. 6.5. At the junction between the saline and the platinum, there is a liquid-metal interface, represented by a distributed resistance of $0.001 \Omega \text{ m}^2$. If the platinum electrode is at a steady-state equipotential level of 1 V, the local potential in the saline adjacent to the electrode will be less, owing to the voltage drop across the distributed resistance. Using COMSOL, plot the inward current density as a function of radial position along the disc, comparing against the theoretical solution with no distributed resistance.

References

1. Aidley DJ (1978) *The physiology of excitable cells*, 2nd edn. Cambridge University Press, Cambridge
2. Barnes FS, Greenebaum B (eds) (2007) *Handbook of biological effects of electromagnetic fields: bioengineering and biophysical aspects of electromagnetic fields*, 3rd edn. CRC Press, Boca Raton
3. Fleisch D (2008) *A student's guide to Maxwell's equations*. Cambridge University Press, Cambridge
4. Hille B (2001) *Ion channels of excitable membranes*, 3rd edn. Sinauer, Sunderland
5. Malmivuo J, Plonsey R (1995) *Bioelectromagnetism: principles and applications of bioelectric and biomagnetic fields*. Oxford University Press, Oxford
6. Plonsey R, Barr RC (2007) *Bioelectricity: a quantitative approach*. Springer, New York
7. Pryor RW (2011) *Multiphysics modeling using COMSOL: a first principles approach*. Jones and Bartlett, Sudbury
8. Rogers JM, McCulloch AD (1994) A collocation-Galerkin finite element model of cardiac action potential propagation. *IEEE Trans Biomed Eng* 41:743–757

Chapter 7

Models of Diffusion and Heat Transfer

7.1 Diffusion

Diffusion refers to the net transport of molecules from regions of high concentration to low concentration as a result of random molecular motion. Diffusion processes play an important role in a range of biological systems and bioengineering applications. Typical examples include:

- exchange of metabolites between a cell and its environment
- diffusion of water through a semi-permeable membrane (osmosis)
- exchange of oxygen and carbon dioxide in the alveoli of mammalian lungs
- drug delivery medical devices

7.1.1 Fick's Laws of Diffusion

In 1855, Adolf Fick¹ formulated two important laws governing diffusion processes based on his experimental observations using salt solutions. The first law states that the amount of substance transported by diffusion per unit time is proportional to the gradient in substance concentration. The rate of substance transported per unit area is also known as the *material flux*, and is a vector quantity having both magnitude and direction. Fick's first law can be stated mathematically in terms of the material flux as

$$\mathbf{\Gamma} = -D\nabla c \tag{7.1}$$

where $\mathbf{\Gamma}$ is the material flux in SI units of $\text{mol m}^{-2} \text{s}^{-1}$, c is the concentration with SI units of mol m^{-3} , and D is the diffusion coefficient of the substance in a given medium in SI units of $\text{m}^2 \text{s}^{-1}$. Note that the negative sign in Eq. 7.1 indicates that diffusion occurs down the concentration gradient, from regions of high to low concentration.

¹German physiologist and physician, 1829–1901.

Fick's second law describes the transient changes of concentration with time, and is written mathematically as a PDE in the concentration variable c . It can be derived from the divergence theorem (Eq. 4.5) for any vector field \mathbf{F} :

$$\int_V (\nabla \cdot \mathbf{F}) \, dV = \int_S \mathbf{F} \cdot d\mathbf{S}$$

where V denotes a closed region with boundary S . Substituting the material flux for \mathbf{F} , we obtain:

$$\begin{aligned} \int_V (\nabla \cdot \mathbf{\Gamma}) \, dV &= \int_S \mathbf{\Gamma} \cdot d\mathbf{S} = -\frac{\partial}{\partial t} \int_V c \, dV \\ \int_V \nabla \cdot (-D\nabla c) \, dV &= -\int_V \frac{\partial c}{\partial t} \, dV \end{aligned}$$

Swapping the left and right-hand sides, and cancelling the negative sign from both, we obtain:

$$\int_V \frac{\partial c}{\partial t} \, dV = \int_V \nabla \cdot (D\nabla c) \, dV$$

Since this holds for any arbitrary volume V , it follows that the integrands must be identically equivalent. Namely,

$$\frac{\partial c}{\partial t} = \nabla \cdot (D\nabla c) \tag{7.2}$$

Typical values for the diffusion coefficient D are $0.6 \times 10^{-9} - 2 \times 10^{-9} \text{ m}^2 \text{ s}^{-1}$ for individual ions and $10^{-11} - 10^{-10} \text{ m}^2 \text{ s}^{-1}$ for biological molecules. D itself is proportional to the velocity squared of the diffusing molecules and inversely proportional to their size. Equation 7.2 is automatically implemented in COMSOL using the Chemical Species Transport|Transport of Diluted Species physics interface,² but can also be readily implemented using the Mathematics PDE interfaces in the base COMSOL package.

7.1.2 Example: Diffusion and Uptake into a Spherical Cell

A spherical cell of radius $50 \mu\text{m}$ is taking in a nutrient from the interstitial space and metabolising it. The rate of uptake through the cell surface is $0.0005c \text{ mol s}^{-1} \text{ m}^{-2}$, where c is the nutrient concentration in mM. If the initial concentration in the interstitial space is 100 mM , find the concentration of the nutrient $0.5 \mu\text{m}$ from the cell surface as a function of time. The diffusion constant of the nutrient in the interstitial space is $5 \times 10^{-11} \text{ m}^2 \text{ s}^{-1}$.

²This interface is part of the optional Chemical Reaction Engineering Module.

To solve this problem in COMSOL, we utilize a 2D axisymmetric geometry as implemented in the following steps:

Model Wizard

1. Open the Model Wizard and select the 2D axisymmetric spatial dimension.
2. In the Select Physics panel, choose Chemical Species Transport| Transport of Diluted Species. Leave c as the default dependent variable.
3. Click the Study arrow (👉) to open the Select Study panel. Select the Time Dependent, and click “Done”. This will exit the Model Wizard, displaying the main COMSOL interface.

Geometry

1. Select the Geometry 1 node in the model tree. In the settings window, change the default length unit to ‘ μm ’.
2. Right-click Geometry 1 and select Circle. Specify the radius as $50\ \mu\text{m}$ and a sector angle of 90° . Click Build Selected (🏗️).
3. Right-click Geometry 1 again and select Circle a second time. Specify the radius as $70\ \mu\text{m}$ and the sector angle as 90° . Click Build Selected (🏗️), followed by the Zoom Extents button (🔍).
4. Right-click Geometry 1 again and select Booleans and Partitions| Difference. Specify circle c2 as the object to add, and circle c1 as the object to subtract. Click Build Selected (🏗️). The resulting COMSOL interface with geometry and model tree should look like that shown in Fig. 7.1.

Transport of Diluted Species

1. Select the Transport Properties 1 node of the model tree and set the isotropic diffusion coefficient D_c to the user-defined value of $5\text{e-}11\ \text{m}^2/\text{s}$. On the Initial Values 1 sub-node, specify the initial value of c as $100\ \text{mol}/\text{m}^3$.
2. Right click to add a ‘Flux’ boundary condition. Select boundary 3 and check the Species c checkbox. Specify the inward flux as $-(0.0005\ [\text{m/s}])*c$.

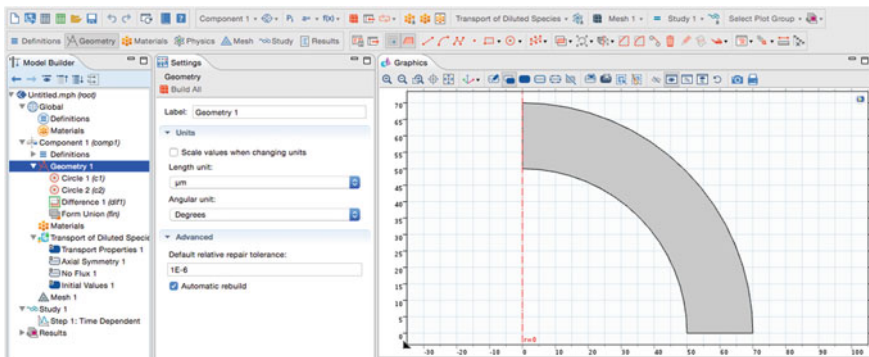


Fig. 7.1 COMSOL interface for axisymmetric spherical cell diffusion example, showing 2D axisymmetric geometry (right) and model tree (far left)

Fig. 7.2 Zoomed-in view of COMSOL-generated mesh for axisymmetric spherical cell diffusion model, showing boundary mesh layers at bottom adjacent to the cell boundary

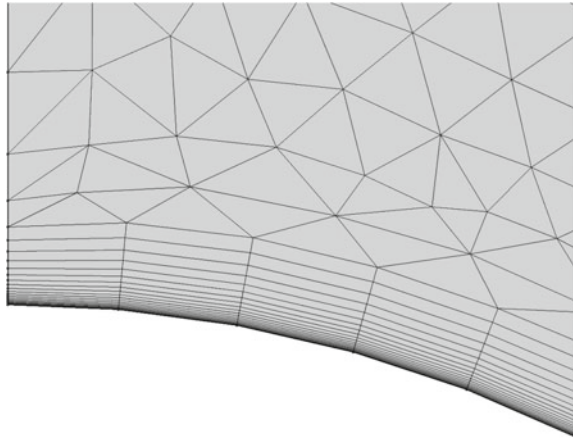
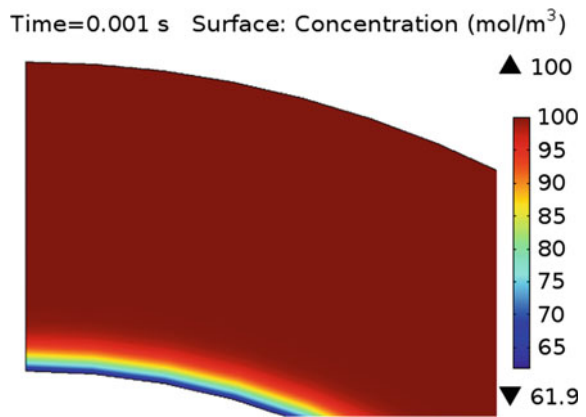


Fig. 7.3 Zoomed-in view of COMSOL concentration solution for axisymmetric spherical cell diffusion model at 0.001 s



Mesh

1. Right-click the Mesh 1 node and add a 'Boundary Layers' node. Under the Boundary Layer Properties sub-node, select boundary 3, and specify the number of boundary layers to be 20. Click the 'Build All' button (☒) to build the mesh. A zoomed-in view of the resulting mesh is shown in Fig. 7.2.

Study

1. Under the Study 1 node, select the Time Dependent solver. Specify the time range as 0 to 0.001 s with a step size of 1e-5 s.
2. Right-click Study 1 and select Show Default Solver. Under Study 1 | Solver Configurations | Solver 1 | Time Dependent Solver 1, select 'Strict' for the Time steps taken by solver (under the Time Stepping tab).
3. To solve the model, right-click Study 1 and select Compute (=). COMSOL will display the default plot of concentration at 0.001 s, similar to that shown in Fig. 7.3.

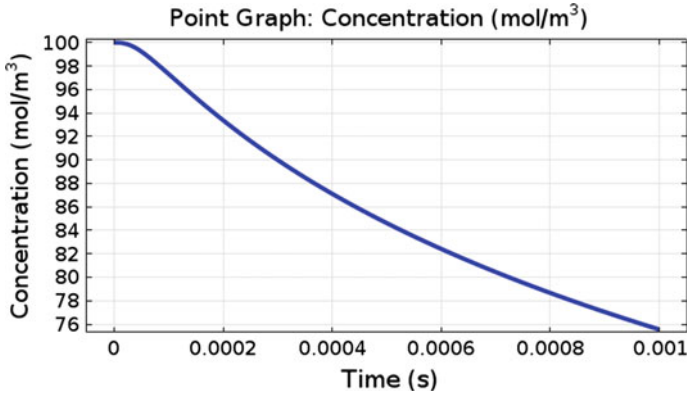


Fig. 7.4 Concentration as a function of time at a point located $0.5\mu\text{m}$ from the spherical cell boundary

Results

1. Right-click the Data Sets sub-node of Results in the model tree, and select Cut Point 2D. In the Settings window, specify the coordinates of the cut point as $r = 0$, $z = 50.5\mu\text{m}$.
2. Right-click the Result node and select '1D Plot Group'. Right-click the newly-created 1D Plot Group 3 sub-node and select 'Point Graph'. In the settings panel, specify the Data set as 'Cut Point 2D 1'. Click the Plot button (🖨️) to display the plot of concentration c against time at this point, as shown in Fig. 7.4.

7.1.3 Convective Transport

When a diffusing substance (solute) is present in a moving liquid, there will be an additional component of flux due to the velocity of the fluid itself. Transport due to fluid movement is referred to as *convection*. If a fluid is moving with velocity magnitude u perpendicularly through an infinitesimal area ΔA , then the volume of fluid passing through this area over infinitesimal time Δt is the fluid displacement ($u\Delta t$) multiplied by ΔA , or

$$\Delta V = (u\Delta t) \Delta A$$

For a substance of concentration c , this represents a total amount of substance $c\Delta V$ passing through the area in the direction of fluid velocity \mathbf{u} . Since flux represents the amount of substance flowing per unit area per unit time, the convective flux, expressed as a vector quantity, is given by

$$\mathbf{\Gamma} = \left(\frac{c\Delta V}{\Delta t \Delta A} \right) \left(\frac{\mathbf{u}}{u} \right) = c\mathbf{u} \quad (7.3)$$

Adding this convective flux component to the diffusional flux (Eq. 7.1), we obtain the total flux in the presence of diffusion and convection as

$$\mathbf{\Gamma} = c\mathbf{u} - D\nabla c \quad (7.4)$$

For a fixed closed region V in the fluid with boundary S , we can employ the divergence theorem (Eq. 4.5) to obtain

$$\begin{aligned} \int_V (\nabla \cdot \mathbf{\Gamma}) dV &= \int_S \mathbf{\Gamma} \cdot d\mathbf{S} = -\frac{\partial}{\partial t} \int_V c dV \\ \int_V \nabla \cdot (c\mathbf{u} - D\nabla c) dV &= -\int_V \frac{\partial c}{\partial t} dV \end{aligned} \quad (7.5)$$

For an incompressible fluid, we can simplify this expression by noting that

$$\nabla \cdot (c\mathbf{u}) = \mathbf{u} \cdot \nabla c + c\nabla \cdot \mathbf{u} = \mathbf{u} \cdot \nabla c$$

since $\nabla \cdot \mathbf{u} = 0$ for incompressibility (see Eq. 4.11). Substituting this into Eq. 7.5, we obtain

$$\begin{aligned} \int_V [\mathbf{u} \cdot \nabla c - \nabla \cdot (D\nabla c)] dV &= -\int_V \frac{\partial c}{\partial t} dV \\ \int_V \left[\frac{\partial c}{\partial t} + \mathbf{u} \cdot \nabla c - \nabla \cdot (D\nabla c) \right] dV &= 0 \end{aligned}$$

and since this expression must hold for any arbitrary fixed volume V , it follows that the integrand must be identically equal to zero, or

$$\frac{\partial c}{\partial t} + \mathbf{u} \cdot \nabla c = \nabla \cdot (D\nabla c) \quad (7.6)$$

7.1.4 Example: Drug Delivery in a Coronary Stent

A drug-eluting stent is a wire scaffold placed into an occluded artery (typically a coronary artery) in order to re-open the vessel for blood flow (Fig. 7.5). Such stents slowly release drugs, typically paclitaxel or sirolimus and its analogues including everolimus [9], which prevent scar tissue formation and keep the artery open. We wish to implement a simplified model of such a system to determine the blood drug distribution as a function of time along the artery. Assuming that blood is travelling parallel to the axis everywhere, and that its velocity is zero along the walls (no-

Fig. 7.5 Occluded coronary artery segment with plaque deposits shown in grey (*top*). Insertion of a coronary stent opens the artery (*bottom*), allowing blood to flow freely again. Drug-eluting stents slowly release pharmaceutical agents that inhibit scar tissue formation and prevent re-occlusion

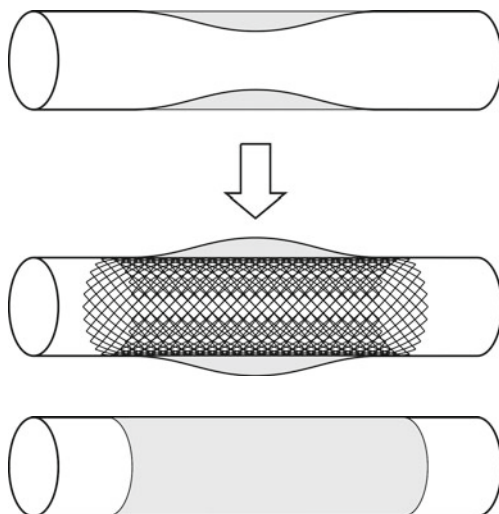


Fig. 7.6 Simplified stent model geometry, with uniform stent surface shown in grey

slip condition) and maximum along the central axis, its motion can be reasonably approximated by a parabolic velocity profile according to

$$u = \frac{u_{\max}}{R^2} (R^2 - r^2) \quad (7.7)$$

where u is the velocity in the axial direction, u_{\max} is the maximum velocity at the axis ($= 50 \text{ cm}^{-1}$), R is the arterial radius ($= 1 \text{ mm}$), and r is the radial distance from the central axis. We take the length of the arterial segment as 9 mm and the length of the stent as 6 mm. Although the stent wire mesh is a complex structure, we can simplify it as a uniform, continuous cylindrical surface along the boundary of the artery (Fig. 7.6). We utilize the reported free diffusion coefficient of paclitaxel as $2.56 \times 10^{-4} \text{ cm}^2 \text{ min}^{-1}$ [4]. Furthermore, we assume the initial total content of paclitaxel in the stent as 42 μg . Since the molecular weight of paclitaxel is 853.9 g mol^{-1} , this corresponds to an initial drug content of 49.2 nmol.³ In rabbit iliac artery experiments, the drug content of the stent has been reported to decrease from 42 to 12.4 μg 6 hours post-implantation [6]. Assuming mono-exponential decay of stent drug-content


$$M = M_0 \exp(-kt)$$

where M_0 is the initial content, this data corresponds to a drug release rate k of 0.2 hr^{-1} .

³<http://pubchem.ncbi.nlm.nih.gov/compound/paclitaxel>.

To implement this model in COMSOL, we use the following steps:

Model Wizard

1. Open the Model Wizard and select the 2D axisymmetric spatial dimension.
2. In the Select Physics panel, choose Chemical Species Transport| Transport of Diluted Species. Click ‘Add’, and leave c as the default dependent variable.
3. Click the Study arrow () to open the Select Study panel. Select Time Dependent, and click “Done”. This will exit the Model Wizard, displaying the main COMSOL interface.



Global Definitions

1. Right-click Global Definitions in the model tree and select Parameters. Enter the following details in the Parameters table of the Settings window:

| Name | Expression | Description |
|---------------------|----------------------------------|----------------------------|
| u_{\max} | 50 [cm/s] | Maximum blood velocity |
| D_{drug} | $2.56e-4$ [cm ² /min] | Drug diffusion coefficient |
| k | 0.2 [1/hour] | Drug release rate |
| R | 1 [mm] | Artery radius |
| L_{artery} | 9 [mm] | Arterial segment length |
| L_{stent} | 6 [mm] | Stent length |
| M_0 | 49.2 [nmol] | Initial stent drug content |

2. Right-click Global Definitions again and select Variables. Specify a new variable M with expression $M_0 \cdot \exp(-k \cdot t)$. This represents the total drug content of the stent as function of time.

Geometry

1. Select the Geometry 1 node in the model tree. In the settings window, change the default length unit to ‘mm’.
2. Right-click Geometry 1 and select Rectangle. Specify the width and height as R and L_{artery} respectively. Click Build Selected ()
3. Right-click Geometry 1 again and select Point. In the Settings panel, enter R and R in the r field (these values should be separated by a space), followed by $(L_{\text{artery}} - L_{\text{stent}}) / 2$ and $(L_{\text{artery}} + L_{\text{stent}}) / 2$ in the z field. This specifies two points enclosing the stent boundary. Click the Build Selected () button. The resulting COMSOL interface with geometry and model tree should look like that shown in Fig. 7.7.

Transport of Diluted Species

1. Expand the Transport of Diluted Species node of the model tree and select the Transport Properties 1 sub-node. In the Settings panel, set the isotropic diffusion coefficient D_c to the user-defined value of D_{drug} . For the velocity field, specify 0 for the r component and $(u_{\max} / R^2) \cdot (R^2 - r^2)$ for the z component.

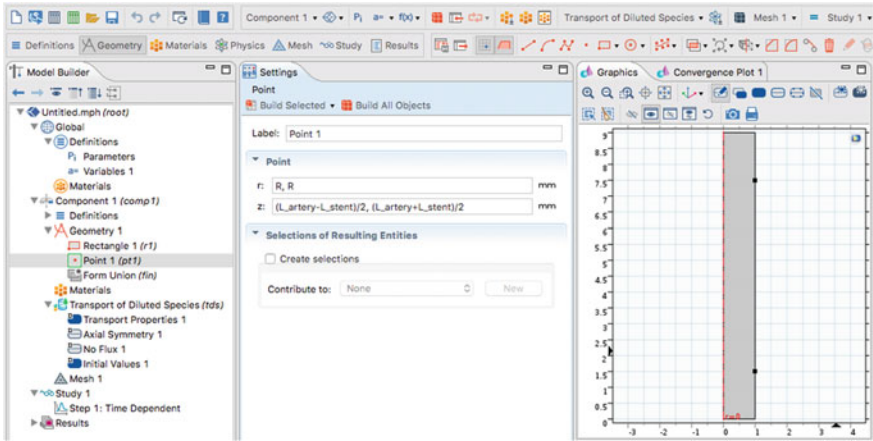


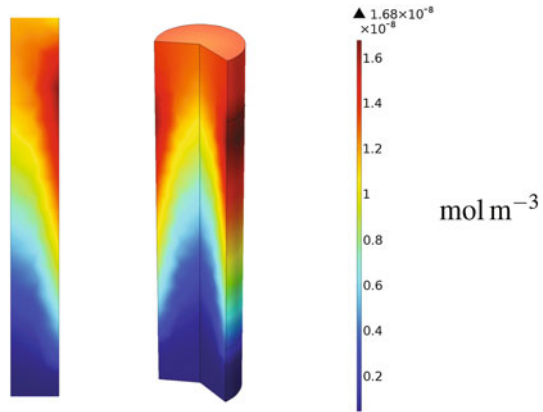
Fig. 7.7 COMSOL interface for stent diffusion and convection example, showing 2D axisymmetric geometry (right) and model tree (far left). The axis of rotational symmetry on the geometry is shown as a vertical red dashed line corresponding to $r = 0$, where r is the radial coordinate. The vertical coordinate is denoted by z . The two points on the interior of the right boundary correspond to each end of the stent

2. Right click the Transport of Diluted Species node to add an ‘Inflow’ boundary condition. Select boundary 2 (i.e. the lower boundary) and specify the boundary condition type as Flux (Danckwerts) with a default concentration of 0. This boundary condition specifies that there there is no diffusive flux normal to the boundary, only convective inflow. Furthermore, since we have specified a concentration of 0 at this boundary, this is equivalent to a zero-flux boundary condition.
3. Right click the Transport of Diluted Species node again to add an ‘Outflow’ boundary condition. Select boundary 3 (i.e. the upper boundary). This boundary condition specifies that there there is no diffusive flux normal to the boundary, only convective outflow.
4. Right click the Transport of Diluted Species node a third time to add a ‘Flux’ boundary condition. Select boundary 5, and click the species ‘c’ checkbox. Specify an inward flux of $k \cdot M / (2 \cdot \pi \cdot R \cdot L_{\text{stent}})$, where we have divided the total transport rate per unit time (kM) by the surface area of the stent ($2\pi R L_{\text{stent}}$).
5. Click the COMSOL Show button (👁) just above the Model Tree and select ‘Stabilization’. This allows model stabilization settings to be viewed. Click the Transport of Diluted Species node and under the Inconsistent Stabilization tab, select the ‘Isotropic diffusion’ checkbox. This adds a small artificial amount of diffusion proportional to the mesh element size, and is necessary to avoid numerical oscillations when convection flow dominates the diffusional flux.

Study

1. Under the Study 1 node, select the Time Dependent solver. Click the range button (📏) and specify the time range as 0 to 5 in steps of 0.001. Spec-

Fig. 7.8 Drug concentration in arterial segment at 99,999 s (≈ 27.8 hrs). At *left* is shown the concentration in the axisymmetric model plane, whilst at *right* is the 3D concentration profile formed by rotating this plane around the axis. Note that the *top* and *bottom* boundaries represent the outflow and inflow respectively



ify the function to apply to all values as 'exp10'. Click the 'Replace' button. This will result in the expression for the output times of the solver as $10^{\text{range}(0, 0.001, 5)}$. Manually append '-1' to this expression so that it reads $10^{\text{range}(0, 0.001, 5)} - 1$: this will create a geometrically increasing time sequence from 0 to $(10^5 - 1)$ s.

2. Right-click Study 1 and select Show Default Solver. Under Study 1 | Solver Configurations | Solution 1 | Time Dependent Solver 1, select the Time Stepping tab and choose 'Strict' for the Time steps taken by solver. Select the Initial step checkbox and specify an initial step value of $1e-6$ s.
3. To solve the model, right-click Study 1 and select Compute (=). COMSOL will display the default plot of concentration at 99,999 s, similar to that shown in Fig. 7.8. Clicking on the Concentration (tds) 1 volume plot (under the Results node of the model tree) will generate a 3D representation of the axisymmetric concentration profile, as shown also in Fig. 7.8.

Results

1. Right-click the Data Sets sub-node of Results in the model tree, and select Cut Point 2D. In the Settings window, specify the coordinates of the cut point as $r = R/2$, $z = L_{\text{artery}}$. This corresponds to a point half-way across the outflow boundary.
2. Right-click the Result node and select '1D Plot Group'. Right-click the newly-created 1D Plot Group 3 sub-node and select 'Point Graph'. In the settings panel, specify the Data set as 'Cut Point 2D 1'. Click the Plot button (=) as well as the x-axis log scale button (III) to display the plot of concentration c against time at this point, as shown in Fig. 7.9. It can be seen that the concentration of the drug at the outflow rapidly increases to its maximum value of $\approx 3.35 \times 10^{-6} \text{ mol m}^{-3}$, before declining again to 0 after 10^5 s, when the drug content of the stent has depleted.

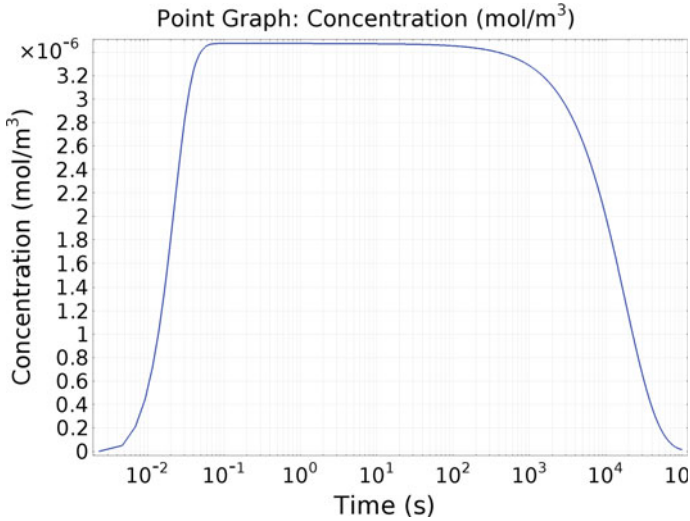


Fig. 7.9 Drug concentration as a function of time at a point located half-way across the outflow boundary

Although this example used a simplified representation of stent structure, we will return to this model again in Chap. 9, Sect. 9.2.1, where we will utilize fluid mechanics to also solve for the actual fluid flow around the struts of the stent, utilizing more fully the multiphysics capabilities of COMSOL.

7.2 Heat Transfer

Heat transfer within a medium takes place by direct transfer via contact of particles vibrating randomly due to their thermal energy. This process is similar to that of diffusion, and therefore the underlying PDEs characterising the two phenomena have many similarities. In a medical device context, the physics of heat transfer is relevant for understanding, among others, heat treatment of tumours, thermal ablation of cardiac tissue for managing arrhythmia, tissue burns, as well as the thermal safety of electronic devices.

7.2.1 Heat Conduction and Convection

The law of thermal conduction was first formulated by Jean-Baptiste Joseph Fourier⁴ as

$$\mathbf{q} = -k\nabla T \quad (7.8)$$

where \mathbf{q} is the *heat flux* in SI units of W m^{-2} , T is the *temperature* in SI units of Kelvin (or K), and k is the thermal conductivity of the medium in SI units of $\text{W K}^{-1} \text{m}^{-1}$. Temperature is a scalar quantity that indicates the energy of molecular vibration, and heat flux is the rate of molecular vibrational energy transport per unit time per unit area. In a given medium, the relationship between the molecular energy of vibration and temperature is given by

$$E = Tmc_p \quad (7.9)$$

where E is the energy, m is the mass of the medium, and c_p is its *specific heat capacity* in units of $\text{J kg}^{-1} \text{K}^{-1}$. Specific heat capacity is a fundamental material property of the medium, independent of its geometric shape or size.

Similar to diffusion, we can formulate the underlying PDE governing heat flow in a medium by combining the conductive heat flux given by Eq. 7.8 with transport by convection i.e. through physical movement of the medium, typically a fluid. If the fluid is moving with velocity magnitude u perpendicularly through an infinitesimal area ΔA , then the volume of fluid ΔV passing through this area over infinitesimal time Δt is the fluid displacement ($u\Delta t$) multiplied by ΔA , or $\Delta V = (u\Delta t)\Delta A$. For a medium of temperature T and mass density ρ , and using Eq. 7.9, this represents a total heat energy $T\rho\Delta Vc_p$ passing through the area in the direction of fluid velocity \mathbf{u} . Since heat flux represents the amount of heat energy flowing per unit area per unit time, the convective heat flux, expressed as a vector quantity, is given by

$$\mathbf{q} = \left(\frac{T(\rho\Delta V)c_p}{\Delta t\Delta A} \right) \left(\frac{\mathbf{u}}{u} \right) = \left(\frac{T\rho u\Delta t\Delta A c_p}{\Delta t\Delta A} \right) \left(\frac{\mathbf{u}}{u} \right) = \mathbf{u}T\rho c_p \quad (7.10)$$

Adding this convective flux component to the conductive heat flux Eq. 7.8, we obtain the total heat flux in the presence of conduction and convection as

$$\mathbf{q} = \mathbf{u}T\rho c_p - k\nabla T \quad (7.11)$$

For a fixed closed region V in the fluid with boundary S , and containing a heat source Q (in W m^{-3}), we can integrate the outward heat flux over S to obtain

⁴French mathematician (1768–1830), best known for his instigation of Fourier series and Fourier transforms which he applied to problems of heat transfer.

$$\begin{aligned}\int_S \mathbf{q} \cdot d\mathbf{S} &= \int_V Q \, dV - \frac{\partial}{\partial t} \int_V \rho c_p T \, dV \\ &= \int_V Q \, dV - \int_V \frac{\partial(\rho c_p T)}{\partial t} \, dV \quad (\text{since } V \text{ is fixed})\end{aligned}\quad (7.12)$$

where the first term on the right-hand side denotes the amount of heat produced by the heat source, and the second term denotes the rate of heat loss. We can then invoke the divergence theorem (Eq. 4.5) to state

$$\int_S \mathbf{q} \cdot d\mathbf{S} = \int_V \nabla \cdot \mathbf{q} \, dV$$

Substituting this identity into Eq. 7.12, we obtain

$$\begin{aligned}\int_V \nabla \cdot \mathbf{q} \, dV &= \int_V Q \, dV - \int_V \frac{\partial(\rho c_p T)}{\partial t} \, dV \\ \int_V \left(\frac{\partial(\rho c_p T)}{\partial t} + \nabla \cdot \mathbf{q} \right) \, dV &= \int_V Q \, dV\end{aligned}$$

Since the above integral is true for any arbitrary volume V , the integrands on both sides must be identically equal. That is:

$$\frac{\partial(\rho c_p T)}{\partial t} + \nabla \cdot \mathbf{q} = Q \quad (7.13)$$

For incompressible fluid flow, ρ and c_p will be constant at each point in the medium. Equation 7.13 is then equivalent to

$$\rho c_p \frac{\partial T}{\partial t} + \nabla \cdot \mathbf{q} = Q$$

Substituting the expression for heat flux (Eq. 7.11) into this equation, we obtain

$$\rho c_p \frac{\partial T}{\partial t} + \nabla \cdot (\mathbf{u}T\rho c_p - k\nabla T) = Q$$

which for incompressible fluid flow, reduces to the following PDE:

$$\rho c_p \frac{\partial T}{\partial t} + \nabla \cdot (-k\nabla T) + \rho c_p \mathbf{u} \nabla \cdot T = Q \quad (7.14)$$

which is the governing PDE for heat transfer in a medium, including convection under incompressible flow.

7.2.2 The Bioheat Equation

For many biological applications, the heat source Q in Eq. 7.14, representing heat energy supplied to the medium per unit time per unit volume, consists of several components:

- tissue metabolism, namely the energy output of chemical reactions underlying cellular function, including maintenance, growth and reproduction.
- blood perfusion, which transports heat to the tissue at a rate proportional to the difference between the temperatures of the blood and tissue.
- external heat sources, such as *Joule heating* from the flow of electric current in a conductive medium. Such external heat sources typically require the implementation of a coupled physics mode to determine the heat source at each point, feeding the value into Eq. 7.14. In the case of Joule heating, for example, the external heat source Q_{ext} would be determined by solving the electrostatic equations for the electric potential, according to

$$\begin{aligned} Q_{ext} &= \mathbf{J} \cdot \mathbf{E} \\ &= \sigma |\mathbf{E}|^2 \text{ for conductive media} \end{aligned} \quad (7.15)$$

where \mathbf{J} , \mathbf{E} are the root mean square (RMS) vector quantities of current density and electric field respectively, and σ is the electrical conductivity.

In biological tissues, another source of heat from applied electric fields arises from *dielectric heating*, due to the flow of polarizing currents displacing charged molecules in the tissue. The total current which flows in the tissue is the sum of conductive and displacement currents, as given by Ampère's Law (Eq. 6.2)

$$\mathbf{J}_{tot} = \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t} = \sigma \mathbf{E} + \varepsilon \frac{\partial \mathbf{E}}{\partial t}$$

For most dielectrics, including biological tissues, the permittivity ε is a frequency-dependent complex quantity, producing a further phase shift in the displacement current. Writing this permittivity as

$$\varepsilon = \varepsilon' - j\varepsilon''$$

where $j = \sqrt{-1}$, the total heat produced by both conductive and dielectric heating is given by

$$Q_{ext} = \sigma |\mathbf{E}|^2 + \omega \varepsilon'' |\mathbf{E}|^2 \quad (7.16)$$

where \mathbf{E} is the RMS electric field, and the angular frequency factor ω in the dielectric heat term is due to the time-derivative of \mathbf{E} in the displacement current. At high frequencies where $\omega \varepsilon'' \gg \sigma$, dielectric heating will dominate.

These sources of heat can be incorporated into the *bioheat equation*,⁵ which modifies Eq. 7.14, ignoring convective heat flow, into

$$\rho c_p \frac{\partial T}{\partial t} + \nabla \cdot (-k \nabla T) = \rho_b c_b \omega_b (T_b - T) + Q_{met} + Q_{ext} \quad (7.17)$$

where ρ_b is the density of blood (in kg m^{-3}), c_b is the specific heat capacity of blood (in $\text{J kg}^{-1} \text{K}^{-1}$), ω_b is the blood perfusion rate (in s^{-1}), T_b is the arterial blood temperature (in K), Q_{met} is the heat source from metabolism (in W m^{-3}), and Q_{ext} is the external heat source (in W m^{-3}).

The bioheat equation can also be used to determine the spatial extent of tissue damage due to heat. The most common measure of tissue damage is based on Arrhenius' equation⁶:

$$\theta_d = 1 - \exp(-\alpha) \\ \alpha = \int_0^t A \cdot \exp\left[\frac{-E_a}{RT}\right] dt \quad (7.18)$$

where θ_d is the fraction of necrotic tissue, A is the tissue frequency factor (in s^{-1}), E_a is the activation energy for irreversible tissue damage (in J mol^{-1}), T is the temperature (in K), and R is the gas constant ($8.31 \text{ J K}^{-1} \text{ mol}^{-1}$). Parameters A and E_a are tissue-dependent, with typical values for liver reported as $A = 7.39 \times 10^{39} \text{ s}^{-1}$ and $E_a = 2.577 \times 10^5 \text{ J mol}^{-1}$ [8], and $A = 3 \times 10^{23} \text{ s}^{-1}$, $E_a = 1.62 \times 10^5 \text{ J mol}^{-1}$ for myocardium [1].

7.2.3 Example: RF Atrial Ablation

Disturbances in the electrical rhythm of the atria of the heart represent the most common type of cardiac arrhythmia. Such arrhythmias can be treated using *RF ablation*, in which a percutaneous catheter containing an active electrode placed onto the endocardial surface of the atria, injecting radio frequency (RF) electrical current at 500 kHz to form necrotic scars in the atria. A large dispersive return electrode is typically placed on the back of the chest. A simplified axisymmetric geometrical representation of a spherical RF ablation electrode indenting the atrial endocardium is shown in Fig. 7.10. We assume here that the myocardium is essentially conductive, and that the applied RF voltage is equivalent to direct current (DC) stimulation applied at the RMS value.⁷ Values of all model parameters are given in Table 7.1.

⁵Also referred to as the Pennes bioheat equation, named after Harry H. Pennes (1918–1963), who first introduced the mathematical description of the blood perfusion heat source component [10].

⁶Due to Svante Arrhenius (1859–1927), a founder of the modern science of physical chemistry.

⁷The effect of dielectric heating will be investigated further in Problem 7.4.

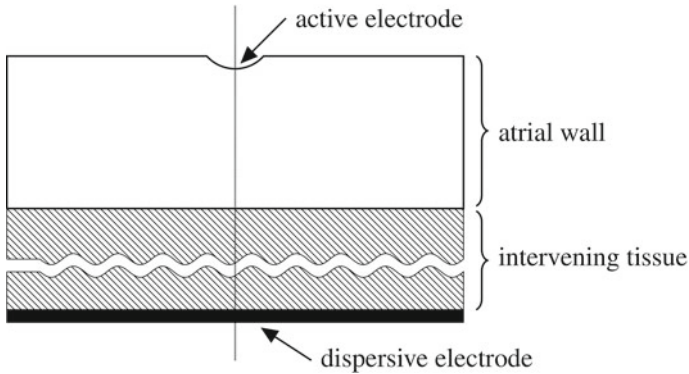


Fig. 7.10 Cross-section of RF atrial ablation model geometry, with rotational axis of symmetry shown *dashed*. The active electrode is shown as a circular arc indenting the upper (endocardial) atrial boundary. Between the lower (epicardial) atrial boundary and the electrode layer of intervening tissue of thickness much greater than the atrial wall

Table 7.1 Parameters for RF atrial ablation model

| Parameter | Value | Description |
|------------|---|--|
| L_{atr} | 6 mm | Atrial wall thickness |
| R | 9 mm | Radius (i.e. width) of atrial tissue segment |
| R_e | 1.5 mm | Active electrode radius |
| d_e | 0.5 mm | Electrode indentation depth |
| V_e | 80 V | Active electrode voltage |
| L_{int} | 20 cm | Intervening tissue thickness |
| σ | 0.61 S m^{-1} | Atrial electrical conductivity [3] |
| ρ | 1200 kg m^{-3} | Atrial tissue density [3] |
| C_p | $3200 \text{ J kg}^{-1} \text{ K}^{-1}$ | Atrial specific heat capacity [3] |
| k | $0.7 \text{ W m}^{-1} \text{ K}^{-1}$ | Atrial thermal conductivity [3] |
| T_b | 37°C | Blood temperature |
| ρ_b | 1000 kg m^{-3} | Blood density [3] |
| C_b | $4180 \text{ J kg}^{-1} \text{ K}^{-1}$ | Blood specific heat capacity [3] |
| ω_b | 0.005 s^{-1} | Blood perfusion rate [12] |
| Q_{met} | 0 W m^{-3} | Atrial metabolic heat source [12] |
| A | $3 \times 10^{23} \text{ s}^{-1}$ | Myocardial frequency factor [1] |
| E_a | 162 kJ mol^{-1} | Myocardial activation energy [1] |

To implement this model in COMSOL, we use the following steps:

Model Wizard

1. Open the Model Wizard and select the 2D axisymmetric spatial dimension.
2. In the Select Physics panel, choose Heat Transfer| Heat Transfer in Solids.⁸ Click ‘Add’, and leave T as the default temperature variable.
3. Staying within the Select Physics panel, choose AC/DC| Electric Currents, and click ‘Add’. Leave V as the default voltage variable.
4. Again from the Select Physics panel, choose Mathematics| PDE Interfaces| General Form PDE, and click ‘Add’. Leave u as the default variable, which will represent the α tissue damage variable of Eq. 7.18. Leave the default units of u as dimensionless, but change the units of the source term to 1/s.
5. Click the Study arrow (🔍) to open the Select Study panel. Select Time Dependent, and click “Done”. This will exit the Model Wizard, displaying the main COMSOL interface.

Global Definitions

1. Right-click Global Definitions in the model tree and select Parameters. Enter the following details in the Parameters table of the Settings window:

| Name | Expression | Description |
|---------|---------------------------|--------------------------------|
| L_atr | 6 [mm] | Atrial thickness |
| R | 9 [mm] | Atrial tissue width |
| R_e | 1.5 [mm] | Electrode radius |
| d_e | 0.5 [mm] | Electrode indentation depth |
| V_e | 80 [V] | Electrode voltage |
| L_int | 20 [cm] | Intervening tissue depth |
| sigma | 0.61 [S/m] | Tissue electrical conductivity |
| rho | 1200 [kg/m ³] | Atrial density |
| C_p | 3200 [J/(kg*K)] | Atrial specific heat |
| k | 0.7 [W/(m*K)] | Atrial thermal conductivity |
| T_b | 37 [degC] | Blood temperature |
| rho_b | 1000 [kg/m ³] | Blood density |
| C_b | 4180 [J/(kg*K)] | Blood specific heat |
| omega_b | 0.005 [1/s] | Blood perfusion rate |
| Q_met | 0 [W/m ³] | Metabolic heat source |
| A | 3e23 [1/s] | Atrial frequency factor |
| E_a | 162 [kJ/mol] | Atrial activation energy |

⁸If you have installed COMSOL’s optional Heat Transfer module, it includes a bioheat transfer module incorporating the Arrhenius measure of tissue damage and other tissue damage indicators. In our example, however, we use COMSOL’s default Heat Transfer module and will implement the Bioheat equation and Arrhenius tissue damage measure directly.

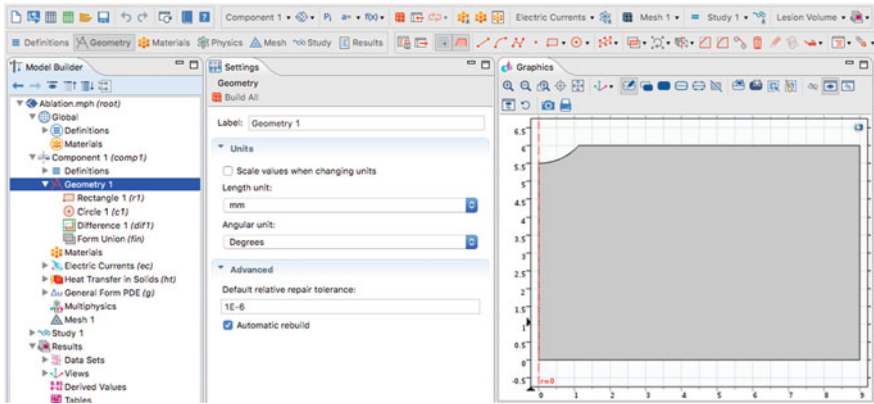





Fig. 7.11 COMSOL interface for atrial ablation model, showing 2D axisymmetric geometry (right) and model tree (far left)

Geometry

1. Select the Geometry 1 node in the model tree. In the settings window, change the default length unit to ‘mm’.
2. Right-click Geometry 1 and select Rectangle. Specify the width and height as R and L_{atr} respectively. Click Build Selected ().
3. Right-click Geometry 1 again and select Circle. In the Settings panel, enter R_e for the radius and 0 and L_{atr}+R_e-d_e in the r and z fields for the centre coordinates respectively. Click the Build Selected () button.
4. Right-click Geometry 1 a final time and select Booleans and Partitions| Difference. In the Settings panel, select rectangle (r1) as the object to add. Activate the ‘Objects to subtract’ panel and select circle (c1) as the object to subtract. Click the Build Selected () button. The resulting COMSOL interface with geometry and model tree should look like that shown in Fig. 7.11.

Component Definitions

1. Right-click the Definitions sub-node of Component 1 in the model tree and select Component Couplings| Integration. In the Settings panel, select domain 1. This specifies a user-defined integration operator ‘intop1’ for integrating expressions over the atrial wall domain.
2. Right-click the Definitions sub-node of Component 1 again and select Variables. Enter the following details in the Variables table of the Settings window:

| Name | Expression | Description |
|--------|-----------------------------|---------------|
| damage | 1-exp(-u) | Tissue damage |
| lesion | intop1(2*pi*r*(damage>0.5)) | Lesion volume |

Note that the variable `damage` corresponds to θ_d in Eq. 7.18. To understand the expression for the `lesion` variable, COMSOL evaluates conditional expressions such as `damage>0.5` to 1 if true, and 0 if false. The volume of the lesion, defined here as those tissue regions with $\theta_d > 0.5$, is then given by the axisymmetric integral expression

$$V_{lesion} = \iint_{\Omega} 2\pi r (\theta_d > 0.5) dr dz$$

where Ω denotes the axisymmetric tissue domain and $(\theta_d > 0.5)$ represents the conditional evaluation described above.

Electric Currents

1. Select the Current Conservation 1 sub-node of the Electric Currents node. In the Settings panel, specify a user defined electrical conductivity of `sigma` for domain 1 and a user defined relative permittivity of 1.
2. Right-click the Electric Currents node to add an ‘Electric Potential’ boundary condition. Select boundary 5 (i.e. the active electrode circular arc) and specify the value of the electric potential as `V_e`.
3. Right-click the Electric Currents node again to add a ‘Distributed Impedance’ boundary condition. Select boundary 2 (i.e. the lower boundary). This boundary condition specifies that there is a resistive layer between the boundary and a reference voltage V_{ref} on the other side of the layer, such that the outward normal component of current density J_n is given by

$$J_n = \frac{\sigma}{d_s} (V - V_{ref})$$

where V is the voltage at the boundary, σ is the electrical conductivity of the layer and d_s is its thickness. Using such a distributed impedance boundary condition allows us to specify the electrical properties of the intervening tissue without physically incorporating its geometry into the model. In the Settings panel of the Distributed Impedance 1 sub-node, specify V_{ref} as 0, the surface thickness d_s as `L_int`, the electrical conductivity as `sigma` and the relative permittivity as 1. These values assume that the intervening tissue has the same electrical properties as myocardium, and that the dispersive electrode is at ground voltage.

4. All remaining boundaries will be set to electrically insulating by default.

Heat Transfer in Solids

1. Select the Heat Transfer in Solids 1 sub-node of the Heat Transfer in Solids node. In the Settings panel, specify the thermal conductivity as `k`, the density as `rho`, and the heat capacity as `C_p`.
2. Right click the Heat Transfer in Solids node to add Heat Source. Select the newly-created Heat Source 1 sub-node and specify a user defined general heat source with expression:

$$\text{sigma} * \text{ec} . \text{normE}^2 + \text{rho}_b * \text{C}_b * \text{omega}_b * (T_b - T) + Q_{\text{met}}.$$

3. Right-click the Heat Transfer in Solids node again to add a ‘Heat Flux’ boundary condition. Select boundary 2 (i.e. the lower boundary). We will use this boundary condition to specify a thermally-resistive layer between the boundary and an external temperature T_{ext} on the other side of the layer, such that the inward normal component of heat flux q_n is given by

$$q_n = \frac{k}{L_{int}} (T_{ext} - T)$$

where T is the temperature at the boundary, k is the thermal conductivity of the intervening tissue layer and L_{int} is its thickness. In COMSOL, we can conveniently specify this boundary condition by selecting the ‘Convective heat flux’ option in the Heat Flux 1 sub-node settings, specifying k/L_{int} as the heat transfer coefficient and T_{b} as the external temperature, which corresponds to typical body (& blood) temperature. These parameter values assume that the intervening tissue has the same thermal properties as the myocardium.

4. Select the Initial Values 1 sub-node of the Heat Transfer in Solids node, and specify the initial value of temperature as T_{b} .

General Form PDE

1. Select the General Form PDE 1 sub-node of the General Form PDE node. In the Settings panel, specify the Conservative flux Γ as 0 and 0 for both the r and z components, the source term as $A \cdot \exp(-E_a / (R_{const} \cdot T))$, the damping coefficient d_a as 1, and the mass coefficient e_a as 0. This represents the PDE



$$\frac{\partial u}{\partial t} = A \cdot \exp\left(\frac{-E_a}{RT}\right)$$

which is equivalent to the integral form of Eq. 7.18 with $u \equiv \alpha$.

Mesh

1. Select the Mesh node of the model tree and choose an ‘Extra fine’ element size in the Settings panel.

Study

1. Under the Study 1 node, select the Time Dependent solver. Click the range button () and specify the time range as 0 to 20 s in steps of 0.1 s.
2. To solve the model, right-click Study 1 and select Compute (). COMSOL will display the default axisymmetric plot of voltage distribution at 20 s, similar to that shown in Fig. 7.12. Clicking on the Temperature, 3D (ht) volume plot (under the Results node of the model tree) will generate a 3D representation of the axisymmetric temperature at this time, as shown also in Fig. 7.12.

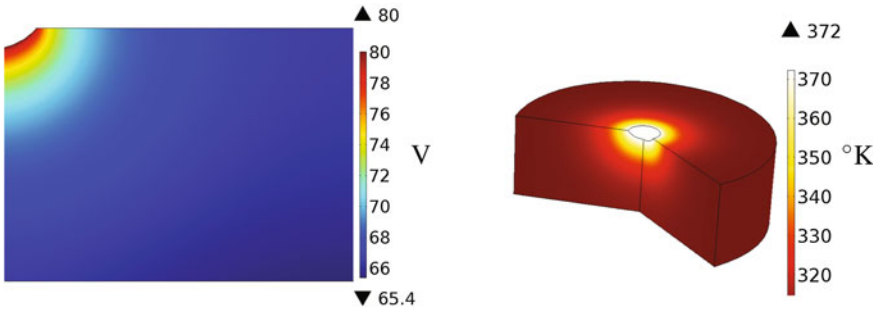


Fig. 7.12 (Left) Atrial wall voltage distribution at 20s in the axisymmetric model plane. (Right) 3D temperature profile at 20s formed by rotating the axisymmetric plane around its axis

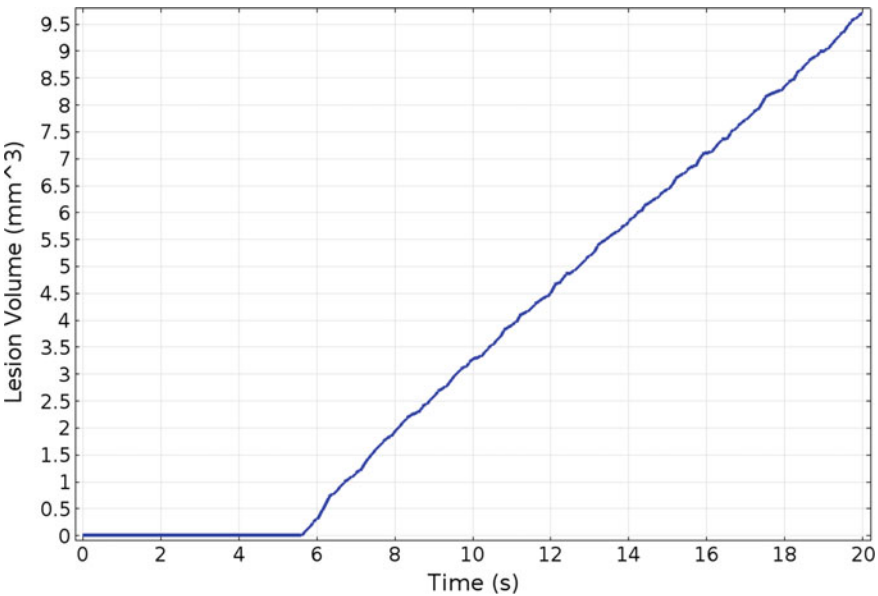



Fig. 7.13 Lesion volume against time, as determined by the RF ablation model

Results

1. Right-click the Result node and select '1D Plot Group'. Right-click the newly-created 1D Plot Group 3 sub-node and select 'Global'. In the settings panel, specify the y-Axis Data expression to plot as `lesion`, and specify the units `mm3` by typing this directly into the Units column, overwriting the default units.
2. Select the 1D Plot Group 3 sub-node and select the y-axis label checkbox. Type `Lesion Volume (mm3)` as the label to display on this axis.

- Click the Plot button () to display the plot of lesion volume against time, as shown in Fig. 7.13. It can be seen that the lesion volume begins to increase from a time of about 5.6 s reaching a value of $\approx 9.7 \text{ mm}^{-3}$ at 20 s.

Although this model is based on a simplified geometrical representation of the ablation electrode and atrial wall, a significant feature of the implementation is the use of distributed impedance and heat flux boundary conditions to represent flow of electrical current and heat into the intervening tissue between the atrial wall and the dispersive electrode. The choice of appropriate boundary conditions that mimic system behaviour forms a crucial aspect of the overall modelling process.

7.3 Further Reading

A good resource on modelling diffusion and heat transfer in biomedical systems, with a focus on COMSOL implementations, is the text of Datta and Rakesh [5]. Further examples of COMSOL bioheat modelling may be found in the text of Pryor [11]. The text of Zemansky and Dittman [14] provides a general overview of the physics of thermodynamics, whilst those of Barnes and Greenebaum [2] and Vorst et al. [13] cover the principles of electromagnetic heating of biological tissues.

Problems

7.1 A transdermal patch is placed on the arm of a patient to deliver a specific dose of medication through the skin. The equation governing the concentration c of the drug is

$$\frac{\partial c}{\partial t} = D \frac{\partial^2 c}{\partial x^2} - k_{up} c$$

where x is the depth through the skin, D is the diffusion coefficient and k_{up} is the rate of uptake of the drug into the bloodstream. Find the steady-state concentration as a function of x given the boundary conditions $c = C_0$ at $x = 0$ (skin surface) and $c = 0$ at $x = d_c$ (critical depth). D , k_{up} , C_0 and d_c are fixed parameters.

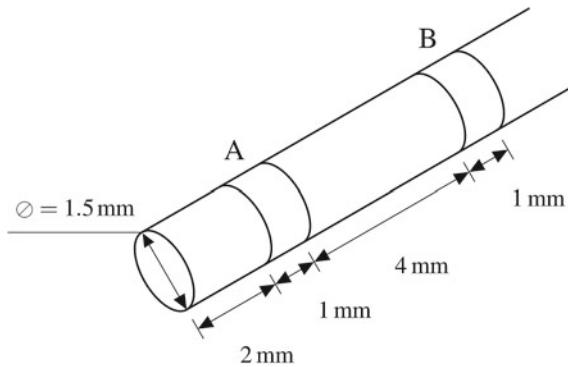
7.2 Cardiac output can be determined clinically using the method of indicator dilution, whereby a known quantity of an indicator, typically a contrast agent, is injected into the bloodstream, and its concentration measured downstream from the site of injection as a function of time. Using COMSOL, we can numerically investigate the validity of this method by employing a simplified cylindrical blood vessel of length 50 cm and diameter 1 cm, and assuming 1 mol of indicator is injected through one end of the vessel over a brief time interval of 100 ms. We then solve for the concentration profile of the indicator at the downstream boundary centre, assuming a parabolic velocity profile of the blood (Eq. 7.7), and a diffusion coefficient for the

indicator of $1 \times 10^{-9} \text{ m}^2 \text{ s}^{-1}$. Using a parametric sweep to vary the total volume flow rate (i.e. cardiac output) from 0.1 to 1 L min^{-1} in steps of 0.1 L min^{-1} , verify that the cardiac output, Q satisfies

$$Q = M_0 \left[\int_0^\infty c \, dt \right]^{-1}$$

where M_0 is the total amount of injected indicator (in mol) and c is the indicator concentration downstream. Take the upper limit of the above integral as $t = 20 \text{ s}$.

7.3 A proposed cylindrical probe for the heat treatment of hepatic tumours is shown below.



Joule heating due to current flow between electrodes A and B (with B at ground) kills cancerous tissue when the Arrhenius damage proportion reaches above 0.5.

| Parameter | Value | Description |
|------------|---|------------------------------|
| σ | 0.61 S m^{-1} | Electrical conductivity |
| ρ | 1200 kg m^{-3} | Density |
| C_p | $3200 \text{ J kg}^{-1} \text{ K}^{-1}$ | Specific heat capacity |
| k | $0.7 \text{ W m}^{-1} \text{ K}^{-1}$ | Thermal conductivity |
| T_b | 37°C | Blood temperature |
| ρ_b | 1000 kg m^{-3} | Blood density |
| C_b | $4180 \text{ J kg}^{-1} \text{ K}^{-1}$ | Blood specific heat capacity |
| ω_b | 0.05 s^{-1} | Blood perfusion rate |
| Q_{met} | 0 W m^{-3} | Metabolic heat source |
| A | $7.39 \times 10^{39} \text{ s}^{-1}$ | Frequency factor |
| E_a | $257.7 \text{ kJ mol}^{-1}$ | Activation energy |

Implement a COMSOL model of this probe inserted into a spherical tumour of radius 2.5 cm , such that the mid-point between the probe electrodes lies at the centre of the tumour. Assume that the temperature of the outer surface of the tumour is maintained at a body temperature of $T_b = 37^\circ \text{C}$. Plot the volume of tissue destroyed by the heat treatment as a function of time (up to 8 min) for DC probe voltages of 22 V and 30 V . Use the tissue parameters shown in the table above.

7.4 For the RF atrial ablation model of this chapter (Sect. 7.2.3), the atrial tissue was assumed to be purely conductive, with the external heat source given by $\sigma |\mathbf{E}|^2$, where σ is the electrical conductivity and $|\mathbf{E}|$ is the electric field magnitude. In reality, the permittivity of atrial tissue may not be negligible, particularly at the RF frequency of 500kHz. To investigate the contribution of the additional dielectric heating component, we note that the relative permittivity of heart muscle has been reported to satisfy the following empirical relation [7]

$$\varepsilon_r(\omega) = \varepsilon_\infty + \sum_{n=1}^4 \frac{\Delta\varepsilon_n}{1 + (j\omega\tau_n)^{(1-\alpha_n)}} \quad (7.19)$$

where ω is the angular frequency and $j = \sqrt{-1}$. Using the parameters of this relation given in the following table, determine the negative complex component of the permittivity ε'' at 500kHz, and re-solve the RF atrial ablation model accordingly, plotting the new lesion volume against time. Is the purely conductive assumption justified when simulating RF cardiac ablation?

| Parameter | Value | Parameter | Value |
|-----------------------|-----------|-----------------------|-------------------|
| ε_∞ | 4 | $\Delta\varepsilon_3$ | 4.5×10^5 |
| $\Delta\varepsilon_1$ | 50 | τ_3 | 72.34 μ s |
| τ_1 | 7.96 ps | α_3 | 0.22 |
| α_1 | 0.1 | $\Delta\varepsilon_4$ | 2.5×10^7 |
| $\Delta\varepsilon_2$ | 1200 | τ_4 | 4.547 ms |
| τ_2 | 159.15 ns | α_4 | 0 |
| α_2 | 0.05 | | |

References

1. Agah R, Gandjbakhche AH, Motamedi M, Nossal R, Bonner RF (1996) Dynamics of temperature dependent optical properties of tissue: dependence on thermally induced alteration. *IEEE Trans Biomed Eng* 43:839–846
2. Barnes FS, Greenebaum B (eds) (2007) *Handbook of biological effects of electromagnetic fields: bioengineering and biophysical aspects of electromagnetic fields*, 3rd edn. CRC Press, Boca Raton
3. Berjano EJ, Hornero F (2004) Thermal-electrical modeling for epicardial atrial radiofrequency ablation. *IEEE Trans Biomed Eng* 51:1348–1357
4. Cremasco MA, Wang LN-H (2012) Estimation of partition, free and specific diffusion coefficients of paclitaxel and taxanes in a fixed bed by moment analysis: experimental, modeling and simulation studies. *Acta Sci Technol* 34:33–40
5. Datta A, Rakesh V (2010) *An introduction to modelling of transport processes: applications to biomedical systems*. Cambridge University Press, Cambridge
6. Farb A, Heller PF, Shroff S, Cheng L, Kolodgie FD, Carter AJ, Scott DS, Froehlich J, Virmani R (2001) Pathological analysis of local delivery of paclitaxel via a polymer-coated stent. *Circulation* 104:473–479

7. Gabriel S, Lau RW, Gabriel C (1996) The dielectric properties of biological tissues: III. Parametric models for the dielectric spectrum of tissues. *Phys Med Biol* 41:2271–2293
8. Jacques S, Rastegar S, Thomsen S, Motamedi M (1996) Nonlinear finite-element analysis of the role of dynamic changes in blood perfusion and optical properties in laser coagulation of tissue. *IEEE J Sel Top Quantum Electron* 2:922933
9. Kaul U, Bangalore S, Seth A, Arambam P, Abhaychand RK, Patel TM, Banker D, Abhyankar A, Mulasari AS, Shah S, Jain R, Kumar PR, Bahuleyan CG (2015) Paclitaxel-eluting versus everolimus-eluting coronary stents in diabetes. *N Engl J Med* 373(18):1709–1719
10. Pennes HH (1948) Analysis of tissue and arterial blood temperature in the resting forearm. *J Appl Physiol* 1:93122
11. Pryor RW (2011) Multiphysics modeling using COMSOL: a first principles approach. Jones and Bartlett, Sudbury
12. Shahidi AV, Savard P (1994) A finite element model for radiofrequency ablation of the myocardium. *IEEE Trans Biomed Eng* 41:963–968
13. Vorst AV, Rosen A, Kotsuka Y (2006) RF/microwave interaction with biological tissues. Wiley, Hoboken
14. Zemansky MW, Dittman RH (1997) Heat and thermodynamics, 7th edn. McGraw-Hill, New York

Chapter 8

Solid Mechanics

8.1 Biomechanics

The application of mechanics to biological systems is referred to as *biomechanics*. Biomechanics can be further subdivided into the fields of *solid mechanics* and *fluid mechanics*, the latter which will be dealt with in Chap.9. Solid mechanics deals with the deformation of solid bodies under applied loads. In biomedical engineering applications, solid mechanics is used for simulating mechanical properties of soft tissues and bone to understand their normal and pathological function, as well as for the development of rehabilitation devices and mechanical prostheses such as hip joint replacements.

8.2 Tensor Fundamentals

To understand the basic principles of solid mechanics, including the notions of stress and strain, it is necessary to be familiar with the concept of *tensors*. We have already seen that some physical quantities, including electrical conductivity in anisotropic materials (see Eq. 6.16), can be represented as a tensor.

8.2.1 Tensor Definition

Tensors represent a natural extension to vectors. Defining a set of three orthogonal basis unit vectors in 3D space \mathbf{e}_1 , \mathbf{e}_2 , \mathbf{e}_3 , any vector quantity \mathbf{a} can be expressed in terms of these as

$$\mathbf{a} = a_1\mathbf{e}_1 + a_2\mathbf{e}_2 + a_3\mathbf{e}_3$$

where a_1, a_2, a_3 are the components of the vector \mathbf{a} with respect to the basis set. Given two vectors \mathbf{a}, \mathbf{b} , we can formally define their scalar dot and vector cross products according to:

$$\mathbf{a} \cdot \mathbf{b} = a_1 b_1 + a_2 b_2 + a_3 b_3 \quad (8.1)$$

$$\mathbf{a} \times \mathbf{b} = \begin{vmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3 \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix} \quad (8.2)$$

where the a_i, b_i ($i = 1, 2, 3$) are the components of these vectors, and the determinant expansion in Eq. 8.2 is understood to be taken by the first row.

In addition to these products, we can also define the *dyadic* product of two vectors $\mathbf{a} \otimes \mathbf{b}$, which forms a second-order tensor. For any three vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$ and scalar quantity α , the dyadic product satisfies the following properties:

$$\begin{aligned} (\alpha \mathbf{a}) \otimes \mathbf{b} &= \mathbf{a} \otimes (\alpha \mathbf{b}) = \alpha (\mathbf{a} \otimes \mathbf{b}) \\ \mathbf{a} \otimes (\mathbf{b} + \mathbf{c}) &= \mathbf{a} \otimes \mathbf{b} + \mathbf{a} \otimes \mathbf{c} \\ (\mathbf{b} + \mathbf{c}) \otimes \mathbf{a} &= \mathbf{b} \otimes \mathbf{a} + \mathbf{c} \otimes \mathbf{a} \end{aligned}$$

Furthermore, we define the dot product between a dyad and a vector to yield a vector, according to

$$\begin{aligned} (\mathbf{a} \otimes \mathbf{b}) \cdot \mathbf{c} &= \mathbf{a} (\mathbf{b} \cdot \mathbf{c}) \\ \mathbf{a} \cdot (\mathbf{b} \otimes \mathbf{c}) &= (\mathbf{a} \cdot \mathbf{b}) \mathbf{c} \end{aligned}$$

Using the above rules, we can express the dyadic product of vectors \mathbf{a} and \mathbf{b} as

$$\begin{aligned} \mathbf{a} \otimes \mathbf{b} &= \sum_{i=1}^3 \sum_{j=1}^3 (a_i \mathbf{e}_i) \otimes (b_j \mathbf{e}_j) \\ &= \sum_{i=1}^3 \sum_{j=1}^3 a_i b_j \mathbf{e}_i \otimes \mathbf{e}_j \\ &= \sum_{i=1}^3 \sum_{j=1}^3 a_i b_j \mathbf{e}_i \otimes \mathbf{e}_j \end{aligned} \quad (8.3)$$

The dyadic products of the basis vectors $\mathbf{e}_i \otimes \mathbf{e}_j$ are also tensors, and are referred to as *unit dyads*. Hence the second-order tensor quantity $\mathbf{a} \otimes \mathbf{b}$ consists of the linear combination of unit dyads $\mathbf{e}_i \otimes \mathbf{e}_j$ ($i, j = 1, 2, 3$), each weighted by a tensor component $a_i b_j$. In fact, any second-order tensor \mathbf{A} can be expressed in terms of the sum of the product of its components and the nine unit dyads as:

$$\mathbf{A} = \sum_{i=1}^3 \sum_{j=1}^3 A_{ij} \mathbf{e}_i \otimes \mathbf{e}_j \tag{8.4}$$

where the A_{ij} are the tensor components. From Eq. 8.3, we see that in general, $\mathbf{a} \otimes \mathbf{b} \neq \mathbf{b} \otimes \mathbf{a}$, since the ij th components of the former and latter are $a_i b_j$ and $b_i a_j$ respectively, which are not equal.

We can also define third-order tensors using the *triadic* product of three vectors $\mathbf{a} \otimes \mathbf{b} \otimes \mathbf{c}$ whose 27 components are $a_i b_j c_k$ ($i, j, k = 1, 2, 3$). In particular, a third order tensor \mathbf{C} is given by the sum of the product of its components C_{ijk} and the unit triads as

$$\mathbf{C} = \sum_{i=1}^3 \sum_{j=1}^3 \sum_{k=1}^3 C_{ijk} \mathbf{e}_i \otimes \mathbf{e}_j \otimes \mathbf{e}_k \tag{8.5}$$

with analogous definitions for even higher-order tensors.¹

8.2.2 *Indicial Notation*

When dealing with cumbersome tensor equations such as Eqs. 8.4 and 8.5, it is convenient to utilize a shorthand indicial notation² governed by the following rules:

- Terms with a single index such as x_i denote $x_1, x_2,$ or x_3 .
- Terms with multiple indices such σ_{ij} denote $\sigma_{11}, \sigma_{12}, \sigma_{13}, \sigma_{21}, \sigma_{22}, \sigma_{23}, \sigma_{31}, \sigma_{32},$ or σ_{33} .
- Unless explicitly stated otherwise, repetition of an index in any *single term* denotes summation over the range of the index. For example:

$$x_i x_i \equiv x_1 x_1 + x_2 x_2 + x_3 x_3$$

and

$$y_i = A_{ij} x_j$$

is shorthand for

$$y_i = A_{i1} x_1 + A_{i2} x_2 + A_{i3} x_3$$

¹The descriptions here and in the following sections pertain to 3D space by default, however these descriptions are readily generalizable to lower dimensions. For example in 2D, vectors have two components, second-order tensors have four components, and third-order tensors have eight components.

²This notation was introduced by Albert Einstein (1879–1955) in his 1916 paper on the General Theory of Relativity [5].

which is in turn shorthand for

$$\begin{aligned}y_1 &= A_{11}x_1 + A_{12}x_2 + A_{13}x_3 \\y_2 &= A_{21}x_1 + A_{22}x_2 + A_{23}x_3 \\y_3 &= A_{31}x_1 + A_{32}x_2 + A_{33}x_3\end{aligned}$$

Using this indicial notation, Eq. 8.4 may be conveniently written as $\mathbf{A} = A_{ij} \mathbf{e}_i \otimes \mathbf{e}_j$ and $\mathbf{C} = C_{ijk} \mathbf{e}_i \otimes \mathbf{e}_j \otimes \mathbf{e}_k$ respectively. We can also use indicial notation to express the dot product between a second-order tensor \mathbf{A} and a vector \mathbf{b} to yield a new vector \mathbf{c} according to

$$c_i = A_{ik}b_k \quad (\mathbf{c} = \mathbf{A} \cdot \mathbf{b}) \quad (8.6)$$

Similarly, we can also define the dot product between two second-order tensors, \mathbf{A} and \mathbf{B} , to yield another second-order tensor \mathbf{C} according to:

$$C_{ij} = A_{ik}B_{kj} \quad (\mathbf{C} = \mathbf{A} \cdot \mathbf{B}) \quad (8.7)$$

8.2.3 Tensor Transformation Law

Consider a set of orthogonal unit basis vectors $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$. Since these form an orthogonal unit set, we can write

$$\mathbf{e}_i \cdot \mathbf{e}_j = \delta_{ij} \quad (8.8)$$

where δ_{ij} is the *Kronecker delta*, defined as

$$\delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

We can form a new set of basis vectors $\bar{\mathbf{e}}_1, \bar{\mathbf{e}}_2, \bar{\mathbf{e}}_3$ by an appropriate rotational transformation given by

$$\bar{\mathbf{e}}_k = M_{ki}\mathbf{e}_i \quad (8.9)$$

where we have used here (and from now on) the indicial notation, and M_{ki} represents the cosine of the angle between $\bar{\mathbf{e}}_k$ and \mathbf{e}_i (the so-called *direction cosines*). Since this new set of vectors also forms an orthogonal unit set, we also have

$$\begin{aligned}\delta_{ij} &= \bar{\mathbf{e}}_i \cdot \bar{\mathbf{e}}_j \\ &= (M_{is}\mathbf{e}_s) \cdot (M_{jr}\mathbf{e}_r) \quad (\text{using Eq. 8.9}) \\ &= M_{is}M_{jr}\delta_{sr} \\ &= M_{ir}M_{jr}\end{aligned} \quad (8.10)$$

Any vector \mathbf{a} with components a_i with respect to the basis $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ will have components \bar{a}_i with respect to the basis $\bar{\mathbf{e}}_1, \bar{\mathbf{e}}_2, \bar{\mathbf{e}}_3$. A relation between these components may be found using

$$\mathbf{a} = a_i \mathbf{e}_i = \bar{a}_s \bar{\mathbf{e}}_s = \bar{a}_s M_{si} \mathbf{e}_i$$

where we have made use of Eq. 8.9 in the last term. Equating the coefficients of \mathbf{e}_i in the last two terms above, we have

$$a_i = M_{si} \bar{a}_s$$

Multiplying both sides by M_{ki} and continuing with the usual summation convention, the above becomes

$$\begin{aligned} M_{ki} a_i &= M_{ki} M_{si} \bar{a}_s \\ &= \delta_{ks} \bar{a}_s \\ &= \bar{a}_k \end{aligned}$$

Hence,

$$\bar{a}_k = M_{ki} a_i \tag{8.11}$$

which follows the same transformation law as Eq. 8.9. Vectors are typically expressed in terms of their components in vector-matrix form as

$$\mathbf{a} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix}$$

However, any set of three scalars arranged in this form will not necessarily yield a physical vector: they will only form a vector if they satisfy the transformation law given by Eq. 8.11 for a change of coordinate basis.

We can also derive a similar transformation law for a second-order tensor \mathbf{A} . With respect to the basis $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$, this tensor has components A_{ij} , and with respect to the basis $\bar{\mathbf{e}}_1, \bar{\mathbf{e}}_2, \bar{\mathbf{e}}_3$, it has components \bar{A}_{ij} . Both representations of the same tensor are equivalent. Hence,

$$\begin{aligned} \mathbf{A} &= A_{ij} \mathbf{e}_i \otimes \mathbf{e}_j \\ &= \bar{A}_{rs} \bar{\mathbf{e}}_r \otimes \bar{\mathbf{e}}_s \\ &= \bar{A}_{rs} (M_{ri} \mathbf{e}_i) \otimes (M_{sj} \mathbf{e}_j) \\ &= \bar{A}_{rs} M_{ri} M_{sj} \mathbf{e}_i \otimes \mathbf{e}_j \end{aligned}$$

Equating the coefficients of $\mathbf{e}_i \otimes \mathbf{e}_j$ in the first and last lines above, we obtain

$$A_{ij} = M_{ri} M_{sj} \bar{A}_{rs}$$

Multiplying both sides of the above by $M_{ki}M_{mj}$, and continuing with the default summation convention, we obtain

$$\begin{aligned} M_{ki}M_{mj}A_{ij} &= M_{ki}M_{mj}A_{ij}M_{ri}M_{sj}\bar{A}_{rs} \\ &= M_{ki}M_{ri}M_{mj}M_{sj}\bar{A}_{rs} \\ &= \delta_{kr}\delta_{ms}\bar{A}_{rs} \\ &= \bar{A}_{km} \end{aligned}$$

From which we obtain the *tensor transformation law*:

$$\bar{A}_{km} = M_{ki}M_{mj}A_{ij} \quad (8.12)$$

We can write a second-order tensor in equivalent vector-matrix form as

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

As with vectors, any set of nine quantities a_{ij} will not necessarily form a second-order tensor. It will be such a tensor only if it satisfies the transformation law of Eq. 8.12. Expressing the components of the rotational transform M_{ij} as a matrix \mathbf{M} , we obtain the following matrix equivalent expressions of the above transformation rules:

$$\bar{\mathbf{a}} = \mathbf{M}\mathbf{a} \quad (\text{From Eq. 8.11})$$

$$\bar{\mathbf{A}} = \mathbf{M}\mathbf{A}\mathbf{M}^T \quad (\text{From Eq. 8.12})$$

$$\mathbf{M}\mathbf{M}^T = \mathbf{I} \quad (\text{From Eq. 8.10})$$

where \mathbf{I} is the identity matrix.

8.2.4 Tensor Invariants

When a vector \mathbf{a} is subjected to an orthogonal coordinate transformation $\bar{\mathbf{a}} = \mathbf{M}\mathbf{a}$, we can show that the magnitude-squared of the vector will not change. In terms of the vector components, this is expressed as:

$$\bar{a}_1^2 + \bar{a}_2^2 + \bar{a}_3^2 = a_1^2 + a_2^2 + a_3^2$$

This result can be shown as follows:

$$\begin{aligned}
\bar{a}_1^2 + \bar{a}_2^2 + \bar{a}_3^2 &= \bar{\mathbf{a}} \cdot \bar{\mathbf{a}} \\
&= (\mathbf{M}\mathbf{a}) \cdot (\mathbf{M}\mathbf{a}) \\
&= (\mathbf{M}\mathbf{a})^T (\mathbf{M}\mathbf{a}) \\
&= (\mathbf{a}^T \mathbf{M}^T) (\mathbf{M}\mathbf{a}) \\
&= \mathbf{a}^T (\mathbf{M}\mathbf{M}^T) \mathbf{a} \\
&= \mathbf{a}^T \mathbf{a} \\
&= a_1^2 + a_2^2 + a_3^2
\end{aligned}$$

Hence, the expression

$$I = a_1^2 + a_2^2 + a_3^2 \quad (8.13)$$

is an example of a vector *invariant*, that is a scalar property of the vector whose value remains unchanged for any orthogonal transformation of the coordinate basis.

For any second-order tensor \mathbf{A} , we will also determine a set of invariants whose value remains unchanged under the orthogonal coordinate transformation $\bar{\mathbf{A}} = \mathbf{M}\mathbf{A}\mathbf{M}^T$. In fact, one such set is the three eigenvalues of \mathbf{A} as well as any function of these. To see this, we note that the eigenvalues λ of $\bar{\mathbf{A}}$ can be determined by solving its characteristic equation:

$$\det(\bar{\mathbf{A}} - \lambda\mathbf{I}) = 0$$

and using $\bar{\mathbf{A}} = \mathbf{M}\mathbf{A}\mathbf{M}^T$, we have:

$$\begin{aligned}
\det(\mathbf{M}\mathbf{A}\mathbf{M}^T - \lambda\mathbf{I}) &= 0 \\
\det(\mathbf{M}\mathbf{A}\mathbf{M}^T - \mathbf{M}\lambda\mathbf{I}\mathbf{M}^T) &= 0 \quad (\text{since } \mathbf{M}\mathbf{M}^T = \mathbf{I}) \\
\det(\mathbf{M}(\mathbf{A} - \lambda\mathbf{I})\mathbf{M}^T) &= 0
\end{aligned}$$

And using the determinant identities $\det(\mathbf{A}\mathbf{B}) = \det \mathbf{A} \det \mathbf{B}$, $\det(\mathbf{B}^T) = \det \mathbf{B}$, the above becomes

$$\det \mathbf{M} \det(\mathbf{A} - \lambda\mathbf{I}) \det \mathbf{M} = 0$$

and since $\mathbf{M}\mathbf{M}^T = \mathbf{I}$, we have $(\det \mathbf{M})^2 = 1$. Therefore, we have

$$\det(\mathbf{A} - \lambda\mathbf{I}) = 0 \quad (8.14)$$

which is the characteristic equation of \mathbf{A} . Hence we have shown that the characteristic equation, and therefore the eigenvalues, of both $\bar{\mathbf{A}}$ and \mathbf{A} are the same. Expanding Eq. 8.14 by writing out all the components of \mathbf{A} , we obtain

$$\begin{aligned}
& \begin{vmatrix} A_{11} - \lambda & A_{12} & A_{13} \\ A_{21} & A_{22} - \lambda & A_{23} \\ A_{31} & A_{32} & A_{33} - \lambda \end{vmatrix} = 0 \\
(A_{11} - \lambda) & \begin{vmatrix} A_{22} - \lambda & A_{23} \\ A_{32} & A_{33} - \lambda \end{vmatrix} - A_{12} \begin{vmatrix} A_{21} & A_{23} \\ A_{31} & A_{33} - \lambda \end{vmatrix} + A_{13} \begin{vmatrix} A_{21} & A_{22} - \lambda \\ A_{31} & A_{32} \end{vmatrix} = 0 \\
(A_{11} - \lambda) & \{ (A_{22} - \lambda)(A_{33} - \lambda) - A_{23}A_{32} \} - A_{12} \{ A_{21}(A_{33} - \lambda) - A_{23}A_{31} \} + \\
& A_{13} \{ A_{21}A_{32} - (A_{22} - \lambda)A_{31} \} = 0 \\
(A_{11} - \lambda) & \left\{ A_{22}A_{33} - A_{23}A_{32} - \lambda(A_{22} + A_{33}) + \lambda^2 \right\} - \\
A_{12} & \{ A_{21}A_{33} - A_{23}A_{31} - \lambda A_{21} \} + A_{13} \{ A_{21}A_{32} - A_{22}A_{31} + \lambda A_{31} \} = 0 \\
A_{11}A_{22}A_{33} & - A_{11}A_{23}A_{32} - \lambda(A_{11}A_{22} + A_{11}A_{33}) + A_{11}\lambda^2 - \\
& \lambda(A_{22}A_{33} - A_{23}A_{32}) + \lambda^2(A_{22} + A_{33}) - \lambda^3 - \\
A_{12}A_{21}A_{33} & + A_{12}A_{23}A_{31} + \lambda A_{12}A_{21} + A_{13}A_{21}A_{32} - A_{13}A_{22}A_{31} + \lambda A_{13}A_{31} = 0 \\
A_{11}A_{22}A_{33} & - A_{11}A_{23}A_{32} - A_{12}A_{21}A_{33} + A_{12}A_{23}A_{31} + A_{13}A_{21}A_{32} - A_{13}A_{22}A_{31} - \\
& \lambda \{ A_{11}A_{12} + A_{11}A_{33} + A_{22}A_{33} - A_{23}A_{32} + A_{12}A_{21} + A_{13}A_{31} \} + \\
& \lambda^2 \{ A_{11} + A_{22} + A_{33} \} - \lambda^3 = 0
\end{aligned}$$

which is a cubic polynomial in λ . Multiplying throughout by -1 , we obtain the characteristic polynomial

$$\lambda^3 - I_1\lambda^2 + I_2\lambda - I_3 = 0$$

with

$$I_1 = A_{11} + A_{22} + A_{33}$$

$$I_2 = A_{11}A_{12} + A_{11}A_{33} + A_{22}A_{33} - A_{23}A_{32} - A_{12}A_{21} - A_{13}A_{31}$$

$$I_3 = A_{11}A_{22}A_{33} - A_{11}A_{23}A_{32} - A_{12}A_{21}A_{33} + A_{12}A_{23}A_{31} + A_{13}A_{21}A_{32} - A_{13}A_{22}A_{31}$$

As noted above, this characteristic polynomial remains unchanged under orthogonal coordinate transformation, hence I_1 , I_2 and I_3 are also invariants of \mathbf{A} . We can simplify their expressions further using

$$\begin{aligned}
\text{trace } \mathbf{A} &= A_{11} + A_{22} + A_{33} = I_1 \\
\frac{1}{2} \{ (\text{trace } \mathbf{A})^2 - \text{trace } \mathbf{A}^2 \} &= \frac{1}{2} \{ (A_{11} + A_{22} + A_{33})^2 - (A_{11}^2 + A_{12}A_{21} + A_{13}A_{31} + \\
& A_{21}A_{12} + A_{22}^2 + A_{23}A_{32} + A_{31}A_{13} + A_{32}A_{23} + A_{33}^2) \} \\
&= \frac{1}{2} \{ 2A_{11}A_{22} + 2A_{11}A_{33} + 2A_{22}A_{33} - \\
& 2A_{12}A_{21} - 2A_{13}A_{31} - 2A_{23}A_{32} \} \\
&= A_{11}A_{22} + A_{11}A_{33} + A_{22}A_{33} - \\
& A_{12}A_{21} - A_{13}A_{31} - A_{23}A_{32} \\
&= I_2 \\
\det \mathbf{A} &= \begin{vmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{vmatrix}
\end{aligned}$$

$$\begin{aligned}
&= A_{11} \begin{vmatrix} A_{22} & A_{23} \\ A_{32} & A_{33} \end{vmatrix} - A_{12} \begin{vmatrix} A_{21} & A_{23} \\ A_{31} & A_{33} \end{vmatrix} + A_{13} \begin{vmatrix} A_{21} & A_{22} \\ A_{31} & A_{32} \end{vmatrix} \\
&= A_{11} (A_{22}A_{33} - A_{23}A_{32}) - A_{12} (A_{21}A_{33} - A_{23}A_{31}) + \\
&\quad A_{13} (A_{21}A_{32} - A_{22}A_{31}) \\
&= A_{11}A_{22}A_{33} - A_{11}A_{23}A_{32} - A_{12}A_{21}A_{33} + A_{12}A_{23}A_{31} + \\
&\quad A_{13}A_{21}A_{32} - A_{13}A_{22}A_{31} \\
&= I_3
\end{aligned}$$

Hence, the three invariants for second order tensors may be written as

$$I_1 = \text{trace} \mathbf{A} \quad (8.15)$$

$$I_2 = \frac{1}{2} \{(\text{trace} \mathbf{A})^2 - \text{trace} \mathbf{A}^2\} \quad (8.16)$$

$$I_3 = \det \mathbf{A} \quad (8.17)$$

Finally, we note that it is possible to choose an orthogonal transformation \mathbf{M} such that in the new basis $\bar{\mathbf{e}}_i$, the transformed tensor $\bar{\mathbf{A}}$ is diagonal:

$$\bar{\mathbf{A}} = \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix}$$

where the λ_i values on the diagonal correspond to the eigenvalues of \mathbf{A} (and $\bar{\mathbf{A}}$). The set of basis vectors $\bar{\mathbf{e}}_i$ which render $\bar{\mathbf{A}}$ diagonal are known as the *principal axes* of \mathbf{A} , and the eigenvalues λ_i are known as its *principal values*.

8.3 Mechanics Principles

Having covered the concept of tensors, we are now in position to introduce the mechanical notions of *stress* and *strain*, which respectively characterize loads and deformations in solids.

8.3.1 Stress

Stress is defined as force per unit area, having SI units of Pascals (Pa), which is equivalent to Nm^{-2} . These are the same units as the physical scalar quantity of *pressure*. Stress however is a tensor, as will be clarified further below. When applied to an external surface of a body, stress is referred to as *traction*, a vector quantity proportional to the applied force, as shown in Fig. 8.1, equal to

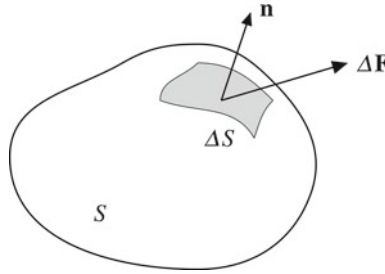


Fig. 8.1 Force $\Delta\mathbf{F}$ acting on an infinitesimal surface of area ΔS with outward normal \mathbf{n} . The limit of $\Delta\mathbf{F}/\Delta S$ as $\Delta S \rightarrow 0$ is defined as the traction

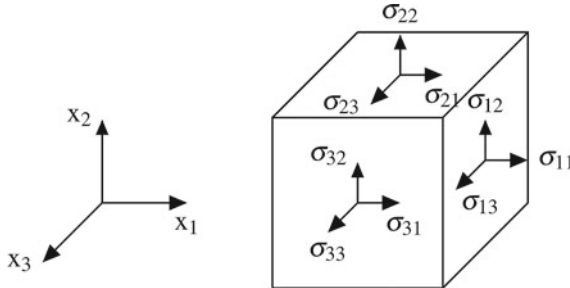


Fig. 8.2 Components of stress acting on the faces of an infinitesimal cube within a solid. The edges of the cube are aligned with the global Cartesian axes x_1, x_2, x_3 . For each stress component, σ_{ij} , the first index i denotes the face of the cube perpendicular to the x_i axis, and the second index j denotes the stress component acting on that face along the x_j axis

$$\mathbf{T}^{\mathbf{n}} = \lim_{\Delta S \rightarrow 0} \frac{\Delta\mathbf{F}}{\Delta S}$$

where $\Delta\mathbf{F}$ is the applied force and ΔS is the area of the infinitesimal surface element having outward normal \mathbf{n} . pressure ΔS , the traction is not necessarily aligned with \mathbf{n} , but can be in any direction.

Traction is also known as *the stress vector*, whose magnitude and direction depend on the orientation of the surface element on which it acts (hence the \mathbf{n} overscript in $\mathbf{T}^{\mathbf{n}}$). With this in mind, it is also possible to define an internal stress vector at any point within a solid for any arbitrary internal surface of normal \mathbf{n} through it. Defining an infinitesimal cube around an internal point, whose edges are aligned with the Cartesian coordinate axes x_1, x_2, x_3 , we can identify nine stress components σ_{ij} ($i, j = 1, 2, 3$) acting on the faces of such a cube, as shown in Fig. 8.2. These can be arranged to form a stress tensor, expressed in matrix form as

$$\boldsymbol{\sigma} = \begin{pmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{pmatrix}$$

also known as the *Cauchy stress*³ or *state of stress* at any given point within a solid medium.

Given the Cauchy stress tensor at any point, the stress vector acting on any imaginary internal surface of normal \mathbf{n} through that point is given by

$$\mathbf{T} = \boldsymbol{\sigma} \cdot \mathbf{n} \quad (T_i = \sigma_{ik}n_k) \tag{8.18}$$

From considerations of finite angular acceleration, we can show that the stress tensor must be symmetric: that is, $\sigma_{12} = \sigma_{21}$, $\sigma_{13} = \sigma_{31}$, and $\sigma_{23} = \sigma_{32}$. If the sidelength of the infinitesimal cube is Δx , we can determine the total right-hand moment around the x_1 axis (M_{x1}) through the centre of the cube (see Fig. 8.3) as follows:

$$\begin{aligned} M_{x1} &= \overbrace{\Delta^2 x (2\sigma_{23} - 2\sigma_{32})}^{\text{force components}} \times \overbrace{\left(\frac{\Delta x}{2}\right)}^{\text{distance to axis}} \\ &= (\sigma_{23} - \sigma_{32}) \Delta^3 x \end{aligned}$$

This moment is equal to the applied torque, which is in turn equal moment of inertia of the cube about the x_1 -axis multiplied by the angular acceleration. For a cube of density ρ , and sidelength Δx , its moment of inertia around the x_1 -axis is given by the volume integral:

$$\begin{aligned} I_{x1} &= \int_{-\Delta x/2}^{\Delta x/2} \int_{-\Delta x/2}^{\Delta x/2} \int_{-\Delta x/2}^{\Delta x/2} \rho r^2 dx_1 dx_2 dx_3 \\ &= \int_{-\Delta x/2}^{\Delta x/2} \int_{-\Delta x/2}^{\Delta x/2} \int_{-\Delta x/2}^{\Delta x/2} \rho (x_2^2 + x_3^2) dx_1 dx_2 dx_3 \\ &= \rho \frac{\Delta^5 x}{6} \end{aligned}$$

Hence the angular acceleration (α) is given by

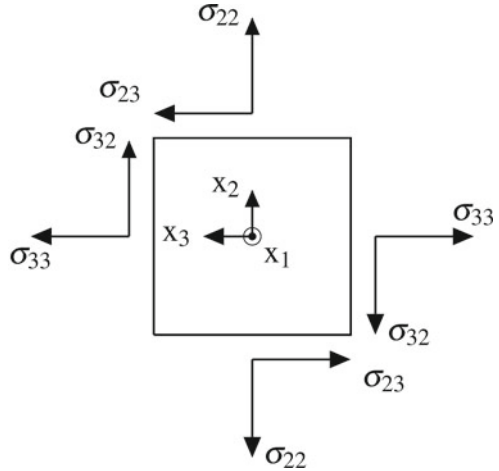
$$\alpha = \frac{M_{x1}}{I_{x1}} = \frac{6(\sigma_{23} - \sigma_{32})}{\rho \Delta^2 x}$$

In the infinitesimal limit as $\Delta x \rightarrow 0$, α will be infinitely large, unless $\sigma_{23} - \sigma_{32} = 0$. Hence, to maintain angular acceleration finite, we must have $\sigma_{23} = \sigma_{32}$. A similar analysis can be made for the other axes of the infinitesimal cube, leading to $\sigma_{13} = \sigma_{31}$ and $\sigma_{12} = \sigma_{21}$. Hence, the Cauchy stress tensor is symmetric.

Now that we have defined the stress tensor, we can formulate the PDE governing stress distribution within a solid using Newton’s second law of motion. To do so, we consider an arbitrary volume V in a solid medium enclosed by a surface S , as

³Augustin-Louis Cauchy (1789–1857), prolific French mathematician who contributed numerous works in mathematical analysis and the theory of elasticity.

Fig. 8.3 Cross-sectional view along x_1 -axis of stress components acting on the faces of infinitesimal cube of sidelength Δx



shown earlier in Fig. 8.1. In addition, we denote a body force acting per unit volume throughout the solid as \mathbf{f} . An example of such a body force is gravity, whereby $\mathbf{f} = \rho \mathbf{g}$, where ρ is the density and \mathbf{g} is the acceleration due to gravity. From the definition of the stress tensor, the total force acting on the surface S due to stress imparted by the surrounding medium is

$$\mathbf{F} = \int_S \boldsymbol{\sigma} \cdot \mathbf{n} \, dS = \int_S \boldsymbol{\sigma} \cdot d\mathbf{S}$$

Furthermore, the force acting on V due to the body force is

$$\mathbf{F} = \int_V \mathbf{f} \, dV$$

Hence, the total force acting on V is

$$\mathbf{F} = \int_S \boldsymbol{\sigma} \cdot d\mathbf{S} + \int_V \mathbf{f} \, dV$$

From Newton's second law, the total force acting on a body equals the time-derivative of its momentum. Denoting the displacement field of points within V as \mathbf{u} , the total momentum of V is given by

$$\mathbf{p} = \int_V \rho \frac{\partial \mathbf{u}}{\partial t} \, dV$$

Newton's second law of motion for V is then

$$\frac{d}{dt} \int_V \rho \frac{\partial \mathbf{u}}{\partial t} \, dV = \int_S \boldsymbol{\sigma} \cdot d\mathbf{S} + \int_V \mathbf{f} \, dV$$

Using the principle of conservation of mass, the total mass of V is constant, hence $\rho \, dV$ in the left-most integral above is independent of time. The time-derivative can then be transferred within the integral to form

$$\int_V \rho \frac{\partial^2 \mathbf{u}}{\partial t^2} \, dV = \int_S \boldsymbol{\sigma} \cdot d\mathbf{S} + \int_V \mathbf{f} \, dV$$

and making use of the divergence theorem, the surface integral $\int_S \boldsymbol{\sigma} \cdot d\mathbf{S}$ can be transformed to $\int_V \nabla \cdot \boldsymbol{\sigma} \, dV$. The entire equation therefore becomes

$$\begin{aligned} \int_V \rho \frac{\partial^2 \mathbf{u}}{\partial t^2} \, dV &= \int_V \nabla \cdot \boldsymbol{\sigma} \, dV + \int_V \mathbf{f} \, dV \\ &= \int_V (\nabla \cdot \boldsymbol{\sigma} + \mathbf{f}) \, dV \end{aligned}$$

Since this integral holds true for any arbitrary volume V with the solid medium, the integrands must be identically equal. Hence, we obtain *Cauchy's momentum equation* as

$$\rho \frac{\partial^2 \mathbf{u}}{\partial t^2} = \nabla \cdot \boldsymbol{\sigma} + \mathbf{f} \quad (8.19)$$

For a body at rest under equilibrium conditions, the left-hand side of Eq. 8.19 is zero, and we obtain the *elastostatics PDE*:

$$\nabla \cdot \boldsymbol{\sigma} + \mathbf{f} = 0 \quad (8.20)$$

8.3.2 Strain

The deformation of a solid body is described by strain. For a 1D object such as an elastic bar, strain may be simply defined as the change in length relative to the initial length, or

$$\varepsilon = \frac{L - L_0}{L_0} \quad (8.21)$$

where ε is the strain, L is the new length of the bar after a load is applied, and L_0 is its initial length. For 3D solids, the actual deformation can be far more complex than this example, with strain consisting of several components forming a second-order tensor quantity, as will be seen below.

To characterise the complex deformation of a solid, consider two closely-spaced material ‘particles’ in a solid body, originally located at points M and N , with spatial coordinates \mathbf{X} and $\mathbf{X} + d\mathbf{X}$ respectively, as shown in Fig. 8.4. After some deformation, these particles are now displaced to the new points M' , N' with coordinates \mathbf{x} and $\mathbf{x} + d\mathbf{x}$. We assume the deformation throughout the solid can be characterised by a

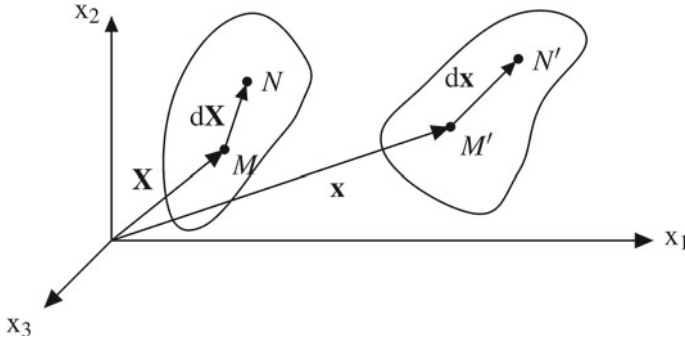


Fig. 8.4 Displacement of two material particles M, N to M', N' due to deformation of a solid body. The spatial coordinates of M prior to deformation are \mathbf{X} . After deformation, the spatial coordinates of the same material particle, now at position M' , are \mathbf{x} . Prior to deformation, particle N is offset from M by a distance $\Delta\mathbf{X}$. After deformation, particle N' is offset from M' a distance $\Delta\mathbf{x}$

continuous function $\mathbf{x}(\mathbf{X})$ that returns the new coordinates of the particle given its initial coordinates \mathbf{X} . The *deformation tensor* $\mathbf{F}(\mathbf{X})$ is defined as

$$\mathbf{F} = \left(\frac{\partial \mathbf{x}}{\partial \mathbf{X}} \right), \quad F_{ij} = \frac{\partial x_i}{\partial X_j}, \quad d\mathbf{x} = \mathbf{F}d\mathbf{X} \tag{8.22}$$

and its inverse

$$\mathbf{F}^{-1} = \left(\frac{\partial \mathbf{X}}{\partial \mathbf{x}} \right), \quad (\mathbf{F}^{-1})_{ij} = \frac{\partial X_i}{\partial x_j}, \quad d\mathbf{X} = \mathbf{F}d\mathbf{x} \tag{8.23}$$

If the original length of segment $M-N$ is denoted by L_0 , and the post-deformation length $M'-N'$ by L , then these square lengths are given by

$$\begin{aligned} L_0^2 &= d\mathbf{X}^T d\mathbf{X} \\ L^2 &= d\mathbf{x}^T d\mathbf{x} = d\mathbf{X}^T \mathbf{F}^T \mathbf{F} d\mathbf{X} \end{aligned}$$

with the change in square length being:

$$L^2 - L_0^2 = d\mathbf{X}^T (\mathbf{F}^T \mathbf{F} - \mathbf{I}) d\mathbf{X}$$

Defining a new tensor $\mathbf{C} = \mathbf{F}^T \mathbf{F}$ as the right Cauchy–Green⁴ deformation tensor, and denoting the directional vector of $d\mathbf{X}$ as \mathbf{m} , with $d\mathbf{X} = L_0 \mathbf{m}$, we have:

⁴Co-named after George Green (1793–1841), British mathematical physicist. We can also define the left Cauchy–Green deformation tensor as $\mathbf{B} = \mathbf{F}\mathbf{F}^T$.

$$\begin{aligned}
L^2 - L_0^2 &= d\mathbf{X}^T (\mathbf{F}^T \mathbf{F} - \mathbf{I}) d\mathbf{X} \\
&= L_0 \mathbf{m}^T (\mathbf{C} - \mathbf{I}) \mathbf{m} L_0 \\
\frac{L^2 - L_0^2}{L_0^2} &= \mathbf{m}^T (\mathbf{C} - \mathbf{I}) \mathbf{m}
\end{aligned} \tag{8.24}$$

where \mathbf{I} is the identity tensor. For small changes in length, we can take the Taylor expansion of the left-hand side in L about L_0 to yield the first-order approximation:

$$\frac{L^2 - L_0^2}{L_0^2} \approx \frac{2}{L_0} (L - L_0)$$

and substituting into Eq. 8.24, we obtain

$$\frac{L - L_0}{L} \approx \frac{1}{2} \mathbf{m}^T (\mathbf{C} - \mathbf{I}) \mathbf{m}$$

Comparing this expression to Eq. 8.21, we can define the analogous generalized strain measure, known as *Green's strain tensor*, as

$$\mathbf{E} = \frac{1}{2} (\mathbf{C} - \mathbf{I}) \tag{8.25}$$

Recalling the definition of \mathbf{F} from Eq. 8.22 as

$$\mathbf{F} = \frac{\partial \mathbf{x}}{\partial \mathbf{X}} = \nabla_{\mathbf{x}}$$

and defining the displacement vector field throughout the solid body as

$$\mathbf{u} = \mathbf{x} - \mathbf{X} \tag{8.26}$$

we obtain:

$$\begin{aligned}
\mathbf{F} &= \nabla_{\mathbf{x}} = \nabla (\mathbf{X} + \mathbf{u}) \\
&= \mathbf{I} + \nabla_{\mathbf{u}}
\end{aligned}$$

which can be substituted into Green's strain tensor (Eq. 8.25) to yield

$$\begin{aligned}
\mathbf{E} &= \frac{1}{2} (\mathbf{C} - \mathbf{I}) \\
&= \frac{1}{2} (\mathbf{F}^T \mathbf{F} - \mathbf{I}) \\
&= \frac{1}{2} (\nabla_{\mathbf{u}} + \nabla_{\mathbf{u}}^T + \nabla_{\mathbf{u}}^T \nabla_{\mathbf{u}})
\end{aligned}$$

which can be written in indicial notation from the components of displacement u_i as

$$E_{ij} = \frac{1}{2} \left(\frac{\partial u_i}{\partial X_j} + \frac{\partial u_j}{\partial X_i} + \frac{\partial u_i}{\partial X_j} \frac{\partial u_j}{\partial X_i} \right) \quad (8.27)$$

In linear elasticity theory, strains are assumed to be small ($\nabla \mathbf{u} \ll 1$). Green's strain tensor can then be approximated by *Cauchy's infinitesimal strain tensor* $\boldsymbol{\varepsilon}$ as

$$\boldsymbol{\varepsilon} = \frac{1}{2} (\nabla \mathbf{u} + \nabla \mathbf{u}^T) \quad (8.28)$$

which can be expressed using indicial notation as

$$\varepsilon_{ij} = \frac{1}{2} \left(\frac{\partial u_i}{\partial X_j} + \frac{\partial u_j}{\partial X_i} \right) \quad (8.29)$$

However in biomechanics, we often deal with large deformations. Green's strain (Eqs. 8.25 and 8.27) is therefore used by default in such applications.

Finally, if dV_0 denotes the original volume of an infinitesimal material element, and dV is its corresponding volume after deformation, then we can define the *volume strain* as

$$\frac{dV}{dV_0} = J = \det \mathbf{F} \quad (8.30)$$

This relationship can be verified by noting that the volume of a parallelepiped with edge vectors \mathbf{a} , \mathbf{b} , \mathbf{c} is given by

$$V = \mathbf{a} \cdot (\mathbf{b} \times \mathbf{c}) = \begin{vmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{vmatrix} = |\mathbf{a} \ \mathbf{b} \ \mathbf{c}|$$

Hence an infinitesimal element with edges given by $d\mathbf{a}$, $d\mathbf{b}$, $d\mathbf{c}$ has a volume

$$dV_0 = |d\mathbf{a} \ d\mathbf{b} \ d\mathbf{c}|$$

On deformation, these edges are transformed (using Eq. 8.22) into $\mathbf{F}d\mathbf{a}$, $\mathbf{F}d\mathbf{b}$, $\mathbf{F}d\mathbf{c}$. Hence the new volume after deformation is

$$\begin{aligned} dV &= |\mathbf{F}d\mathbf{a} \ \mathbf{F}d\mathbf{b} \ \mathbf{F}d\mathbf{c}| \\ &= |\mathbf{F}| |d\mathbf{a} \ d\mathbf{b} \ d\mathbf{c}| \\ &= \det \mathbf{F} \ dV_0 \end{aligned}$$

from which we readily obtain Eq. 8.30. Note that for incompressible solids, there is no volume change anywhere in the material: consequently $\det \mathbf{F} = 1$.

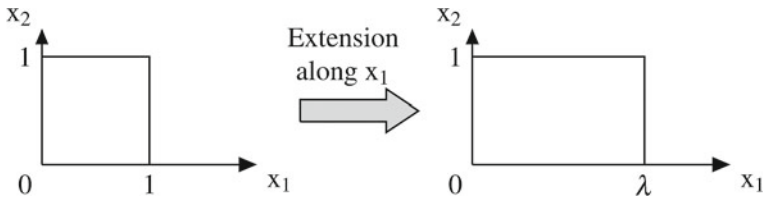


Fig. 8.5 Example of 2D deformation by simple extension along x_1

Example 8.1 A 2D square of sidelength 1 is stretched along the x_1 -axis to a new sidelength of λ , as shown in Fig. 8.5. Determine the 2D Cauchy and Green strain tensors.

Answer: To find the strain, we first need to determine the displacement field. The simplest such field is a linear interpolation between the boundaries along x_1 , noting also that there is no displacement along x_2 , according to:

$$\begin{aligned} u_1 &= (\lambda - 1) x_1 \\ u_2 &= 0 \end{aligned}$$

The components of the Cauchy strain are therefore:

$$\varepsilon_{11} = \frac{1}{2} \left(\frac{\partial u_1}{\partial x_1} + \frac{\partial u_1}{\partial x_1} \right) = \lambda - 1, \quad \varepsilon_{12} = \varepsilon_{21} = \varepsilon_{22} = 0$$

Note that in 2D, there are only 4 components of strain. The corresponding Green strain components are determined from:

$$\begin{aligned} E_{11} &= \frac{1}{2} \left(\frac{\partial u_1}{\partial x_1} + \frac{\partial u_1}{\partial x_1} + \frac{\partial u_1}{\partial x_1} \frac{\partial u_1}{\partial x_1} \right) \\ &= \frac{1}{2} (2(\lambda - 1) + (\lambda - 1)^2) \\ &= \frac{1}{2} ((\lambda - 1)(2 + \lambda - 1)) \\ &= \frac{1}{2} ((\lambda - 1)(\lambda + 1)) \\ &= \frac{1}{2} (\lambda^2 - 1) \\ E_{12} &= E_{21} = E_{22} = 0 \end{aligned}$$

In matrix form, these strain tensors are written as:

$$\boldsymbol{\varepsilon} = \begin{pmatrix} \lambda - 1 & 0 \\ 0 & 0 \end{pmatrix} \quad \mathbf{E} = \begin{pmatrix} \frac{1}{2} (\lambda^2 - 1) & 0 \\ 0 & 0 \end{pmatrix}$$

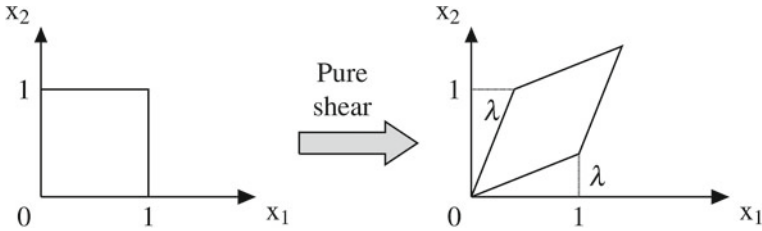


Fig. 8.6 Example of 2D deformation by pure shear

It can be seen that the diagonal components of the strain correspond to the components of pure extension. The off-diagonal components of strain correspond to another type of deformation - *shear* - as will be seen in the next example. \square

Example 8.2 A 2D square of sidelength 1 is subjected to the pure shear deformation shown in Fig. 8.6. Determine the 2D Cauchy and Green strain tensors.

Answer: The displacement field for this deformation may be written as:

$$u_1 = \lambda x_2$$

$$u_2 = \lambda x_1$$

The components of the Cauchy strain are therefore:

$$\varepsilon_{11} = \frac{1}{2} \left(\frac{\partial u_1}{\partial x_1} + \frac{\partial u_1}{\partial x_1} \right) = 0$$

$$\varepsilon_{12} = \frac{1}{2} \left(\frac{\partial u_2}{\partial x_1} + \frac{\partial u_1}{\partial x_2} \right) = \lambda$$

$$\varepsilon_{21} = \frac{1}{2} \left(\frac{\partial u_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1} \right) = \lambda$$

$$\varepsilon_{22} = \frac{1}{2} \left(\frac{\partial u_2}{\partial x_2} + \frac{\partial u_2}{\partial x_2} \right) = 0$$

The corresponding Green strain components are:

$$E_{11} = \frac{1}{2} \left(\frac{\partial u_1}{\partial x_1} + \frac{\partial u_1}{\partial x_1} + \frac{\partial u_1}{\partial x_1} \frac{\partial u_1}{\partial x_1} \right) = 0$$

$$E_{12} = \frac{1}{2} \left(\frac{\partial u_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1} + \frac{\partial u_1}{\partial x_2} \frac{\partial u_2}{\partial x_1} \right) = \lambda + \frac{1}{2} \lambda^2$$

$$E_{21} = \frac{1}{2} \left(\frac{\partial u_2}{\partial x_1} + \frac{\partial u_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1} \frac{\partial u_1}{\partial x_2} \right) = \lambda + \frac{1}{2} \lambda^2$$

$$E_{22} = \frac{1}{2} \left(\frac{\partial u_2}{\partial x_2} + \frac{\partial u_2}{\partial x_2} + \frac{\partial u_2}{\partial x_2} \frac{\partial u_2}{\partial x_2} \right) = 0$$

In matrix form, these strain tensors are therefore:

$$\boldsymbol{\varepsilon} = \begin{pmatrix} 0 & \lambda \\ \lambda & 0 \end{pmatrix} \quad \mathbf{E} = \begin{pmatrix} 0 & \lambda + \frac{1}{2}\lambda^2 \\ \lambda + \frac{1}{2}\lambda^2 & 0 \end{pmatrix}$$

It can be seen that the off-diagonal components correspond to the components of pure shear. All deformations therefore are a combination of stretch (i.e. extension) and shear components. \square

8.4 Linear Elasticity

In order to solve for the deformation of a solid body under an applied external load, Cauchy's momentum equation (Eq. 8.19) or its elastostatics stationary form (Eq. 8.20) are insufficient. What is required is a formulation linking stress to strain in a given material. Such a formulation is known as a *constitutive law*. For materials exhibiting *linear elasticity*, the constitutive law follows the generalised Hooke's law⁵ which linearly relates the stresses to the infinitesimal strains via a tensor of elastic moduli \mathbf{C} :

$$\sigma_{ij} = C_{ijmn}\varepsilon_{mn} \quad (8.31)$$

where C_{ijmn} is a fourth-order tensor with 81 constants specific to the material. If the material is isotropic, these 81 material constants can be reduced to just two independent parameters, λ and μ , also known as *Lamé's constants*,⁶ with constitutive law given by:

$$\sigma_{ij} = \lambda\varepsilon_{\alpha\alpha}\delta_{ij} + 2\mu\varepsilon_{ij} \quad (8.32)$$

where we have used the indicial notation to signify $\varepsilon_{\alpha\alpha} = \text{trace}(\boldsymbol{\varepsilon})$, and δ_{ij} is the Kronecker delta. Equation 8.32 describes the constitutive law for an isotropic Hookean elastic solid, with λ known as Lamé's first parameter and μ often referred to as the shear modulus. Another way of writing this constitutive law is:

$$\varepsilon_{ij} = \frac{1+\nu}{E}\sigma_{ij} - \frac{\nu}{E}\sigma_{\alpha\alpha}\delta_{ij} \quad (8.33)$$

where E is *Young's modulus*⁷ (or modulus of elasticity), and ν is *Poisson's ratio*⁸.

⁵Robert Hooke, 1635–1703, English scientist, mathematician and architect who made numerous contributions to the fields of physics, biology, astronomy and mechanics.

⁶Named after the French mathematician Gabriel Léon Jean Baptiste Lamé (1795–1870), who made important contributions to the mathematical theory of elasticity.

⁷Thomas Young (1773–1829), English scientist who made contributions to many fields, including solid mechanics.

⁸Siméon Denis Poisson (1781–1840), French mathematician and physicist.

In order to understand the significance of the various terms in Eq. 8.33, we note firstly that stresses and strains can be separated into *hydrostatic* and *deviatoric* components, such that the hydrostatic stress is akin to an internal pressure in the medium, containing only equal diagonal entries and the hydrostatic strains containing only equal diagonal terms that simply scale the shape of the material element. That is,

$$\begin{aligned}\boldsymbol{\sigma} &= \boldsymbol{\sigma}^H + \boldsymbol{\sigma}^D, & \sigma_{ij}^H &= \frac{1}{3}\sigma_{\alpha\alpha}\delta_{ij} \\ \boldsymbol{\varepsilon} &= \boldsymbol{\varepsilon}^H + \boldsymbol{\varepsilon}^D, & \varepsilon_{ij}^H &= \frac{1}{3}\varepsilon_{\alpha\alpha}\delta_{ij}\end{aligned}$$

Note (using indicial summation) that $\sigma_{\alpha\alpha}^H = \sigma_{\alpha\alpha}$ and $\sigma_{\alpha\alpha}^D = 0$, with similar identities for the strain tensor. Splitting the stress into these components, Eq. 8.33 can be written as:

$$\begin{aligned}\varepsilon_{ij} &= \frac{1+\nu}{E}\sigma_{ij} - \frac{3\nu}{E}\sigma_{ij}^H \\ &= \frac{1+\nu}{E}(\sigma_{ij}^H + \sigma_{ij}^D) - \frac{3\nu}{E}\sigma_{ij}^H \\ &= \frac{1-2\nu}{E}\sigma_{ij}^H + \frac{1+\nu}{E}\sigma_{ij}^D\end{aligned}\tag{8.34}$$

The hydrostatic component of strain can then be determined from

$$\begin{aligned}\varepsilon_{ij}^H &= \frac{1}{3}\varepsilon_{\alpha\alpha}\delta_{ij} \\ &= \frac{1}{3}\left(\frac{1-2\nu}{E}\sigma_{\alpha\alpha}^H + \frac{1+\nu}{E}\sigma_{\alpha\alpha}^D\right)\delta_{ij} \\ &= \frac{1-2\nu}{E}\sigma_{ij}^H\end{aligned}\tag{8.35}$$

where we have used $\sigma_{\alpha\alpha}^D = 0$. This equation shows that hydrostatic stresses and strains are proportional to each other in an isotropic Hookean elastic material. Note that when Poisson's ratio $\nu = 0.5$, the solid will be incompressible, since the hydrostatic strain components will be zero regardless of the hydrostatic pressure. To find the deviatoric strain, we subtract the hydrostatic component from Eq. 8.34 to obtain:

$$\begin{aligned}\varepsilon_{ij}^D &= \varepsilon_{ij} - \varepsilon_{ij}^H \\ &= \left(\frac{1-2\nu}{E}\sigma_{ij}^H + \frac{1+\nu}{E}\sigma_{ij}^D\right) - \left(\frac{1-2\nu}{E}\sigma_{ij}^H\right) \\ &= \frac{1+\nu}{E}\sigma_{ij}^D\end{aligned}\tag{8.36}$$

which shows that the deviatoric stresses and strains are also proportional to each other.

Most materials are well able to withstand hydrostatic stresses and strains, but will fail when the deviatoric stresses are too high. Since stress is a tensor quantity, it

would be useful if a single representative scalar measure of stress could be used as an indicator of high deviatoric stress levels. One such representative stress indicator is the *von Mises stress*.⁹ To derive this stress measure, consider the work done per unit volume in deforming a solid, known as the *strain energy*, determined from

$$W = \int \sigma_{ij} d\varepsilon_{ij} \quad (8.37)$$

which for a Hookean elastic material evaluates to

$$W = \frac{1}{2} \sigma_{ij} \varepsilon_{ij} \quad (8.38)$$

Substituting in the hydrostatic and deviatoric components of stress and strain, the above becomes:

$$\begin{aligned} W &= \frac{1}{2} (\sigma_{ij}^H + \sigma_{ij}^D) (\varepsilon_{ij}^H + \varepsilon_{ij}^D) \\ &= \frac{1}{2} \sigma_{ij}^H \varepsilon_{ij}^H + \frac{1}{2} \sigma_{ij}^H \varepsilon_{ij}^D + \frac{1}{2} \sigma_{ij}^D \varepsilon_{ij}^H + \frac{1}{2} \sigma_{ij}^D \varepsilon_{ij}^D \\ &= \frac{1}{2} \sigma_{ij}^H \varepsilon_{ij}^H + \frac{1}{2} \sigma_{ij}^D \varepsilon_{ij}^D \end{aligned}$$

where we have used the fact that the scalar product $A_{ij}B_{ij}$ of any hydrostatic and deviatoric tensor is equal to zero (see Problem 8.1b). This equation shows that the total strain energy can be separated into hydrostatic and deviatoric components. Examining the deviatoric strain energy W^D component, we can write:

$$\begin{aligned} W^D &= \frac{1}{2} \sigma_{ij}^D \varepsilon_{ij}^D \\ &= \frac{1}{2} \sigma_{ij}^D \left(\frac{1+\nu}{E} \sigma_{ij}^D \right) \quad (\text{from Eq. 8.36}) \\ &= \frac{1+\nu}{2E} \sigma_{ij}^D \sigma_{ij}^D \end{aligned}$$

We can choose a representative scalar stress value σ_{rep} such that $W^D = \frac{1+\nu}{2E} \sigma_{\text{rep}}^2$. That is,

$$\sigma_{\text{rep}}^2 = \sigma_{ij}^D \sigma_{ij}^D$$

Now consider the case of a uniaxial stress in a solid:

$$\boldsymbol{\sigma} = \begin{pmatrix} \sigma_a & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

⁹Named after Richard Edler von Mises (1883–1953), German scientist and mathematician who made important contributions to the fields of solid mechanics, fluid mechanics, statistics and probability theory.

In this case, the hydrostatic and deviatoric stresses are given by

$$\boldsymbol{\sigma}^H = \begin{pmatrix} \sigma_a/3 & 0 & 0 \\ 0 & \sigma_a/3 & 0 \\ 0 & 0 & \sigma_a/3 \end{pmatrix} \quad \boldsymbol{\sigma}^D = \begin{pmatrix} 2\sigma_a/3 & 0 & 0 \\ 0 & -\sigma_a/3 & 0 \\ 0 & 0 & -\sigma_a/3 \end{pmatrix}$$

with the representative stress given by

$$\begin{aligned} \sigma_{\text{rep}}^2 &= \frac{4\sigma_a^2}{9} + \frac{\sigma_a^2}{9} + \frac{\sigma_a^2}{9} \\ &= \frac{6\sigma_a^2}{9} \\ \therefore \sigma_{\text{rep}} &= \sigma_a \sqrt{\frac{2}{3}} \end{aligned}$$

It would, however, be more intuitive if the representative stress for this uniaxial case was σ_a instead of the above. Scaling the representative stress by $\sqrt{3/2}$ yields the von Mises stress value:

$$\sigma_{VM} = \sqrt{\frac{3}{2} \sigma_{ij}^D \sigma_{ij}^D} \quad (8.39)$$

When simulating structural mechanics problems in COMSOL, the default plot produced is typically the von Mises stress distribution.

8.4.1 Example: Detecting Tension in a Respirator Strap

Many artificial respirators use a face mask to deliver oxygen, with the mask attached using straps placed around the head. In order to improve mask design, it is useful to measure tension in the strap using a simple steel three-pronged sensor shown in Fig. 8.7 that can be easily applied to the strap in-situ. The middle prong contains a strain gauge whose electrical resistance changes in proportion to the change in length of the gauge. As the strap tension increases, the middle prong deflects, leading to a measurable response in the strain gauge.

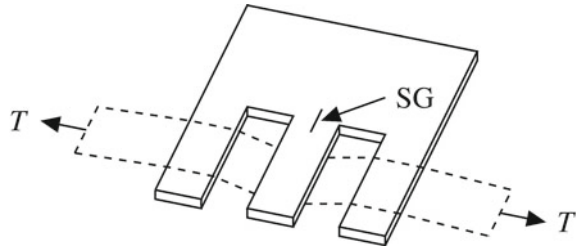
Using COMSOL, determine the change in strain gauge length as a function of strap tension T from 0 to 100 N, assuming that the strain gauge is positioned in the centre of the device. Also, for a strap tension of $T = 50$ N, determine the optimal placement of the strain gauge along the central device axis that produces a maximum change in strain gauge length.


Answer: To model this device in COMSOL, we can implement the following steps:

Model Wizard

1. Open the Model Wizard and select the 3D spatial dimension.

Fig. 8.7 Strap tension measurement device with strain gauge SG. The strap, with tension T , is shown as a dashed outline. Overall dimensions of the device are $50 \times 40 \times 0.4$ mm, with each fork (or prong) being 10 mm wide and 20 mm deep. The strain gauge is 5 mm in length







2. In the Select Physics panel, choose Structural Mechanics|Solid Mechanics, and click “Add”.
3. Click the Study arrow () to open the Select Study panel. Select Stationary, and click “Done”. This will exit the Model Wizard, displaying the main COMSOL interface.

Global Definitions

1. Right-click Global Definitions in the model tree and select Parameters. Enter the following two parameters in the Parameters table of the Settings window:

| Name | Expression | Description |
|------|------------|---------------|
| L | 17.5 [mm] | SG position |
| T | 50 [N] | Strap tension |

Geometry

1. Select the Geometry 1 node in the model tree. In the settings window, change the length unit to ‘mm’.
2. Right-click Geometry 1 and select Work Plane. Leave all settings to their default values and click Build Selected (). This creates an xy -workplane located at $z = 0$ mm.
3. Right-click the Plane Geometry sub-node of the newly-created Work Plane 1 node and select Polygon. Specify the width as 1 cm and the height as 10 cm. Click Build Selected (). In the settings window, specify the vertex coordinates by entering xw values of 0 0 25 25 20 20 10 10 (include spaces between the entries) and yw values of 0 40 40 0 0 20 20 0. Click Build Selected () to produce a plot of the 2D geometry in the work plane.
4. Right-click Geometry 1 and select Extrude. In the settings window, specify the Distances from Plane as 0.4 mm. The workplane $wp1$ object will have already been specified by default. Click Build Selected () to extrude the work plane geometry into 3D.
5. Right-click Geometry 1 again and select More Primitives| Point. In the settings window, specify the x -values as 25, 25, the y -values as L, L+(5 [mm]),

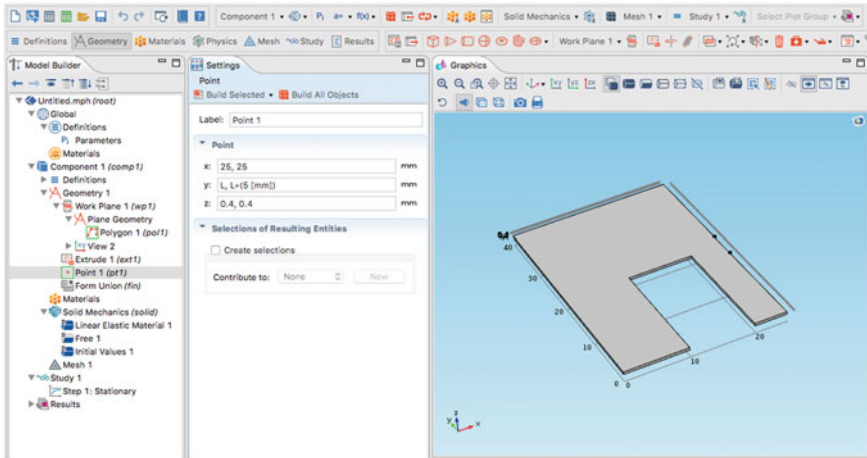


Fig. 8.8 COMSOL interface for strap tension device example, showing geometry and model tree. Note that due to device symmetry, only half of the geometry is implemented

and the z-values as 0.4 , 0.4 mm. These define two points marking the extents of the strain gauge. Click Build Selected (). Once complete, the geometry and model tree will look like that shown in Fig. 8.8.

Component Definitions

1. Right-click the Definitions sub-node of Component 1 and select Component Couplings| Integration. In the Settings window, specify the Geometric entity level as 'Edge', and select edge 24, corresponding to the strain gauge. Leave the Frame setting to its default Spatial (x, y, z). This defines an integration operator, intop1, acting over the strain gauge length. It will be used to determine the strain gauge length during deformation.
2. Right-click the Definitions sub-node of Component 1 again and select Component Couplings| Integration a second time. In the Settings window, specify the Geometric entity level as 'Point', and select point 6, corresponding to the upper inner vertex of the outer device prong. As above, leave the Frame setting to its default Spatial (x, y, z) value. This defines an integration operator, intop2, acting over point 6 - this will be used to determine the displacement due to the strap and hence the tension transmitted to the prongs of the device.
3. Right-click the Definitions sub-node of Component 1 a third time and select Component Couplings| Integration again. As above, specify the Geometric entity level as 'Point', and this time select point 9, corresponding to the lower vertex of the inner prong. Leave the Frame setting to its default Spatial (x, y, z) value. This defines an integration operator, intop3, acting over point 9, which will also be used to determine the tension transmitted to the prongs of the device.

- Right-click the Definitions sub-node of Component 1 a final time and select Variables. Leave the geometric entity level to its default as ‘Entire model’, and enter the following variables in the settings table:

| Name | Expression |
|------|--|
| SG | $\text{intop1}(1)$ |
| w1 | $\text{intop2}(w)$ |
| w2 | $\text{intop3}(w)$ |
| F | $T * (w1 - w2 + (0.4 \text{ [mm]})) / (10 \text{ [mm]})$ |

Note that the F variable corresponds to the vertical projection of the strap tension, the latter being diagonally oriented between upper and lower edges on the outer and middle prongs separated horizontally by 10 mm and vertically by 0.4 mm.

Materials

- Right-click the Materials node of the model tree and select Add Material. Expand the list of in-built materials and select ‘Steel AISI 4300’. Click ‘Add to Component’ to add this material to the model. This feature allows ready access to a range of pre-defined material parameters to be used in various physics applications.

Solid Mechanics

- Select the Linear Elastic Material 1 sub-node of Solid Mechanics, and leave all material parameters to their default setting of “From material”.
- Right-click Solid Mechanics and select the Fixed Constraint boundary condition. In the settings panel, specify boundary 5 corresponding to the rear boundary of the device, forcing it to be fixed in place.
- Right-click Solid Mechanics again and select Edges| Edge Load. In the Settings window, select edge 16 (i.e. the lower edge on the middle prong), and specify the x, y, z components of load force per unit length as 0, 0, $F / (20 \text{ [mm]})$ respectively. This assumes that the tension from the strap is transmitted over the entire length of this edge.
- As above, right-click Solid Mechanics again and select Edges| Edge Load. Specify edge 11 (i.e. the upper inner edge on the outer prong), and specify the x, y, z components of load force per unit length as 0, 0, $-F / (20 \text{ [mm]})$ respectively.
- Right-click Solid Mechanics a final time and select More Constraints| Symmetry. Specify boundary 10 as the symmetric boundary.

Study

- Right-click the Study node and select Parametric Sweep. In the Settings panel, click the Add parameter button (+) and specify parameter T. Click the Range button (📏) and specify start, step and stop values of 0, 5, and 100 respectively. For the parameter units, enter N.

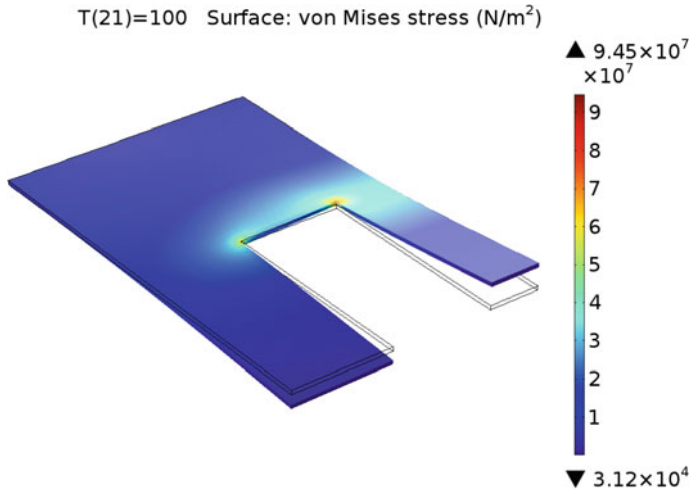


Fig. 8.9 von Mises stress distribution in deformed device for a strap tension level of 100 N. The deformation shown has been increased by a default scale factor of ≈ 9.7 to visualise the displacements more clearly. The original device edges prior to deformation are outlined in *black*

2. Select the Step 1: Stationary sub-node of Study 1 and click the ‘Include geometric nonlinearity’ checkbox. Click Compute (=) to solve the model. Select the Progress tab in the Information window to see the solver progress.

Results

1. When the model has completed solving, the Graphics window will display a default plot of the von Mises stress in the deformed solid at the final parameter value of $T = 100$ N, similar to that shown in Fig. 8.9.
2. To visualise the full 3D symmetric model solution at this strap tension, click the Data Sets sub-node of Results and select More Data Sets| Mirror 3D. In the settings panel, leave the default data set as Study1/Solution 1, as well as quick YZ-plane for the mirror plane. For the X-coordinate, specify a value of 25 mm. Now right-click the Results node and select 3D Plot Group. In the settings panel, specify the data set as Mirror 3D 1. Right-click the newly-created 3D Plot Group 2 node and select Surface. In the settings panel, specify the expression to plot as `solid.mises`, corresponding to the von Mises stress. Right-click the Surface 1 sub-node of 3D Plot Group 2 and select Deformation. The resulting full-geometry visualization is shown in Fig. 8.10.
3. To plot the change in strain gauge length against strap tension, right-click the Results node and select ‘1D Plot Group’. Right-click the 1D Plot Group 3 sub-node and select ‘Global’. Under the y-Axis Data tab, specify `SG-(5 [mm])` as the expression to plot, with units of ‘nm’.
4. Select the 1D Plot Group 3 sub-node again, and in the Plot Settings window, check the ‘x-axis label’ and ‘y-axis label’ checkboxes. Specify the x-axis label

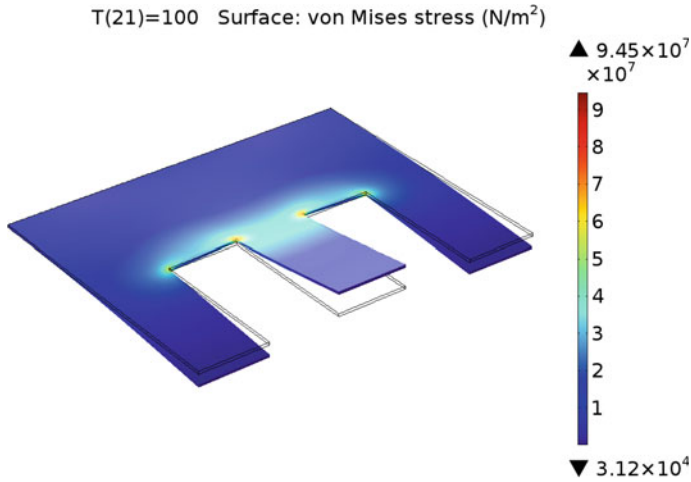


Fig. 8.10 von Mises stress distribution for a strap tension of 100 N visualised in the full symmetric 3D model representation, even though only half this geometry was actually used to solve the model. All deformations have been magnified by a scale factor of ≈ 13.4

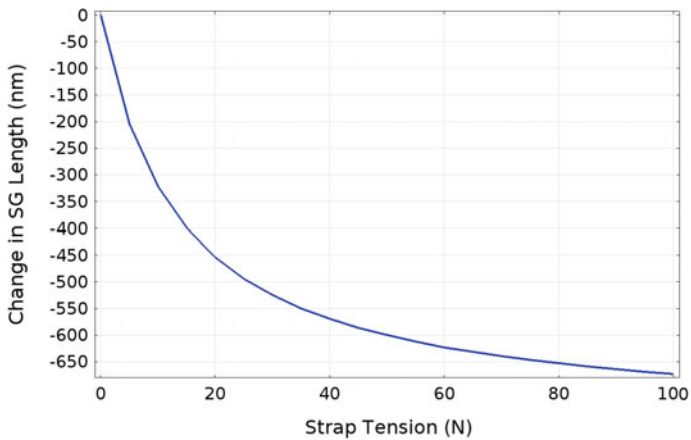



Fig. 8.11 Plot of change in strain gauge (SG) length against strap tension. Note that the overall change in length is negative, indicating a reduction of length due to local compression of the material at the strain gauge

as ‘Strap Tension (N)’ and the y-axis label as ‘Change in SG Length (nm)’. Click the Plot button () to display the graph, as shown in Fig. 8.11.

Add Study

1. To determine the optimal placement of the strain gauge, right-click the root node of the model tree and select Add Study. Specify Stationary and click ‘Add Study’.

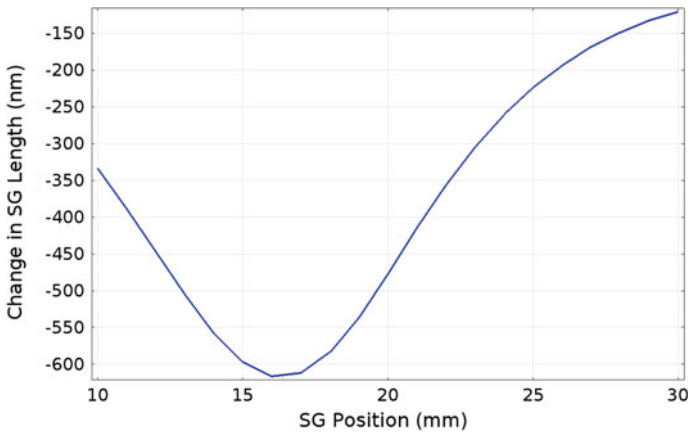


Fig. 8.12 Plot of change in strain gauge (SG) length against SG Position. The greatest change in strain gauge length occurs at a position of 16 mm from the end of the middle prong

A newly-created Study 2 node will appear in the model tree. In the Settings panel for Study 2, click the ‘Include geometric nonlinearity’ checkbox.

2. Right-click Study 2 and select Parametric Sweep. In the Settings panel, click the Add parameter button (+) and specify parameter L. Click the Range button (▭) and specify start, step and stop values of 10, 1, and 30 respectively. For the parameter units, enter mm.
3. To solve the model again, right-click Study 2 and select compute (=). Select the Progress tab in the Information window to see the solver progress.
4. When the model has finished solving, right-click the Results node and select ‘1D Plot Group’. Specify the Data set to be ‘Study 2/Parametric Solutions 1’. Right-click the 1D Plot Group 5 sub-node and select ‘Global’. Under the y-Axis Data tab, specify $SG - (5 \text{ [mm]})$ as the expression to plot, with units of ‘nm’.
5. Select the 1D Plot Group 5 sub-node again, and in the Plot Settings window, check the ‘x-axis label’ and ‘y-axis label’ checkboxes. Specify the x-axis label as ‘SG Position (mm)’ and the y-axis label as ‘Change in SG Length (nm)’. Click the Plot button (🖨) to display the graph, as shown in Fig. 8.12. It can be seen that the optimal position for the strain gauge, producing the greatest change in its length, is 16 mm from the end of the prong.

8.5 Linear Viscoelasticity

Viscoelastic materials, including most biological tissues, exhibit a time-dependent response in their stress or strain response to an applied load, even if this load remains constant. In a linear viscoelastic material, the deviatoric stress depends linearly on the deviatoric strain and its time derivatives. One common model of linear viscoelasticity

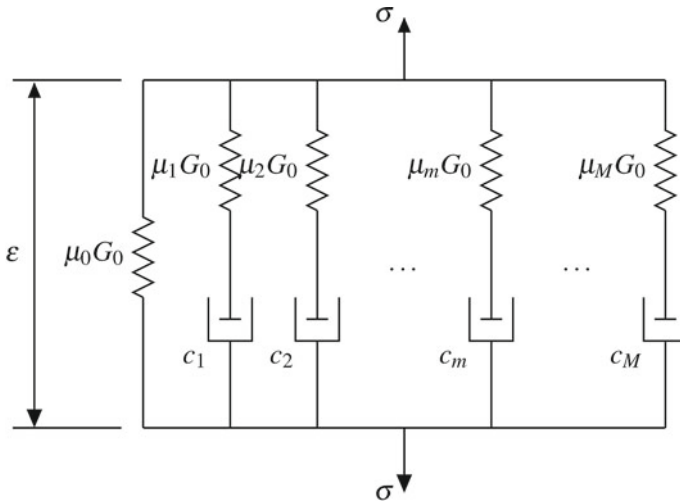


Fig. 8.13 Generalized Maxwell model with M branches)

is the generalized Maxwell model,¹⁰ as shown in Fig. 8.13, employing a number of parallel branches, each consisting of a linear spring in series with a dashpot. For each spring element, stress is proportional to strain, whereas for each dashpot, stress is proportional to the time-derivative of strain across it. The overall relationship between deviatoric stress and strain for this model satisfies

$$\sigma^D + \sum_{m=1}^M c_m \frac{\partial \sigma^D}{\partial t} = \mu_0 G_0 \epsilon^D + \sum_{m=1}^M \mu_m G_0 c_m \frac{\partial \epsilon^D}{\partial t}, \quad \sum_{m=1}^M \mu_m = 1$$

where G_0 , μ_m and c_m are material parameters, and M is the total number of parallel branches in the Maxwell model.

8.6 Hyperelastic Materials

Hyperelastic materials are defined by the existence of a strain energy function W , analogous to Eq. 8.37, as

$$\sigma_{ij}^D = \frac{\partial W}{\partial E_{ij}} \tag{8.40}$$

where W is a function of the Green strain components E_{11} , E_{22} , etc. For example, an elastic solid whose stress components are linear functions of the strains would have

¹⁰After James Clerk Maxwell who, in addition to formulating his famous equations of electromagnetism (Eqs. 6.1–6.4), also made important contributions to structural mechanics.

a strain energy function of the form

$$W = \frac{q}{2}$$

$$q = a_1 E_{11}^2 + a_2 E_{22}^2 + a_3 E_{33}^2 + 2a_4 E_{11} E_{22} + 2a_5 E_{22} E_{33} + 2a_6 E_{33} E_{11} + a_7 E_{12}^2 + a_8 E_{23}^2 + a_9 E_{31}^2$$

where a_1, a_2, \dots, a_9 are material parameters. For isotropic hyperelastic materials, whose properties are independent of direction, the strain energy must be of the form:

$$W = W(I_1, I_2, I_3)$$

where I_1, I_2, I_3 are the strain tensor invariants (see Eqs. 8.15 and 8.16), since the strain energy of such a material is independent of coordinate rotation. We recall from Sect. 8.2.4 that the principal values of the strain tensor are also invariants. Denoting these principal strains by E_1, E_2 and E_3 , we note that it is possible to rotate the reference coordinate system so that the strain tensor contains only these diagonal terms, and is equal to

$$\bar{\mathbf{E}} = \begin{pmatrix} E_1 & 0 & 0 \\ 0 & E_2 & 0 \\ 0 & 0 & E_3 \end{pmatrix}$$

Since the stress tensor is related to the deformation tensor \mathbf{F} by $\bar{\mathbf{E}} = \frac{1}{2} (\bar{\mathbf{F}}^T \bar{\mathbf{F}} - \mathbf{I})$, we readily deduce that

$$\bar{\mathbf{F}} = \begin{pmatrix} \sqrt{2E_1 + 1} & 0 & 0 \\ 0 & \sqrt{2E_2 + 1} & 0 \\ 0 & 0 & \sqrt{2E_3 + 1} \end{pmatrix}$$

The diagonal elements of $\bar{\mathbf{F}}$ are known as the *principal stretch ratios*, and are given by

$$\lambda_1 = \sqrt{2E_1 + 1}$$

$$\lambda_2 = \sqrt{2E_2 + 1}$$

$$\lambda_3 = \sqrt{2E_3 + 1}$$

and are also invariant under coordinate rotation. We can therefore define a new set of strain invariants as functions of the above principal stretches according to:

$$\bar{I}_1 = \lambda_1^2 + \lambda_2^2 + \lambda_3^2$$

$$\bar{I}_2 = \lambda_1^{-2} + \lambda_2^{-2} + \lambda_3^{-2}$$

$$\bar{I}_3 = \lambda_1 \lambda_2 \lambda_3$$

For incompressible materials, we note from Eq. 8.30 that $\det \mathbf{F} = 1$. From the above,

$$\det \bar{\mathbf{F}} = \begin{vmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{vmatrix} = 1$$

$$\therefore \lambda_1 \lambda_2 \lambda_3 = 1 = \bar{I}_3$$

Hence, for incompressible materials, $\bar{I}_3 = 1$. The strain energy function will depend only on \bar{I}_1 and \bar{I}_2 . An example is the *Mooney–Rivlin*¹¹ strain energy function used to describe materials such as rubber:

$$W = C_1 (\bar{I}_1 - 3) + C_2 (\bar{I}_2 - 3) \quad (8.41)$$

where C_1, C_2 are material constants.¹² In COMSOL, the incompressibility constraint is handled by adding an additional volumetric strain term to the strain energy function according to:

$$W_{\text{vol}} = \frac{1}{2} K (J - 1)^2 \quad (8.42)$$

where K is the *bulk modulus*. When K is sufficiently large, this volumetric strain energy contributes sufficient additional stress to maintain $J = \det \mathbf{F} = 1$ for incompressibility.

Unlike the Mooney–Rivlin hyperelastic description, many biological tissues exhibit anisotropic mechanical properties. Cardiac muscle, for example, consists of a microstructure defined by muscle fibres with mechanical properties different along the fibre direction than in the transverse directions. Defining the local fibre orientation as a unit vector \mathbf{f} with components f_i , a suitable transverse anisotropic hyperelastic strain energy function for myocardium, based on a formulation introduced by Holzapfel et al. [10], is:

$$W = \frac{c_0}{2} (\bar{I}_1 - 3) + \frac{c_{f1}}{2c_{f2}} \left[e^{c_{f2}(\bar{I}_f - 1)^2} - 1 \right] \quad (8.43)$$

$$\text{with } \bar{I}_f = \bar{C}_{ij} f_i f_j \quad (8.44)$$

where \bar{I}_1 is the first invariant of the *isochoric* right Cauchy–Green deformation tensor¹³ $\bar{\mathbf{C}} (= J^{-2/3} \mathbf{F}^T \mathbf{F})$, and c_0, c_{f1}, c_{f2} are material parameters specific to myocardium. Since most biological tissues, including myocardium, are incompressible, a

¹¹Named after American rheologists Melvin Mooney (1893–1968) and Ronald Samuel Rivlin (1915–2005).

¹²It should also be noted that in some descriptions, the invariants of Eq. 8.41 refer to the standard first and second invariants of the Cauchy–Green deformation tensor.

¹³So named because this modified deformation preserves local volume. Defining $\bar{\mathbf{F}} = J^{-1/3} \mathbf{F}$, where $J = \det \mathbf{F}$, then $\det \bar{\mathbf{F}} = 1$, and $\bar{\mathbf{C}} = \bar{\mathbf{F}}^T \bar{\mathbf{F}} = J^{-2/3} \mathbf{F}^T \mathbf{F}$.

volumetric strain energy function such as that of Eq. 8.42 should be added to the above strain energy when simulating the mechanical behaviour of tissues in COMSOL).

8.6.1 Example: Myocardial Shear

During contraction of the heart, the left ventricle (LV) ejects approximately 75% of its blood volume, reducing LV cavity volume primarily through wall-thickening than by reduction of its outer (epicardial) dimensions. Such a mode of deformation suggests substantial myocardial shear occurs during the contraction process. In order to investigate the shear properties of passive LV myocardium, simple shear experiments were undertaken on small $3 \times 3 \times 3$ mm cubic blocks of myocardial tissue at different orientations of underlying local microstructure [3]. Using the transversely anisotropic hyperelastic constitutive law of Eq. 8.43 and the volume strain energy function Eq. 8.42, use COMSOL to simulate the similar myocardial simple shear experiments shown in Fig. 8.14, plotting shear force F against shear displacements from 0 to 0.4, normalised against cube sidelength. Use the model parameters given in Table 8.1.

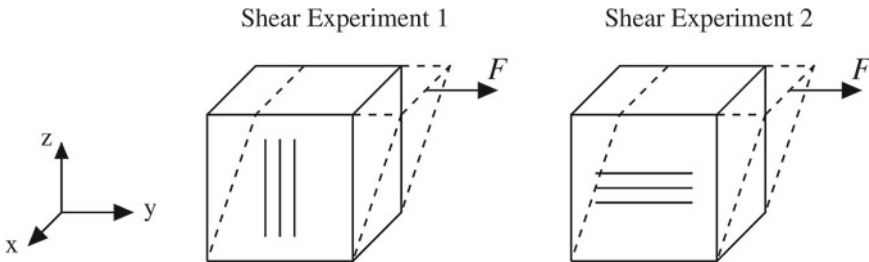


Fig. 8.14 Simple shear experiments on a myocardial cubic block of sidelength 3 mm. The orientation of myocardial fibres in the resting (undeformed) configuration are shown in each case as lines on the front face. In Shear Experiment 1, the fibres are oriented vertically from top to bottom faces, whilst in Shear Experiment 2, the fibres are oriented horizontally from left to right

Table 8.1 Parameters for myocardial shear model

| Parameter | Value | Description |
|-----------|-----------------------------|--------------------------------------|
| ρ | 1200 kg m^{-3} | Myocardial density |
| K | $1 \times 10^6 \text{ kPa}$ | Bulk modulus |
| c_0 | 3 kPa | Hyperelastic material parameter [10] |
| c_{f1} | 4.51 kPa | Hyperelastic material parameter [11] |
| c_{f2} | 23.4 | Hyperelastic material parameter [11] |

Answer: To model user-defined hyperelastic strain energies in COMSOL, including the myocardial strain energy function of Eq. 8.43, we require COMSOL's Nonlinear Structural Materials Module, an optional add-on to the base COMSOL package. The required simple shear simulations can then be implemented using the following steps:

Model Wizard

1. Open the Model Wizard and select the 3D spatial dimension.
2. In the Select Physics panel, choose Structural Mechanics|Solid Mechanics, and click “Add”.
3. Click the Study arrow (👉) to open the Select Study panel. Select Stationary, and click “Done”. This will exit the Model Wizard, displaying the main COMSOL interface.

Global Definitions

1. Right-click Global Definitions in the model tree and select Parameters. Enter the following list of parameters in the Parameters table of the Settings window:

| Name | Expression | Description |
|------------|---------------------------|----------------------|
| d_shear | 0.4 | Shear displacement |
| experiment | 1 | Shear experiment no. |
| rho | 1200 [kg/m ³] | Density |
| K | 1e6 [kPa] | Bulk modulus |
| c_0 | 3 [kPa] | Material parameter |
| c_f1 | 4.51 [kPa] | Material parameter |
| c_f2 | 23.4 | Material parameter |

Geometry

1. Select the Geometry 1 node in the model tree. In the settings window, change the length unit to ‘mm’.
2. Right-click Geometry 1 and select Block. Specify the width, depth and height to be 1.5, 3 and 3 mm respectively. Note that due to model symmetry, we will only be simulating half of the cubic block. Click Build Selected (🏗️). Once complete, the geometry and model tree will look like that shown in Fig. 8.15.

Component Definitions

1. Right-click the Definitions sub-node of Component 1 and select Component Couplings|Integration. In the Settings window, specify the Geometric entity level as ‘Boundary’, and select boundary 4, corresponding to the top face of the block. This integration operator will be used to determine the shear force from the y-component of calculated traction for each shear displacement.

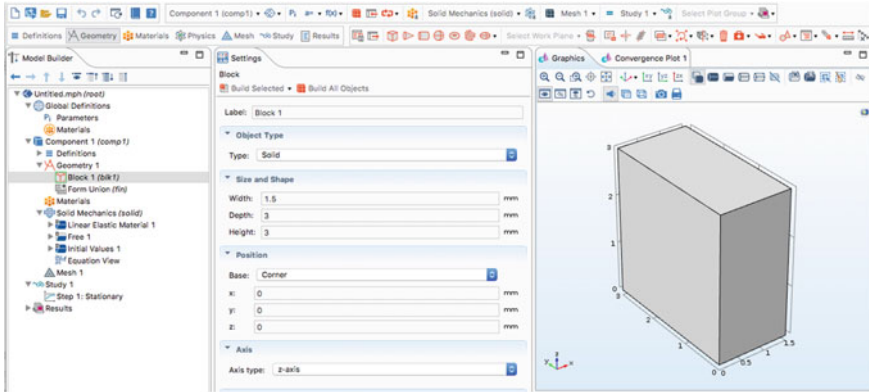


Fig. 8.15 COMSOL interface for myocardial simple shear model, showing geometry and model tree

- Right-click the Definitions sub-node of Component 1 again and select select Variables. Leave the geometric entity level to its default as ‘Entire model’, and enter the following variables in the settings table¹⁴:

| Name | Expression |
|----------|---|
| f_x | 0 |
| f_y | $\text{if}(\text{experiment}==1, 0, 1)$ |
| f_z | $\text{if}(\text{experiment}==1, 1, 0)$ |
| W_1 | $c_0 * (\text{solid.I1CIel}-3)$ |
| I_{f1} | $\text{solid.CIel11} * f_x * f_x + \text{solid.CIel12} * f_x * f_y$ |
| I_{f2} | $\text{solid.CIel13} * f_x * f_z + \text{solid.CIel12} * f_y * f_x$ |
| I_{f3} | $\text{solid.CIel22} * f_y * f_y + \text{solid.CIel23} * f_y * f_z$ |
| I_{f4} | $\text{solid.CIel13} * f_z * f_x + \text{solid.CIel23} * f_z * f_y$ |
| I_{f5} | $\text{solid.CIel33} * f_z * f_z$ |
| I_f | $I_{f1} + I_{f2} + I_{f3} + I_{f4} + I_{f5}$ |
| W_f | $c_{f1} / (2 * c_{f2}) * (\exp(c_{f2} * (I_f - 1)^2) - 1)$ |
| W | $W_1 + W_f$ |
| F | $2 * \text{intop1}(\text{solid.Tay})$ |

where the f_x , f_y , and f_z variables correspond to the x-, y-, and z-components of the fibre orientation vector f_i in Eq. 8.44, solid.I1CIel denotes the first invariant of the isochoric right Cauchy–Green tensor, and solid.CIel11 , solid.CIel12 etc. denote its components.¹⁵


¹⁴Many of these expressions will appear orange-coloured in COMSOL when first entered, indicating that some terms have not yet been defined. This is because we have not yet added the hyperelastic material description, which we will do in the next step.

¹⁵Many of COMSOL’s solid mechanics in-built variables include the term ‘el’, which is short for *elastic*. COMSOL separates deformations into elastic and inelastic components, with inelastic



Solid Mechanics

1. Right-click the Solid Mechanics node and select Material Models|Hyperelastic Material. In the Settings panel, select domain 1 as the domain to which this material applies. Under the Hyperelastic Material tab, select 'User defined' from the Material model dropdown list. Click the 'Nearly incompressible material' checkbox. For the isochoric strain energy density, enter the expression W , and for the volumetric strain energy density, enter $0.5 * K * (\text{solid.Jel}-1)^2$. The latter corresponds to Eq. 8.42, and the former to Eq. 8.43. Finally, for the density, enter the user defined value of ρ .
2. Right-click Solid Mechanics and select the Fixed Constraint boundary condition. Specify boundary 3 corresponding to the lower face of the block, forcing it to be fixed.
3. Right-click Solid Mechanics again and select Prescribed Displacement. In the Settings window, select boundary 4 corresponding to the upper face of the cube. Click the 'Prescribed in x direction' checkbox and specify a value of 0. Click the 'Prescribed in y direction' checkbox and specify a value of $d_shear * (3 \text{ [mm]})$. Finally, click the 'Prescribed in z direction' checkbox and specify a value of 0.
4. Right-click Solid Mechanics a final time and select More Constraints|Symmetry. Specify boundary 1, corresponding to the left-face of the block, located at $x = 0$. This boundary condition prevents displacements normal to the boundary, as required for a symmetry-plane.

Mesh

1. Right-click the Mesh 1 node and select More Operations|Mapped. In the Settings panel, select boundary 6, corresponding to the right face of the block. Right-click the newly created Mapped 1 sub-node and select Distribution. In the settings panel, select edges 11 and 12, and specify the fixed number of elements to be 10.
2. Right-click the Mesh 1 node again and select Swept. Specify the source face as boundary 6 and the destination face as boundary 1. Right-click the Swept 1 sub-node and select Distribution. Specify domain 1 and the fixed number of elements to be 5. Click the build all button () to display the resulting mesh made up of cubic elements.


Study

1. Right-click the Study 1 node and select Parametric Sweep. In the Settings panel, click the Add parameter button () and specify parameter d_shear . Click the Range button () and specify start, step and stop values of 0, 0.05, and 0.4 respectively. Leave the parameter unit field as empty (i.e. dimensionless unit). Click 'Replace' to add this range to the parameter. Click Add parameter a second

(Footnote 15 continued)

components arising from non-elastic effects such as thermal expansion and plasticity. For the models presented in this chapter, there are no inelastic effects, hence COMSOL terms such as the elastic isochoric right Cauchy Green tensor are equivalent to the standard definitions presented.

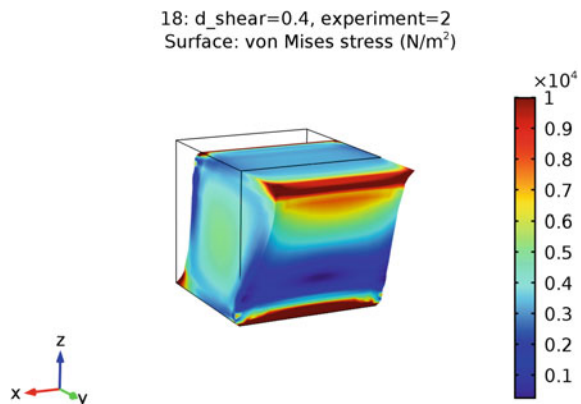
time and specify parameter `experiment`. In the Parameter value list, manually enter 1, 2 and leave the parameter units field blank. For the Sweep type, select 'All combinations' from the dropdown list.

2. Right-click Study 1 and select Show Default Solver. Select Study 1| Solver Configurations| Solution 1| Stationary Solver 1| Parametric 1. In the Settings panel, find the 'Run continuation for' option, and select 'Manual' from the dropdown list. Specify the continuation parameter as `d_shear`. For the 'Reuse solution from previous step' option, select 'Yes'.
3. Select Study 1| Solver Configurations| Solution 1| Stationary Solver 1| Fully Coupled 1. Under the Method and Termination tab, specify the Nonlinear method to be Automatic (Newton) and enter values for the Initial damping factor and Minimum damping factor as $1e-4$ and $1e-8$ respectively.
4. Right-click Study 1 again and select compute () to solve the model. Select the Progress tab in the Information window to see the solver progress.

Results

1. When the model has completed solving, the Graphics window will display a default plot of the von Mises stress in the deformed solid at the final parameter values of `d_shear` and `experiment` of 0.4 and 2 respectively. To visualise the full 3D symmetric model solution at this shear, click the Data Sets sub-node of Results and select More Data Sets| Mirror 3D. In the settings panel, leave the default data set as Study1/Solution 1, as well as quick YZ-plane for the mirror plane with the X-coordinate value of 0 mm. Now right-click the Results node and select 3D Plot Group. In the settings panel, specify the data set as Mirror 3D 1. Right-click the newly-created 3D Plot Group 2 node and select Surface. In the settings panel, specify the expression to plot as `solid.mises`, corresponding to the von Mises stress. Right-click the Surface 1 sub-node of 3D Plot Group 2 and select Deformation. Click on the Scale factor checkbox and specify a scale factor of 1. Select the Surface 1 sub-node again and under the Range tab in the Settings, click the Manual color range checkbox and select a maximum value

Fig. 8.16 von Mises stress distribution in myocardial block at a normalised shear displacement of 0.4 for Simple Shear Experiment 2



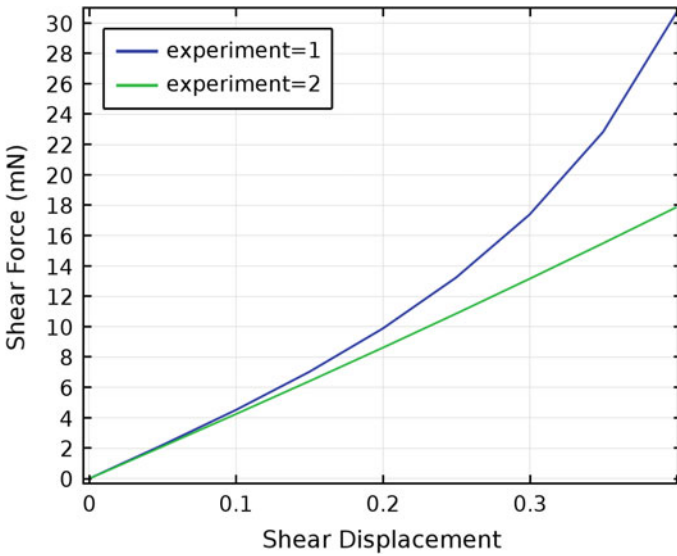


Fig. 8.17 Simulated myocardial shear force against normalised shear displacement in LV tissue block for the two simple shear experiments described

of $1e4$. Click the Plot button (🖨️). The resulting plot of von Mises stress on the full-geometry visualization is shown in Fig. 8.16.

- To plot the shear force against displacement for both experiments, right-click the Results node and select '1D Plot Group'. Right-click the 1D Plot Group 3 sub-node and select 'Global'. Under the y-Axis Data tab, specify F as the expression to plot, with units of 'mN'.
- Select the 1D Plot Group 3 sub-node again, and in the Plot Settings window, check the 'x-axis label' and 'y-axis label' checkboxes. Specify the x-axis label as 'Shear Displacement' and the y-axis label as 'Shear Force (mN)'. Under the Legend tab, specify the legend position as 'Upper Left'. Click the Plot button (🖨️) to display the graph, as shown in Fig. 8.17.

The simple shear results from this model (see Fig. 8.17) indicate that myocardial tissue is mechanically anisotropic, with greater forces produced in shear modes where fibres are actually stretched (i.e. shear experiment 1), as has been verified experimentally [3].

8.7 Further Reading

Excellent texts on solid mechanics with some biomechanics examples, are those of Fung [6–8] and Holzapfel [9]. Even more concise and readable texts, although not biomechanics-based, are those of Spencer [12], Chadwick [2], and Atkin and Fox [1].

A thorough coverage of the principles of continuum mechanics applied to a diverse range of physical phenomena, with detailed problems and worked solutions, is the two-part volume edited by Eglit and Hodges [4].

Problems

8.1 (a) Show that the triple scalar product $(\mathbf{a} \times \mathbf{b}) \cdot \mathbf{c}$ is given by

$$(\mathbf{a} \times \mathbf{b}) \cdot \mathbf{c} = \begin{vmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{vmatrix} = \varepsilon_{ijk} a_i b_j c_k$$

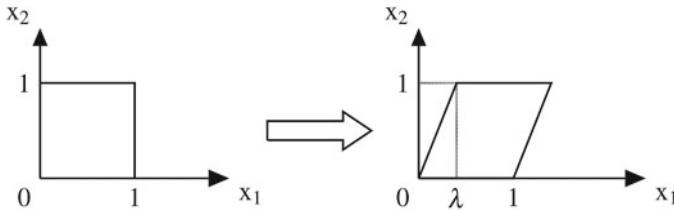
where ε_{ijk} is the permutation symbol (a third-order tensor) defined by

$$\begin{aligned} \varepsilon_{123} &= \varepsilon_{231} = \varepsilon_{312} = 1 \\ \varepsilon_{321} &= \varepsilon_{132} = \varepsilon_{213} = -1 \\ \varepsilon_{ijk} &= 0 \quad (\text{if any two of } i, j, k \text{ are equal}) \end{aligned}$$

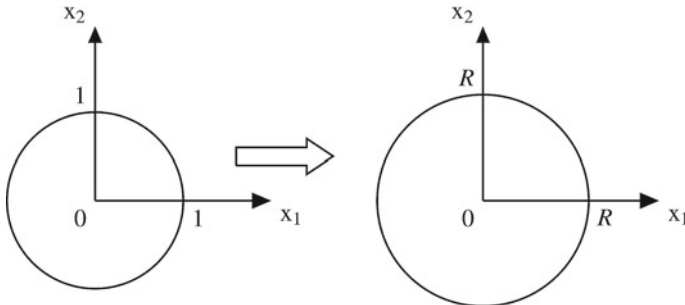
(b) Verify that $\sigma_{ij}^H \varepsilon_{ij}^D = \sigma_{ij}^D \varepsilon_{ij}^H = 0$.

8.2 Determine the Cauchy and Green strain tensors for the following deformations:

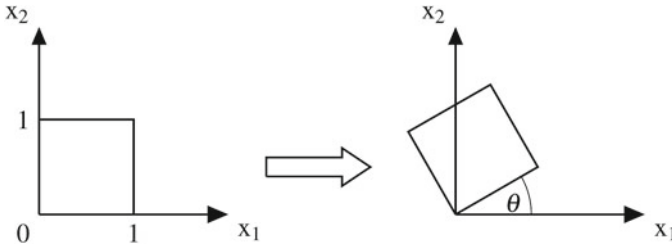
(a) Simple shear:



(b) Uniform inflation:



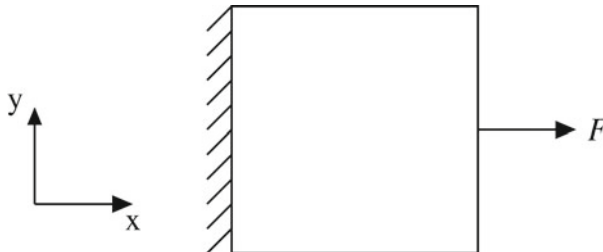
(c) Rotation:



8.3 Consider the left ventricle to be approximated by a spherical shell with inner and outer radii A and B . A uniform pressure is applied to the endocardial (i.e. inner) surface such that the ventricle passively expands to new inner and outer radii of a and b respectively.

- (a) Assuming that the myocardium is incompressible, find b as a function of a , A , and B .
- (b) Determine the resulting Cauchy strain tensor in the ventricular wall.

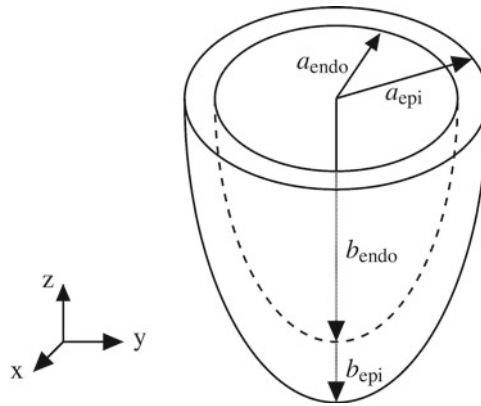
8.4 A $100 \times 100 \times 1$ mm square piece of rabbit skin tissue is subjected to a tensile test in the laboratory as shown schematically below:



Assume that the constitutive law of the skin tissue obeys a Mooney–Rivlin relationship with material parameters C_1 and C_2 equal to 20 kPa, density $\rho = 1100 \text{ kg m}^{-3}$ and bulk modulus $K = 1 \times 10^6$ kPa. Assume that the leftmost edge of the tissue is clamped (fixed) in place, and that the rightmost edge is subjected to a prescribed displacement only in the x -direction. All other edges are free.

- (a) Use COMSOL’s stationary solver to simulate this tensile test and plot the applied force magnitude F (in mN) against skin displacement from 0 to 50 mm.
- (b) Plot the spatial distribution of strain energy in the deformed skin sample at its maximum displacement of 50 mm.

8.5 The left ventricle of the heart can be approximated by a circular half-ellipsoidal shell, with endocardial semi-axes of a_{endo} and b_{endo} , and epicardial semi-axes of a_{epi} and b_{epi} , as shown in the figure below:



Assuming the myocardium follows a two-parameter Mooney–Rivlin strain energy formulation, use COMSOL to simulate passive inflation of the ventricle using a 2D axisymmetric implementation, and determine endocardial volume in ml as a function of filling pressures from 0 to 50 mmHg. Use the model parameters in the table below, and employ a stationary solver for each pressure. Assume the upper boundary (base) of the ventricle is held fixed, and take the external pressure on the epicardial surface to be zero.

| Parameter | Value | Description |
|-------------------|-----------------------------|----------------------------------|
| a_{endo} | 2 cm | Endocardial radius |
| a_{epi} | 3 cm | Epicardial radius |
| b_{endo} | 6 cm | Endocardial long semi-axis |
| b_{epi} | 7 cm | Epicardial long semi-axis |
| ρ | 1200 kg m^{-3} | Myocardial density |
| K | $1 \times 10^6 \text{ kPa}$ | Bulk modulus |
| C_1 | 3 kPa | Mooney–Rivlin material parameter |
| C_2 | 5 kPa | Mooney–Rivlin material parameter |

References

1. Atkin RJ, Fox N (2005) An introduction to the theory of elasticity, Dover edn. Dover, Mineola
2. Chadwick P (1999) Continuum mechanics: concise theory and problems, Dover edn. Dover, Mineola
3. Dokos S, Smaill BH, Young AA, LeGrice IJ (2002) Shear properties of passive ventricular myocardium. *Am J Physiol Heart Circ Physiol* 283:H2650–H2659
4. Eglit ME, Golublatnikov AN, Kamenjarzh JA, Karlikov VP, Kulikovskiy AG, Petrov AG, Shikina IS, Sveshnikova EI (1996) In: Eglit ME, Hodges DH (eds) Continuum mechanics via

- problems and exercises. World scientific series on nonlinear science. Series A, vol 19. World Scientific, Singapore
5. Einstein A (1916) The foundation of the general theory of relativity (Translated by Engel A). In: Kox AJ, Klein MJ, Schulmann R (eds) The collected papers of Albert Einstein. The Berlin years: Writings, 1914–1917, vol 6, Princeton University press, Princeton, pp 146–200 (1997)
 6. Fung YC (1984) Biomechanics: circulation, 2nd edn. Springer, New York
 7. Fung YC (1994) A first course in continuum mechanics, 3rd edn. Prentice-Hall, Upper Saddle River
 8. Fung YC, Tong P (2001) Classical and computational solid mechanics. World Scientific, Singapore
 9. Holzapfel GA (2000) Nonlinear solid mechanics: a continuum approach for engineering. Wiley, Chichester
 10. Holzapfel G, Gasser T, Ogden R (2000) A new constitutive framework for arterial wall mechanics and a comparative study for material models. *J Elast* 61:1–48
 11. Schmid H, Wang YK, Ashton J, Ehret AE, Krittian SBS, Nash MP, Hunter PJ (2009) Myocardial material parameter estimation: a comparison of invariant based orthotropic constitutive equations. *Comput Meth Biomech Biomed Eng* 12:283–295
 12. Spencer AJM (1980) Continuum mechanics. Longman, Burnt Mill

Chapter 9

Fluid Mechanics

Fluid mechanics deals with the motion of fluids and the forces acting on them, where *fluid* refers to both liquids and gases. Numerically solving the equations of fluid mechanics can be difficult, requiring finer mesh discretization than other physics applications such as diffusion or electric current flow. In biomedical engineering, examples of fluid mechanics modelling include:

- Simulating the flow of blood in the heart and circulation.
- Flow of air in the pulmonary tracts.
- Peristaltic motion of lymph fluid.
- Prosthetic devices such as artificial heart valves, ventricular assist pumps, drug eluting stents and implantable drug delivery devices.

9.1 Fluid Motion

In contrast to solids, where deformation and movement is described in terms of strain and displacement, the motion of fluids is described by velocity and *strain rate*. For continuous flow, let a particle of fluid located at (x_1, x_2, x_3) have components of velocity

$$v_1(x_1, x_2, x_3), \quad v_2(x_1, x_2, x_3), \quad v_3(x_1, x_2, x_3)$$

then the strain rate tensor is defined as

$$V_{ij} = \frac{1}{2} \left(\frac{\partial v_j}{\partial x_i} + \frac{\partial v_i}{\partial x_j} \right)$$

In the case of solid mechanics, strain was linked to stress via the particular constitutive law of the material. In the case of fluids, the constitutive law links stress to strain rate.

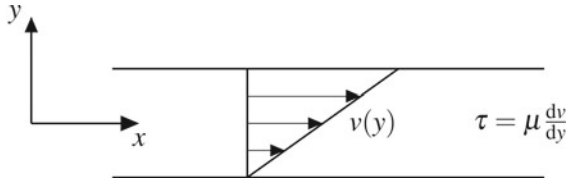


Fig. 9.1 Concept of viscosity for the case of shear flow with a uniform velocity gradient. In this case, the fluid velocity increases linearly with y . The viscous stress in the fluid, τ , is proportional to the velocity gradient dv/dy , with the coefficient of proportionality equal to the viscosity μ

The most common constitutive law for fluids is that for the incompressible, isotropic case:

$$\sigma_{ij} = -p\delta_{ij} + 2\mu V_{ij} \quad (9.1)$$

where σ_{ij} are the components of stress, p denotes the pressure, V_{ij} are the components of strain rate, δ_{ij} is the Kronecker delta, and μ denotes the *viscosity coefficient*. If μ is a constant, then the fluid is *Newtonian*.¹ Viscosity can be understood as akin to internal friction, in which stress is induced whenever regions of fluid move relative to each other, as illustrated in Fig. 9.1.

When fluid in motion has a strain rate that remains constant with time, the flow is defined as *steady*. When the fluid is moving in parallel layers, with no penetration between layers, the flow is said to be *laminar* or *streamline*.

9.1.1 Example: Laminar Flow Through a Circular Tube

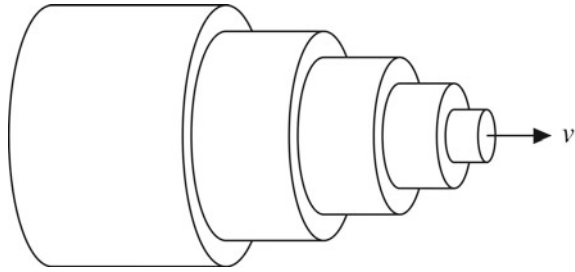
Consider the case of a fluid flowing axially along a long cylindrical tube such as, for example, blood travelling through an artery. Let the diameter of the tube be D , its length L , and the viscosity of fluid within it μ . Furthermore, there are pressures acting at the upstream and downstream ends of the tube, P_{up} and P_{down} , that cause the fluid to flow.

To analytically determine the steady-state fluid motion, we note that due to radial symmetry, and since the diameter of the tube is constant, the steady-state fluid velocity will only be a function of radial position. We can represent such fluid motion with a series of sliding tubes as shown in Fig. 9.2, each tube travelling a distance in a given time proportional to its own velocity. Consider one such sliding tube of radius r . The three forces acting on the surface boundaries of this tube will be:

1. the force on the upstream end due to the upstream pressure. This force acts in a positive direction along the axis of the tube and is given by $F_{up} = \pi r^2 P_{up}$.

¹Named after English physicist and mathematician Sir Isaac Newton (1642–c.1726) who, in addition to formulating his laws of motion, gravitational action and other foundational principles of physics and mathematics, also introduced the concept of fluid viscosity.

Fig. 9.2 Fluid flow in a circular tube, where the fluid layers are represented by concentric cylinders, the length of each proportional to the axial velocity v of fluid in that layer



2. the force on the downstream end due to the downstream pressure. This acts in a negative direction along the axis of the tube and is given by $F_{down} = -\pi r^2 P_{down}$.
3. the traction acting on the curved surface of the tube, due to the viscous force from the relative velocities of layers sliding past each other. Using the definition given in Fig. 9.1, this viscous force equals the viscous stress τ multiplied by the curved surface area, or

$$\begin{aligned}
 F_{viscous} &= 2\pi r L \tau \\
 &= 2\pi r L \mu \frac{dv}{dr}
 \end{aligned}$$

where v is the fluid velocity and the positive sign of the force indicates that a positive dv/dr , that is the layer surrounding the inner tube is moving with higher velocity, then this viscous force will act to increase the velocity of this inner layer.

Adding the above three forces together yields the total force on the inner tube. For steady-state flow, the fluid is not accelerating, and the total force must be zero. That is,

$$\pi r^2 P_{up} - \pi r^2 P_{down} + \mu \frac{dv}{dr} 2\pi r L = 0$$

Denoting the pressure differential between the upstream and downstream ends as $\Delta P (= P_{up} - P_{down})$, and dividing throughout by πr , we have:

$$\begin{aligned}
 r \Delta P + \mu \frac{dv}{dr} 2L &= 0 \\
 \frac{dv}{dr} &= -\frac{r \Delta P}{2\mu L} \\
 v &= -\frac{r^2 \Delta P}{4\mu L} + C
 \end{aligned}$$


where C is a constant of integration. Its value can be determined from the experimental observation that fluids tend to ‘stick’ to solid boundaries, such that the velocity of the fluid is zero at the boundary. This boundary state is known as a *no-slip* boundary condition. If the diameter of the tube is D , then this condition states that at $r = D/2$, $v = 0$. This implies that $C = D^2 \Delta P / 16\mu$, or

$$v = \frac{\Delta P}{4\mu L} \left(\frac{D^2}{4} - r^2 \right) \quad (9.2)$$

In other words, the fluid follows a *parabolic* velocity profile in the tube (see also Eq. 7.7 of Sect. 7.1.4), with maximum axial velocity corresponding to $r = 0$ at the centre of the tube.

We can also use COMSOL to simulate this system to confirm that the velocity profile is indeed parabolic.² In particular, we will utilize an axisymmetric geometry with the following parameters: $D = 2$ cm, $L = 15$ cm, $\mu = 3.5$ mPa s, and $\Delta P = 100$ Pa. To setup the model in COMSOL, implement the following steps:

Model Wizard


1. Open the Model Wizard and select the 2D axisymmetric spatial dimension.
2. In the Select Physics panel, choose Fluid Flow|Single-Phase Flow|Laminar Flow. Click ‘Add’, and leave variables u , v , w as the default dependent variables for the velocity field components, and p as the default dependent variable for pressure.
3. Click the Study arrow () to open the Select Study panel. Select Stationary, and click “Done”. This will exit the Model Wizard, displaying the main COMSOL interface.

Global Definitions

1. Right-click Global Definitions in the model tree and select Parameters. Enter the following details in the Parameters table of the Settings window:

| Name | Expression | Description |
|---------|---------------|-----------------------|
| D | 2 [cm] | Diameter |
| L | 15 [cm] | Length of tube |
| mu | 3.5 [mPa*s] | Viscosity coefficient |
| rho | 1000 [kg/m^3] | Density |
| Delta_P | 100 [Pa] | Pressure differential |

Geometry

1. Select the Geometry 1 node in the model tree. In the settings window, change the default length unit to ‘mm’.
2. Right-click Geometry 1 and select Rectangle. Specify the width and height as $D/2$ and L respectively. Click Build Selected ().

²The Navier-Stokes equations that COMSOL solves for will be described more fully in the next section.

Component 1 Definitions

1. Right-click the Definitions sub-node of Component 1 and select Variables. Define a single variable V_{th} with expression $\Delta_P / (4 * \mu * L) * (D^2 / 4 - r^2)$ to denote the theoretical velocity profile given by Eq. 9.2.


Laminar Flow

1. Select the Fluid Properties 1 sub-node of Laminar Flow. In the Settings panel, set the fluid density and dynamic viscosity fields to the user-defined values of ρ_0 and μ respectively.
2. Right click the Laminar Flow node to add an ‘Inlet’ boundary condition. Select boundary 2 (i.e. the lower boundary) and specify the boundary condition type as Pressure with a pressure value of Δ_P .
3. Right click the Laminar Flow node again to add an ‘Outlet’ boundary condition. Select boundary 3 (i.e. the upper boundary). Leave the boundary condition type to the default value of Pressure, and leave the pressure value at 0.

Mesh

1. Select the Mesh 1 node of the model tree. In the Settings panel, specify the element size as ‘Extra Fine’. This will create an extra fine mesh consisting of almost 108,000 elements!

Study

1. Right-click the Study 1 and select Compute () to solve the model. COMSOL will display the default plot of fluid velocity magnitude, similar to that shown in Fig. 9.3 (left). Clicking on the Velocity (spf) 1 volume plot (under the Results node of the model tree) will generate a 3D representation of the axisymmetric velocity profile, as shown also in Fig. 9.3.

Results

1. Right-click the Data Sets sub-node of Results in the model tree, and select Cut Line 2D. In the Settings window, specify the (r, z) coordinates of the two points defining the cut line as $r = 0, z = L/2$ and $r = D/2, z = L/2$. This corresponds to a line perpendicular to the axis of the tube at a position half-way along it.
2. Right-click the Results node and select ‘1D Plot Group’. Right-click the newly-created 1D Plot Group 4 sub-node and select ‘Line Graph’. In the settings panel, specify the Data set as ‘Cut Line 2D 1’. For the expression to plot, enter w , corresponding to the z- (i.e. axial) component of velocity. In the Legends tab, select the Show legends checkbox, and specify ‘Manual’. Enter the expression ‘Computed’.
3. Right-click the 1D Plot Group 4 node again and select Line Graph. As above, enter the Data set as ‘Cut Line 2D 1’. This time, specify the expression to plot as V_{th} . In the Legends tab, select the Show legends checkbox, and specify ‘Manual’. Enter the expression ‘Theoretical’.

Fig. 9.3 Steady-state velocity magnitude in circular tube, as computed by COMSOL in the 2D axisymmetric geometry (*left*), and in the 3D representation (*right*), obtained by rotating the axisymmetric 2D data about its axis. Dimensions shown are in mm

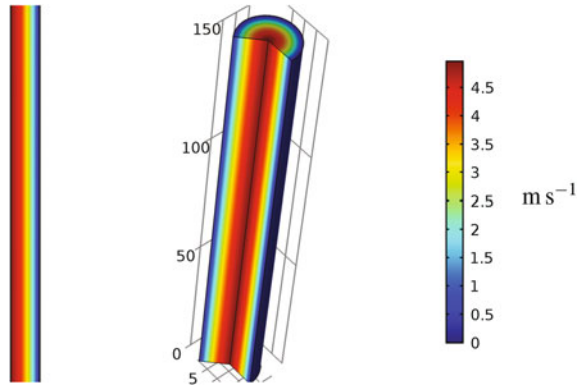
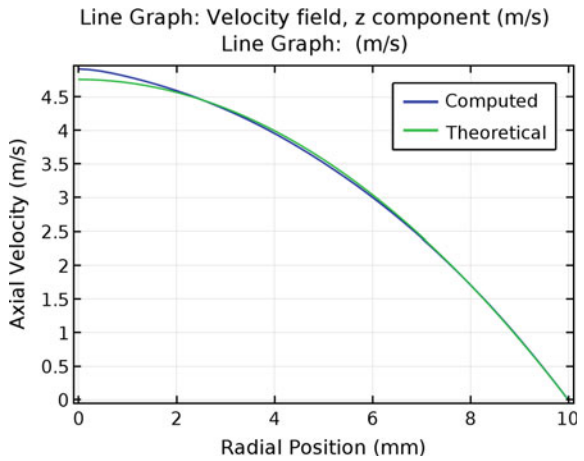



Fig. 9.4 Laminar parabolic velocity profile in circular tube as computed by COMSOL, compared with the theoretical profile given by Eq. 9.2



4. Select the 1D Plot Group 4 node and click the x-axis and y-axis label checkboxes. Enter the x label as 'Radial Position (mm)' and the y label as 'Axial Velocity (m/s)'. Click the Plot button (). The resulting plot should be similar to that shown in Fig. 9.4.

This simulation in COMSOL verifies that the laminar fluid velocity profile in a circular tube is indeed parabolic, although the number of mesh elements required for an accurate solution is large. It is left as an exercise for the reader to perform a mesh convergence analysis on this simple model, plotting the error in the axial velocity component as a function of maximum element size (see Problem 9.2).

9.2 Navier-Stokes Equations

To determine the equations governing motion of an incompressible Newtonian fluid, we first introduce two spatial frameworks for defining time derivatives of various quantities such as pressure and temperature in a moving fluid. This is necessary because, in a moving fluid, such quantities will generally vary in a complex manner with respect to both time and space, and it will be convenient to switch between two separate viewpoints for spatial position.

Consider a scalar field f that represents some quantity of interest in a moving fluid. Denote the spatial coordinates of each point in the fluid by \mathbf{x} relative to a fixed global Cartesian set of axes. We can characterise the time derivative of f according to the following two frameworks:

- the *Eulerian* framework, in which we take the time derivative holding \mathbf{x} fixed, so that $\partial f/\partial t$ represents the time rate of change of f at a fixed point in space.
- the *Lagrangian* framework, where we describe the rate of change of f as viewed by a given ‘particle’ in the fluid. Labelling each fluid particle with its starting position \mathbf{X} , we can parameterise the scalar field as $f(\mathbf{X}, t)$. We then take the time derivative of f holding \mathbf{X} fixed: that is, the time derivative - Df/Dt - as viewed by a particle flowing along with the fluid, where D/Dt is referred to as the *material derivative*.

We can transform between these two frameworks by utilizing the chain rule as follows:

$$\begin{aligned} \frac{Df}{Dt} &= \frac{\partial f}{\partial t} + \frac{dx_1}{dt} \frac{\partial f}{\partial x_1} + \frac{dx_2}{dt} \frac{\partial f}{\partial x_2} + \frac{dx_3}{dt} \frac{\partial f}{\partial x_3} \\ &= \frac{\partial f}{\partial t} + u_1 \frac{\partial f}{\partial x_1} + u_2 \frac{\partial f}{\partial x_2} + u_3 \frac{\partial f}{\partial x_3} \end{aligned}$$

which can be simplified to

$$\frac{Df}{Dt} = \frac{\partial f}{\partial t} + \mathbf{u} \cdot \nabla f \tag{9.3}$$

where $\mathbf{x} = (x_1, x_2, x_3)^T$ and $\mathbf{u} = (u_1, u_2, u_3)^T$ is the fluid velocity. Equation 9.3 also holds for vector fields. For example, acceleration in the Lagrangian frame of reference is related to its description in the Eulerian frame according to:

$$\mathbf{a} = \frac{D\mathbf{u}}{Dt} = \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \tag{9.4}$$

where \mathbf{a} is the acceleration and \mathbf{u} is the velocity.

Using this concept of the material derivative, we can apply Newton’s second law of motion to regions within a fluid: this law states that the time rate of change of momentum equals the applied force. In our case, we consider a region of fluid V enclosed by a surface S . In this context, V is understood to be part of a small region

within a larger overall fluid domain. Newton's second law of motion applied to V yields:

$$\frac{D}{Dt} \int_V \rho \mathbf{u} dV = \int_S \mathbf{T}^{\mathbf{n}} dS + \int_V \mathbf{f} dV$$

where ρ is the fluid density, \mathbf{u} is the velocity, $\mathbf{T}^{\mathbf{n}}$ is the stress vector acting on the region of S with outward normal direction \mathbf{n} , and \mathbf{f} is the body force per unit volume, typically due to gravity. Noting that $\mathbf{T}^{\mathbf{n}} = \boldsymbol{\sigma} \cdot \mathbf{n}$, we can substitute this into the above, employing the divergence theorem (Eq.4.5), to obtain:

$$\frac{D}{Dt} \int_V \rho \mathbf{u} dV = \int_V \nabla \cdot \boldsymbol{\sigma} dV + \int_V \mathbf{f} dV$$

Furthermore for an incompressible fluid, ρ will be constant. We can therefore take the material derivative of the first term inside its integral to obtain

$$\int_V \rho \frac{D\mathbf{u}}{Dt} dV = \int_V \nabla \cdot \boldsymbol{\sigma} dV + \int_V \mathbf{f} dV$$

Since this must hold for any arbitrary volume V in the fluid, then the integrands on both sides of the equation must be equal. Namely,

$$\rho \frac{D\mathbf{u}}{Dt} = \nabla \cdot \boldsymbol{\sigma} + \mathbf{f} \quad (9.5)$$

We now recall the constitutive law for an incompressible Newtonian fluid (Eq.9.1):

$$\sigma_{ij} = -p\delta_{ij} + 2\mu V_{ij}$$

where V_{ij} is the strain rate tensor satisfying

$$V_{ij} = \frac{1}{2} \left(\frac{\partial u_j}{\partial x_i} + \frac{\partial u_i}{\partial x_j} \right)$$

The above can also be written in matrix form as

$$\begin{aligned} \mathbf{V} &= \frac{\nabla \mathbf{u} + (\nabla \mathbf{u})^T}{2} \\ \boldsymbol{\sigma} &= -p\mathbf{I} + 2\mu \mathbf{V} \\ &= -p\mathbf{I} + \mu [\nabla \mathbf{u} + (\nabla \mathbf{u})^T] \end{aligned}$$

where \mathbf{I} is the identity matrix and $\boldsymbol{\sigma}$, \mathbf{V} are the stress and strain rate respectively. Substituting these expressions into Eq.9.5, and employing the material derivative expression (Eq.9.3), we obtain:

$$\rho \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right) = \nabla \cdot (-p\mathbf{I} + \mu [\nabla \mathbf{u} + (\nabla \mathbf{u})^T]) + \mathbf{f} \quad (9.6)$$

We note that for a Newtonian fluid, the viscosity μ is constant. Furthermore,

$$\nabla \cdot (p\mathbf{I}) = \begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \end{pmatrix} \cdot \begin{pmatrix} p & 0 & 0 \\ 0 & p & 0 \\ 0 & 0 & p \end{pmatrix} = \begin{pmatrix} \frac{\partial p}{\partial x} \\ \frac{\partial p}{\partial y} \\ \frac{\partial p}{\partial z} \end{pmatrix} = \nabla p$$

In addition, since the fluid is incompressible, we have from the equation of continuity for incompressible flow (see Eq. 4.11 of Sect. 4.5):

$$\nabla \cdot \mathbf{u} = 0$$

or

$$\frac{\partial u_1}{\partial x} + \frac{\partial u_2}{\partial y} + \frac{\partial u_3}{\partial z} = 0 \quad (9.7)$$

With this result, we can also determine the remaining quantities of Eq. 9.6, namely:

$$\nabla \cdot \nabla \mathbf{u} = \begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \end{pmatrix} \cdot \begin{pmatrix} \frac{\partial u_1}{\partial x} & \frac{\partial u_1}{\partial y} & \frac{\partial u_1}{\partial z} \\ \frac{\partial u_2}{\partial x} & \frac{\partial u_2}{\partial y} & \frac{\partial u_2}{\partial z} \\ \frac{\partial u_3}{\partial x} & \frac{\partial u_3}{\partial y} & \frac{\partial u_3}{\partial z} \end{pmatrix} = \begin{pmatrix} \frac{\partial^2 u_1}{\partial x^2} + \frac{\partial^2 u_1}{\partial y^2} + \frac{\partial^2 u_1}{\partial z^2} \\ \frac{\partial^2 u_2}{\partial x^2} + \frac{\partial^2 u_2}{\partial y^2} + \frac{\partial^2 u_2}{\partial z^2} \\ \frac{\partial^2 u_3}{\partial x^2} + \frac{\partial^2 u_3}{\partial y^2} + \frac{\partial^2 u_3}{\partial z^2} \end{pmatrix} = \begin{pmatrix} \nabla^2 u_1 \\ \nabla^2 u_2 \\ \nabla^2 u_3 \end{pmatrix} = \nabla^2 \mathbf{u}$$

and

$$\begin{aligned} \nabla \cdot (\nabla \mathbf{u})^T &= \begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \end{pmatrix} \cdot \begin{pmatrix} \frac{\partial u_1}{\partial x} & \frac{\partial u_2}{\partial x} & \frac{\partial u_3}{\partial x} \\ \frac{\partial u_1}{\partial y} & \frac{\partial u_2}{\partial y} & \frac{\partial u_3}{\partial y} \\ \frac{\partial u_1}{\partial z} & \frac{\partial u_2}{\partial z} & \frac{\partial u_3}{\partial z} \end{pmatrix} = \begin{pmatrix} \frac{\partial^2 u_1}{\partial x^2} + \frac{\partial^2 u_2}{\partial y \partial x} + \frac{\partial^2 u_3}{\partial z \partial x} \\ \frac{\partial^2 u_1}{\partial x \partial y} + \frac{\partial^2 u_2}{\partial y^2} + \frac{\partial^2 u_3}{\partial z \partial y} \\ \frac{\partial^2 u_1}{\partial x \partial z} + \frac{\partial^2 u_2}{\partial y \partial z} + \frac{\partial^2 u_3}{\partial z^2} \end{pmatrix} \\ &= \begin{pmatrix} \frac{\partial}{\partial x} \left[\frac{\partial u_1}{\partial x} + \frac{\partial u_2}{\partial y} + \frac{\partial u_3}{\partial z} \right] \\ \frac{\partial}{\partial y} \left[\frac{\partial u_1}{\partial x} + \frac{\partial u_2}{\partial y} + \frac{\partial u_3}{\partial z} \right] \\ \frac{\partial}{\partial z} \left[\frac{\partial u_1}{\partial x} + \frac{\partial u_2}{\partial y} + \frac{\partial u_3}{\partial z} \right] \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} = \mathbf{0} \end{aligned}$$

where we have made use of Eq. 9.7. Substituting all the above relationships into Eq. 9.6, we obtain:

$$\rho \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right) = -\nabla p + \mu \nabla^2 \mathbf{u} + \mathbf{f} \quad (9.8)$$

which represents a vector equation in the components of velocity \mathbf{u} and the scalar pressure p . Equation 9.8 represents a system of PDEs known as the incompressible

*Navier-Stokes equations.*³ In 3D, there are four unknowns to be solved for (i.e. the components of \mathbf{u} and p) in only three equations. Therefore, an additional equation is needed, and this is provided by the equation of continuity (Eqs. 4.11 and 9.7). Hence, the full set of equations governing motion of incompressible Newtonian fluids, as solved in COMSOL's single phase laminar flow physics are:

$$\rho \left(\overbrace{\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u}}^{\text{total acceleration}} \right) = \underbrace{-\nabla p}_{\text{pressure gradient}} + \underbrace{\mu \nabla^2 \mathbf{u}}_{\text{viscosity}} + \underbrace{\mathbf{f}}_{\text{body force}}$$

$$\nabla \cdot \mathbf{u} = 0 \quad (\text{incompressibility})$$

Note that the total acceleration of the fluid consists of both unsteady and convective components. Unsteady acceleration occurs when the fluid velocity at a point in space is changing with time. In some instances of laminar flow, the fluid may still be undergoing acceleration even if $\partial \mathbf{u} / \partial t$ is zero everywhere. To understand this phenomenon, consider the steady-state case of an incompressible fluid flowing in a tube with a narrowing at some point along its length. At the narrowing section, the cross-sectional area of the tube is lower, resulting in a greater fluid velocity to maintain the required incompressible volumetric flow rate. If the fluid velocity at each point in space is not changing with time, the flow is steady, and $\partial \mathbf{u} / \partial t = 0$. That is, in the Eulerian framework, the acceleration is zero. However, from the Lagrangian point of view of an individual fluid particle, the fluid will accelerate as it travels through the occlusion. This component of acceleration is referred to as the convective component $\mathbf{u} \cdot \nabla \mathbf{u}$, and occurs due to the spatial gradient of velocity.

9.2.1 Example: Drug Delivery in a Coronary Stent Revisited

In this example, we revisit the coronary stent model described in Sect. 7.1.4, in which we simulate the concentration of drug eluted from a stent in a coronary artery. In this case, however, we will incorporate the mechanics of fluid flow around the individual stent struts, and not assume the fluid is flowing in a prescribed velocity profile along a continuum uniform boundary representing the stent. We will use similar model parameters to the former example: namely the maximum blood velocity along the arterial axis is 50 cm^{-1} , the arterial radius is 1 mm, the length of the arterial segment is 9 mm and the length of the stent is 6 mm. Although the stent wire mesh is a complex structure, we will make some simplifications for computational efficiency: namely,

³Named after French engineer and mathematician Claude-Louis Navier (1785–1836), and the Irish-born mathematical physicist Sir George Gabriel Stokes (1819–1903).



Fig. 9.5 Simplified axisymmetric stent model geometry showing cylindrical arterial segment and stent strut hoops (dashed)

we take the overall stent geometry to be axisymmetric, with individual stent struts represented by regularly-spaced ‘hoops’ along the artery. Each strut consists of a square cross-section $100\ \mu\text{m}$ in sidelength, spaced $0.5\ \text{mm}$ apart centre-centre. The overall geometry is shown in Fig. 9.5. As in the earlier example, we take the diffusion coefficient of the drug to be $2.56 \times 10^{-4}\ \text{cm}^2\ \text{min}^{-1}$, and the initial total content of drug in the stent as $49.2\ \text{nmol}$. We also assume mono-exponential decay of stent drug-content

$$M = M_0 \exp(-k_{drug}t)$$


where M_0 is the initial content with a drug release rate k_{drug} of $0.2\ \text{hr}^{-1}$. For the blood parameters, we use a density of $1100\ \text{kg}\ \text{m}^{-3}$ and a viscosity of $3.5\ \text{mPa}\ \text{s}$.

To implement this model in COMSOL, we take advantage of the weakly-coupled physics and solve first for the fluid flow, then use this solution to solve for the diffusion/convection of the drug. To implement the desired axial velocity, we impose a uniform pressure boundary condition on the upstream end, using a global PDE to determine the appropriate pressure value as follows:

$$\frac{dP}{dt} = G_{in} (w_{max} - w_{in})$$

where P is the upstream pressure, w_{max} is the specified fluid axial velocity, w_{model} is the actual fluid axial velocity determined by the model, and G_{in} is a gain factor, nominally set to a value of $1000\ \text{Pa}\ \text{m}^{-1}$. We expect that in the steady-state, P will adjust itself so that $dP/dt = 0$ and $w_{model} = w_{max}$, as desired. To implement this model in COMSOL, we can use the following steps:

Model Wizard





1. Open the Model Wizard and select the 2D axisymmetric spatial dimension.
2. In the Select Physics panel, choose Fluid Flow|Single-Phase Flow|Laminar Flow, and click ‘Add’. Choose also Chemical Species Transport|Transport of Diluted Species, and click ‘Add’ again. Finally, choose Mathematics|ODE and DAE Interfaces|Global ODEs and DAEs and click ‘Add’ a third time. There will now be three physics interfaces that will be added to the model.
3. Click the Study arrow () to open the Select Study panel. Select Time Dependent, and click “Done”. This will exit the Model Wizard, displaying the main COMSOL interface.

Global Definitions

1. Right-click Global Definitions in the model tree and select Parameters. Enter the following details in the Parameters table of the Settings window:

| Name | Expression | Description |
|--------|----------------------------------|----------------------------|
| w_max | 50 [cm/s] | Maximum blood velocity |
| D_drug | $2.56e-4$ [cm ² /min] | Drug diffusion coefficient |
| k_drug | 0.2 [1/hour] | Drug release rate |
| M0 | 49.2 [nmol] | Initial stent drug content |
| rho | 1100 [kg/m ³] | Blood density |
| mu | 3.5 [mPa*s] | Blood viscosity |
| G_in | 1000 [Pa/m] | Input pressure gain factor |

Geometry

1. Select the Geometry 1 node in the model tree. In the settings window, change the default length unit to 'mm'.
2. Right-click Geometry 1 and select Rectangle. Specify the width and height as 1 mm and 9 mm respectively. Click Build Selected ().
3. Right-click Geometry 1 again and select Square. In the Settings panel, enter 0.1 mm for the sidelength, and for the corner position, enter 0.9 mm and 1.45 mm in the r and z fields respectively. Click Build Selected ().
4. Right-click Geometry 1 again and select Transforms|Array. In the Settings panel, specify the input object as the square strut (sq1), the Array type as Linear, and the Size as 13. This specifies that the strut will be copied 13 times and placed in a linear array. For the Displacement settings, enter 0 and 0.5 mm into the r and z fields respectively. Click Build Selected ().
5. Right-click Geometry 1 a final time and select Booleans and Partitions|Difference. For the objects to add, specify the large rectangle (r1). For the objects to subtract, specify the 13 squares (struts) formed from the previous linear array operation. Click Build Selected (). The resulting COMSOL interface with geometry and model tree should look like that shown in Fig. 9.6.

Component 1 Definitions

1. Right-click the Definitions sub-node of Component 1 and select Component Couplings|Maximum. In the settings panel, specify the geometric entity level as 'Boundary', and select boundary 1 corresponding to the symmetry axis. This will create a maximum operator (maxop1) located on the symmetry axis of the model, which will be used to evaluate the maximum fluid velocity on the artery axis, making its value globally available to the rest of the model.
2. Right-click again the Definitions sub-node of Component 1 and select Component Couplings|Integration. In the settings panel, specify the geometric entity level as 'Point', and select point 2. This will create a point integration operator (intop1)

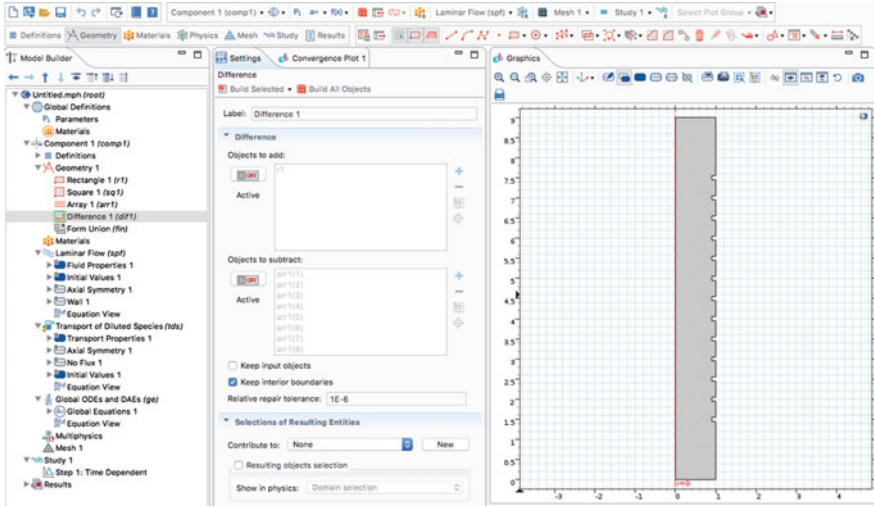


Fig. 9.6 COMSOL interface for stent drug delivery model, showing 2D axisymmetric geometry (right) and model tree (far left). The tiny square insets on the rightmost boundary of the geometry correspond to the individual stent struts

located on the symmetry axis of the outlet boundary, which will be later used to evaluate and plot the drug concentration at that point.

3. Right-click the Definitions sub-node of Component 1 a third time and select Component Couplings|Integration again. In the settings panel, specify the geometric entity level as ‘Boundary’, and select boundaries 4–42, corresponding to all the strut surfaces. This integration operator (intop2) will be used to evaluate the surface area of the struts to determine the boundary flux of drug release.
4. Right-click the Definitions sub-node of Component 1 a final time and select Variables. Specify the following variables and expressions:

| Name | Expression |
|---------|--------------------------------------|
| M | $M0 \cdot \exp(-k \cdot t)$ |
| w_model | $\maxop1(w)$ |
| c_out | $\text{intop1}(c)$ |
| Area | $\text{intop2}(2 \cdot \pi \cdot r)$ |

where variable M represents the total drug content of the stent as function of time, w_model is the maximum z-component of velocity along the symmetry axis, and Area is the surface area of the struts given by $\int 2\pi r ds$ for an axisymmetric geometry, where s is the arc length along the boundary.

Global ODEs and DAEs

1. Select the Global Equations 1 sub-node of Global ODEs and DAEs. In the Settings panel under the Units tab, set the dependent variable quantity to ‘Pressure (Pa)’ from the drop-down list. For the source term quantity, manually enter the units as Pa/s. Under the Global Equations tab, enter the following details on a single row of the equations table:

| Name | f(u,ut,utt,t) | Initial value (u_0) | Description |
|------|-------------------------|---------------------|----------------|
| P | Pt-G_in*(w_max-w_model) | 0 | Inlet pressure |

This defines the global ODE for the inlet pressure $dP/dt - G_{in}(w_{max} - w_{model}) = 0$, with initial value $P(0) = 0$.

Laminar Flow

1. Select the Fluid Properties 1 sub-node of Laminar Flow. In the Settings panel, set the fluid density and dynamic viscosity fields to the user-defined values of ρ_{ho} and μ respectively.
2. Right click the Laminar Flow node to add an ‘Inlet’ boundary condition. Select boundary 2 (i.e. the lower boundary) and specify the boundary condition type as Pressure with a pressure value of P.
3. Right click the Laminar Flow node again to add an ‘Outlet’ boundary condition. Select boundary 3 (i.e. the upper boundary). Leave the boundary condition type to the default value of Pressure, and leave the pressure value at 0.

Study




1. Under the Study 1 node, select the Time Dependent solver. Click the range button () and specify the time range as 0 to 10 in steps of 0.1, and click ‘Replace’. Under the Physics and Variables Selection tab, uncheck the Transport of Diluted Species Interface, leaving Laminar Flow and Global ODEs and DAEs as the only two physics modes solved for in this study.
2. To solve the model, right-click Study 1 and select Compute (). COMSOL will display the default plot of fluid velocity magnitude at 10s, similar to that shown in Fig. 9.7. Clicking on the Zoom Box button () and dragging a zoom box around the bottom two struts closest to the inlet will show a magnified view of the velocity there, as shown also in Fig. 9.7.
3. COMSOL also generates a default 1D global plot of the inlet pressure Global ODE variable, as shown in Fig. 9.8: it can be seen that the value of this pressure has stabilized after 10s, indicating the model has already reached steady-state.

Fig. 9.7 Fluid velocity magnitude across coronary stent at 10 s, showing zoomed-in view around the two struts closest to the inlet boundary

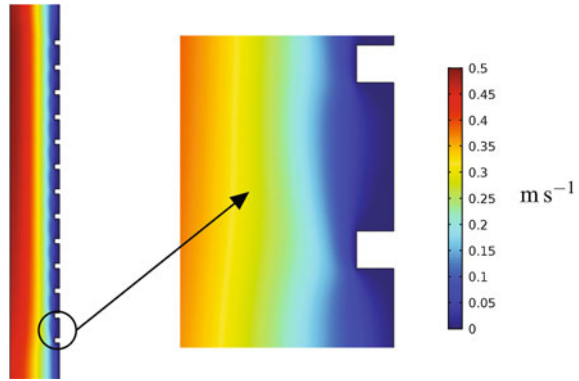
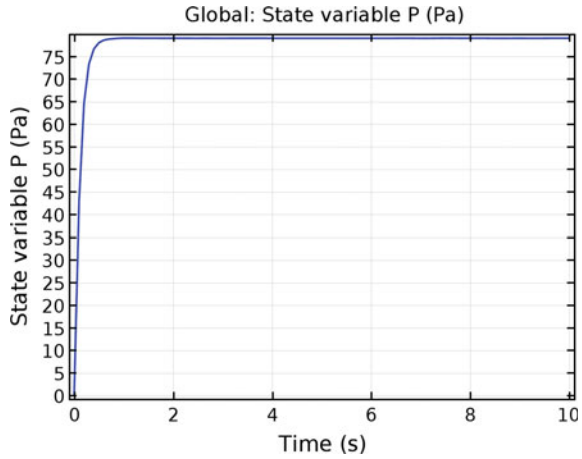



Fig. 9.8 Inlet pressure P as a function of time. It can be seen that the pressure has stabilized at $t = 10$ s, indicating that the desired axial velocity $w_{max} = 0.5 \text{ cm s}^{-1}$ has been reached






Transport of Diluted Species

1. Select the Transport Properties 1 sub-node of the Transport of Diluted Species node. In the Settings panel, set the isotropic diffusion coefficient D_c to the user-defined value of D_{drug} . For the velocity field, specify u for the r component and w for the z component. These are the COMSOL fluid velocity variables from the Laminar Flow physics interface. We will be using these to compute the convection component of drug transport.
2. Right click the Transport of Diluted Species node to add an 'Inflow' boundary condition. Select boundary 2 (i.e. the lower boundary) and specify the boundary condition type as Flux (Danckwerts) with a default concentration of 0. This boundary condition specifies that there is no diffusive flux normal to the boundary, only convective inflow. Furthermore, since we have specified a concentration of 0 at this boundary, this is equivalent to a zero-flux boundary condition.

3. Right click the Transport of Diluted Species node again to add an ‘Outflow’ boundary condition. Select boundary 3 (i.e. the upper boundary). This boundary condition specifies that there is no diffusive flux normal to the boundary, only convective outflow.
4. Right click the Transport of Diluted Species node a third time to add a ‘Flux’ boundary condition. Select boundaries 4–42 corresponding to the strut surfaces, and click the species ‘c’ checkbox. Specify an inward flux of $k_{drug}M/Area$, where we have divided the total transport rate per unit time ($k_{drug}M$) by the surface area of the stent.
5. Click the COMSOL Show button () just above the Model Tree and select ‘Stabilization’. This allows model stabilization settings to be viewed. Click the Transport of Diluted Species node and under the Inconsistent Stabilization tab, select the ‘Isotropic diffusion’ checkbox. This adds a small artificial amount of diffusion proportional to the mesh element size, and is necessary to avoid numerical oscillations whenever convection flow dominates the diffusional flux. Leave the default tuning parameter to its value of 0.25.

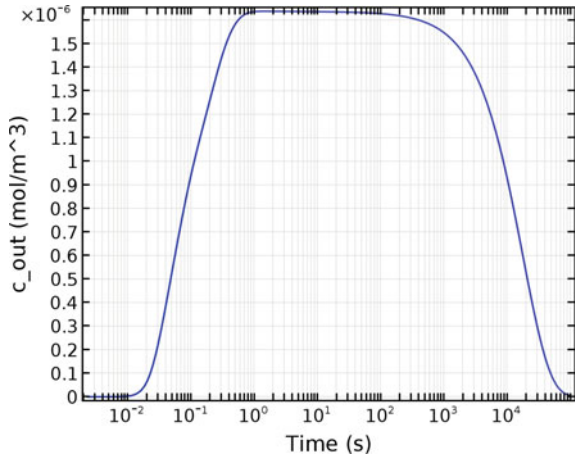
Study



1. Right-click the root node of the model tree and Select ‘Add Study’. Specify ‘Time Dependent’ and click the add study button ().
2. Under the newly created Study 2 node, select the Time Dependent solver. Click the range button () and specify the time range as 0–5 in steps of 0.001. Specify the function to apply to all values as ‘exp10’. Click the ‘Replace’ button. This will result in the expression for the output times of the solver as $10^{\{range(0, 0.001, 5)\}}$. Manually append ‘-1’ to this expression so that it reads $10^{\{range(0, 0.001, 5)\}}-1$: this will create a geometrically increasing time sequence from 0 to $(10^5 - 1)$ s. Under the Physics and Variables Selection tab, uncheck the Laminar Flow and Global ODEs and DAEs physics checkboxes, leaving only the Transport of Diluted Species physics mode to be solved for in this study. Under the Values of Dependent Variables tab, specify ‘User controlled’ for the Values of variables not solved for. For the Method, select ‘Solution’, and for the Study, select ‘Study 1, Time Dependent’, For the Times(s) field, specify ‘Last’.
3. Right-click Study 2 and select Show Default Solver. Under Study 2| Solver Configurations| Solution 2| Time-Dependent Solver 1, select the Time Stepping tab and choose ‘Strict’ for the Time steps taken by solver. Select the Initial step checkbox and specify an initial step value of 1e-6s.
4. To solve for the diffusion/convection physics, right-click Study 2 and select Compute ().

Results

1. Right-click the Result node and select ‘1D Plot Group’. Click on the newly-created 1D Plot Group 5 sub-node and specify the Data set as Study 2/Solution 2.

Fig. 9.9 Drug concentration eluted by stent as a function of time at the axial point on the outflow boundary



2. Right-click the 1D Plot Group 5 sub-node and select ‘Global’. In the settings panel under the y-Axis Data tab, specify the expression to plot as c_{out} . Click the Plot button () as well as the x-axis log scale button () to display the plot of concentration c_{out} against time, as shown in Fig. 9.9. It can be seen that the concentration of the drug at the outflow rapidly increases to its maximum value of $\approx 1.64 \times 10^{-6} \text{ mol m}^{-3}$, before declining again to 0 after 10^5 s , when the drug content of the stent has depleted. This represents about half the peak drug concentration of the former model (see Fig. 7.9), although the time to peak release value has also been delayed, most likely due to the smaller surface area of this model compared with the earlier example.

To solve this model, we utilized a coupled set of physics modes involving fluid mechanics, diffusion and convection as well a global ODE. We also took advantage of the weakly-coupled nature of the model (i.e. drug concentration has no effect on the fluid motion) to solve first for the fluid motion using one study, then use a second study to solve for the diffusion/convection transport from the results of the first. The results of this model compare favourably with those of the earlier example (see Sect. 7.1.4), although peak drug concentration was reduced and time to peak concentration was delayed, likely due to the reduced surface area of the stent with struts compared to the uniform stent surface of the former example.

9.3 Non-laminar Flow

To generalize the behaviour of Newtonian fluids, we can rewrite the Navier-Stokes equations (Eq. 9.8) in their dimensionless form by scaling all variables to dimensionless quantities. This is achieved by choosing characteristic quantities peculiar to a given fluid system, namely:

- a characteristic length L
- a characteristic velocity v
- a characteristic frequency ω

We can then form the dimensionless variables

$$\mathbf{x}' = \frac{\mathbf{x}}{L}, \quad \mathbf{u}' = \frac{\mathbf{u}}{v}, \quad p' = \frac{p}{\rho v^2}, \quad t' = \omega t$$

along with the following:

$$\begin{aligned} \frac{\partial \mathbf{u}'}{\partial t'} &= \frac{\partial \mathbf{u}}{\partial t} \cdot \frac{d\mathbf{u}'}{d\mathbf{u}} \cdot \frac{dt}{dt'} = \left(\frac{1}{v\omega} \right) \frac{\partial \mathbf{u}}{\partial t} \\ \frac{\partial \mathbf{u}'}{\partial x'_1} &= \frac{\partial \mathbf{u}}{\partial x_1} \cdot \frac{d\mathbf{u}'}{d\mathbf{u}} \cdot \frac{dx_1}{dx'_1} = \left(\frac{L}{v} \right) \frac{\partial \mathbf{u}}{\partial x_1} \end{aligned}$$

where x'_1, x_1 are the first components of \mathbf{x}', \mathbf{x} respectively. We obtain a similar relationship for derivatives with respect to the other two components, namely:

$$\begin{aligned} \frac{\partial \mathbf{u}'}{\partial x'_2} &= \left(\frac{L}{v} \right) \frac{\partial \mathbf{u}}{\partial x_2} \\ \frac{\partial \mathbf{u}'}{\partial x'_3} &= \left(\frac{L}{v} \right) \frac{\partial \mathbf{u}}{\partial x_3} \end{aligned}$$

Defining the new del operator ∇' with respect to the dimensionless spatial coordinates x'_1, x'_2, x'_3 , the above expressions lead to the following relationships:

$$\begin{aligned} \nabla' \mathbf{u}' &= \left(\frac{\partial \mathbf{u}'}{\partial x'_1}, \frac{\partial \mathbf{u}'}{\partial x'_2}, \frac{\partial \mathbf{u}'}{\partial x'_3} \right) = \left(\frac{L}{v} \right) \nabla \mathbf{u} \\ \nabla'^2 \mathbf{u}' &= \frac{\partial^2 \mathbf{u}'}{\partial x'^2_1} + \frac{\partial^2 \mathbf{u}'}{\partial x'^2_2} + \frac{\partial^2 \mathbf{u}'}{\partial x'^2_3} \\ &= \frac{\partial}{\partial x'_1} \left[\frac{\partial \mathbf{u}'}{\partial x'_1} \right] + \frac{\partial}{\partial x'_2} \left[\frac{\partial \mathbf{u}'}{\partial x'_2} \right] + \frac{\partial}{\partial x'_3} \left[\frac{\partial \mathbf{u}'}{\partial x'_3} \right] \\ &= \left(\frac{L}{v} \right) \frac{\partial}{\partial x'_1} \left[\frac{\partial \mathbf{u}}{\partial x_1} \right] + \left(\frac{L}{v} \right) \frac{\partial}{\partial x'_2} \left[\frac{\partial \mathbf{u}}{\partial x_2} \right] + \left(\frac{L}{v} \right) \frac{\partial}{\partial x'_3} \left[\frac{\partial \mathbf{u}}{\partial x_3} \right] \\ &= \left(\frac{L}{v} \right) \left\{ \frac{\partial}{\partial x_1} \left[\frac{\partial \mathbf{u}}{\partial x_1} \right] \cdot \frac{dx_1}{dx'_1} + \frac{\partial}{\partial x_2} \left[\frac{\partial \mathbf{u}}{\partial x_2} \right] \cdot \frac{dx_2}{dx'_2} + \frac{\partial}{\partial x_3} \left[\frac{\partial \mathbf{u}}{\partial x_3} \right] \cdot \frac{dx_3}{dx'_3} \right\} \\ &= \left(\frac{L}{v} \right) \left\{ \frac{\partial^2 \mathbf{u}}{\partial x^2_1} \cdot L + \frac{\partial^2 \mathbf{u}}{\partial x^2_2} \cdot L + \frac{\partial^2 \mathbf{u}}{\partial x^2_3} \cdot L \right\} \\ &= \left(\frac{L^2}{v} \right) \nabla^2 \mathbf{u} \end{aligned}$$

Finally, we also obtain

$$\frac{\partial p'}{\partial x'_1} = \frac{\partial p}{\partial x_1} \cdot \frac{dx_1}{dp} \cdot \frac{dp}{dx'_1} = \left(\frac{L}{\rho v^2}\right) \frac{\partial p}{\partial x_1}$$

with similar relationships for the derivative of p' with respect to x'_2 and x'_3 . Hence, we obtain

$$\nabla' p' = \left(\frac{L}{\rho v^2}\right) \nabla p$$

Substituting all these relationships into Eq. 9.8, and ignoring the body force \mathbf{f} , we obtain

$$\begin{aligned} \rho \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right) &= -\nabla p + \mu \nabla^2 \mathbf{u} \\ \rho \left(v\omega \frac{\partial \mathbf{u}'}{\partial t'} + v\mathbf{u}' \cdot \left(\frac{v}{L}\right) \nabla' \mathbf{u}' \right) &= -\left(\frac{\rho v^2}{L}\right) \nabla' p' + \mu \left(\frac{v}{L^2}\right) \nabla'^2 \mathbf{u}' \\ \rho v\omega \frac{\partial \mathbf{u}'}{\partial t'} + \frac{\rho v^2}{L} \mathbf{u}' \cdot \nabla' \mathbf{u}' &= -\frac{\rho v^2}{L} \nabla' p' + \frac{\mu v}{L^2} \nabla'^2 \mathbf{u}' \\ \frac{\omega L}{v} \frac{\partial \mathbf{u}'}{\partial t'} + \mathbf{u}' \cdot \nabla' \mathbf{u}' &= -\nabla' p' + \frac{\mu}{vL\rho} \nabla'^2 \mathbf{u}' \end{aligned}$$

For convenience, we write $\nabla' = \nabla$, with the understanding that the del operator is taken with respect to the dimensionless spatial variables. The above may therefore be re-arranged and written as the dimensionless equation

$$\frac{N_S}{N_R} \frac{\partial \mathbf{u}'}{\partial t'} + \mathbf{u}' \cdot \nabla \mathbf{u}' = -\nabla p' + \frac{1}{N_R} \nabla^2 \mathbf{u}' \tag{9.9}$$

where

$$N_R = \frac{vL\rho}{\mu}, \quad N_S = \frac{\omega L^2\rho}{\mu}$$

are two dimensionless parameters that completely characterise the fluid system. N_R and N_S are known as the *Reynolds number*⁴ and the *Stokes number*⁵ respectively. As the characteristic velocity v increases, N_R also increases, decreasing the relative contribution of the viscous force component $1/N_R \nabla^2 \mathbf{u}'$ in Eq. 9.9. For high values of N_R above 2000, experimental observations indicate that flow becomes unstable and chaotic, forming small swirls or eddies at multiple spatial scales. Such flow is said to be *turbulent*, and is highly sensitive to initial conditions and any surface roughness on the boundaries. Under these conditions, the Navier-Stokes equations (Eq. 9.8) become computationally inefficient, and alternative formulations approximating the eddy viscosities are required for simulating turbulent flow [2], including

⁴Named after the Irish-born Engineer Osborne Reynolds (1842-1912).

⁵After Sir George Gabriel Stokes (1819–1903), of Navier-Stokes equation fame.

the $k - \varepsilon$ model [5] used in COMSOL's optional computational fluid dynamics (CFD) module.)

Bioengineering applications of turbulence include modelling blood flow in the large arteries and heart, where velocities are high enough even under normal conditions to induce turbulence [6].

9.4 Modelling Blood Flow

As noted above, an important application of CFD in bioengineering is the simulation of blood flow. This section will outline the use of electrical analogues of flow, which can be coupled to CFD models as boundary conditions to simulate flow in the cardiovascular system. The non-Newtonian characteristics of blood flow will also be briefly outlined.

9.4.1 Electric Circuit Analogues for Blood Flow

Electric circuit analogues of the circulation have already been introduced in Chap. 2 (see Sect. 2.3.1 and Problem 2.6), where it was seen that fluid pressure and flow were analogous to electric voltage and current respectively. With these basic definitions, it is possible to form a hydraulic circuit representation of blood flow, with circuit elements similar to that of an electric circuit. The following hydraulic elements can be defined, where P is the pressure across the element and Q is the flow through it:

- hydraulic resistance R : analogous to electric resistance, with $P = QR$.
- fluid volume V : analogous to electric charge, with

$$Q = \frac{dV}{dt}$$

- compliance C : analogous to capacitance, with

$$Q = C \frac{dP}{dt}$$

- inertance L : analogous to inductance, with

$$P = L \frac{dQ}{dt}$$

- valve element: analogous to a diode, with $Q = 0$ if $P < 0$.

Hydraulic circuit elements can be combined in series and parallel, with analogous equivalent values as per an electric circuit.

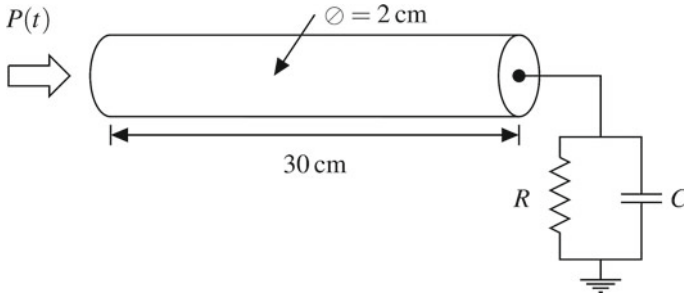


Fig. 9.10 Simplified model of aortic blood flow coupled to hydraulic circuit representation of downstream circulation. Values of R and C are $1.0 \text{ mmHg s cm}^{-3}$ and $2.75 \text{ cm}^3 \text{ mmHg}^{-1}$ respectively

9.4.2 Example: Aortic Blood Flow


Consider the simplified axisymmetric model of the descending thoracic and abdominal aorta shown in Fig. 9.10. Solve this model in COMSOL and plot blood flow against time ($0 \leq t \leq 1 \text{ s}$) for an input half-sine pressure pulse (in mmHg) of:

$$P(t) = \begin{cases} 120 \sin(2\pi t) & 0 \leq t < 0.5 \text{ s} \\ 0 & \text{otherwise} \end{cases}$$

Take the viscosity coefficient of blood to be $\mu = 3.5 \text{ mPa s}$ and density $\rho = 1000 \text{ kg m}^{-3}$.

To implement this model in COMSOL, we can use the following steps:

Model Wizard

1. Open the Model Wizard and select the 2D axisymmetric spatial dimension.
2. In the Select Physics panel, choose Fluid Flow|Single-Phase Flow|Laminar Flow, and click ‘Add’. Select Mathematics| ODEs and DAE interfaces| Global ODEs and DAEs and click ‘Add’ again.
3. Click the Study arrow () to open the Select Study panel. Select Time Dependent, and click “Done”. This will exit the Model Wizard, displaying the main COMSOL interface.

Global Definitions

1. Right-click Global Definitions in the model tree and select Parameters. Enter the following details in the Parameters table of the Settings window:

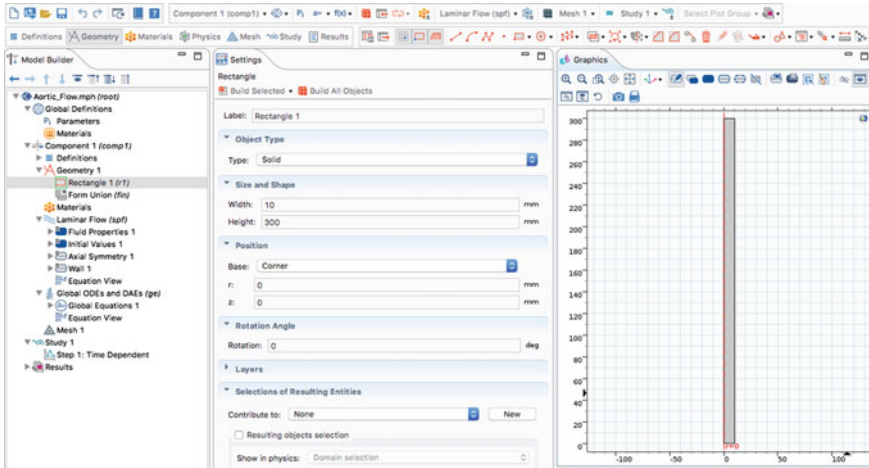


Fig. 9.11 COMSOL interface for aortic flow example model, showing 2D axisymmetric geometry (right) and model tree (far left). The inlet boundary is located at the top of the 2D rectangular domain, and the outlet boundary, which connects to the systemic circulation, is at the bottom

| Name | Expression | Description |
|-------|------------------------------|------------------------|
| mu | 3.5 [mPa*s] | Blood viscosity |
| rho | 1100 [kg/m ³] | Blood density |
| P_max | 120 [mmHg] | Maximum blood pressure |
| R | 1 [mmHg*s/cm ³] | Systemic resistance |
| C | 2.75 [cm ³ /mmHg] | Systemic compliance |

Geometry

1. Select the Geometry 1 node in the model tree. In the settings window, change the default length unit to ‘mm’.
2. Right-click Geometry 1 and select Rectangle. Specify the width and height as 10 and 300 mm respectively. Click Build Selected (). The resulting COMSOL interface with geometry and model tree should look like that shown in Fig. 9.11.

Component 1 Definitions

1. Right-click the Definitions sub-node of Component 1 and select Component Couplings|Integration. In the settings panel, specify the geometric entity level as ‘Boundary’, and select boundary 2 (i.e. the lower boundary). This will create a boundary integration operator (intop1) over the outlet boundary, which will be used to determine the blood flow into the systemic circulation.
2. Right-click the Definitions sub-node of Component 1 again and select Variables. Specify the following variables and expressions:

| Name | Expression |
|------|---|
| tt | mod(t, (1 [s])) |
| P_in | if(tt<0.5, P_max*sin(2*pi*tt/(1 [s])), 0) |
| Q | intop1(-2*pi*r*w) |

where variable w represents the z -component of velocity parallel to the symmetry axis, and Q is the flow out of the lower outlet boundary, given by $\int_0^R -2\pi r w dr$ for an axisymmetric geometry, where R is the radius of the aorta, and w is the z -component of blood velocity (i.e. parallel to the symmetry axis), with positive values denoting upward flow.

Global ODEs and DAEs

1. Select the Global Equations 1 sub-node of Global ODEs and DAEs. In the Settings panel under the Units tab, set the dependent variable quantity to ‘Pressure (Pa)’ from the drop-down list. For the source term quantity, select ‘Volume per time (m³/s)’ from the drop-down list. Under the Global Equations tab, enter the following details on a single row of the equations table:

| Name | f(u,ut,utt,t) | Initial value (u_0) | Description |
|-------|--------------------|---------------------|-----------------|
| P_out | C*P_outt+P_out/R-Q | 0 | Outlet pressure |

This defines the global ODE for the outlet pressure $CdP_{out}/dt + \frac{P_{out}}{R} = Q$, with initial value $P_{out}(0) = 0$. Note that the COMSOL term $P_{out}t$ denotes the derivative of variable P_{out} with respect to time.

Laminar Flow

1. Select the Laminar Flow node and under ‘Compressibility’, select ‘Incompressible flow’.
2. Select the Fluid Properties 1 sub-node of Laminar Flow. In the Settings panel, set the fluid density and dynamic viscosity fields to the user-defined values of ρ and μ respectively.
3. Right click the Laminar Flow node to add an ‘Inlet’ boundary condition. Select boundary 3 (i.e. the upper boundary) and specify the boundary condition type as ‘Pressure’ with a pressure value of P_{in} . Uncheck the ‘Suppress backflow’ checkbox to allow flow to reverse at the inlet.
4. Right click the Laminar Flow node again to add an ‘Outlet’ boundary condition. Select boundary 2 (i.e. the lower boundary). Leave the boundary condition type to the default value of Pressure, with pressure value P_{out} . As above, uncheck the ‘Suppress backflow’ checkbox to allow flow to reverse (due to discharge of the systemic compliance).

Fig. 9.12 Simulation result of blood velocity magnitude in aorta at 1 s (*left*), showing zoomed-in view around the inlet boundary (*right*). In each case, the axis of symmetry is the left-most edge. Note that the velocity profile is not parabolic with maximum velocity at the axis, since there is an oscillatory pressure gradient

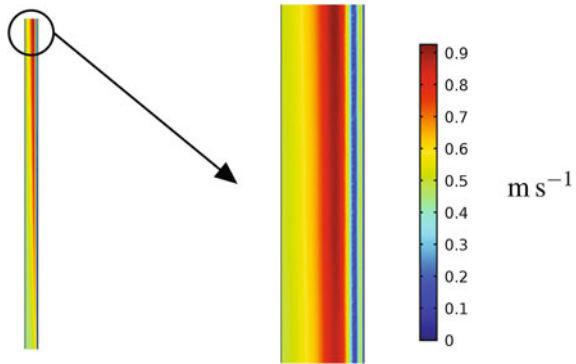
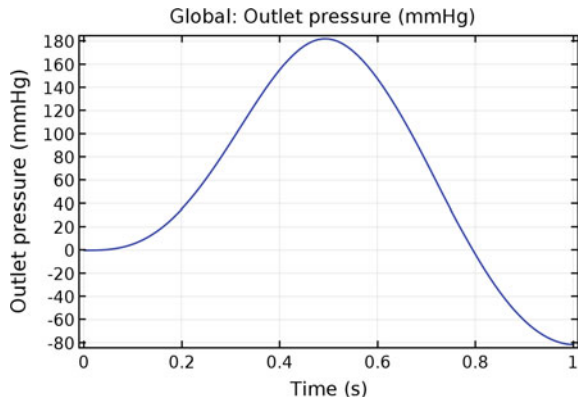





Fig. 9.13 Global outlet pressure variable P_{out} as a function of time



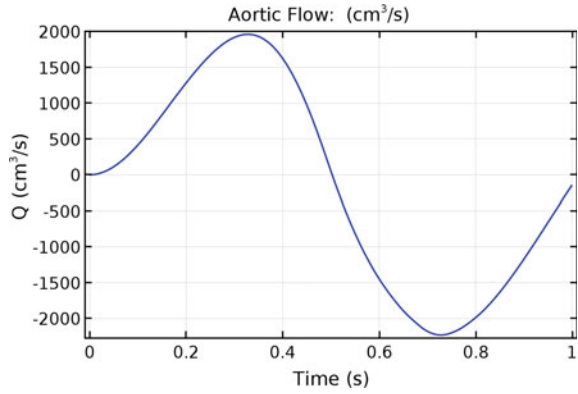
Study


1. Under the Study 1 node, select the Time Dependent solver. Click the range button () and specify the time range as 0 to 1 s in steps of 0.01 s, and click 'Replace'.
2. To solve the model, right-click Study 1 and select Compute (). COMSOL will display the default plot of fluid velocity magnitude at 1 s, similar to that shown in Fig. 9.12.
3. COMSOL also generates a default 1D global plot of the outlet pressure Global ODE variable. Select the Global 1 sub-node of 1D Plot Group 4 (under the Results node), and under the y-Axis data tab, change the default units of the expression 'P_out' from 'Pa' to 'mmHg'. Clicking the Plot button () will generate the graph of outlet pressure, as shown in Fig. 9.13.

Results

1. Right-click the Result node and select '1D Plot Group'. Right-click the 1D Plot Group 5 sub-node and select 'Global'. In the settings panel under the y-Axis Data tab, specify the expression to plot as Q , and modify the units to be ' cm^3/s '.

Fig. 9.14 Aortic blood flow Q as a function of time



2. Select the 1D Plot Group 5 sub-node and under the Plot Settings tab, check the y-axis label checkbox and enter the expression ' Q (cm³/s)'.⁶ Under the Title tab, select the title type to be manual, and edit the expression to be 'Aortic Flow: (cm³/s)'. Finally, under the Legend tab, uncheck the checkbox 'Show legends'. Click the Plot button () to display the plot of aortic blood flow Q against time, as shown in Fig. 9.14.

9.4.3 Blood as a Non-newtonian Fluid

Unlike many fluids which are well-approximated by Newtonian behaviour, blood consists of a *suspension* of red cells (erythrocytes) in plasma, which can lead to a non-linear relationship between shear rate and viscosity, as well as changes in viscosity due to vessel diameter. In most cases however, the Newtonian assumption for blood still remains valid, although for some cases outlined below, there may be a significant departure from Newtonian assumptions.

9.4.3.1 Blood Viscosity and Shear Rate

Recall that the strain rate tensor \mathbf{V} , with components V_{ij} , is defined as

$$V_{ij} = \frac{1}{2} \left(\frac{\partial v_j}{\partial x_i} + \frac{\partial v_i}{\partial x_j} \right)$$

where the v_i are the components of fluid velocity. The off-diagonal terms of \mathbf{V} define the components of *shear rate*. For blood, low values of shear rate less than 100 s^{-1} lead to aggregation or clumping together of red blood cells, leading to an increase

⁶Note that $\langle \sup \rangle 3 \langle \sup \rangle$ will superscript the '3'.

in the viscosity. At high shear rates above 100 s^{-1} (typically $200\text{--}300\text{ s}^{-1}$), blood behaves essentially as a Newtonian fluid.

9.4.3.2 Blood Viscosity and Vessel Diameter

For blood vessels with internal diameter greater than 1 mm, blood viscosity approaches its asymptotic value (Newtonian condition) at shear rates higher than $200\text{--}300\text{ s}^{-1}$, as described above. However for vessels less than 1 mm in diameter, the value of viscosity is constant (i.e. independent of shear rate), but varies with vessel diameter. For vessels of diameter $300\text{ }\mu\text{m}$, the viscosity is about 95% of its asymptotic value at 1 mm. For vessels of diameter $20\text{ }\mu\text{m}$, the viscosity is about 50% of its asymptotic value. This effect was first described by Fåhræus⁷ and Lindqvist⁸ in 1931, and is consequently known as the *Fåhræus-Lindqvist* effect. The effect is due to *axial streaming*, whereby erythrocytes become concentrated at the vessel axis during laminar flow because of the higher velocity near the centre. The velocity gradient pulls the long axis of the erythrocytes parallel to the direction of flow and forces the cell toward the centre. This leaves a cell-free sleeve of plasma near the wall, decreasing the viscosity of the blood. The effect is more pronounced in smaller vessels, due to the larger velocity gradient present over the dimensions of a single blood cell.

9.4.4 Example: Axial Streaming of a Blood Cell

In order to simulate the effect of axial streaming, we consider a simple 2D model of a blood vessel with diameter $80\text{ }\mu\text{m}$, in which is placed a rounded rectangular cell of dimensions of 4 and $10\text{ }\mu\text{m}$, located initially at a radial offset of $20\text{ }\mu\text{m}$ from the vessel axis, as shown in Fig. 9.15. The fluid in the vessel is modelled as blood plasma with density $\rho = 1000\text{ kg m}^{-3}$ and viscosity $\mu = 1.2\text{ mPa s}$. Instead of modelling the entire vessel length, we consider only a segment of length $100\text{ }\mu\text{m}$ which travels along with the cell along the main vessel axis. We assume there is a pressure differential of 0.1 mmHg continually applied between the inlet (left) and outlet (right) domain boundaries. The cell is free to displace laterally in the y -direction through the plasma towards the vessel axis, as well as free to rotate about its centre. To determine the moment of inertia of the cell, we assume it to be approximated by a rectangular prism of dimensions $4 \times 10 \times 10\text{ }\mu\text{m}$, in which its depth perpendicular to the 2D domain is $10\text{ }\mu\text{m}$. The model will demonstrate some useful multiphysics capabilities of COMSOL, including its *moving mesh* interface.

To implement this model of axial streaming in COMSOL, we can employ the following steps:

⁷Robert (Robin) Sanno Fåhræus (1888–1968), Swedish pathologist and haematologist.

⁸Johan Torsten Lindqvist (1906–2007), Swedish physician.

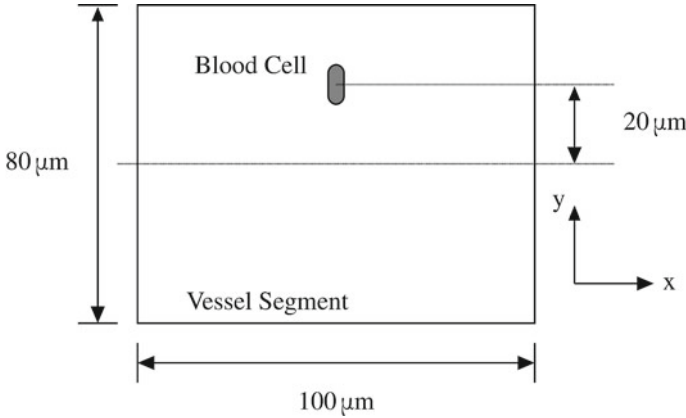



Fig. 9.15 Simplified 2D model of axial streaming in a 100 μm blood vessel segment of diameter 80 μm. A single red blood cell is placed at an offset of 20 μ from the vessel axis in the y-direction. Blood flows from left to right in the x-direction. The vessel domain is assumed to travel in the x-direction along with the blood cell, so that the cell is always centred in the domain with respect to the x-axis. The cell is free to migrate along the y-axis, as well as rotate about its centre

Model Wizard

1. Open the Model Wizard and select the 2D spatial dimension.
2. In the Select Physics panel, choose Fluid Flow|Single-Phase Flow|Laminar Flow, and click ‘Add’. Select Mathematics|ODEs and DAE interfaces|Global ODEs and DAEs and click ‘Add’ again. Select Mathematics|ODEs and DAE interfaces|Global ODEs and DAEs a second time and click ‘Add’ a third time. Finally, select Mathematics|Deformed mesh|Deformed Geometry and click ‘Add’.
3. Click the Study arrow () to open the Select Study panel. Select Time Dependent, and click “Done”. This will exit the Model Wizard, displaying the main COMSOL interface.

Global Definitions

1. Right-click Global Definitions just beneath the root node of the model tree and select Parameters. Enter the following details in the Parameters table of the Settings window:

| Name | Expression | Description |
|------|---|-------------------|
| mu | 1.2 [mPa*s] | Plasma viscosity |
| rho | 1000 [kg/m^3] | Plasma density |
| m | 27 [pg] | Blood cell mass |
| I | $m/12 * ((4 [\text{um}])^2 + (10 [\text{um}])^2)$ | Moment of inertia |

where the last parameter *I* is the moment of inertia of the blood cell, determined from the theoretical moment of inertia of a rectangular prism about its central axis,

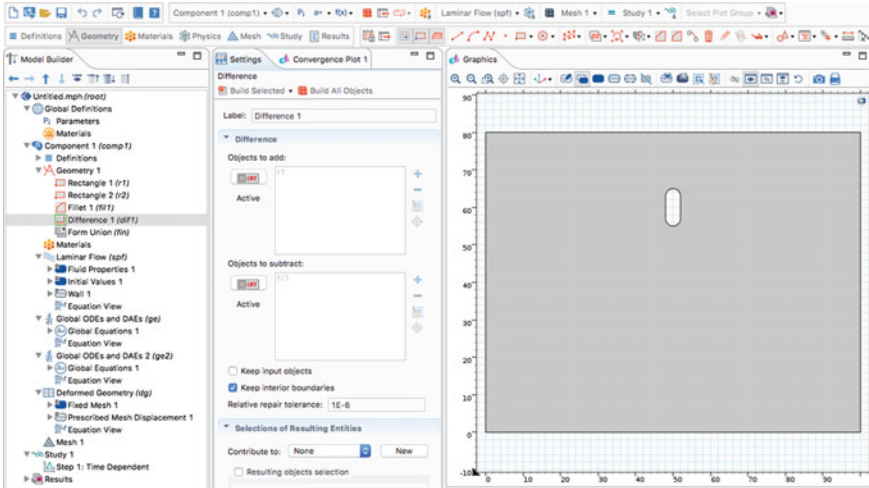






Fig. 9.16 COMSOL interface for axial streaming model, showing 2D vessel segment geometry with blood cell (*rightmost panel*) and model tree (*leftmost panel*). In the model geometry, the inlet boundary is located at the *left edge* of the 2D domain, whilst the outlet boundary is at the *right edge*

given by $I = \frac{1}{12}m(a^2 + b^2)$, where a, b are the dimensions of the rectangular cross-section perpendicular to the axis, and m is the mass.

Geometry

1. Select the Geometry 1 node in the model tree. In the settings window, change the default length unit to ‘ μm ’.
2. Right-click Geometry 1 and select Rectangle. Specify the width and height as 100 and $80\mu\text{m}$ respectively. Click Build Selected ().
3. Right-click the Geometry 1 node again and select Rectangle for a second time. Specify the width and height as 4 and $10\mu\text{m}$ respectively. Also specify the x, y positions of the centre as 50 and $60\mu\text{m}$ respectively. Click Build Selected ().
4. Right-click Geometry 1 again and select Fillet. In the settings window, specify the vertices to be filleted as 1–4, corresponding to the four vertices of the smaller rectangle (i.e. the blood cell). Specify the fillet radius as $2\mu\text{m}$ and click Build Selected ().
5. Right-click Geometry 1 a final time and select Booleans and Partitions| Difference. In the settings panel, select the rectangle (r1) as the object to add. Click the objects to subtract toggle button to make it active, and select the filleted rectangle (fil1). Click Build Selected () to subtract the blood cell from the vessel segment. The resulting COMSOL interface with geometry and model tree should look like that shown in Fig. 9.16.

Component 1 Definitions

1. Right-click the Definitions sub-node of Component 1 and select Component Couplings|Integration. In the settings panel, specify the geometric entity level as ‘Boundary’, and select boundaries 4, 5, 7–10 (i.e. the cell boundaries). This will create a boundary integration operator (intop1) over the cell boundary, which will be used to determine the forces and moments acting on the cell due to the blood plasma.
2. Right-click the Definitions sub-node of Component 1 again and select Variables. Specify the following variables and expressions:

| Name | Expression |
|--------|--|
| Fx | $(-10 [\mu\text{m}]) * \text{intop1}(\text{spf.T_stressx})$ |
| Fy | $(-10 [\mu\text{m}]) * \text{intop1}(\text{spf.T_stressy})$ |
| rr_x | $x - (50 [\mu\text{m}])$ |
| rr_y | $y - (60 [\mu\text{m}]) - \text{cell_y}$ |
| rr_X | $Xg - (50 [\mu\text{m}])$ |
| rr_Y | $Yg - (60 [\mu\text{m}])$ |
| Torque | $(-10 [\mu\text{m}]) * \text{intop1}(\dots - \text{spf.T_stressx} * \text{rr_y} + \text{spf.T_stressy} * \text{rr_x})$ |

where `spf.T_stressx`, `spf.T_stressy` are in-built variables in COMSOL representing the x- and y-components of the stress vector (traction) acting on the fluid at the domain boundaries, and `Xg`, `Yg` are the COMSOL variables defining the initial coordinates of the undeformed mesh within the fluid domain. The remaining user-defined variables are described below:

- `Fx`, `Fy` represent the x- and y-components of force acting on the cell, and result from stresses acting in the negative direction to the traction on the fluid. The components of the traction are integrated over the cell boundary, and multiplied the $10 \mu\text{m}$ depth of the cell perpendicular to the 2D domain, to yield `Fx`, `Fy` in units of Newtons.
- `rr_x`, `rr_y` are the x- and y-components of radial displacement from the cell centre in its moved state. The centre of the cell has shifted from $(50, 60) \mu\text{m}$ to $(50, 60 + \text{cell_y}) \mu\text{m}$, where `cell_y` denotes the displacement of the cell in the y-direction (to be defined later). Note that `x`, `y` denote the standard spatial coordinates, which will shift along with each point as it moves in the deformed mesh.
- `rr_X`, `rr_Y` are the x- and y-components of radial displacement from the cell centre in the undeformed (i.e. initial) state. These variables are useful for describing the displacement of the cell due to its rotation.
- `Torque` represents the torque acting on the cell due to stresses from the fluid. These stresses act in the negative direction from the traction on the fluid boundaries. If the radial vector from the centre of the cell to a point on its boundary is given by $(\text{rr_x}, \text{rr_y})$, then its perpendicular unit

vector is given by $(-rr_y, rr_x)/r$, where r denotes the magnitude of the radial vector. Components of the traction along this vector will contribute to the torque. Since torque is defined by Fr , where r is the radius and F is the magnitude of force acting perpendicular to the radius, the total torque is determined by integrating the negative components of traction along this perpendicular vector, multiplying by the radius and the depth of the cell. The traction along this perpendicular vector is simply $\text{spf.T_stressx} \cdot -rr_y/r + \text{spf.T_stressy} \cdot rr_x/r$. Multiplying by the radius r , the depth of the cell ($10\mu\text{m}$), and taking the negative, yields the total torque as $(-10[\mu\text{m}]) \cdot \text{intop1}(\dots -\text{spf.T_stressx} \cdot rr_y + \text{spf.T_stressy} \cdot rr_x)$.

Note that some of these variables will appear in ‘orange’ as a COMSOL warning, since variables `cell_x` and `cell_y` have not yet been defined. These variables will be implemented next in the Global ODEs and DAE interface below.

Global ODEs and DAEs

1. Select the Global Equations 1 sub-node of Global ODES and DAEs. In the Settings panel under the Units tab, set the dependent variable quantity to ‘Displacement field (m)’ from the drop-down list. For the source term quantity, select ‘Force load (N)’ from the drop-down list. Under the Global Equations tab, enter the following details on two rows of the equations table, leaving all initial values to their default setting of zero:

| Name | f(u,ut,utt,t) | Description |
|--------|----------------------------------|---------------------|
| cell_x | $m \cdot \text{cell_xtt} - F_x$ | Cell x-displacement |
| cell_y | $m \cdot \text{cell_ytt} - F_y$ | Cell y-displacement |

This defines two global ODEs for the cell displacements $m \cdot d^2c_x/dt^2 = F_x$, and $m \cdot d^2c_y/dt^2 = F_y$, where m is the mass of the cell and $c_x \equiv \text{cell_x}$, $c_y \equiv \text{cell_y}$. Note that the terms `cell_xtt`, `cell_ytt` in COMSOL denote the second-derivatives of variables `cell_x`, `cell_y` with respect to time.

Global ODEs and DAEs 2

1. Select the Global Equations 1 sub-node of Global ODES and DAEs 2. In the Settings panel under the Units tab, set the dependent variable quantity to ‘None’ from the drop-down list and manually enter the units as ‘rad’. For the source term quantity, select ‘Torque (N*m)’ from the drop-down list. Under the Global Equations tab, enter the following details on a single row of the equation table, leaving initial values to their default setting of zero:

| Name | f(u,ut,utt,t) | Description |
|-------|------------------|-----------------------|
| theta | I*thetatt-Torque | Cell angular rotation |

This defines a global ODE for the angle of rotation of the cell `theta`, such that $I \cdot d^2\theta/dt^2 = T$, where $\theta \equiv \text{theta}$, $T \equiv \text{Torque}$, and I is the moment of inertia. Note that `thetatt` denotes the second-derivative of `theta` with respect to time. Also note that a second Global ODEs and DAEs node is implemented in the model, since the units of variable `theta` are different from variables `cell_x`, `cell_y` defined in the first Global ODEs and DAEs node.

Deformed Geometry

1. Select the Deformed Geometry node and under ‘Mesh smoothing type’, select ‘Yeoh’. Leave the mesh stiffening parameter to its default value of 100. This setting solves for the mesh deformation within the fluid domain by minimising a strain energy function of the form

$$W = \frac{1}{2} \int_{\Omega} (I_1 - 3) + k (I_1 - 3)^2 + \kappa (J - 1)^2 \, dV$$

where Ω is the mesh domain, k is the stiffening factor (set to a default value of 100), I_1 is the first invariant of the mesh strain tensor (see Eq. 8.15), κ is an artificial bulk modulus, and J is the volumetric strain (see Eq. 8.30). This mesh smoothing approach allows large deformations of the mesh, whilst preserving overall mesh quality.

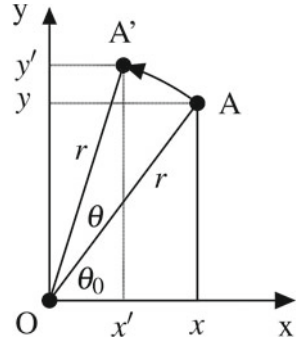
2. Right click the Deformed Geometry node and select ‘Free Deformation’. In the Settings panel, specify `domain1` (i.e. the blood plasma domain).
3. Right click the Deformed Geometry node again and select ‘Prescribed Mesh Displacement’. Select boundaries 4, 5, 7–10 (i.e. the cell surface boundaries). Specify the x and y displacements of the mesh on these boundaries as

- $d_x: \text{rr_X} * (\cos(\text{theta}) - 1) - \text{rr_Y} * \sin(\text{theta})$
- $d_y: \text{cell_y} + \text{rr_X} * \sin(\text{theta}) + \text{rr_Y} * (\cos(\text{theta}) - 1)$

where d_x and d_y denote the x and y displacements respectively. These displacement expressions are derived from the rotation of the cell counter-clockwise by angle `theta`, added to a displacement in the y -direction by `cell_y`.⁹ The displacements due to rotation can be understood in reference to Fig. 9.17. Assume the cell centre is located at point O, and a point on the cell boundary is located at A with coordinates (x, y) relative to O. The radial segment OA with length r initially makes an angle θ_0 with respect to the x -axis. After counter-clockwise rotation by θ about O, point A moves to A' with coordinates (x', y') , such that segment OA' is also of length r . Coordinates x', y' can be determined from

⁹Note that the mesh is not displaced in the x -direction by `cell_x`, since the domain is moving along with the cell in this direction.

Fig. 9.17 Displacement of point A to A' due to counter-clockwise rotation about the origin O by an angle θ



$$\begin{aligned}
 x' &= r \cos(\theta + \theta_0) \\
 &= r \cos \theta \cos \theta_0 - r \sin \theta \sin \theta_0 \\
 &= r \cos \theta \left(\frac{x}{r}\right) - r \sin \theta \left(\frac{y}{r}\right) \\
 &= x \cos \theta - y \sin \theta
 \end{aligned}$$

and

$$\begin{aligned}
 y' &= r \sin(\theta + \theta_0) \\
 &= r \sin \theta \cos \theta_0 + r \cos \theta \sin \theta_0 \\
 &= r \sin \theta \left(\frac{x}{r}\right) + r \cos \theta \left(\frac{y}{r}\right) \\
 &= x \sin \theta + y \cos \theta
 \end{aligned}$$

with x, y displacements given by:

$$\begin{aligned}
 d_x &= x' - x = x (\cos \theta - 1) - y \sin \theta \\
 d_y &= y' - y = x \sin \theta + y (\cos \theta - 1)
 \end{aligned}$$

Replacing x and y in these expressions with the initial relative coordinates of each point on the cell boundary relative to the cell centre, namely rr_X and rr_Y respectively, we obtain the mesh displacement expressions used earlier for the cell boundaries due to rotation of the cell by an angle $\theta = \text{theta}$.


Laminar Flow

1. Select the Laminar Flow node and under 'Compressibility', leave the default setting as 'Incompressible flow'.
2. Select the Fluid Properties 1 sub-node of Laminar Flow. In the Settings panel, set the fluid density and dynamic viscosity fields to the user-defined values of ρ and μ respectively.

3. Right click the Laminar Flow node to add an ‘Inlet’ boundary condition. Select boundary 1 (i.e. the left boundary) and specify the boundary condition type as ‘Pressure’ with a pressure value of 0.1 [mmHg].
4. Right click the Laminar Flow node again to add an ‘Outlet’ boundary condition. Select boundary 6 (i.e. the right boundary). Leave the boundary condition type to the default value of Pressure, with pressure value of 0.
5. Right click the Laminar Flow node a third time and select ‘Wall’. In the Settings panel, select boundaries 4, 5, 7–10 (i.e. the cell boundaries). Specify the boundary condition type to Moving wall from the dropdown list, with the following velocity components:

- v_x : $\text{cell_xt} - \text{thetat} * \text{rr_y}$
- v_y : $\text{cell_yt} + \text{thetat} * \text{rr_x}$

where v_x, v_y are the components of velocity in the x and y directions respectively. These components are determined from the translational velocity of the cell, with x, y components of cell_xt and cell_yt respectively, as well as the rotational velocity components. The latter are determined from $v = \omega r$, where ω is the angular velocity (i.e. thetat) and r is radius. This velocity is directed perpendicular to the radial vector, that is, in the direction of the unit vector $(-\text{rr_y}/r, \text{rr_x}/r)$. Hence the x-component of velocity due to rotation is $\omega \times -\text{rr_y}$, with the corresponding y-component of velocity being $\omega \times \text{rr_x}$. Adding these to the translation velocities gives the moving wall velocities above.

6. Click the View button () just above the model tree and select ‘Advanced Physics Options’. Select the Wall 2 node, and in the Settings panel, under Constraint Settings, select the ‘Use weak constraints checkbox’. This allows COMSOL to use time-derivatives of variables (namely cell_xt , cell_yt and thetat) in the moving wall velocity settings.

Mesh


1. Right-click the Mesh 1 node and select ‘Size’. In the Settings panel, specify the Geometric entity level as ‘Boundary’, and select boundaries 4, 5, 7–10. Under the Element Size tab, select ‘Custom’ and specify a maximum element size of $0.5 \mu\text{m}$.
2. Select the Size sub-node of Mesh 1 (just above the Size 1 sub-node), and specify a pre-defined mesh size of ‘Extremely fine’ from the drop-down list. Now check the ‘Custom’ radio button and overwrite the default maximum element size of $1 \mu\text{m}$ to the new setting of $0.7 \mu\text{m}$.
3. Right-click Mesh 1 and select ‘Boundary Layers’. Select the Boundary Layer Properties sub-node of Boundary Layers 1. In the Settings panel, specify boundaries 4, 5, 7–10 (i.e. the cell boundaries) as well boundaries 2 and 3 (the bottom and top domain boundaries). Under the Boundary Layer Properties tab, specify the number of boundary layers as 16.
4. Right-click Mesh 1 again and select ‘Free Triangular’. Leave all settings to their default values. Clicking the Build All button () will result in a plot of the mesh

Fig. 9.18 Zoomed-in view of initial mesh around the blood cell, showing boundary mesh layers surrounded by free-triangular elements

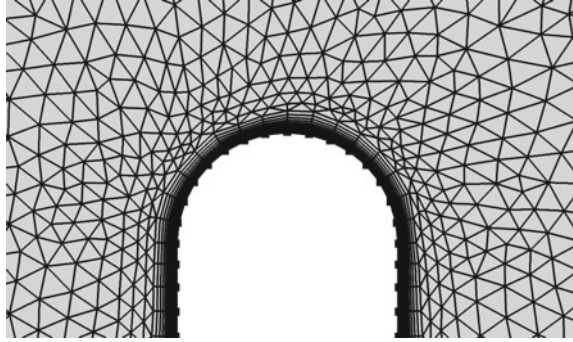
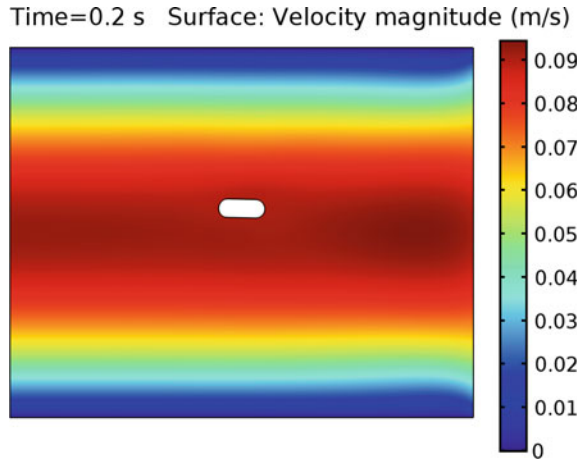


Fig. 9.19 Blood plasma velocity magnitude in vessel segment at 0.2 s. Note that the blood cell has rotated and shifted from its initial geometric specification towards the vessel centre

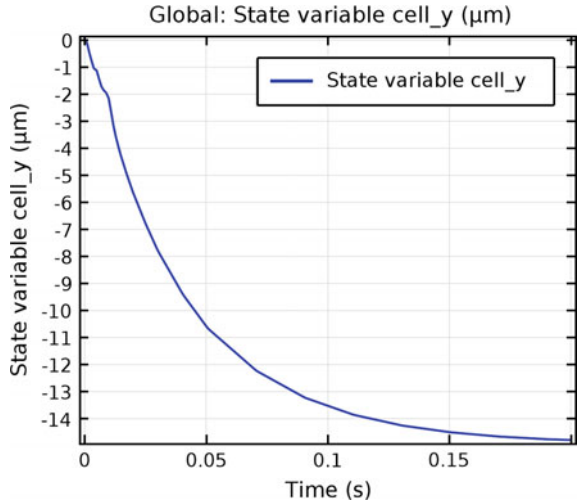


elements, for which a zoomed-in view in the vicinity of the blood cell is shown in Fig. 9.18.



Study

1. Under the Study 1 node, select the Time Dependent solver. Click the range button (📏) and specify the time range as 0–0.2 s in steps of 0.001 s, and click ‘Replace’.
2. Right-click Study 1 and select ‘Show Default Solver’. Under Solver Configurations|Solution 1|Time-Dependent Solver 1|Fully Coupled 1, select the ‘Method and Termination’ tab and specify the Nonlinear method as ‘Automatic (Newton)’.
3. Select the Step 1: Time Dependent sub-node of Study 1 and under the Results While Solving tab, click the ‘Plot’ checkbox. This provides an intermediate plot of results during the solution process.
4. To solve the model, right-click Study 1 and select Compute (🔍). COMSOL will display the default plot of fluid velocity magnitude at 0.2 s, similar to that shown

Fig. 9.20 Plot of state variable `cell_y` as a function of time, representing the lateral displacement of the blood cell from its starting position. At 0.2s, the cell has displaced $15\ \mu\text{m}$ towards the vessel axis (located at $-20\ \mu\text{m}$), in accordance with the axial streaming effect



in Fig. 9.19, where it can be seen that the blood cell has aligned itself with the flow and drifted toward the centre of the vessel.

5. COMSOL also generates default plots of the Global ODE variables `cell_x`, `cell_y` and `theta`. To visualize the lateral displacement of the cell (i.e. `cell_y`) as a function of time, select the Global 1 sub-node of 1D Plot Group 3 (under the Results node), and under the y-Axis data tab, select the row containing the expression `cell_x` and click Delete (). Clicking the Plot button () will generate the graph of lateral cell displacement, as shown in Fig. 9.20, where it can be seen that the cell is moving towards the vessel axis, corresponding to a displacement of $-20\ \mu\text{m}$ from its starting position.

9.5 Further Reading

An excellent introductory text on fluid mechanics is that of Massey and Ward-Smith [3]. A more general treatment of fluid and solid mechanics, with interesting examples covering a range of physics, is the text of Trefil [7]. More specific texts on fluid mechanics principles underlying blood flow in the circulation are those of Nichols and O'Rourke [4] and Fung [1].

Problems

9.1 Consider a circular tube of diameter D and length L containing an incompressible Newtonian fluid of density ρ and viscosity μ . Initially, the upstream and

downstream pressures at each end of the tube are 0 and the fluid is at rest. Suddenly, the upstream pressure is instantaneously stepped to a value of P . By approximating this system as a concentric set of circular tubes of fluid sliding past each other, determine the governing 1D PDE and associated boundary conditions for the axial fluid velocity v in terms of radial position r and time t . Use COMSOL to solve this PDE for the following parameters: $D = 10$ mm, $L = 100$ mm, $\rho = 1000$ kg m⁻³, $\mu = 2$ mPa s, and $P = 10$ mmHg. Plot the axial velocity as a function of radial position for times 0, 0.5, 1, and 1.5 s.

9.2 In the example of Sect. 9.1.1, a simple axisymmetric COMSOL model was implemented to determine the steady-state fluid velocity profile in a cylindrical tube of diameter 2 cm, length 15 cm, fluid viscosity 3.5 mPa s, and pressure differential of 100 Pa between the ends of the tube. Use COMSOL to perform a mesh convergence analysis on this model, solving for a free-triangular mesh automatically generated at maximum element sizes of 0.1–1 mm in steps of 1 mm. Plot the axial velocity at the inlet against maximum element size. What maximum element size is required to achieve an axial velocity error of less than 5%?

9.3 The descending thoracic and abdominal aorta may be modelled as a cylindrical tube of diameter 20 mm and length 30 cm, terminated at its distal end by a simple RC hydraulic-circuit, similar to the example of Sect. 9.4.2. In this case, however, the inlet pressure $P(t)$ is a pressure pulse whose values are shown below as a function of time:

| Time (ms) | Pressure (mmHg) | Time (ms) | Pressure (mmHg) | Time (ms) | Pressure (mmHg) |
|-----------|-----------------|-----------|-----------------|-----------|-----------------|
| 0 | 72 | 360 | 104 | 720 | 86 |
| 40 | 78 | 400 | 93 | 760 | 86 |
| 80 | 90 | 440 | 81 | 800 | 83 |
| 120 | 112 | 480 | 77 | 840 | 79 |
| 160 | 132 | 520 | 77 | 880 | 77 |
| 200 | 148 | 560 | 78 | 920 | 75 |
| 240 | 149 | 600 | 81 | 960 | 73 |
| 280 | 135 | 640 | 85 | 1000 | 72 |
| 320 | 122 | 680 | 86 | | |

As in Sect. 9.4.2, assume the material parameters of blood are $\mu = 3.5$ mPa s and $\rho = 1100$ kg m⁻³. Also assume the blood is initially at rest: for this to be the case, the initial values of pressure at both ends of the tube must be the same (72 mmHg). Simulate this model using COMSOL and plot aortic blood flow as a function of time over 1 s.

HINT: Use COMSOL's interpolation function feature to specify the inlet pressure waveform as a table.

References

1. Fung YC (1997) *Biomechanics: circulation*, 2nd edn. Springer, New York
2. Layton W (2008) *Introduction to the numerical analysis of incompressible viscous flows*. SIAM, Pittsburgh
3. Massey BS, Ward-Smith J (2012) *Mechanics of fluids*, 9th edn. Spon Press, New York
4. Nichols WW, O'Rourke MF (2005) *McDonald's blood flow in arteries: theoretical, experimental and clinical principles*, 5th edn. Hodder Arnold, London
5. Pope SB (2000) *Turbulent flows*. Cambridge University Press, Cambridge
6. Sabbah HN, Stein PD (1976) Turbulent blood flow in humans: its primary role in the production of ejection murmurs. *Circ Res* 38:513–525
7. Trefil JS (2010) *Introduction to the physics of fluids and solids*, Dover edn. Dover, Mineola

Appendix A

Matlab Fundamentals

This appendix provides an overview of Matlab¹ mathematical software, widely used in scientific computation applications to simulate physical systems, run computational algorithms, as well as perform comprehensive data analysis and visualisation. Optional toolboxes extend Matlab functionality to specialised applications including neural networks, signal processing, bioinformatics, system identification, image processing and systems biology.

A.1 Matlab Overview

Matlab provides an interpreter environment for executing an extensive library of in-built mathematical commands, as well as user-defined code scripts and functions. This section provides an overview of the Matlab interface and basic functionality.

A.1.1 User Interface

The default Matlab interface consists of several windows, as shown in Fig. A.1. These are the *command window* where commands are entered and executed by the Matlab interpreter, the *workspace* which lists variables defined in the current session, the *command history* which lists recent commands, and the *current folder* window and *file path* where user-defined scripts and functions are saved and accessed. Recent commands can be re-typed in the command window by using the up-arrow keyboard shortcut. Repeated use of the up-arrow will cycle through several commands, beginning from the most recent entered.

¹The Mathworks Inc, Natick, Massachusetts, U.S.A.

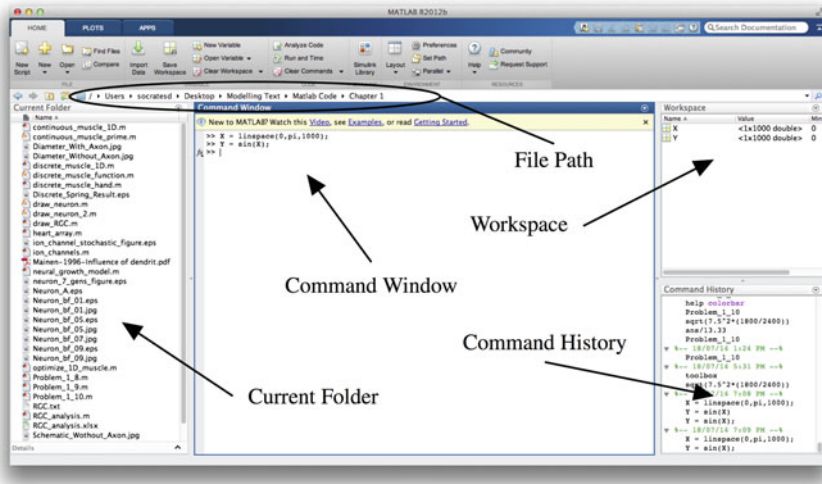


Fig. A.1 Default Matlab interface. The command window represents the main work area where commands are entered for execution by the Matlab interpreter. The workspace window displays current variables, the command history shows recent commands entered, and the current folder window specified by the file path lists local files

A.1.2 Working with Variables and Arrays

Variables do not need to be declared first: simply assign their value directly. For example, the following commands entered in the command window:

```
r = 2;
c = 2*pi*r;
```

will create the variables r and c in the current workspace and assign their values to 2 and $2 \times \pi \times 2 = 4\pi$ respectively (note that `pi` is the in-built Matlab constant for $\pi \approx 3.1416$). Note that each command ends with a semicolon. Although not strictly necessary, the semicolon prevents command results from being echoed in the command window.

To define an array, values can be entered element by element, as in the following:

```
x = [2, 3, 1, 0, 0];
y = [1; 2; 3; 4; 5];
A = [1, 0; 0, 1];
```

which define x to be a five-element row array, y a five-element column array, and A as the 2×2 identity matrix. The subsequent command


```
d = x*y;
```

would perform array multiplication of a row and column vector, yielding the scalar $d = 11$. Reversing the order of the arrays in the command $E = y*x$ would assign a 5×5 matrix to E .

Individual elements of the above arrays can be accessed using commands such as $x(1)$ (returning a value of 2) and $A(1,2)$ (returning a value of 0). To append elements to existing arrays, use commands like

```
z = [x,3];
w = [y;8];
```

which add extra elements of value 3 and 8 to the end of the above-defined arrays x and y respectively. Note that a comma appends to rows, whilst a semi-colon appends to columns. Similar principles apply when concatenating two-arrays. Thus, for example

```
z = [x,x];
w = [y;y];
```

would double the length of both x and y defined above.

Arrays can also be defined using

```
x = 0:0.001:1;
```

which creates a row array of 1001 uniformly-spaced elements, 0 as the first and 1 as the last, in increments of 0.001. An alternative is to use Matlab's `linspace` function:

```
x = linspace(0,1,1001);
```

which yields the same result. Note that this function takes three arguments, the first and last values of the array, and the total number of elements.

To square each element of x , use the `.^` exponent operator which acts on each element of x individually:

```
y = x.^2;
```

Analogous element by element array operators also defined for multiplication (`.*`) and division (`./`). Thus, for example, the following sequence of commands:

```
A = [1,2;3,4];
B = [5,6;7,8];
```

```
C = A*B;
D = A.*B;
E = A/B;
F = A./B;
```

would yield

$$C = \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}, \quad D = \begin{bmatrix} 5 & 12 \\ 21 & 32 \end{bmatrix}, \quad E = \begin{bmatrix} 3 & -2 \\ 2 & -1 \end{bmatrix}, \quad F = \begin{bmatrix} 0.2 & 0.3333 \\ 0.4286 & 0.5 \end{bmatrix}.$$

Note that the division operator (/) for calculating E denotes matrix division, such that $A/B = A \cdot \text{inv}(B)$ where $\text{inv}(B)$ is the inverse of matrix B.

In addition to real number data types, Matlab allows other variable types including strings and complex numbers. For example, the commands

```
a = 'This is some text';
b = complex(1,2);
```

define a and b to be string and complex data types respectively.

A short list of basic Matlab operators and functions is given in Table A.1. More comprehensive documentation on Matlab operators, functions and advanced features can be found in the in-built documentation, which can be accessed from the command window using

```
doc matlab
```

Help on any command can be obtained in the command window by typing help followed by the command, e.g.

```
help linspace
```

Typing doc followed by the command will display html-formatted documentation instead:

```
doc linspace
```

A.1.3 *Matlab Programming*

Matlab provides an extensive set of high-level programming features for implementing complex automated numerical computations and algorithms.

Table A.1 List of basic Matlab operators and functions

| Operator(s) | Description |
|----------------------|--|
| * + - / | Basic arithmetic operators |
| ^ | Exponent operator e.g. $3^2 (= 9)$ |
| .* ./ .^ | Element by element array operators |
| mod(x, y) | Modulus operator, yielding the remainder on division of x by y |
| sin(x) cos(x) tan(x) | Trigonometric functions |
| exp(x) | Exponential function e^x |
| log(x) | Natural logarithm |
| \textbackslash{ } | Array division |
| inv(A) | Returns the inverse of square matrix A |
| > < >= <= == ~= | Comparison operators, returning a value of 1 if true, or 0 otherwise |
| ~ && | NOT, AND, OR logical operators |
| linspace(x1, x2, N) | Generates row array of N equi-spaced values from x1 to x2 |
| zeros(N, M) | Returns an N × M matrix of zero elements |
| ones(N, M) | Returns an N × M matrix with all elements equal to 1 |
| rand(N, M) | Returns an N × M matrix of random uniformly-distributed elements between 0 and 1 |
| randn(N, M) | Returns an N × M matrix of normally-distributed random elements with mean 0 and standard deviation 1 |
| plot(x, y) | Plots array y against x |

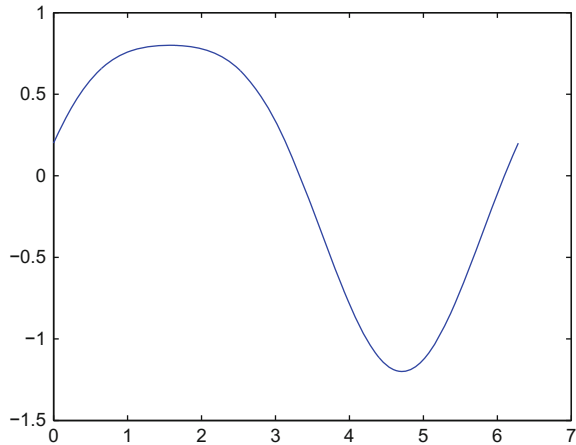
A.1.3.1 Scripting

Matlab command sequences can also be saved as *scripts*; text files having a .m extension. Scripts can be written using the in-built Matlab *editor*, invoked from the Matlab File menu or from the Toolbar, depending on the version of Matlab. To execute the script, enter the name of the script (i.e. the filename without the .m extension) in the command window. For example, to generate a plot of $y = \sin(x) + 0.2 \cos(2x)$, the following commands can be saved to a script named `my_waveform.m`:

```
% initialise x from 0 to 2*pi:
x = 0:2*pi/1000:2*pi;

% calculate waveform:
y = sin(x)+0.2*cos(2*x);
```

Fig. A.2 Plot of $y = \sin(x) + 0.2\cos(2x)$ using the `my_waveform` Matlab script



```
% plot graph:
plot(x,y);
```

Note that text following the `%` character in a line is a *comment*, useful for documenting code function, and is ignored by the Matlab interpreter. Entering `my_waveform` in the command window produces the plot shown in Fig. A.2.

A.1.3.2 Conditional Branching and Loops

As with all high-level programming languages, Matlab provides several conditional branching and loop structures, including `if... else` and `case` structures, as well as `for` and `while` loops. These can be used in scripts as well as user-defined functions (see Sect. A.1.5). For example, the following code generates, rather clumsily, a square-wave input stimulus current I from an array of time values, such

that $I = \begin{cases} 50 & t \leq 10 \\ 0 & \text{otherwise} \end{cases}$:

```
t = 0:1:100;
I = zeros(1,101);
for i=1:101
    if (t(i)<=10)
        I(i)=50;
    else
        I(i)=0;
    end
end
end
```

Note that the same result could be generated using the far more compact code:

```
t = 0:1:100;  
I = 50*(t<=10);
```

A.1.3.3 Code Debugging

The in-built Matlab editor provides continuous, automated code checking to alert the user to coding errors and warnings, as well as additional tools for code debugging. Fig. A.3 illustrates the editor view for the previous for-loop generating a square-wave stimulus, however this time with a missing `end` statement. An error indicator in the top right margin of the editor window alerts the user to a serious code error. The indicator colour can be red, orange or green and indicates either (1) a syntax error (red) in which the code will not run, (2) a warning (orange) in which the code will run but the user should heed the given suggestion, or (3) no error (green). Also shown in the right margin are line markers indicating the relevant location of errors and warnings.

It is also possible to insert breakpoints into the code by clicking in the left margin of the editor. When run, the code will pause execution at the breakpoint, allowing variables to be examined. In fact, any Matlab command can be executed from the command window whilst the code has paused, providing a very powerful debugging feature. Editor toolbuttons allow the user to subsequently step through the code, one line at a time, or continue execution until the next breakpoint.

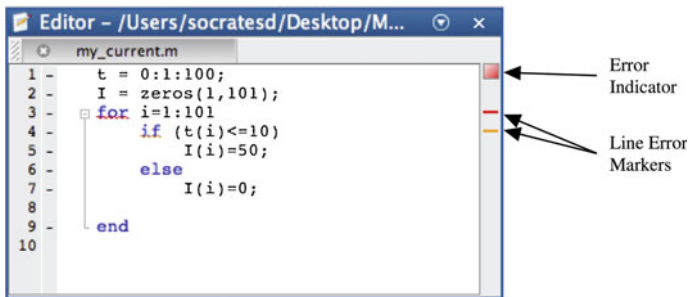


Fig. A.3 Matlab editor, with red indicator (*top right*) alerting the user to a syntax error, due here to a missing `end` statement. Also shown in the *right* margin are line markers indicating specific locations of code errors and warnings. In this case, a red marker at line 3 indicates that the `for` statement has a missing `end`, and the orange marker provides a warning that the `if` statement on line 4 does not have a matching `end`

A.1.4 Solving Linear Systems of Equations

The following linear system of equations:

$$\begin{aligned} 2x + 3y - 4z &= 7 \\ x + 5y - z &= 2 \\ x + y &= 1 \end{aligned}$$

can be represented by the equivalent array equation

$$\mathbf{Ax} = \mathbf{b}$$

$$\mathbf{A} = \begin{bmatrix} 2 & 3 & -4 \\ 1 & 5 & -1 \\ 1 & 1 & 0 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 7 \\ 2 \\ 1 \end{bmatrix}$$

which has the solution

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}.$$

In Matlab, the above system can be solved for using the backslash (\) operator:

```
A = [2, 3, -4; 1, 5, -1; 1, 1, 0];
b = [7; 2; 1];
x = A\b;
```

which yields, correct to four decimal places,

$$\mathbf{x} = \begin{bmatrix} 1.0667 \\ -0.0667 \\ -1.2667 \end{bmatrix}.$$

Use of the backslash operator is equivalent to the Matlab command

```
x = inv(A)*b;
```

which inverts matrix **A** and multiplies by **b**. However, Matlab's backslash operator is more efficient and accurate than direct matrix inversion, particularly for large systems. Using this operator, Matlab can easily solve systems consisting of thousands of matrix elements, as in the following example:

```
A = rand(1000);
b = ones(1000,1);
x = A\b;
```

which only takes a fraction of a second to solve for on a current standard desktop or laptop computer! In the above code, A consists of a 1000×1000 matrix of uniformly-distributed random elements between 0 and 1, and b is a 1000-element column array consisting of 1's.

A.1.5 User-Defined Functions

In addition to hundreds of in-built mathematical functions, Matlab allows the user to define custom functions which can take multiple arguments, and produce multiple outputs. User-defined functions are saved in `.m` files whose first line contains the `function` reserved word. For example, to create a function to solve the system of equations $Ax = b$, the following code can be used:

```
function x = solve_my_system(A, b)
    x = A\b;
end
```

which must be saved in a `.m` file having the same name as the function: in this case, `solve_my_system.m`. Note that this function takes two arguments, A and b , and returns a single output x . The following command can then be invoked from the command window, or within other code:

```
C = [2, 3; 1, 4];
d = [3; 8];
z = solve_my_system(C, d);
```

To define a function with multiple outputs, use code such as:

```
function [x y] = solve_my_systems(A, b, c)
    x = A\b;
    y = A\c;
end
```

which would be invoked from the command window using

```
C = [2, 3; 1, 4];
d = [3; 8];
e = [1; 2];
[u v] = solve_my_systems(C, d, e);
```

A.1.6 Solving Systems of ODEs in Matlab

Matlab provides powerful functions for numerically solving systems of ordinary differential equations (ODEs). As an example, consider the following ODE system:

$$\begin{aligned}\frac{dx}{dt} &= -2x - 3y - 4z \\ \frac{dy}{dt} &= -x + 5z \\ \frac{dz}{dt} &= -x - 2y - 3z\end{aligned}$$

with initial values $x(0) = y(0) = z(0) = 1$. This can be written in matrix form as

$$\frac{d\mathbf{x}}{dt} = \mathbf{A}\mathbf{x}, \quad \mathbf{A} = \begin{bmatrix} -2 & -3 & -4 \\ -1 & 0 & 5 \\ -1 & -2 & -3 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad \text{with } \mathbf{x}(0) = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

To solve such a system in Matlab, we write a function to output the time-derivative evaluations as a function of both t and \mathbf{x} :

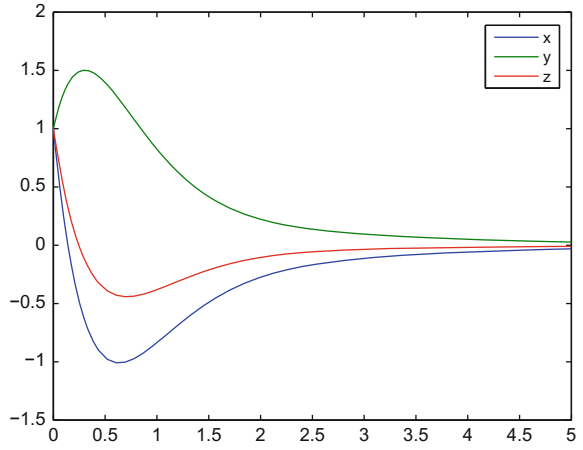
```
function dxdt = derivs(t,x)
    A = [-2, -3, -4; -1, 0, 5; -1, -2, -3];
    dxdt = A*x;
end
```

The system can then be numerically-solved using Matlab's built-in ODE solver `ode15s` by coding the following in a separate script:

```
x_start = [1; 1; 1];
t_range = [0 5];
[t, y] = ode15s('derivs', t_range, x_start);
plot(t,y), legend('x','y','z');
```

Executing this code produces the plot shown in Fig. A.4. Note that the user-defined `derivs` function above included both t and \mathbf{x} as arguments, even though only \mathbf{x} was strictly required in this example (the time-derivatives of this system are functions only of \mathbf{x}). However, `ode15s` requires the user-specified derivative-evaluation function to include both t and \mathbf{x} as arguments.

Fig. A.4 Numerical solution of ODE system using Matlab's `ode15s` function



Appendix B

Overview of COMSOL Multiphysics

COMSOL Multiphysics² is a versatile finite-element software package providing a convenient means for implementing a wide range of multiphysics models. These include standard physics modalities such as electromagnetism, structural and fluid mechanics, diffusion and heat transfer, as well as user-defined systems. Its multiphysics coupling capabilities render COMSOL an increasingly popular choice for bioengineering modelling. Optional add-on modules provide user-interfaces and functionality for additional physics implementations including electromagnetics, microelectromechanical systems (MEMS), heat transfer, nonlinear structural materials, fluid mechanics, microfluidics, as well as interfaces to other software such as the LiveLink for Matlab interface, which allows COMSOL models to be implemented from within Matlab.

B.1 COMSOL Basics

COMSOL has undergone several changes to its user-interface since early versions pre-2005. This section provides an overview of COMSOL v5.2, the most recent release at the time of writing.

B.1.1 User Interface

The default COMSOL user-interface consists of several windows, as shown in Fig. B.1. These include the *Model Builder Window*, which provides a tree representation of the current model, the *Settings Window* which reports associated model settings when clicking a node in the model tree, the *Graphics Window*, which displays the model geometry, mesh and results, and the *Information Windows* which provide

²COMSOL Inc., Burlington, MA.

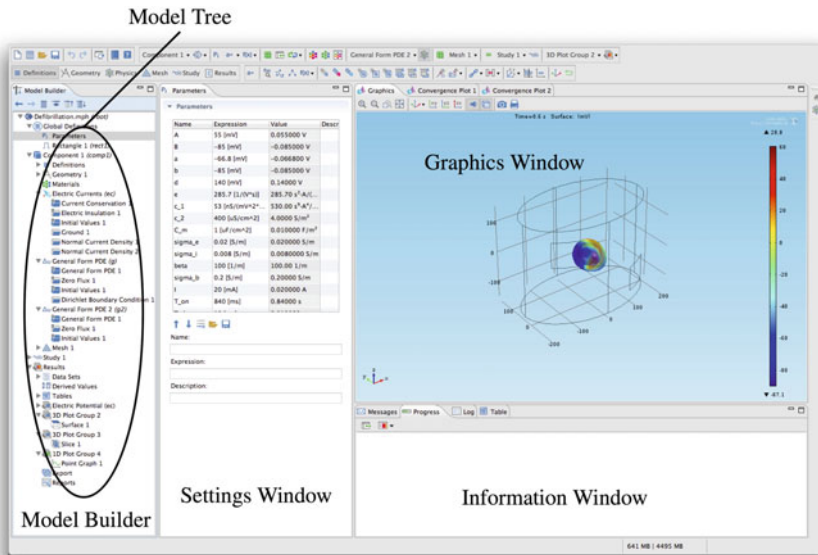


Fig. B.1 Default COMSOL interface (Mac OS X, version 5.2). From *left to right*, the model builder window displays the model tree, the settings window presents various model settings, the graphics window displays model geometry, mesh and results, and the information window displays non-graphical model information including solver progress details, error messages and post-processing evaluations. Across the *top* of the interface are various toolbars and menus

various model information including solution progress, solver logs, error messages, as well as the results of post-processing evaluations. Across the top of the interface are various toolbars and menus.

Central to the COMSOL interface is the *Model Tree* displayed in the Model Builder window. The model tree allows all aspects of a model to be specified and adjusted, including the model geometry, physics, equations, mesh and solver settings, as well as visualisation of results. When solving a model, it is useful to regard the model tree as being executed from top to bottom. Thus, settings in higher nodes in the tree will be visible to all subsequent nodes and sub-nodes.

The Model Builder, Settings and Graphics windows are fully-interactive. Thus, clicking on a node in the model tree will display its associated settings in the Settings window, allowing these to be specified. If the node pertains to the model geometry, mesh or results, the Graphics window will also be updated as appropriate. Right-clicking a node in the model tree will create a new sub-node associated with that node. To set model boundary conditions or domain properties, relevant domains, boundaries, edges or points can be specified by selecting these directly from the Graphics window.

Context-sensitive help can be obtained at any time by selecting the help button (H) at the top of the COMSOL interface. COMSOL also provides an extensive *Model*

Library containing a range of models with step by step instructions for implementation.

B.1.2 Specifying Models

COMSOL provides a series of tools and interfaces for implementing models from scratch, including the Model Wizard, geometry tools, physics and user-equation interfaces, mesh and solver settings, parameters, variables and model couplings, as well as post-processing analysis and visualisation.

B.1.2.1 The Model Wizard

The Model Wizard provides for rapid configuration of new models, and is accessed from the COMSOL start-up screen (or from the File|New menu) as shown in Fig. B.2. Clicking Model Wizard will bring up the Select Space Dimension panel, allowing a choice of 3D, 2D, 1D, 0D, as well as 2D and 1D axisymmetric space dimensions. Selecting the space dimension will then open the Select Physics panel, from which a number of physics interfaces can be added to the model, including the Mathematics

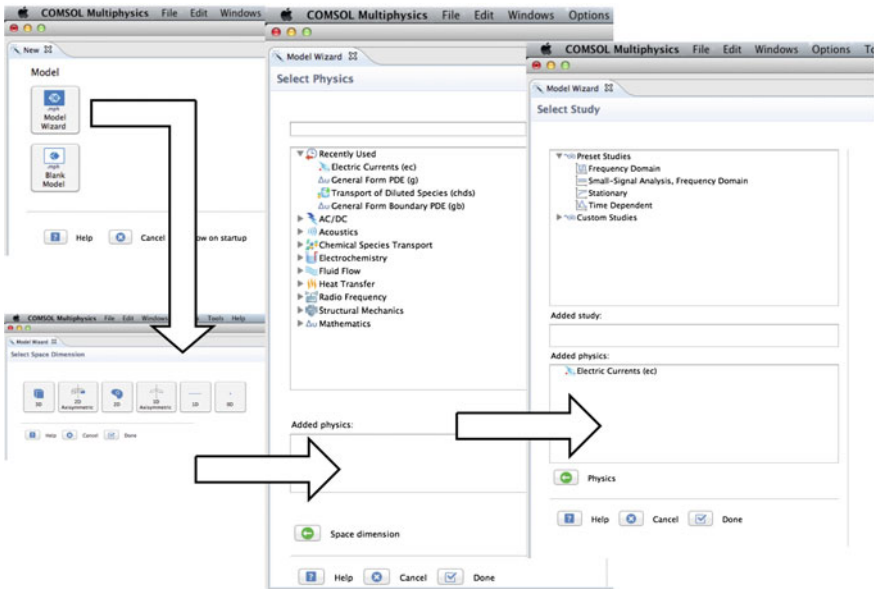


Fig. B.2 COMSOL model wizard. Shown at top left is the new model startup screen. Clicking model wizard will in turn bring up select space dimension, select physics and select study panels

interface for specifying user-defined equations. The list of physics interfaces displayed will depend on which optional COMSOL modules have been installed. Interfaces can be added to the model by clicking the “Add” button. Additional physics interfaces can also be added later from the model tree.

Once the required physics interfaces have been added, clicking the Study forward arrow button (➡) will open the Select Study panel for specifying a default study (i.e. solver) for the model. Depending on the physics interface(s) selected, the choice of solver can include Stationary, Time Dependent or Frequency Domain. Additional studies can also be added later to the same model. Clicking “Done” will exit the Model Wizard and display the main COMSOL interface with model tree configured according to the specified Model Wizard settings.

B.1.2.2 Creating a Geometry

Once the model tree has been initialised, the model geometry can be specified by right-clicking on the geometry node. Depending on the space dimension, the geometry interface allows basic geometric objects, known as *primitives*, to be defined and added to the model. Figure B.3(left) illustrates some of the 3D geometric primitives available on right clicking the geometry node: analogous interfaces are available for other spatial dimensions. 3D primitives include blocks, cones, cylinders, spheres, ellipsoids, toroids, as well as parametric surfaces. On selecting a primitive, its dimensions and position can be specified from within the Settings window. Clicking the Build Selected (🏗️) or Build All Objects (🏗️) buttons in the Settings window will build the object, display the resulting geometry in the Graphics window. To automatically adjust the zoom and view the entire geometry, click the Zoom Extents button (🔍) in the Graphics window.

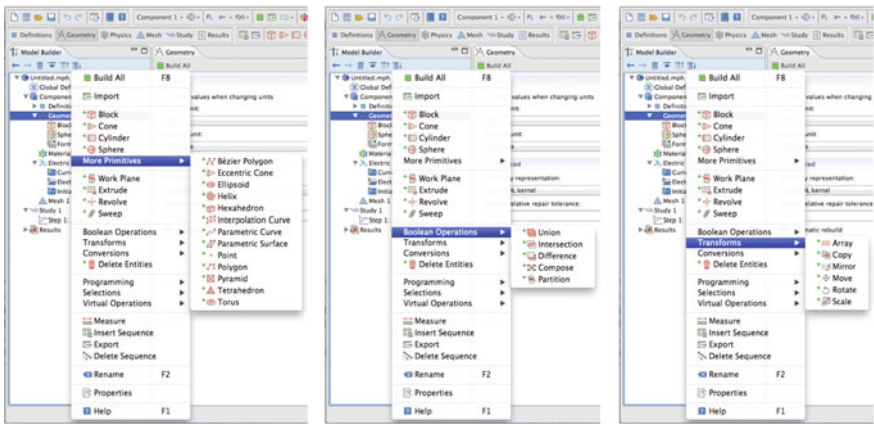


Fig. B.3 COMSOL 3D geometry tools available from the model tree geometry node. From left to right shows the geometry submenus for more primitives, Boolean operations and transforms

By right-clicking the geometry node, it is also possible to perform Boolean operations on geometric primitives, including subtracting objects from each other, forming unions and intersections, as well as custom combinations of these (Fig. B.3, middle). It is also possible to transform objects by moving, scaling, rotating, mirroring or copying (Fig. B.3, right). Geometric objects for Boolean operations and transformations can be selected using the Graphics window.

In 3D, it is also possible to define a *Work Plane*, a 2D plane embedded in the 3D geometry, again by right-clicking the geometry node. 2D primitives can be specified by right-clicking the associated plane geometry sub-node. These 2D objects can then be revolved or extruded into the 3D geometry.

Using a combination of COMSOL’s in-built geometry tools, it is possible to construct fairly complex objects and shapes. For specifying more complex geometries, it is also possible to import CAD, STL and VRML files.

All geometric primitives, operations and transforms appear as geometry sub-nodes in the model tree. It is possible to click on an existing node in any order and modify its settings. Clicking the “Build All Objects” button in the Settings window will then update the geometry.

B.1.2.3 User-Defined Parameters, Functions and Variables

Right-clicking on the Global Definitions node in the model tree, just below the root node, and selecting Parameters allows global parameters to be defined, as shown in Fig. B.4. These parameters can be accessed and used anywhere in the model, including within the geometry settings (e.g. for specifying the size of objects). Parameters are entered in the Settings window by specifying their name, numerical value, and an optional brief description. It is also possible to enter expressions for values using Matlab-type syntax (without the semicolon), provided these

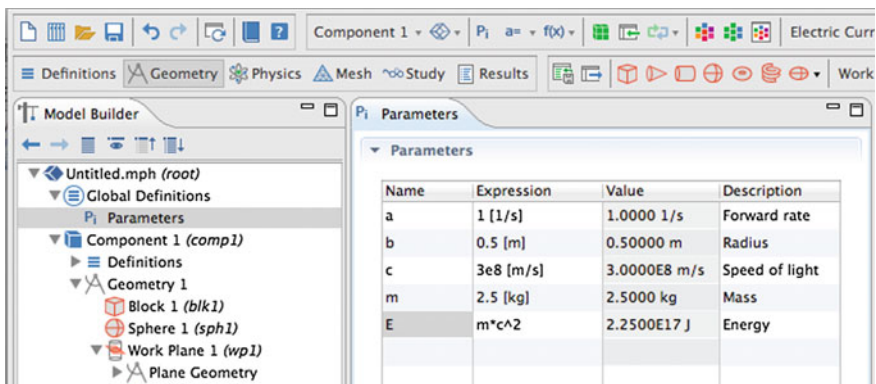


Fig. B.4 COMSOL global parameters interface

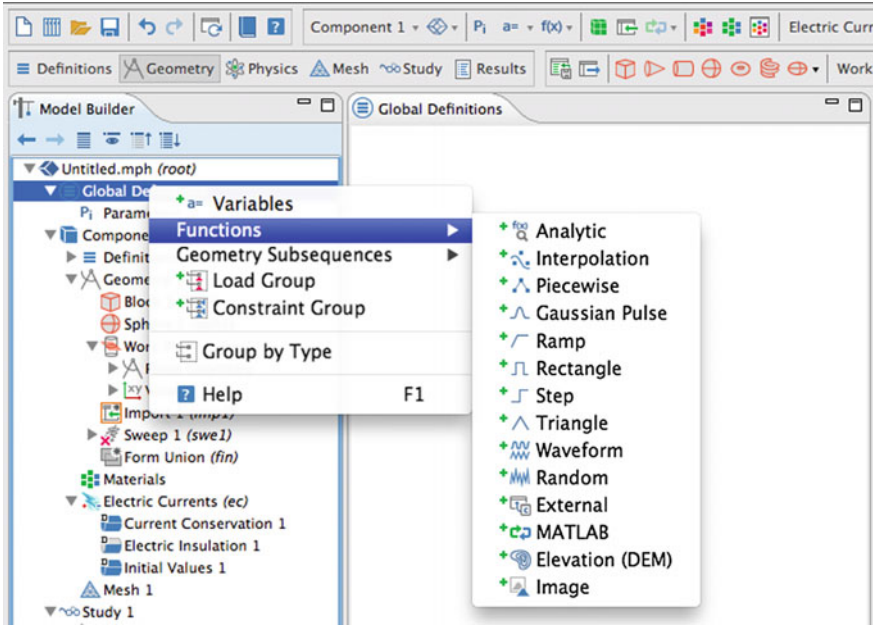


Fig. B.5 COMSOL user-defined function options

expressions yield constants. Examples, of allowable parameter expressions are $\sqrt{2}$, $a \cdot \sin(\pi/3)$, a/b etc., where a and b are themselves parameters.

COMSOL allows physical units for parameters to be specified using square brackets following the parameter value, as seen in Fig. B.4. COMSOL's units feature is very convenient, particularly for bioengineering models which typically involve multiple physical units. COMSOL supports standard SI unit nomenclature as well as other units, in addition to prefixes such as nano-, micro-, milli-, kilo-, or mega-. Thus for example, a value of 1 km would be expressed as 1 [km], and 2 mA would be written as 2 [mA]. For expressions involving other terms with units, COMSOL automatically determines the unit of the dependent quantity. For example, a parameter value expression of $(2 \text{ [m]}) * (1 \text{ [1/s]})$ would have units of ms^{-1} .

In addition to parameters, user-defined functions can also be specified, with a number of function type options, as shown in Fig. B.5. These are accessible from anywhere in the model, and include Interpolation functions which interpolate a set of supplied data values, as well as Rectangle and Step functions. To specify a rectangle function for example, select this option and enter a function name along with upper and lower limits from the Settings window. The function will output a value equal to 1 if its argument lies between these limits, or 0 otherwise. The interface also allows the user to enter a 'smoothing factor', which defines the length of a transition zone region for continuous change from 0 to 1. This is a useful feature, since discontinuous function values can lead to model convergence issues. For user-defined functions,

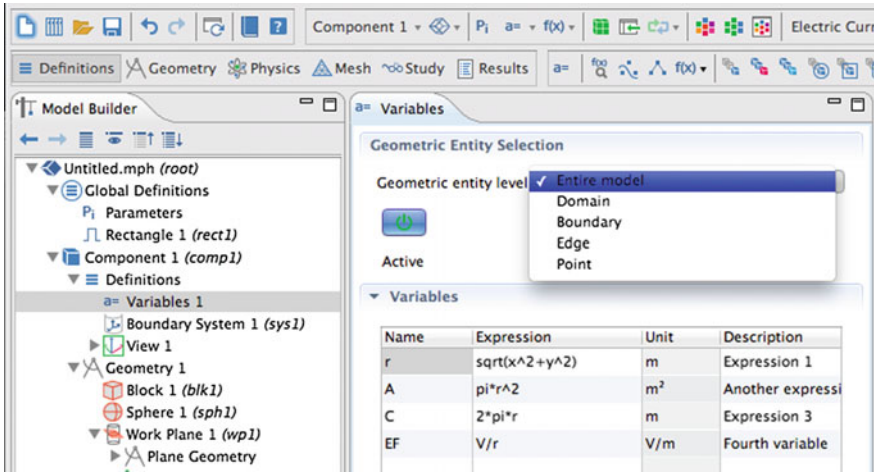


Fig. B.6 COMSOL user-defined variables interface. In the *top* part of the settings window, a geometric entity level for the variables can be specified

COMSOL assumes their inputs and outputs are dimensionless. The only exception is the Interpolation function type, where units for inputs and outputs can be specified.

When specifying physics interfaces, COMSOL will assign default names to the associated dependent variables. Thus, for example, the default variable for voltage in the AC/DC physics interface is v . These variable names can be overridden by the user. To specify additional user-defined variables, right-click the Definitions sub-node of the Component node in the model tree (typically named “Component 1”) and select the Variables option.³ This will create a table in the settings window, where new variables can be defined, as shown in Fig. B.6. Similar to the parameters interface, Matlab-like expressions can be entered to define values. Unlike parameters, however, the expressions don’t have to yield constant values, but can include other variables that vary in space and/or time. Furthermore, user-defined variables can be associated with a *geometric entity level* to specify their *scope*. Depending on the spatial dimension of the model, this scope can include the entire model, specific domains, boundaries, edges or points. Multiple variable sub-nodes can be created within the Definitions node, to group variables into different scopes.

When naming user-defined parameters and variables, COMSOL is case-sensitive. A number of reserved names should be avoided, such as those shown in Table B.1.

³It is also possible to define variables in the Global Definitions node, but these can only be of global scope, unlike variables defined within a component node.

Table B.1 Examples of reserved COMSOL variables and parameters. Depending on the spatial dimension of the model and the study type, not all of these may be reserved in a given model. Variable u is a generic dependent variable defined in a physics interface, and should be replaced with the actual variable name

| Name | Description | Name | Description |
|--------------------------|-----------------------------|--------------------------|--|
| t | Time | ut | $\partial u / \partial t$ |
| x, y, z, r, X, Y, Z, R | Position | ux, uy, uz | $\partial u / \partial x, \partial u / \partial y, \partial u / \partial z$ |
| $freq$ | Frequency | utt | $\partial^2 u / \partial t^2$ |
| $lambda$ | Eigenvalues | uxx, uyy, uzz | $\partial^2 u / \partial x^2, \partial^2 u / \partial y^2, \partial^2 u / \partial z^2$ |
| $phase$ | Phase angle | uxy, uyz, \dots etc. | $\partial^2 u / \partial x \partial y, \partial^2 u / \partial y \partial z, \dots$ |
| pi | π (≈ 3.14159) | uxt, uyt, uzt | $\partial^2 u / \partial x \partial t, \partial^2 u / \partial y \partial t, \partial^2 u / \partial z \partial t$ |
| i, j | $\sqrt{-1}$ | $uxxt, uxyt, \dots$ etc. | $\partial^3 u / \partial^2 x \partial t, \partial^3 u / \partial x \partial y \partial t, \dots$ |
| h | Mesh size | $uxxtt, uxytt, \dots$ | $\partial^4 u / \partial^2 x \partial t^2, \partial^4 u / \partial x \partial y \partial t^2, \dots$ |
| nx, ny, nz | Normal vector components | uTx, uTy, uTz | Tangential derivatives |

B.1.2.4 Assigning Materials

Right-clicking on the Materials node in the model tree allows the optional selection, definition and assignment of various physical materials to parts of the model. The materials include physical parameters utilised by COMSOL's physics interfaces such electrical conductivity, density, Young's modulus etc. To add a material from COMSOL's in-built material libraries, right-click the Materials node and select Add Material. Once the material is selected, click Add to Component to insert that material as a sub-node of the Materials node. The domains of the model in which the material is active can then be selected from the Graphics window, as shown in Fig. B.7.

It is not necessary to explicitly define materials in a model: material constants can manually be inserted into the appropriate physics settings by simply entering user-defined values. The latter can include global parameters defined under the Global Definitions node.

B.1.2.5 Physics and User-Defined Equation Settings

Physics interfaces added during the Model Wizard are visible as nodes in the model tree. If desired, additional physics nodes can be inserted by right-clicking on the Component node in the model tree (typically named "Component 1") and selecting "Add Physics". Right-clicking a physics node will then bring-up all available settings for that interface, including domain settings and boundary conditions. Figure B.8 lists

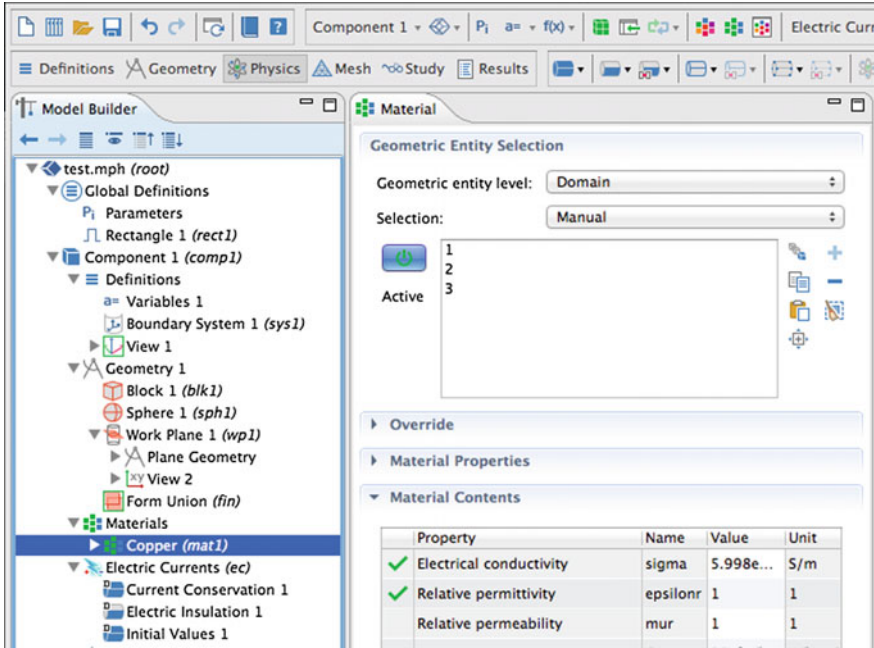




Fig. B.7 COMSOL materials interface. Added materials can be assigned to various parts of a model, and include material properties used by the physics interfaces

the options available when right-clicking the Electric Currents physics node, part of the AC/DC interface.

Physics options are mainly grouped into domain settings (shown for 3D with a solid-filled shape icon ) and boundary conditions (shown for 3D as a boundary patch icon ). Selecting any of these will allow specific settings to be entered in the Settings window. This includes specifying regions where that physics setting is applicable. Each selected setting will appear in the model tree as a sub-node of that physics node. Settings can be modified at any time by returning to that node.

To enter user-defined model equations, COMSOL provides several Mathematics interfaces for specifying a number of equation types, including the *Coefficient Form* and *General Form* PDE interfaces. The general form PDE is represented as

$$e_a \left(\frac{\partial^2 u}{\partial t^2} \right) + d_a \left(\frac{\partial u}{\partial t} \right) + \nabla \cdot \mathbf{\Gamma} = f$$

where u is the dependent variable, e_a is the mass coefficient, d_a is the damping coefficient, f is the source term, and $\mathbf{\Gamma}$ is the conservative flux associated with u . By default, as shown in Fig. B.9, $\mathbf{\Gamma}$ is set to the negative gradient of u , $-\nabla u$, with components (in 3D) given by:

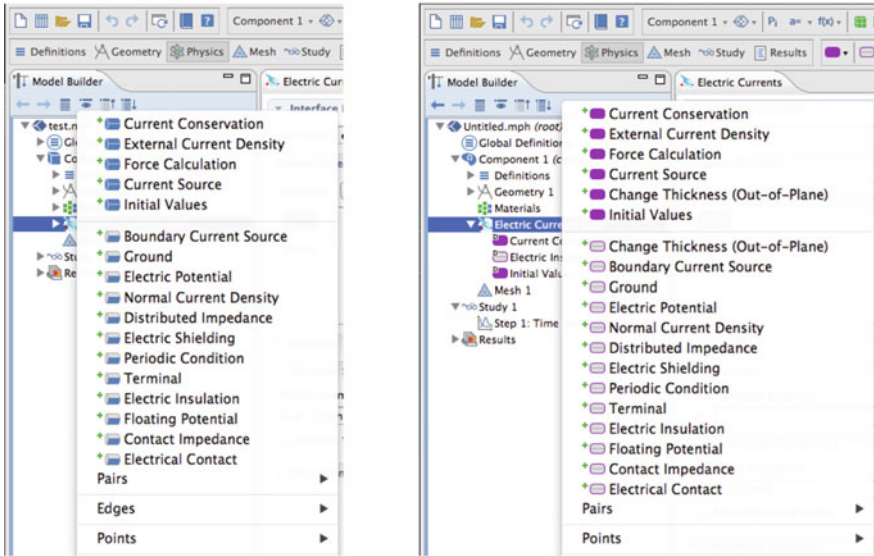


Fig. B.8 Example COMSOL physics options, in this case made available by right-clicking the electric currents node for 3D (*left*) and 2D (*right*) models. Physics options are mainly grouped into domain settings (upper options with solid-filled shape icons) and boundary conditions (lower options with highlighted boundary patch or edge)

$$\mathbf{\Gamma} = \begin{pmatrix} -\partial u / \partial x \\ -\partial u / \partial y \\ -\partial u / \partial z \end{pmatrix} = \begin{pmatrix} -u_x \\ -u_y \\ -u_z \end{pmatrix}$$

where the rightmost column-vector is written using COMSOL notation (see Table B.1). The e_a , d_a , f , and $\mathbf{\Gamma}$ terms can be modified by the user as required. Note that u can be replaced by any other variable name as required. Furthermore, u can consist of several variables, in which case both u and f are replaced by column-arrays, and e_a , d_a and $\mathbf{\Gamma}$ are matrices (the interface is updated accordingly).

As with the other physics interfaces, a range of domain settings and boundary conditions can be specified for the general PDE form. It is also possible (and recommended) to assign physical units to the dependant variable u and source term f .

B.1.2.6 Component Couplings

Right-clicking the component definitions sub-node allows a range of component couplings to be defined, as shown in Fig. B.10. Coupling operators evaluate expressions or integrals over one part of a model (i.e. component), and make these available globally, or to another part of the model. The integration coupling operators, for example,

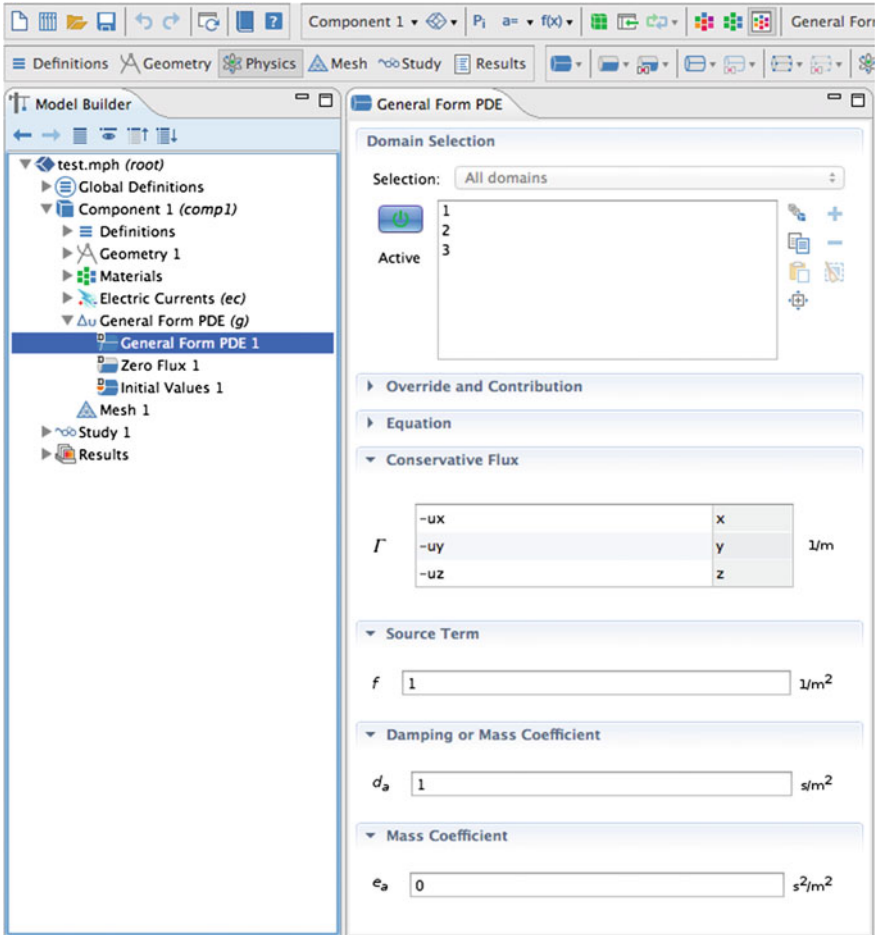


Fig. B.9 COMSOL general form PDE interface. Right-clicking the general form domain setting displays the general form terms in the settings window, which can be modified by the user

specify integration over one or more domains, boundaries, edges or points, and once defined, can be used in any COMSOL expression. Integration of an expression over a point simply returns the value at that point, and is useful for making pointwise values globally-available to other expressions.

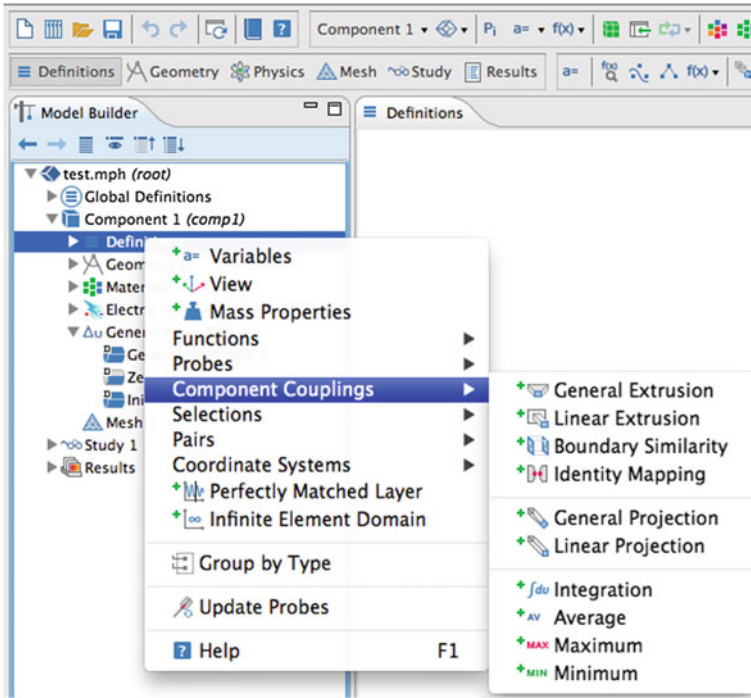


Fig. B.10 COMSOL component coupling options. These define coupling and integration operators that link parts of a model together or assign the evaluations to global scope

B.1.3 Solving and Visualisation

B.1.3.1 Mesh Settings

In order to solve a PDE model, COMSOL automatically generates a finite element mesh to spatially discretize the geometry. It consists of freely-generated tetrahedral elements (3D) or triangular elements (2D) according to default settings, but it is also possible to specify additional settings to mesh, for example, more finely over a given region or boundary. Mesh settings can be specified by right-clicking the mesh node and sub-nodes in the model tree, as shown in Fig. B.11. For example, to globally refine the mesh everywhere, select Size and choose from a number of predefined sizes including ‘Fine’, ‘Finer’, and ‘Extra Fine’. To specify a custom mesh size over part of a model, choose the appropriate geometric entity level (e.g. edge, boundary, or entire model), select the region of interest, then choose “Custom” in the mesh size Settings window. This allows custom sizing to be applied to that part of the model. After custom sizes have been specified, select the “Free Tetrahedral” option to freely mesh the remaining geometry. Finally, click the Build All button (🏗️) to build and visualise the mesh in the Graphics window.

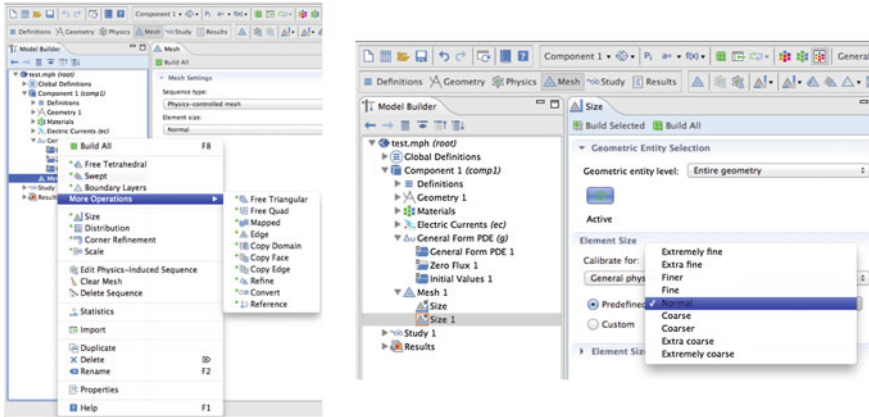


Fig. B.11 COMSOL meshing options. *Left* mesh options available on right-clicking the Mesh node of the model tree (typically named “Mesh 1”). *Right* selecting “size” provides options for predefined mesh sizes, as well as custom size settings

B.1.3.2 Solver Settings

Solver settings can be modified from the Study node in the model tree. Right clicking this node and selecting Show Default Solver will display basic solver settings which can be adjusted by the user (see Fig. B.12). For a time-dependent solver, for example, these settings include the output time steps as well as time stepping behaviour.

Once the solver settings have been specified, right-clicking the study node and selecting Compute (=) will solve the model. Solution progress can be examined in the Progress Information window.

COMSOL also allows parameter sweeps which generate multiple solutions for various values of one or more global parameters. Simply right-click the study node and select “Parametric Sweep” to specify the parameter(s) and their values. This feature is very useful and can be used to generate, for example, a mesh analysis plot by specifying a mesh size global parameter, assigning the maximum mesh size to this parameter, and then performing a parameter sweep to examine how the solution changes with mesh size.

B.1.3.3 Visualisation of Results

On solving a model, COMSOL will automatically generate a plot of the primary dependent variable in the Graphics window. Depending on the spatial dimension of the model, the default plot is either a multislice plot (3D model), a surface plot (2D model), or a line plot (1D model). Right-clicking the Results node in the model tree allows additional plots to be defined, including arrow and streamline plots, contour

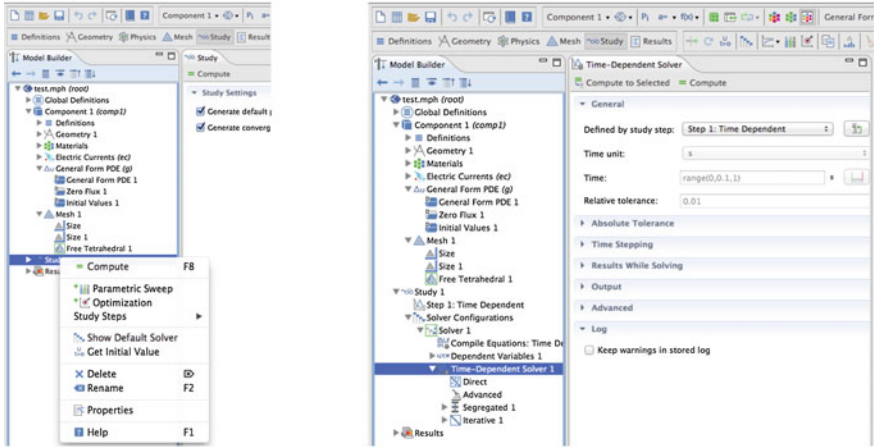
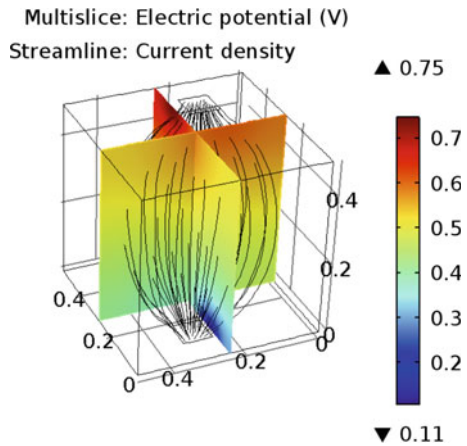


Fig. B.12 COMSOL solver options. *Left* options available on right-clicking the study node (typically named “study 1”). *Right* selecting “show default solver” provides options for the solver. Including time stepping behaviour and absolute tolerance

Fig. B.13 Example of COMSOL multislice plot combined with streamline plot. The plot shows the voltage distribution in a cubic volume conductor of sidelength 0.5 m. There are two electrodes at the *top* and *bottom* boundaries, with one electrode held at ground (0 V) and the other at a fixed potential (1 V). The two slices and colourbar show the electric potential (in V), and the streamlines show the direction of current flow



plots, as well as surface and line plots. It is possible to combine multiple plot types together into a single plot, as shown in Fig. B.13 for a slice and streamline plot.

On right-clicking the Results node, options may appear for 3D, 2D and 1D plot groups. Selecting one of these will create the appropriate sub-node in Results. Right-clicking on these sub-nodes provides further plot choices appropriate to the plot group. COMSOL allows for lower-dimensional plots than that of the model space-dimension. This allows, for example, plots along edges or surfaces in a 3D model. Using the 1D plot group, it is also possible to plot global variables or expressions against time or against a global parameter following a parametric sweep.

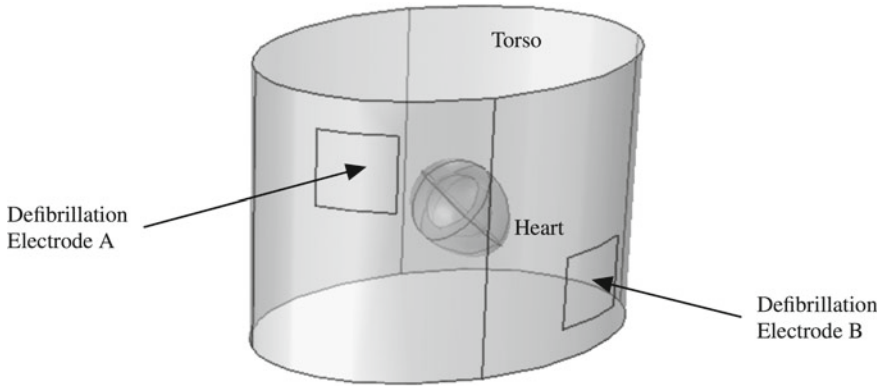


Fig. B.14 Cardiac defibrillation model geometry. The torso is an elliptic cylinder of height 300 mm, with two defibrillation electrodes, A and B, on its surface. The heart is embedded within the torso, and is electrically-active

B.2 Example Model: Cardiac Defibrillation

To illustrate many of COMSOL’s features, this section will present a fully-implemented model of cardiac defibrillation, in which abnormal reentrant activation of the heart is “reset” by an external current applied to electrodes on the wall of the chest. The heart, torso and defibrillating electrodes are represented using the idealised geometry shown in Fig. B.14.

Outside the walls of the heart, the electric potential (V) is governed by

$$\nabla \cdot (-\sigma_b \nabla V) = 0$$

where σ_b is the electrical conductivity of the torso. Within the walls of the heart, extracellular (V_e) and intracellular (V_i) potentials are defined at every point using the *bidomain* formulation, coupled with modified Fitzhugh–Nagumo kinetics for the electrically-active tissue:

$$\beta C_m \left(\frac{\partial V_e}{\partial t} - \frac{\partial V_i}{\partial t} \right) + \nabla \cdot (-\sigma_e \nabla V_e) = \beta i_{ion}$$

$$\beta C_m \left(\frac{\partial V_i}{\partial t} - \frac{\partial V_e}{\partial t} \right) + \nabla \cdot (-\sigma_i \nabla V_i) = -\beta i_{ion}$$

with

$$i_{ion} = c_1(V_m - a)(V_m - A)(V_m - B) + c_2u(V_m - B)$$

$$V_m = V_i - V_e$$

$$\frac{\partial u}{\partial t} = e(V_m - du - b)$$

Table B.2 Parameter values of cardiac defibrillation model

| Parameter | Value | Parameter | Value |
|------------|---------------------------------------|------------|---|
| A | 55 mV | c_1 | 53 nS mV ⁻² cm ⁻² |
| B | -85 mV | c_2 | 400 μS cm ⁻² |
| a | -66.8 mV | C_m | 1 μF |
| b | -85 mV | σ_e | 0.02 S m ⁻¹ |
| d | 140 mV | σ_i | 0.008 S m ⁻¹ |
| e | 285.7 V ⁻¹ s ⁻¹ | β | 100 m ⁻¹ |
| σ_b | 0.2 S m ⁻¹ | T_{ON} | 840 ms |
| I_d | 20 mA | T_{DUR} | 10 ms |

where u is an auxiliary ‘recovery’ variable, σ_e and σ_i are the extracellular and intracellular electrical conductivities within the heart, β is the surface to volume ratio, C_m is cell membrane capacitance per unit area, i_{ion} is the ionic current per unit cell membrane area, and A , B , a , b , d , e , c_1 and c_2 are parameters describing the active electrical activity of the heart.

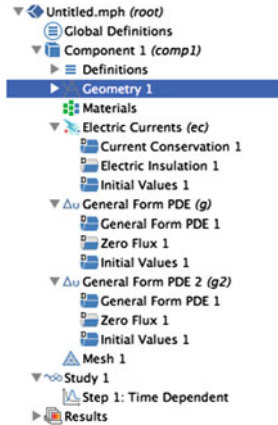
To defibrillate the heart, a rectangular current-pulse of amplitude I_d and duration T_{DUR} is applied to defibrillating electrode A (Fig. B.14) at time $t = T_{ON}$. Defibrillating electrode B is held at ground.

All external boundaries of the torso are electrically-insulating, except at the defibrillating electrodes. At the boundaries of the heart, the extracellular voltage equals the torso potential and the extracellular current density is continuous. For the intracellular potential, the boundaries of the heart are electrically-insulating. All model parameter values are given in Table B.2.




To implement this model in COMSOL, use the following steps:

Model Wizard

1. Open the Model Wizard and select the 3D spatial dimension.
2. In the Select Physics panel, choose AC/DC|Electric Currents. Click “Add”.
3. Next, select Mathematics|PDE Interfaces|General Form PDE. Click “Add”.
4. In the Review Physics panel at right, specify U as the Field name and 2 as the number of dependent variables. In the dependent variables list, enter the names of these variables as V_e and V_i . For the dependent variable quantity, specify the units as Electric potential (V), and the source term quantity as Current source (A/m³).
5. Next, select again Mathematics|PDE Interfaces|General Form PDE, and click “Add”. This will insert a second General Form PDE into the model. In the Review Physics panel, leave the field name as u and the number of dependent variables as 1. Leave the units of the dependent variable as Dimensionless, but enter the source term units manually as 1/s.
6. Click the Study arrow to open the Select Study panel. Select Time Dependent, and click “Done”. This will exit the Model Wizard, displaying the main COMSOL interface. The model tree will look like the following:



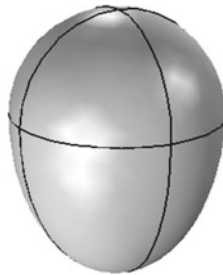
Geometry

1. Select Geometry 1 in the model tree and specify the length unit as mm.
2. Right-click Geometry 1 and select Sphere. Specify the radius as 45 mm. Click Build Selected ().
3. Right-click Geometry 1 and select Sphere again. Specify the radius as 30 mm. Click Build Selected.
4. Right-click Geometry 1 and select Boolean Operations|Difference. In the Objects to add field, select the outer sphere (sph1). In the Objects to subtract field, turn on its Active button and select the inner sphere (sph2). To select this, you may need to hide the outer sphere first by clicking the Select and Hide button(). This will make the inner sphere visible. Deselect the hide button select the inner sphere. Click the reset hiding button () to make both spheres visible. Clicking Build Selected will subtract the inner sphere from the outer.
5. Next, right-click Geometry 1 and select Block. Specify the width, depth and height as 100, 100 and 50mm respectively. Specify the corner coordinates as (-50, -50, -50 mm), and click Build Selected.
6. Right-click Geometry 1 again and select Boolean Operations|Difference. Select the hollowed-out sphere (dif1) as the object to add, and select the block (blk1) as the object to subtract. Click Build Selected. This will result in the hollowed hemispherical ‘shell’ shown below:

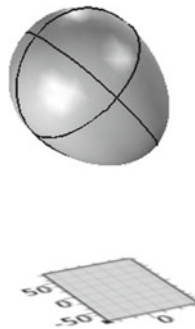


7. Right-click Geometry 1 again and select More Primitives|Ellipsoid. Specify the a-, b- and c-semiaxes as 30, 30, and 46 mm respectively. Click Build Selected.

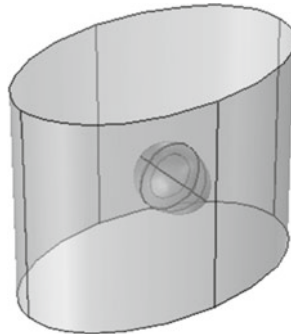
8. Right-click Geometry 1, selecting again More Primitives|Ellipsoid. This time, specify the a-, b- and c-semiaxes as 45, 45, and 69 mm respectively. Click Build Selected.
9. Right-click Geometry 1 and select Block. Specify the width, depth and height to all be 100 mm. Specify the corner of the block to be at $(-50, -50, 0)$ mm. Click Build Selected.
10. Right-click Geometry 1 again and select Boolean Operations|Difference. Select the outer ellipsoid (elp1) as the object to add, and select the block (blk2) as the object to subtract. Click Build Selected.
11. Now right-click Geometry 1 and select Boolean Operations|Difference one more time. Select the outer half-ellipsoid (dif3) as the object to add, and select the inner ellipsoid (elp1) as the object to subtract. Click Build Selected. This will result in a hollowed-out 'heart' object:



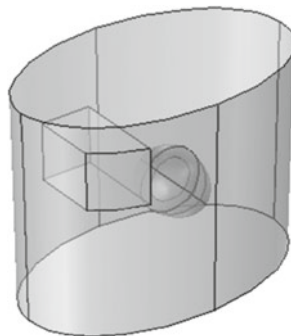
12. Next, rotate the heart by right-clicking Geometry 1 and selecting Transforms|Rotate. Select both halves of the heart (dif2 and dif4) and specify a rotation angle of -45° . Specify the axis of rotation as the y-axis. Click Build Selected. This completes the heart geometry.
13. We now proceed to specify the torso and defibrillating electrodes. Right-click Geometry 1 and select Work Plane. Define a quick xy-plane (the default setting), and specify the z-coordinate as -150 mm. Click Build Selected, followed by the Zoom Extents (🔍) button in the Graphics window to view the whole geometry with work plane, as shown below.



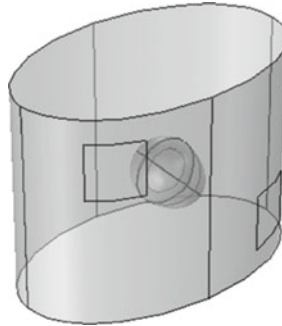
14. Now, right-click on the Plane Geometry subnode of Work Plane 1 and select Ellipse. Specify the a- and b-semiaxes as 200 and 125 mm respectively. Specify the (xw, yw) position of the centre to be (0, 30 mm) and click Build Selected. Click the Zoom Extents (⊕) button to see the whole ellipse.
15. Next right-click Geometry 1 and select Extrude. By default, the input object of the extrude operation will be the work plane (wp1). Specify an extrude distance from the plane as 300 mm and click Build Selected. This will extrude the ellipse to construct the torso elliptic cylinder. Click Zoom Extents followed by the Transparency button (☐) to visualise the heart embedded in the torso as shown below.



16. To build the defibrillation electrodes on the torso surface, we specify elongated blocks that intersect with the torso surface. Right-click Geometry 1 and select Block. Specify the width, depth and height to be 70, 150 and 80 mm respectively. Specify the centre (not corner) of the block to be at (-125, -30, 75 mm) and click Build Selected.
17. Next, right-click Geometry 1 and select Transforms|Copy. Select the torso (ext1) and click Build Selected. This creates a copy of the torso. Right-click Geometry 1 and select Boolean Operations|Intersection. Select both the torso (ext1) and the block (blk3) as input objects and click Build Selected. The resulting geometry will look like:



18. Now we repeat this procedure for the other electrode. Right-click Geometry 1 and select Block. Specify the width, depth and height to be 70, 150 and 80 mm respectively. Specify the centre of the block to now be at (125, -30, -75 mm) and click Build Selected.
19. As before, right-click Geometry 1 and select Transforms|Copy. Select the torso (copy1) and click Build Selected. Right-click Geometry 1 and select Boolean Operations|Intersection. Select both the torso (copy1) and the block (blk4) as input objects and click Build Selected.
20. Finally, right-click Geometry 1 and select Boolean Operations|Union. Specify the two intersection regions (int1 and int2) and the torso (copy2) as the three input objects. Deselect the “Keep interior boundaries” checkbox and click Build Selected. The final geometry obtained is shown below:



Global Definitions

1. Right-click Global Definitions and select Parameters. Enter the following details in the Parameters table of the Settings window:

| Name | Expression | Description |
|---------|---|----------------------------|
| A | 55 [mV] | Model parameter |
| B | -85 [mV] | Model parameter |
| a | -66.8 [mV] | Model parameter |
| b | -85 [mV] | Model parameter |
| d | 140 [mV] | Model parameter |
| e | 285.7 [1/(V*s)] | Model parameter |
| c_1 | 53 [nS/(mV ² *cm ²)] | Model parameter |
| c_2 | 400 [uS/cm ²] | Model parameter |
| C_m | 1 [uF/cm ²] | Membrane capacitance |
| sigma_e | 0.02 [S/m] | Extracellular conductivity |
| sigma_i | 0.008 [S/m] | Intracellular conductivity |
| beta | 100 [1/m] | Surface to volume ratio |
| sigma_b | 0.2 [S/m] | Torso bulk conductivity |
| I | 100 [mA] | Defibrillation amplitude |
| T_on | 760 [ms] | Defibrillation onset |
| T_dur | 100 [ms] | Defibrillation duration |

2. Right-click Global Definitions and select Functions|Rectangle. Specify the lower limit as T_{on} and the upper limit as $T_{on}+T_{dur}$. In the Smoothing tab, specify the size of the transition zone as $T_{dur}/10$. Leave the function name to its default (rect1).

Component Definitions

1. Right-click the Definitions sub-node of Component 1 and select Component Couplings|Integration. Specify the geometric entity level as ‘Boundary’ and select boundary 5. This creates an integration operator for integrating expressions over this boundary. Leave the default operator name as intop1.
2. Right-click the Definitions sub-node again and select Variables. Leave the geometric entity level to its default as ‘Entire model’, and enter the following variables in the settings table:

| Name | Expression | Description |
|--------|------------------------|-----------------|
| Area | intop1(1) | Electrode area |
| J_stim | (I/Area)*rect1(t[1/s]) | Current density |

3. Again, right-click Definitions and select Variables to define a new variables group. Specify the geometric entity level as ‘Domain’ and select domains 2 and 3 corresponding to the heart. It may be easier to use the Select Box button (📦) in the Graphics window to drag a box around the heart to select these domains. Enter the following in the variables settings table:

| Name | Expression |
|-------|---|
| Vm | Vi-Ve |
| i_ion | $c_1 * (V_m - a) * (V_m - A) * (V_m - B) + c_2 * u * (V_m - B)$ |

Electric Currents

1. Select the Electric Currents node in the model tree. By default this physics is set to hold in all domains of the model (1–4). We wish to override this setting, since this physics should apply only to the torso and not the heart. Select domains 2 and 3 and click the Remove from Selection button (➔) to individually remove these domains.
2. Expand the Electric Currents node and select the Current Conservation sub-node. In the Settings window, specify the electrical conductivity to be user defined, and enter a value of sigma_b. Similarly, specify the relative permittivity to be user defined, and leave the default value of 1.
3. Right-click Electric Currents and select Ground. In the Settings window, select boundary 25 (defibrillating electrode B in Fig. B.14), to set this electrode to ground.

4. Right-click Electric Currents again and select Normal Current Density. Select boundary 5 (defibrillating electrode A in Fig. B.14) and specify a normal current density of J_{stim} .
5. Right-click Electric Currents again and select Normal Current Density. This time, select all the boundaries of the heart by dragging a select box around it. These correspond to boundaries 6, 7, 9–18, 20–23. For the inward normal current density, enter the expression $\sigma_e * (Vex * nx + Vey * ny + Vex * nz)$ to specify that the current density flowing into the torso from the heart is equal to the current density flowing out of the heart's extracellular domain.

General Form PDE

1. Select the General Form PDE node. Again by default, this PDE is set to apply to all domains of the model (1–4). However, since it should be applicable only within the heart, remove domains 1 and 4 using the Remove from Selection button (–), leaving only domains 2 and 3.
2. Select the General Form PDE 1 sub-node of General Form PDE, and enter the following expressions for the conservative flux $\mathbf{\Gamma}$ components for variables V_e and V_i respectively:

| Component | Expression |
|-----------|-------------------|
| x | $-\sigma_e * Vex$ |
| y | $-\sigma_e * Vey$ |
| z | $-\sigma_e * Vez$ |
| x | $-\sigma_i * Vix$ |
| y | $-\sigma_i * Viy$ |
| z | $-\sigma_i * Viz$ |

Enter the following expressions for the source term f corresponding to variables V_e and V_i respectively:

$\beta * i_{ion} + stim$

$-\beta * i_{ion}$

Finally, enter the following terms for the damping coefficient matrix d_a :

$\beta * C_m$ $-\beta * C_m$


$-\beta * C_m$ $\beta * C_m$

Leave the mass coefficient matrix e_a entries to their default value of 0.


3. Right-click General Form PDE again and select Dirichlet Boundary Condition. Using the select box, drag a selection around the heart to select all heart boundaries. Remove boundary 8 from the selection, which is the internal boundary between the top and bottom parts of the heart. Deselect the 'Prescribed value of Vi' checkbox, leaving only the 'Prescribed value of Ve' checkbox selected. Enter a value of V in the r_1 field. This constrains the value of V_e at the outer and inner boundaries of the heart to equal the torso potential.

4. Finally, select the Initial Values 1 sub-node of the General Form PDE node. Leave the initial value of V_e to its default value of 0. For V_i however, enter the initial value expression $-0.085 + 0.12 * (y < 0) * (z > 0.02)$. This sets V_i to an initial value of $-0.085 + 0.12 = 0.035$ V for the sector of the heart corresponding to $y < 0$ and $z > 0.02$ m, representing an electrically-excited state. In all other heart regions, V_i is initially set to -0.085 V, corresponding to the resting, non-excited state.



General Form PDE 2

1. Select the General Form PDE 2 node. Select domains 1 and 4 and individually remove these using the Remove from Selection button () , leaving only domains 2 and 3.
2. Select the General Form PDE 1 sub-node of General Form PDE 2, and enter a value of 0 for each of the three components of conservative flux $\mathbf{\Gamma}$, since there are no spatial derivatives in the equation for u . For the source term f , enter the expression $e * (V_m - d * u - b)$ and for the damping and mass coefficients d_a and e_a , leave their value as 1 and 0 respectively.
3. Finally, select the Initial Values 1 sub-node of the General Form PDE 2 node, and enter the initial value expression $5 * (x > 0.03)$. This sets variable u to an initial value of 5 when $x > 0.03$ m, corresponding to heart state that is temporarily inexcitable (i.e. refractory), and 0 elsewhere.

Mesh

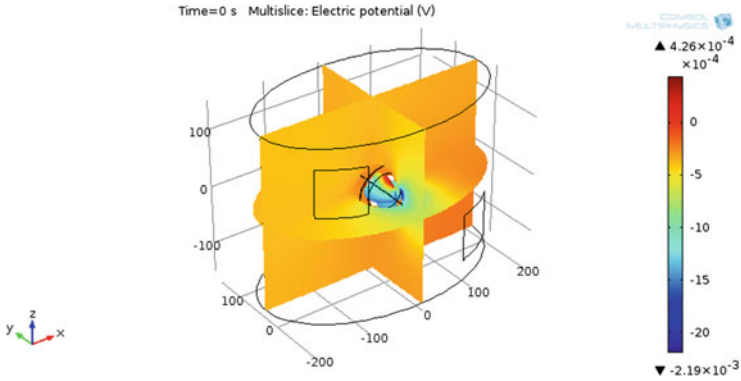
1. Right-click Mesh 1 and select Size. In the Settings window, specify 'Domain' as the geometric entity level, and select domains 2 and 3 corresponding to the heart. Under the Predefined Element Size option, select 'Finer' from the dropdown list.
2. Right-click Mesh 1 again and select Free Tetrahedral. Leave the default geometric entity level as 'Remaining'. This will mesh the remaining parts of the model with a free tetrahedral mesh.
3. Click Build All () in the Settings window to build and display the mesh.


Study

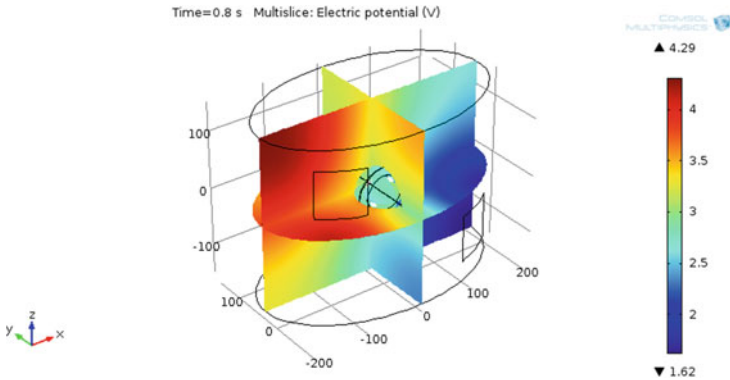
1. Select the Step1: Time Dependent sub-node of the Study 1 node. In the Settings window, Click the Range button () adjacent to the Times field. Leave the entry method as 'Step' and enter Start, Step and Stop values of 0, 0.001 and 2 respectively. Click Replace. This will create a range of output time values from 0 to 2 s in time steps of 0.001 s.
2. Right-click the Study 1 node and select Show Default Solver. Select the Study 1|Solver Configurations|Solution 1|Time-Dependent Solver 1 node. In the Settings window, expand the Advanced tab, and for the 'Singular mass matrix' option, select 'Yes'. Under the Time Stepping tab, select 'Strict' for the Steps taken by solver option.
3. To solve the model, right-click Study 1 and select Compute () . Select the Progress tab in the Information window to see the solver progress.


Results

1. When the model has completed solving,⁴ the Graphics window will display a default multislice plot of the torso potential (variable V) at $t = 0$, as shown below:

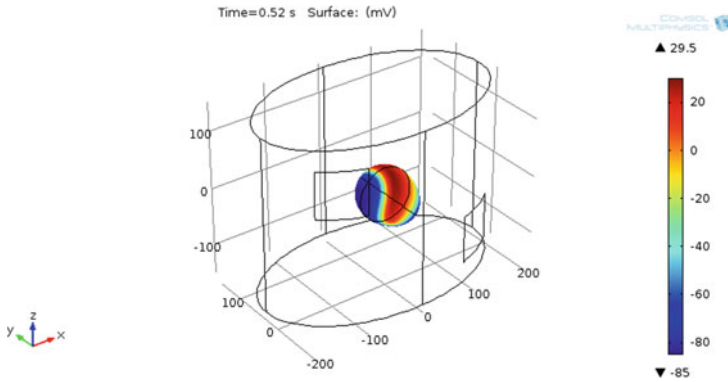



In the Settings window, select a time of 0.8 s from the Time dropdown list and click the Plot button () to display the torso potential at this time, during the defibrillation stimulus:

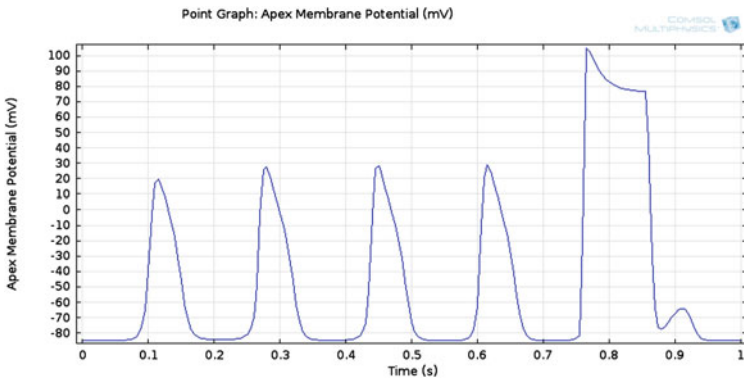


2. Right-click Results and select 3D Plot Group. Right-click this new plot group (3D Plot Group 4) and select Surface. In the Settings window, enter V_m as the expression to plot and select mV as the units from the Unit dropdown list. Left-click the 3D Plot Group 4 node again, and select a time of 0.52 s from the Time dropdown list. Clicking the Plot button () will display the membrane potential on the surface of the heart at this time:

⁴Using my MacBook Air laptop with 8GB RAM and OS X version 10.8.5, it took just under 5 min to solve this model.



3. Right-click Results and select 1D Plot Group. Right-click this new plot group (1D Plot Group 5) and select Point Graph. In the Settings window, enter V_m as the expression to plot and select mV as the units. In the Graphics window, select point 22 corresponding to a point at the apex of the heart. Click the Description checkbox, and type 'Apex Membrane Potential' in the Description field. By default, the x-axis data will be the time values. Clicking the Plot button () will display the membrane potential at the apex against time:



This plot shows a repetitive sequence of action potentials (electrical excitations) at the surface of the heart prior to 0.7 s, due to re-entrant activation from a wave of excitation travelling continuously around the heart. Following application of an extracellular current between $t = 0.76$ and 0.86 s through the torso surface electrodes, these periodic self-excitations are terminated, indicating successful defibrillation.

Solutions

Problems of Chap. 1

1.1 (a) Let ϕ_1 and ϕ_2 be two distinct solutions satisfying the 1D diffusion equation, such that $\frac{\partial \phi_1}{\partial t} = D \frac{\partial^2 \phi_1}{\partial x^2}$ and $\frac{\partial \phi_2}{\partial t} = D \frac{\partial^2 \phi_2}{\partial x^2}$. We then form $u = c_1 \phi_1 + c_2 \phi_2$, to obtain:

$$\begin{aligned} \frac{\partial u}{\partial t} &= c_1 \frac{\partial \phi_1}{\partial t} + c_2 \frac{\partial \phi_2}{\partial t} \\ &= c_1 D \frac{\partial^2 \phi_1}{\partial x^2} + c_2 D \frac{\partial^2 \phi_2}{\partial x^2} \\ &= D \left[\frac{\partial^2 (c_1 \phi_1 + c_2 \phi_2)}{\partial x^2} \right] \\ &= D \frac{\partial^2 u}{\partial x^2} \end{aligned}$$

Hence u is also a solution, and the 1D diffusion equation is linear.

(b) Let ϕ_1 and ϕ_2 be two distinct solutions to the equation, and form $u = c_1 \phi_1 + c_2 \phi_2$, to obtain:

$$\begin{aligned} \frac{du}{dt} &= c_1 \frac{d\phi_1}{dt} + c_2 \frac{d\phi_2}{dt} \\ &= c_1 k \phi_1 \left(1 - \frac{\phi_1}{N_{max}} \right) + c_2 k \phi_2 \left(1 - \frac{\phi_2}{N_{max}} \right) \\ &= k(c_1 \phi_1 + c_2 \phi_2) - \frac{k}{N_{max}} (c_1 \phi_1^2 + c_2 \phi_2^2) \end{aligned}$$

$$\begin{aligned}
 &= ku - \frac{k}{N_{max}} (u^2 - u^2 + c_1\phi_1^2 + c_2\phi_2^2) \\
 &= ku - \frac{ku^2}{N_{max}} + \frac{k}{N_{max}} (u^2 + c_1\phi_1^2 + c_2\phi_2^2) \\
 &= ku \left(1 - \frac{u}{N_{max}}\right) + \frac{k}{N_{max}} (u^2 + c_1\phi_1^2 + c_2\phi_2^2) \\
 &\neq ku \left(1 - \frac{u}{N_{max}}\right)
 \end{aligned}$$

Hence, the equation is non-linear.

1.2 (a) MLT^{-2} (b) L^2T^{-1} (c) $M^{-1}L^{-2}T^4I^2$ (d) $ML^2T^{-3}I^{-2}$ (e) L^3T^{-1} (f) T^{-1}
 (g) The radian angle measure is defined as the circular arc length subtended divided by the radius. Hence its dimensions are $L/L = 1$, i.e. a dimensionless quantity.

1.3 $[B_0] = NL^{-3}T^{-1}$, $[k_1] = T^{-1}$, $[k_2] = T^{-1}$, $[k_3] = T^{-1}$, $[k_4] = L^3N^{-1}T^{-1}$,
 $[k_5] = T^{-1}$, $[k_6] = L^3N^{-1}T^{-1}$.

1.4 (a) $[c_0] = ML^{-1}T^{-2}$, SI units: $Nm^{-2} = Pa$, $[c_1] = ML^8T$, SI units: kgm^8s ,
 $[c_2] = ML^{-1}$, SI units: $Pa s^2 rad^{-2}$
 (b) $a: mV^{-1}s^{-1}$, $b: mV$, $c: mV$, $A: s^{-1}$, $B: mV$, $C: mV$.

1.5 (a) $ML^3T^{-3}I^{-2}$

(b) Physical quantities are R_a (access resistance), D and ρ . From these, we form the products

$$\begin{aligned}
 \pi_i &= R_a^a D^b \rho^c \\
 [\pi_i] &= (ML^2T^{-3}I^{-2})^a L^b (ML^3T^{-3}I^{-2})^c \\
 &= M^{(a+c)} L^{(2a+b+3c)} T^{(-3a-3c)} I^{(-2a-2c)}
 \end{aligned}$$

For these to be dimensionless, we require

$$\begin{aligned}
 a + c &= 0 \\
 2a + b + 3c &= 0 \\
 -3a - 3c &= 0 \\
 -2a - 2c &= 0
 \end{aligned}$$

which has infinitely many solutions of the form

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix} \alpha$$

where α may be freely chosen. Choosing $\alpha = 1$, we have $\pi_1 = R_a D \rho^{-1}$. Hence,

$$R_a = \frac{c\rho}{D}$$

where c is a dimensionless constant.

1.6 We form the dimensionless variables $u^* = u/V$, $x^* = x/L$, $t^* = \omega t$, $p^* = p/(\rho V^2)$. This leads to the scaled equation:

$$\frac{\partial u^*}{\partial t^*} + \left(\frac{V}{\omega L}\right) u^* \frac{\partial u^*}{\partial x^*} = - \left(\frac{V}{\omega L}\right) \frac{\partial p^*}{\partial x^*} + \left(\frac{\mu}{\rho \omega L^2}\right) \frac{\partial^2 u^*}{\partial x^{*2}}$$

with two dimensionless parameters characterising this system:

$$p_1 = \frac{V}{\omega L}$$

$$p_2 = \frac{\mu}{\rho \omega L^2}$$

1.7 The scaled form of the Hodgkin–Huxley equations may be written as:

$$\frac{dV^*}{dt^*} = p_1(V^* - 1) + p_2 V^* + p_3(V^* - p_4)$$

$$\frac{dn}{dt^*} = \alpha_n^*(1 - n) - \beta_n^* n$$

$$\frac{dm}{dt^*} = \alpha_m^*(1 - m) - \beta_m^* m$$

$$\frac{dh}{dt^*} = \alpha_h^*(1 - h) - \beta_h^* h$$

with

$$\alpha_n^* = \frac{p_5(V^* + p_6)}{1 - \exp\left[\frac{-(V^* + p_6)}{p_7}\right]} \quad \beta_n^* = \exp\left[\frac{-(V^* + p_8)}{p_9}\right]$$

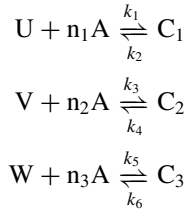
$$\alpha_m^* = \frac{p_{10}(V^* + p_{11})}{1 - \exp\left[\frac{-(V^* + p_{11})}{p_{12}}\right]} \quad \beta_m^* = p_{13} \exp\left[\frac{-(V^* + p_{14})}{p_{15}}\right]$$

$$\alpha_h^* = p_{16} \exp\left[\frac{-(V^* + p_{17})}{p_{18}}\right] \quad \beta_h^* = \frac{p_{19}}{1 + \exp\left[\frac{-(V^* + p_{20})}{p_{21}}\right]}$$

where 21 parameters characterise the system:

$$\begin{aligned}
 p_1 &= -\frac{g_{Na}}{C B_n} & p_2 &= -\frac{g_K}{C B_n} & p_3 &= -\frac{g_L}{C B_n} \\
 p_4 &= \frac{V_L - V_K}{V_{Na} - V_K} & p_5 &= \frac{A_n}{B_n} (V_{Na} - V_K) & p_6 &= \frac{V_K + V_{an}}{V_{Na} - V_K} \\
 p_7 &= S_{an} & p_8 &= \frac{V_K + V_{bn}}{V_{Na} - V_K} & p_9 &= \frac{s_{bn}}{V_{Na} - V_K} \\
 p_{10} &= \frac{A_m}{B_n} (V_{Na} - V_K) & p_{11} &= \frac{V_K + V_{am}}{V_{Na} - V_K} & p_{12} &= \frac{s_{am}}{V_{Na} - V_K} \\
 p_{13} &= \frac{B_m}{B_n} & p_{14} &= \frac{V_K + V_{bm}}{V_{Na} - V_K} & p_{15} &= \frac{s_{bm}}{V_{Na} - V_K} \\
 p_{16} &= \frac{A_h}{B_n} & p_{17} &= \frac{V_K + V_{ah}}{V_{Na} - V_K} & p_{18} &= \frac{s_{ah}}{V_{Na} - V_K} \\
 p_{19} &= \frac{B_h}{B_n} & p_{20} &= \frac{V_K + V_{bh}}{V_{Na} - V_K} & p_{21} &= \frac{s_{bh}}{V_{Na} - V_K}
 \end{aligned}$$

1.8 Assume there are three compounds U, V, and W in the hydrogel, each of which can enter into a chemical reaction with the analyte to produce distinct complexes that fluoresce with wavelengths λ_1 , λ_2 and λ_3 respectively. Furthermore, these reactions are given by:



where n_1, n_2, n_3 are the number of molecules of analyte A needed for each reaction, and $\text{C}_1, \text{C}_2, \text{C}_3$ are the resultant complexes formed. If the total concentration of each compound/complex is denoted by T_1, T_2 and T_3 , then

$$\begin{aligned}
 [\text{U}] + [\text{C}_1] &= T_1 \\
 [\text{V}] + [\text{C}_2] &= T_2 \\
 [\text{W}] + [\text{C}_3] &= T_3
 \end{aligned}$$

Furthermore, if $\lambda_U, \lambda_V, \lambda_W$ are the fluorescence wavelengths of U, V, and W respectively, then the mean wavelength of the hydrogel-analyte system will be given by:

$$\begin{aligned}
 \bar{\lambda} &= \frac{\lambda_U [\text{U}] + \lambda_1 [\text{C}_1] + \lambda_V [\text{V}] + \lambda_2 [\text{C}_2] + \lambda_W [\text{W}] + \lambda_3 [\text{C}_3]}{T_1 + T_2 + T_3} \\
 &= \frac{\lambda_U (T_1 - [\text{C}_1]) + \lambda_V (T_2 - [\text{C}_2]) + \lambda_W (T_3 - [\text{C}_3]) + \lambda_1 [\text{C}_1] + \lambda_2 [\text{C}_2] + \lambda_3 [\text{C}_3]}{T_1 + T_2 + T_3} \\
 &= \frac{\lambda_U T_1 + \lambda_V T_2 + \lambda_W T_3 + (\lambda_1 - \lambda_U) [\text{C}_1] + (\lambda_2 - \lambda_V) [\text{C}_2] + (\lambda_3 - \lambda_W) [\text{C}_3]}{T_1 + T_2 + T_3}
 \end{aligned}$$

Denoting the concentration of analyte A with c , the above reaction scheme can be modelled as the following system of differential equations:

$$\begin{aligned}\frac{d[C_1]}{dt} &= k_1 [U] c^{n_1} - k_2 [C_1] = k_1 (T_1 - [C_1]) c^{n_1} - k_2 [C_1] \\ \frac{d[C_2]}{dt} &= k_3 [V] c^{n_2} - k_4 [C_2] = k_3 (T_2 - [C_2]) c^{n_2} - k_4 [C_2] \\ \frac{d[C_3]}{dt} &= k_5 [W] c^{n_3} - k_6 [C_3] = k_5 (T_3 - [C_3]) c^{n_3} - k_6 [C_3]\end{aligned}$$

The steady-state concentrations of C_1 , C_2 , C_3 can be found by equating the above derivatives to zero, to obtain:

$$[C_1] = \frac{T_1 c^{n_1}}{\left[c^{n_1} + \frac{k_2}{k_1} \right]}, \quad [C_2] = \frac{T_2 c^{n_2}}{\left[c^{n_2} + \frac{k_4}{k_3} \right]}, \quad [C_3] = \frac{T_3 c^{n_3}}{\left[c^{n_3} + \frac{k_6}{k_5} \right]}$$

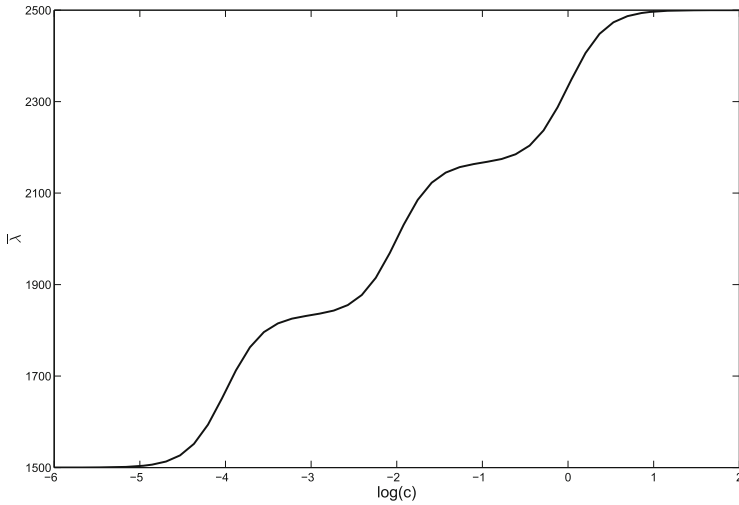
with the steady-state fluorescent wavelength given by:

$$\bar{\lambda} = \lambda_0 + \frac{F_1(\lambda_1 - \lambda_U)c^{n_1}}{\left[c^{n_1} + \frac{k_2}{k_1} \right]} + \frac{F_2(\lambda_2 - \lambda_V)c^{n_2}}{\left[c^{n_2} + \frac{k_4}{k_3} \right]} + \frac{F_3(\lambda_3 - \lambda_W)c^{n_3}}{\left[c^{n_3} + \frac{k_6}{k_5} \right]}$$

where

$$\begin{aligned}\lambda_0 &= \frac{\lambda_U T_1 + \lambda_V T_2 + \lambda_W T_3}{T_1 + T_2 + T_3}, \quad F_1 = \frac{T_1}{T_1 + T_2 + T_3}, \quad F_2 = \frac{T_2}{T_1 + T_2 + T_3}, \\ F_3 &= \frac{T_3}{T_1 + T_2 + T_3}\end{aligned}$$

Choosing, for example, the following parameter values (all in arbitrary units): $T_1 = T_2 = T_3 = 1$, $n_1 = n_2 = n_3 = 2$, $\lambda_U = 1000$, $\lambda_V = 1500$, $\lambda_W = 2000$, $\lambda_1 = 2000$, $\lambda_2 = 2500$, $\lambda_3 = 3000$, $k_1 = 10^8$, $k_3 = 10^4$ and $k_2 = k_4 = k_5 = k_6 = 1$, the following graph for steady-state mean wavelength against log analyte concentration is readily obtained.



1.9 For a cylindrical axon of length L , radius r , axoplasmic resistivity ρ_i , and membrane resistance r_m , the axon can be discretized into N discrete elements, each of length $\Delta x = L/N$. Values of the R_i and R_m components are then given by:

$$R_i = \frac{\rho_i \Delta x}{\pi r^2}, \quad R_m = \frac{r_m}{2\pi r \Delta x}$$

Applying Kirchhoff's current law to each node, the following system of equations can be obtained:

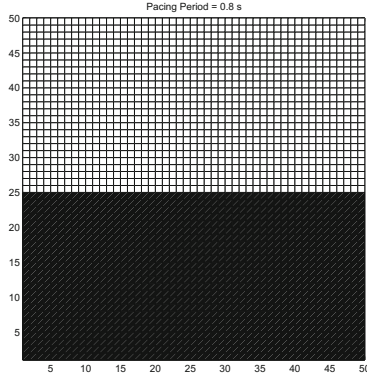
$$\begin{bmatrix} \left(\frac{1}{R_m} + \frac{1}{R_i}\right) & -\frac{1}{R_i} & 0 & \cdots & 0 \\ -\frac{1}{R_i} & \left(\frac{1}{R_m} + \frac{2}{R_i}\right) & -\frac{1}{R_i} & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & \cdots & & -\frac{1}{R_i} & \left(\frac{1}{R_m} + \frac{1}{R_i}\right) \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_N \end{bmatrix} = \begin{bmatrix} I \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

where V_1, V_2, \dots, V_N are the membrane voltages at the nodes, and I is the current injected into the axon at the first node. Using Matlab's `interp1` interpolation function, the length constant at which the membrane voltage has fallen to $V_1 e^{-1}$ can then be determined. This will depend on the number of nodes N as follows:

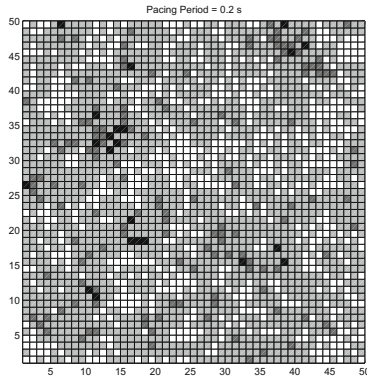
| N | Length Constant (mm) |
|-----|----------------------|
| 5 | 2.733 |
| 10 | 2.570 |
| 20 | 2.510 |
| 40 | 2.506 |
| 80 | 2.506 |
| 160 | 2.505 |

$N = 20$ is sufficient to yield a length constant accuracy of 1%.

1.10 (a) For a heart pacing period of 0.8 s, the plot at $t = 1.65$ s is shown below, where black regions denote state 0 (i.e. quiescent), and white regions denote state 3 (excited):



(b) For a heart pacing period of 0.2 s, the plot at $t = 1.65$ s is shown below, where black regions denote state 0 (i.e. quiescent), dark grey regions denote state 1 (absolute refractory), light grey regions denote state 2 (relative refractory), and white regions denote state 3 (excited):



Some suggestions for implementing this model in Matlab are as follows:

- Define a 50×50 array `H_state` to store the current state of each region in the grid. Update the values of the array during each time increment.
- Define a 50×50 array `S_duration` to store the time elapsed for the current state in each grid region. Whenever a region changes state, its elapsed time should be reset to zero. Whilst remaining in its current state, the time elapsed should be incremented by dT on every time step.
- To plot the grid states, use the command

```
pcolor(H_state), caxis([0 3]), axis('square');
```

- To generate a square array of random total refractory periods, use

```
RP = MRP + SD*randn(size(H_state));
```

- To determine the total number of excited neighbours at any instant, use the code:

```
padded_exc = [zeros(1,52); (H_state(:,50) == 3), ...
(H_state == 3), (H_state(:,1) == 3); zeros(1,52)];

EN = padded_exc(1:50,1:50) + padded_exc(1:50,2:51) + ...
padded_exc(1:50,3:52) + padded_exc(2:51,1:50) + ...
padded_exc(2:51,3:52) + padded_exc(3:52,1:50) + ...
padded_exc(3:52,2:51) + padded_exc(3:52,3:52);
```

This provides an extra “padded” row and column around all edges of `H_state`, allowing the direct summation of the “eight” neighbours.

The full Matlab code listing that generated the above plots (in this instance, for the rapid pacing case) is given below:

```
% Solves a cellular automata model of cardiac electrical
% activation, based on Mitchell et al. (1992), "Cellular
% Automaton Model of Ventricular Fibrillation", IEEE
% Transactions on Biomedical Engineering, 39:253-259.

N = 50;           % number of cells in each column
M = 50;           % number of cells in each row
H_state = zeros(N,M); % state of all cells of the heart
                  % 3 = excited
                  % 2 = absolute refractory
                  % 1 = relative refractory
                  % 0 = quiescent
S_duration = ... % time each cell has spent in its
zeros(size(H_state)); % current state
Dt = 0.002;      % time step (in seconds)
t_end = 1.7;     % final time (in seconds)
MRP = 0.25;      % mean refractory period (in seconds)
SD = 0.1;        % standard deviation of refractory
                  % period (in seconds)
ES = 0.07;       % Excited-state duration
RP = MRP + ...   % Total refractory period
SD*randn(size(H_state)); % (abs. + rel.) for each cell (in seconds)
T = 0.2;         % Heartbeat period (in seconds)

% begin simulation
```

```

for t = 0:Dt:t_end
    % determine number of excited neighbours for each cell
    padded_exc = [zeros(1,M+2); (H_state(:,M) == 3), ...
        (H_state == 3), (H_state(:,1) == 3); zeros(1,M+2)];
    EN = padded_exc(1:N,1:M) + padded_exc(1:N,2:M+1) + ...
        padded_exc(1:N,3:M+2) + padded_exc(2:N+1,1:M) + ...
        padded_exc(2:N+1,3:M+2) + padded_exc(3:N+2,1:M) + ...
        padded_exc(3:N+2,2:M+1) + padded_exc(3:N+2,3:M+2);

    % calculate spread of activation
    for i = 1:M
        for j = 1:N
            if (H_state(i,j) == 0)    % if quiescent
                if (EN(i,j) > 0)
                    H_state(i,j) = 3;
                    S_duration(i,j) = 0;
                else
                    S_duration(i,j) = S_duration(i,j) + Dt;
                end;
            elseif (H_state(i,j) == 3) % if excited
                if (S_duration(i,j) >= ES)
                    H_state(i,j) = 2;
                    S_duration(i,j) = 0;
                else
                    S_duration(i,j) = S_duration(i,j) + Dt;
                end;
            elseif (H_state(i,j) == 2) % if absolute refractory
                if (S_duration(i,j) >= RP(i,j)-0.05)
                    H_state(i,j) = 1;
                    S_duration(i,j) = 0;
                else
                    S_duration(i,j) = S_duration(i,j) + Dt;
                end;
            else % if relative refractory
                if ((S_duration(i,j) <= 0.002)&&(EN(i,j) == 8))
                    H_state(i,j) = 3;
                    S_duration(i,j) = 0;
                elseif ((S_duration(i,j) <= 0.004)&&(EN(i,j) >= 7))
                    H_state(i,j) = 3;
                    S_duration(i,j) = 0;
                elseif ((S_duration(i,j) <= 0.006)&&(EN(i,j) >= 6))
                    H_state(i,j) = 3;
                    S_duration(i,j) = 0;
                elseif ((S_duration(i,j) <= 0.008)&&(EN(i,j) >= 5))
                    H_state(i,j) = 3;
                    S_duration(i,j) = 0;
                elseif ((S_duration(i,j) <= 0.012)&&(EN(i,j) >= 4))
                    H_state(i,j) = 3;
                    S_duration(i,j) = 0;
                elseif ((S_duration(i,j) <= 0.02)&&(EN(i,j) >= 3))
                    H_state(i,j) = 3;
                    S_duration(i,j) = 0;
                elseif ((S_duration(i,j) <= 0.05)&&(EN(i,j) >= 2))

```

```

    H_state(i,j) = 3;
    S_duration(i,j) = 0;
elseif ((S_duration(i,j) > 0.05)&&(EN(i,j) >= 1))
    H_state(i,j) = 3;
    S_duration(i,j) = 0;
elseif (S_duration(i,j) > 0.05)
    H_state(i,j) = 0;
    S_duration(i,j) = 0;
else
    S_duration(i,j) = S_duration(i,j) + Dt;
end;
end;
end; % j loop
end; % i loop
% if necessary, excite the pacemaker cell
if (mod(t,T) < Dt)
    if (H_state(N,round(M/4)) ~= 2)
        H_state(N,round(M/4)) = 3;
        S_duration(N,round(M/4)) = 0;
    end;
end;

% plot states
pause(0.1);
pcolor(H_state), caxis([0 3]), axis('square'), ...
title('Simulation of Cardiac Electrical Activity ');
end; % t loop

```

Problems of Chap. 2

2.1 (a) To solve $\frac{dx}{dt} = \alpha_x(1-x) - \beta_x x$, we re-write it as the non-homogeneous linear equation:

$$\frac{dx}{dt} + (\alpha_x + \beta_x)x = \alpha_x$$

To solve this ODE, we first solve the homogeneous form

$$\frac{dx}{dt} + (\alpha_x + \beta_x)x = 0$$

with characteristic equation

$$m + (\alpha_x + \beta_x) = 0$$

$$\therefore m = -(\alpha_x + \beta_x)$$

Hence, the homogeneous solution is

$$x_h = Ce^{-(\alpha_x + \beta_x)t}$$

where C is a constant.

Next, we solve for a particular solution, which we can conveniently take as the steady-state value of x . This steady-state value can be obtained from the original non-homogeneous ODE by simply setting $\frac{dx}{dt} = 0$,

$$\Rightarrow (\alpha_x + \beta_x)x = \alpha_x$$

Hence, a particular solution is

$$x_p = \frac{\alpha_x}{\alpha_x + \beta_x}$$

The general solution is the sum of the homogeneous and particular solutions:

$$x = Ce^{-(\alpha_x + \beta_x)t} + \frac{\alpha_x}{\alpha_x + \beta_x}$$

When $t = 0$, $x = x_0$

$$\begin{aligned} \therefore x_0 &= C + \frac{\alpha_x}{\alpha_x + \beta_x} \\ \Rightarrow C &= x_0 - \frac{\alpha_x}{\alpha_x + \beta_x} \end{aligned}$$

Thus, the solution of the ODE is

$$x = \left[x_0 - \frac{\alpha_x}{\alpha_x + \beta_x} \right] e^{-(\alpha_x + \beta_x)t} + \frac{\alpha_x}{\alpha_x + \beta_x}$$

(b) From above, the steady-state solution, x_∞ , is given by

$$x_\infty = \frac{\alpha_x}{\alpha_x + \beta_x}$$

and since α_x, β_x are functions of the membrane potential, which equals V_{clamp} during the voltage-clamp, this steady state value is more accurately expressed as

$$x_\infty = \frac{\alpha_x(V_{clamp})}{\alpha_x(V_{clamp}) + \beta_x(V_{clamp})}$$

Hence, a reasonable estimate for x_0 would be the corresponding steady-state value at the holding potential V_{hold} , prior to the onset of the voltage-clamp step, or

$$x_0 = \frac{\alpha_x(V_{hold})}{\alpha_x(V_{hold}) + \beta_x(V_{hold})}$$

2.2 (a) The total force applied to the muscle is given by

$$F = k_1 x_1 = b \frac{dx_2}{dt} + k_2 x_2$$

and since the total length x is held fixed at X_m , then $x_1 = X_m - x_2$. Substituting this value for x_1 into the right-most equality above, we have:

$$\begin{aligned} b \frac{dx_2}{dt} + k_2 x_2 &= k_1 [X_m - x_2] \\ b \frac{dx_2}{dt} + (k_1 + k_2)x_2 &= k_1 X_m \\ \text{or } \frac{dx_2}{dt} + \left(\frac{k_1 + k_2}{b}\right)x_2 &= \frac{k_1}{b} X_m \end{aligned}$$

To solve this ODE, the corresponding homogeneous equation is

$$\frac{dx_2}{dt} + \left(\frac{k_1 + k_2}{b}\right)x_2 = 0$$

with homogeneous solution

$$x_{2,h} = C e^{-\left(\frac{k_1+k_2}{b}\right)t}$$

where C is a constant. A particular solution can be found for the steady-state value of x_2 by setting $\frac{dx_2}{dt} = 0$ in the non-homogeneous ODE. This yields the particular solution

$$x_{2,p} = \left(\frac{k_1}{k_1 + k_2}\right) X_m$$

Hence, the general solution is the sum of the homogeneous and particular solutions:

$$x_2 = C e^{-\left(\frac{k_1+k_2}{b}\right)t} + \left(\frac{k_1}{k_1 + k_2}\right) X_m$$

When $t = 0$, $x_2 = 0$, which implies $C = -\left(\frac{k_1}{k_1+k_2}\right) X_m$. Hence

$$x_2 = \left(\frac{k_1}{k_1 + k_2}\right) X_m \left[1 - e^{-\left(\frac{k_1+k_2}{b}\right)t}\right]$$

Knowing x_2 , we can readily obtain x_1 :

$$\begin{aligned} x_1 &= X_m - x_2 \\ &= \left(\frac{k_2}{k_1 + k_2}\right) X_m + \left(\frac{k_1}{k_1 + k_2}\right) X_m e^{-\left(\frac{k_1+k_2}{b}\right)t} \\ &= \left(\frac{X_m}{k_1 + k_2}\right) \left[k_2 + k_1 e^{-\left(\frac{k_1+k_2}{b}\right)t}\right] \end{aligned}$$

Since $F = k_1 x_1$, the applied force is readily determined as

$$F = \left(\frac{k_1 X_m}{k_1 + k_2} \right) \left[k_2 + k_1 e^{-\left(\frac{k_1 + k_2}{b}\right)t} \right]$$

(b) The fixed applied force, F_m , satisfies

$$F_m = k_1 x_1 = b \frac{dx_2}{dt} + k_2 x_2$$

and working with variable x_2 , we can rewrite the above as

$$\frac{dx_2}{dt} + \frac{k_2}{b} x_2 = \frac{F_m}{b}$$

The homogeneous ODE is

$$\frac{dx_2}{dt} + \frac{k_2}{b} x_2 = 0$$

with solution

$$x_{2,h} = C e^{-\left(\frac{k_2}{b}\right)t}$$

where C is a constant. The particular solution can be found from the steady-state value of x_2 in the non-homogeneous ODE by setting $\frac{dx_2}{dt} = 0$. We therefore obtain the particular solution

$$x_{2,p} = \frac{F_m}{k_2}$$

and the general solution

$$x_2 = C e^{-\left(\frac{k_2}{b}\right)t} + \frac{F_m}{k_2}$$

When $t = 0$, $x_2 = 0$, which implies $C = -\frac{F_m}{k_2}$. Hence

$$x_2 = \frac{F_m}{k_2} \left[1 - e^{-\left(\frac{k_2}{b}\right)t} \right]$$

Furthermore, since $F_m = k_1 x_1$, we have $x_1 = F_m/k_1$. Hence, the total length of the muscle is given by

$$x = x_1 + x_2 = F_m \left[\frac{1}{k_1} + \frac{1}{k_2} - \frac{1}{k_2} e^{-\left(\frac{k_2}{b}\right)t} \right]$$

2.3 (a) The total current flowing through the parallel resistive and capacitive branches is equal to the applied stimulus current. Hence, the ODE for this system is given by

$$C \frac{dV}{dt} + \frac{V}{R} = I$$

or

$$\frac{dV}{dt} + \frac{V}{RC} = \frac{I}{C}$$

The homogeneous ODE for this system is

$$\frac{dV}{dt} + \frac{V}{RC} = 0$$

with homogeneous solution $V_h = Ce^{-\frac{t}{RC}}$, where C is a constant. The particular solution, $V_{2,p}$, can be obtained by setting $\frac{dV}{dt} = 0$ in the original ODE to obtain:

$$V_{2,p} = IR$$

Hence, the general solution, given by the sum of the homogeneous and particular solutions, is

$$V = IR + Ce^{-\frac{t}{RC}}$$

When $t = 0$, $V = 0$, which implies $C = -IR$. Hence,

$$V = IR \left(1 - e^{-\frac{t}{RC}}\right)$$

Now, when $V = V_{th}$, the required stimulus duration T must therefore satisfy

$$V_{th} = IR \left(1 - e^{-\frac{T}{RC}}\right)$$

Hence, the strength-duration characteristic is

$$I = \frac{V_{th}}{R \left(1 - e^{-\frac{T}{RC}}\right)}$$

(b) From the strength-duration characteristic above, the rheobase may be found from the stimulus current I as $T \rightarrow \infty$, or simply

$$I_{rheobase} = \frac{V_{th}}{R}$$

The chronaxie can be found by solving for stimulus duration T corresponding to an applied current of $2I_{rheobase}$. From the previous strength-duration characteristic, we have:

$$\begin{aligned}
 2 \frac{V_{th}}{R} &= \frac{V_{th}}{R \left(1 - e^{-\frac{T}{RC}}\right)} \\
 2 &= \frac{1}{1 - e^{-\frac{T}{RC}}} \\
 1 - e^{-\frac{T}{RC}} &= \frac{1}{2} \\
 e^{-\frac{T}{RC}} &= \frac{1}{2} \\
 \frac{T}{RC} &= \ln 2 \\
 \therefore T_{chronaxie} &= RC \ln 2 \\
 &\approx 0.69RC
 \end{aligned}$$

2.4 (a) The pair of ODEs describing the coupled spring masses is

$$\begin{aligned}
 M\ddot{x}_1 &= -kx_1 + k(x_2 - x_1) \\
 M\ddot{x}_2 &= k(x_1 - x_2) - kx_2
 \end{aligned}$$

or

$$\begin{aligned}
 M\ddot{x}_1 &= -2kx_1 + kx_2 \\
 M\ddot{x}_2 &= kx_1 - 2kx_2
 \end{aligned}$$

(b) Adding the above ODEs together produces

$$M\ddot{x}_1 + M\ddot{x}_2 = -kx_1 - kx_2$$

or simply

$$M\ddot{y}_1 = -ky_1$$

where we have used the substitution $y_1 = x_1 + x_2$. To solve this ODE, we can write its characteristic equation as

$$\begin{aligned}
 Mm^2 + k &= 0 \\
 \Rightarrow m &= \pm i\sqrt{\frac{k}{M}}
 \end{aligned}$$

Hence,

$$\begin{aligned}
 y_1 &= C_1 e^{it\sqrt{\frac{k}{M}}} + C_2 e^{-it\sqrt{\frac{k}{M}}} \\
 &= C_1 \cos\left(t\sqrt{\frac{k}{M}}\right) + C_1 i \sin\left(t\sqrt{\frac{k}{M}}\right) + C_2 \cos\left(t\sqrt{\frac{k}{M}}\right) - C_2 i \sin\left(t\sqrt{\frac{k}{M}}\right)
 \end{aligned}$$

where C_1 and C_2 are constants. Differentiating this expression, we obtain

$$\begin{aligned} \dot{y}_1 = & -C_1\sqrt{\frac{k}{M}}\sin\left(t\sqrt{\frac{k}{M}}\right) + C_1\sqrt{\frac{k}{M}}i\cos\left(t\sqrt{\frac{k}{M}}\right) \\ & - C_2\sqrt{\frac{k}{M}}\sin\left(t\sqrt{\frac{k}{M}}\right) - C_2\sqrt{\frac{k}{M}}i\cos\left(t\sqrt{\frac{k}{M}}\right) \end{aligned}$$

When $t = 0$, $y_1 = u_1 + u_2$ and $\dot{y}_1 = \dot{x}_1 + \dot{x}_2 = 0$. Substituting these into the above equations for y_1 and \dot{y}_1 yields

$$\begin{aligned} u_1 + u_2 &= C_1 + C_2 \\ 0 &= C_1\sqrt{\frac{k}{M}}i - C_2\sqrt{\frac{k}{M}}i \end{aligned}$$

or

$$C_1 = C_2 = \frac{1}{2}(u_1 + u_2)$$

Hence,

$$y_1 = (u_1 + u_2)\cos\left(t\sqrt{\frac{k}{M}}\right)$$

Similarly, we can subtract the two original ODEs in x_1 and x_2 to obtain:

$$M\ddot{x}_1 - M\ddot{x}_2 = -3kx_1 + 3kx_2$$

or

$$M\ddot{y}_2 = -3ky_2$$

where we have used the substitution $y_2 = x_1 - x_2$. The characteristic equation of this ODE is

$$\begin{aligned} Mm^2 + 3k &= 0 \\ \Rightarrow m &= \pm i\sqrt{\frac{3k}{M}} \end{aligned}$$

Hence,

$$\begin{aligned} y_2 &= C_3e^{it\sqrt{\frac{3k}{M}}} + C_4e^{-it\sqrt{\frac{3k}{M}}} \\ &= C_3\cos\left(t\sqrt{\frac{3k}{M}}\right) + C_3i\sin\left(t\sqrt{\frac{3k}{M}}\right) + C_4\cos\left(t\sqrt{\frac{3k}{M}}\right) - C_4i\sin\left(t\sqrt{\frac{3k}{M}}\right) \end{aligned}$$

where C_3 and C_4 are constants. Differentiating this expression, we obtain

$$\begin{aligned} \dot{y}_2 = & -C_3\sqrt{\frac{3k}{M}}\sin\left(t\sqrt{\frac{3k}{M}}\right) + C_3\sqrt{\frac{3k}{M}}i\cos\left(t\sqrt{\frac{3k}{M}}\right) \\ & - C_4\sqrt{\frac{3k}{M}}\sin\left(t\sqrt{\frac{3k}{M}}\right) - C_4\sqrt{\frac{3k}{M}}i\cos\left(t\sqrt{\frac{3k}{M}}\right) \end{aligned}$$

When $t = 0$, $y_2 = u_1 - u_2$ and $\dot{y}_2 = \dot{x}_1 - \dot{x}_2 = 0$. Substituting these into the above equations for y_2 and \dot{y}_2 yields

$$\begin{aligned} u_1 - u_2 &= C_3 + C_4 \\ 0 &= C_3\sqrt{\frac{3k}{M}}i - C_4\sqrt{\frac{3k}{M}}i \end{aligned}$$

or

$$C_3 = C_4 = \frac{1}{2}(u_1 - u_2)$$

Hence,

$$y_2 = (u_1 - u_2)\cos\left(t\sqrt{\frac{3k}{M}}\right)$$

To recover x_1 and x_2 from y_1 and y_2 , we can use the expressions

$$\begin{aligned} x_1 &= \frac{1}{2}(y_1 + y_2) \\ x_2 &= \frac{1}{2}(y_1 - y_2) \end{aligned}$$

to finally obtain

$$\begin{aligned} x_1 &= \frac{1}{2}(u_1 + u_2)\cos\left(t\sqrt{\frac{k}{M}}\right) + \frac{1}{2}(u_1 - u_2)\cos\left(t\sqrt{\frac{3k}{M}}\right) \\ x_2 &= \frac{1}{2}(u_1 + u_2)\cos\left(t\sqrt{\frac{k}{M}}\right) - \frac{1}{2}(u_1 - u_2)\cos\left(t\sqrt{\frac{3k}{M}}\right) \end{aligned}$$

2.5 The ODEs for the glucose-insulin model are

$$\begin{aligned} \frac{dI}{dt} &= k_2 I_p(t) - k_3 I \\ \frac{dG}{dt} &= B_0 - (k_1 + k_4 I + k_5 + k_6 I) \end{aligned}$$

The following Matlab code (GI_solve.m and GI_prime.m) will numerically-integrate these and display the solution:

GI_solve.m

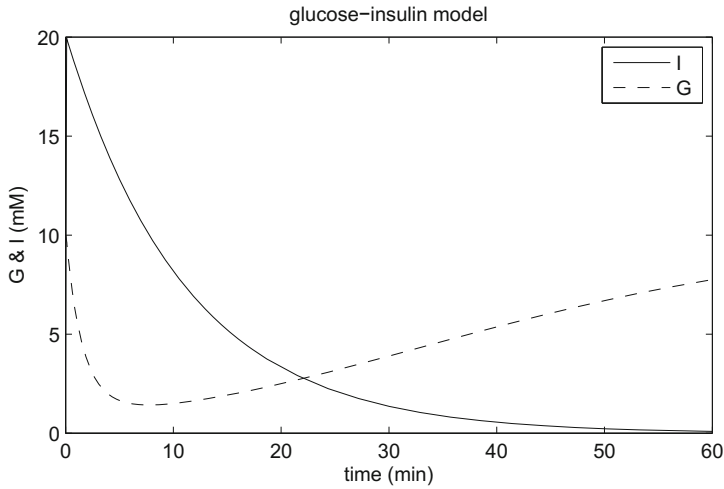
```
[time, y_out] = ode15s('GI_prime', [0 60], [0 10]);

plot(time, y_out(:,1),'k', time, y_out(:,2),'k--'), ...
     xlabel('time (min)'), ylabel('G & I (mM)'), ...
     title('glucose-insulin model'), legend('I', 'G');
```

GI_prime.m

```
function y_prime = GI_prime(t,y)
    y_prime = zeros(2,1);
    k1 = 0.015;
    k2 = 1;
    k3 = 0.09;
    k4 = 0.01;
    k5 = 0.035;
    k6 = 0.02;
    B0 = 0.5;
    I = y(1);
    G = y(2);
    if (t < 0.1)
        Ip = 200;
    else
        Ip = 0;
    end;
    y_prime(1) = k2*Ip - k3*I;
    y_prime(2) = B0 - (k1+k4*I+k5+k6*I)*G;
```

producing the plot:



2.6 (a) Denoting the total outflow from the ventricle as Q , we can write

$$Q = C_s \frac{dP_s}{dt} + \frac{P_s}{R_s}$$

To evaluate Q , there are two cases to consider:

- (1) $P_v \leq P_s$. In this case, the aortic valve (diode) prevents any backflow and $Q = 0$.
- (2) $P_v > P_s$. Denoting the flows through elements R_o and L_o by Q_R and Q_L respectively, we have

$$\begin{aligned} Q &= Q_L + Q_R \\ &= Q_L + \frac{P_v - P_s}{R_o} \end{aligned}$$

Combining both cases above:

$$C_s \frac{dP_s}{dt} + \frac{P_s}{R_s} = \begin{cases} Q_L + \frac{P_v - P_s}{R_o} & P_v > P_s \\ 0 & P_v \leq P_s \end{cases}$$

which can be re-arranged to

$$\frac{dP_s}{dt} = \begin{cases} \frac{Q_L}{C_s} + \frac{P_v - P_s}{R_o C_s} - \frac{P_s}{R_s C_s} & P_v > P_s \\ -\frac{P_s}{R_s} & P_v \leq P_s \end{cases}$$

To evaluate Q_L , we can consider the same two cases:

- (1) $P_v \leq P_s$. In this case, $Q = 0$ and $Q_R = -Q_L$. The pressure drop across L_o is

$$Q_R R_o = -R_o Q_L = L_o \frac{dQ_L}{dt}$$

Hence,

$$\frac{dQ_L}{dt} = -\frac{R_o}{L_o} Q_L$$

(2) $P_v > P_s$. In this case, the pressure drop across L_o is given by:

$$P_v - P_s = L_o \frac{dQ_L}{dt}$$

$$\therefore \frac{dQ_L}{dt} = \frac{P_v - P_s}{L_o}$$

Hence, the pair of ODEs characterising the system is

$$\frac{dP_s}{dt} = \begin{cases} \frac{Q_L}{C_s} + \frac{P_v - P_s}{R_o C_s} - \frac{P_s}{R_s C_s} & P_v > P_s \\ -\frac{P_s}{R_s} & P_v \leq P_s \end{cases}$$

$$\frac{dQ_L}{dt} = \begin{cases} \frac{P_v - P_s}{L_o} & P_v > P_s \\ -\frac{R_o}{L_o} Q_L & P_v \leq P_s \end{cases}$$

(b) The following Matlab code (Windkessel_solve.m and Windkessel_prime.m) will solve these ODEs over a time interval of $100T$ (i.e. 100 heartbeats) to obtain a sufficient steady-state P_s waveform, then use the final state variable values as subsequent initial values to solve for a further time interval of T . The initial values for P_s and Q_L are both zero at the start of the $100T$ simulation:

Windkessel_solve.m

```
% define global parameters
global R0 L0 Cs Rs P T tc
R0 = 0.06;      % mmHg.s/cm^3
L0 = 0.2;      % mmHg.s^2/cm^3
Cs = 1;        % cm^3/mmHg
Rs = 1.4;      % mmHg.s/cm^3
P = 120;       % mmHg
T = 1;         % s
tc = 0.35;     % s

% solve first for 100 heartbeats
Ps_init = 0;
QL_init = 0;
[time, y_out] = ode15s('Windkessel_prime', ...
    [0 100*T], [Ps_init QL_init]);
```

```

% solve for one additional heartbeat for steady-state
Ps_init = y_out(end,1);
QL_init = y_out(end,2);
[time, y_out] = ode15s('Windkessel_prime', ...
    [0 T], [Ps_init QL_init]);

plot(time, y_out(:,1),'k'), ...
    xlabel('time (s)'), ylabel('P_s (mmHg)'), ...
    title('Systemic pressure - windkessel model');

```

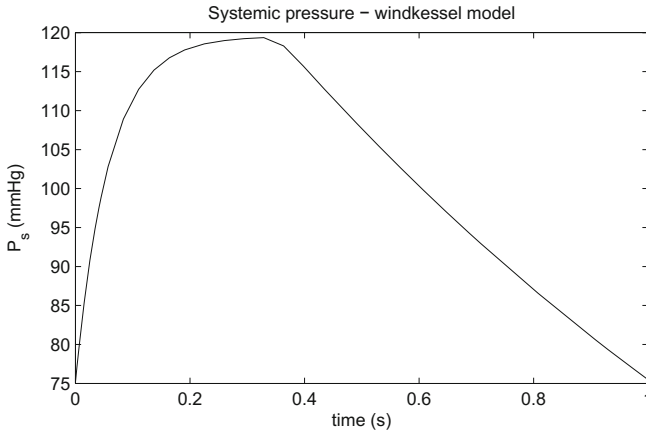
Windkessel_prime.m

```

function y_prime = Windkessel_prime(t,y)
global R0 L0 Cs Rs P T tc
    y_prime = zeros(2,1);
    Ps = y(1);
    QL = y(2);
    tt = mod(t,T);    % to produce a periodic waveform
    if (tt < tc)
        Pv = P;
    else
        Pv = 0;
    end;
    if (Pv > Ps)
        y_prime(1) = QL/Cs + (Pv-Ps)/(R0*Cs) - Ps/(Rs*Cs);
        y_prime(2) = (Pv-Ps)/L0;
    else
        y_prime(1) = -Ps/Rs;
        y_prime(2) = -R0*QL/L0;
    end;
end;

```

producing the following plot:



2.7 The following Matlab code (ECG_solve.m and ECG_prime.m) will solve the ECG model over a time interval of 1 s:

ECG_solve.m

```
% define global parameters
global A B TH w z0;
A = [1.2, -5, 30, -7.5, 0.75];
B = [0.25, 0.1, 0.1, 0.1, 0.4];
TH = [-1/3, -1/12, 0, 1/12, 1/2]*pi;
w = 2*pi;
z0 = 0;

[time, y_out] = ode15s('ECG_prime', [0 1], [-1 0 0]);

plot(time, y_out(:,3),'k'), ...
     xlabel('time (s)'), ylabel('mV'), ...
     title('Synthetic ECG Signal');
```

ECG_prime.m

```
function Y_prime = ECG_prime(t,Y)
global A B TH w z0;
Y_prime = zeros(3,1);
x = Y(1);
y = Y(2);
z = Y(3);
alpha = 1-sqrt(x^2+y^2);
th = atan2(y,x);
Y_prime(1) = alpha*x - w*y;
```



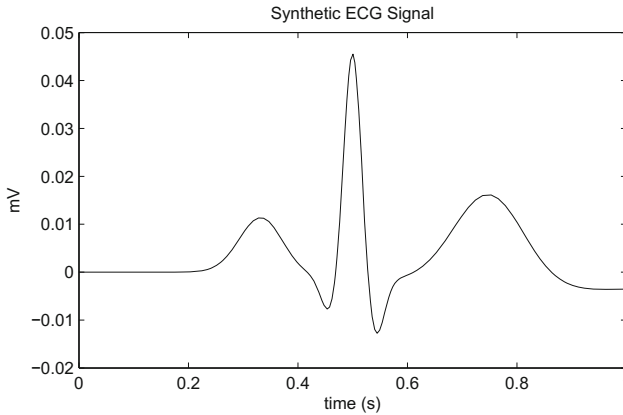
```

Y_prime(2) = alpha*y + w*x;

Y_prime(3) = -(z-z0);
for ii = 1:5
    Y_prime(3) = Y_prime(3)-...
        A(ii)*(th-TH(ii))*exp(-(th-TH(ii))^2/(2*B(ii)^2));
end;

```

to produce the plot



2.8 The Frankenhaeuser–Huxley neural model is a typical example of pitfalls typically encountered when modelling biological systems. A chief difficulty arises from the choice of appropriate units. Since (1) the membrane capacitance is in units of $\mu\text{F cm}^{-2}$, (2) the transmembrane potential is in units of mV and (3) time is in ms, the membrane ionic current i_{ion} in the equation

$$\frac{dV}{dt} = -\frac{i_{ion}}{C_m}$$

should be in units of $\mu\text{A cm}^{-2}$. Naive use of the supplied parameters for i_{Na} , i_K and i_P would result in units of mA cm^{-2} , so these evaluated currents must be multiplied by a factor of 1000 (note that i_L is already in the correct units of $\mu\text{A cm}^{-2}$). Another difficulty is that the stimulus current duration is only 0.12 ms, so that naive use of any of Matlab's ODE solvers such as `ode15s` would likely result in too large a step size, bypassing the stimulus. It is therefore necessary to restrict the step size, which can be achieved by specifying a maximum step with Matlab's `odeset` command. These 'tricks' have been implemented in the following Matlab code (`FH_solve.m` and `FH_prime.m`):

FH_solve.m

```

% solves the Frankenhaeuser--Huxley neural model
global Cm P_Na P_K P_P g_L V_L Na_o;
global Na_i K_o K_i I_s t_on t_dur;
global Er F R T;

Cm = 2;           % uC/cm^2
P_Na = 0.008;    % cm/s
P_K = 0.0012;   % cm/s
P_P = 0.00054;  % cm/s
g_L = 30.3;     % mS/cm^2
V_L = 0.026;    % mV
Na_o = 114.5;   % mM
Na_i = 13.74;   % mM
K_o = 2.5;      % mM
K_i = 120;      % mM
I_s = 1;        % mA/cm^2
t_on = 1;       % ms
t_dur = 0.12;   % ms
Er = -70;       % mV
F = 96.49;      % C/mmol
R = 8.31;       % J/(mol*K)
T = 310;        % K

Y_init = [0, 0.0005, 0.8249, 0.0268, 0.0049];
options = odeset('MaxStep', 0.01);
% solve first for control case (no TTX)
[time_1, Y_out_1] = ode15s('FH_prime', [0 5], Y_init, options);
% next, solve for presence of TTX
P_Na = 0.2*P_Na;
[time_2, Y_out_2] = ode15s('FH_prime', [0 5], Y_init, options);
plot(time_1, Y_out_1(:,1)+Er, 'k', time_2, Y_out_2(:,1)+Er, 'k--'), ...
     xlabel('ms'), ylabel('mV'), legend('Control', 'TTX'), ...
     title('TTX effect on neural action potential');

```

FH_prime.m

```

function y_out = FH_prime(t,y)
% calculates derivatives for solving the
% Frankenhaeuser--Huxley neural model.
global Cm P_Na P_K P_P g_L V_L Na_o;
global Na_i K_o K_i I_s t_on t_dur;
global Er F R T;
y_out = zeros(5,1);

V = y(1);
m = y(2);
h = y(3);
n = y(4);

```

```

p = y(5);

alpha_m = 0.36*(V-22)/(1-exp((22-V)/3));
beta_m = 0.4*(13-V)/(1-exp((V-13)/20));
alpha_h = 0.1*(-10-V)/(1-exp((V+10)/6));
beta_h = 4.5/(1+exp((45-V)/10));
alpha_n = 0.02*(V-35)/(1-exp((35-V)/10));
beta_n = 0.05*(10-V)/(1-exp((V-10)/10));
alpha_p = 0.006*(V-40)/(1-exp((40-V)/10));
beta_p = 0.09*(-25-V)/(1-exp((V+25)/20));

E = V+Er;

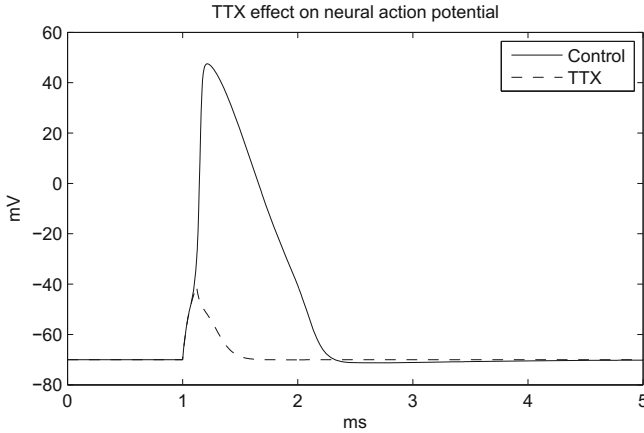
i_Na = 1000*m^2*h*P_Na*(E*F^2/(R*T))*...
      (Na_o - Na_i*exp(E*F/(R*T)))...
      / (1-exp(E*F/(R*T)));
i_K = 1000*n^2*p_K*(E*F^2/(R*T))*...
      (K_o - K_i*exp(E*F/(R*T)))...
      / (1-exp(E*F/(R*T)));
i_P = 1000*p^2*p_P*(E*F^2/(R*T))*...
      (Na_o - Na_i*exp(E*F/(R*T)))...
      / (1-exp(E*F/(R*T)));
i_L = g_L*(V-V_L);

if ((t >= t_on)&&(t<t_on+t_dur))
    i_s = 1000*I_s;
else
    i_s = 0;
end;

y_out(1) = -(i_Na+i_K+i_P+i_L-i_s)/Cm;
y_out(2) = alpha_m*(1-m)-beta_m*m;
y_out(3) = alpha_h*(1-h)-beta_h*h;
y_out(4) = alpha_n*(1-n)-beta_n*n;
y_out(5) = alpha_p*(1-p)-beta_p*p;

```

to produce the plot



2.9 (a) For the isometric contraction case, total muscle length L is held at L_0 . The total tension T is given by

$$\begin{aligned}
 T &= T_p + T_s \\
 &= \beta (e^{\alpha(L-L_0)} - 1) + \beta (e^{\alpha L_s} - 1) \\
 &= \beta (e^{\alpha L_s} - 1) \quad (\text{since } L = L_0) \\
 &= \beta (e^{\alpha(L_0-L_c)} - 1)
 \end{aligned}$$

Taking derivatives of both sides and using the chain rule:

$$\begin{aligned}
 \frac{dT}{dt} &= \frac{d}{dL_c} [e^{\alpha(L_0-L_c)} - 1] \frac{dL_c}{dt} \\
 &= -\alpha e^{\alpha(L_0-L_c)} \frac{dL_c}{dt} \\
 &= -\alpha e^{\alpha(L_0-L_c)} \left(\frac{a [T_s - S_0 f(t)]}{T_s + \gamma S_0} \right)
 \end{aligned}$$

Hence, the pair of ODEs describing this isometric contraction is

$$\begin{aligned}
 \frac{dT}{dt} &= -\alpha e^{\alpha(L_0-L_c)} \left(\frac{a [T_s - S_0 f(t)]}{T_s + \gamma S_0} \right) \\
 \frac{dL_c}{dt} &= \frac{a [T_s - S_0 f(t)]}{T_s + \gamma S_0}
 \end{aligned}$$

These are implemented in following Matlab code (isometric_solve.m and isometric_prime.m):

isometric_solve.m

```

global L0 alpha beta a S0 gamma t_0 t_ip;

L0 = 10;      % mm
alpha = 15;   % mm
beta = 5;     % mN
a = 0.66;    % 1/mm
S0 = 4;      % mN
gamma = 0.45;
t_0 = 0.05;  % s
t_ip = 0.2;  % s

Y_init = [0, 10];
options = odeset('MaxStep', 0.01);

[time, Y_out] = ode15s('isometric_prime', [0 1], Y_init, options);
plot(time, Y_out(:,1), 'k'), ...
     xlabel('s'), ylabel('mN'),
     title('Isometric contraction - muscle tension');

```

isometric_prime.m

```

function y_prime = isometric_prime(t,y)
global L0 alpha beta a S0 gamma t_0 t_ip;
y_prime = zeros(2,1);

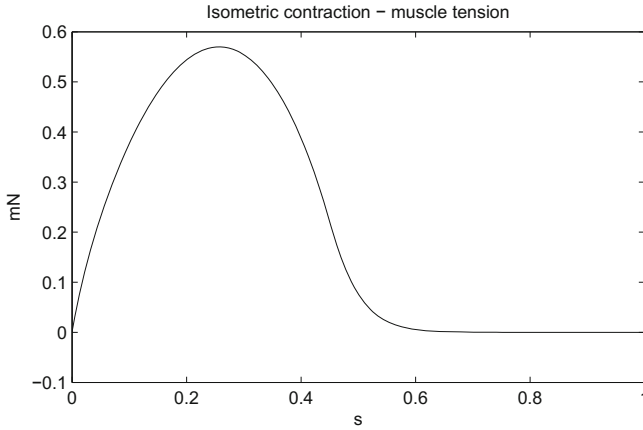
if (t<2*t_ip+t_0)
    f = sin((pi/2)*(t+t_0)/(t_ip+t_0));
else
    f = 0;
end

T = y(1);
Lc = y(2);
Ls = L0-Lc;
Ts = beta*(exp(alpha*Ls)-1);

y_prime(1) = -alpha*exp(L0-Lc)*...
             a*(Ts-S0*f)/(Ts+gamma*S0);
y_prime(2) = a*(Ts-S0*f)/(Ts+gamma*S0);

```

to produce the plot



(b) For the isotonic contraction case, total muscle tension T is held at 0. This total tension T may be written as

$$\begin{aligned} T &= T_p + T_s \\ &= \beta (e^{\alpha(L-L_0)} - 1) + \beta (e^{\alpha L_s} - 1) \end{aligned}$$

Taking derivatives of both sides and using the chain rule:

$$\begin{aligned} \frac{dT}{dt} &= \frac{d}{dL} [e^{\alpha(L-L_0)} - 1] \frac{dL}{dt} + \frac{d}{dL_s} [e^{\alpha L_s} - 1] \frac{dL_s}{dt} \\ &= \alpha e^{\alpha(L-L_0)} \frac{dL}{dt} + \alpha e^{\alpha L_s} \frac{dL_s}{dt} \\ &= \alpha e^{\alpha(L-L_0)} \frac{dL}{dt} + \alpha e^{\alpha(L-L_c)} \frac{d(L-L_c)}{dt} \end{aligned}$$

and since this contraction is isotonic, $\frac{dT}{dt} = 0$. Hence,

$$\begin{aligned} 0 &= \alpha e^{\alpha(L-L_0)} \frac{dL}{dt} + \alpha e^{\alpha(L-L_c)} \frac{d(L-L_c)}{dt} \\ \alpha e^{\alpha(L-L_c)} \frac{dL_c}{dt} &= \alpha e^{\alpha(L-L_0)} \frac{dL}{dt} + \alpha e^{\alpha(L-L_c)} \frac{dL}{dt} \\ \alpha e^{\alpha(L-L_c)} \frac{dL_c}{dt} &= (\alpha e^{\alpha(L-L_0)} + \alpha e^{\alpha(L-L_c)}) \frac{dL}{dt} \\ \therefore \frac{dL}{dt} &= \left(\frac{e^{\alpha(L-L_c)}}{e^{\alpha(L-L_0)} + e^{\alpha(L-L_c)}} \right) \frac{dL_c}{dt} \\ &= \left(\frac{e^{\alpha L_c}}{e^{\alpha L_0} + e^{\alpha L_c}} \right) \frac{dL_c}{dt} \end{aligned}$$

$$= \left(\frac{e^{\alpha L_c}}{e^{\alpha L_0} + e^{\alpha L_c}} \right) \frac{a [T_s - S_0 f(t)]}{T_s + \gamma S_0}$$

Hence, the pair of ODEs describing the isotonic contraction is

$$\begin{aligned} \frac{dL}{dt} &= \left(\frac{e^{\alpha L_c}}{e^{\alpha L_0} + e^{\alpha L_c}} \right) \frac{a [T_s - S_0 f(t)]}{T_s + \gamma S_0} \\ \frac{dL_c}{dt} &= \frac{a [T_s - S_0 f(t)]}{T_s + \gamma S_0} \end{aligned}$$

These are implemented in following Matlab code (isotonic_solve.m and isotonic_prime.m):

isotonic_solve.m

```
global L0 alpha beta a S0 gamma t_0 t_ip;

L0 = 10;           % mm
alpha = 15;        % mm
beta = 5;          % mN
a = 0.66;          % 1/mm
S0 = 4;            % mN
gamma = 0.45;
t_0 = 0.05;        % s
t_ip = 0.2;        % s

Y_init = [10, 10];
options = odeset('MaxStep', 0.01);

[time, Y_out] = ode15s('isotonic_prime', [0 1], Y_init, options);
plot(time, Y_out(:,1), 'k'), ...
    xlabel('s'), ylabel('mm'),
    title('Isotonic contraction - muscle length');
```

isotonic_prime.m

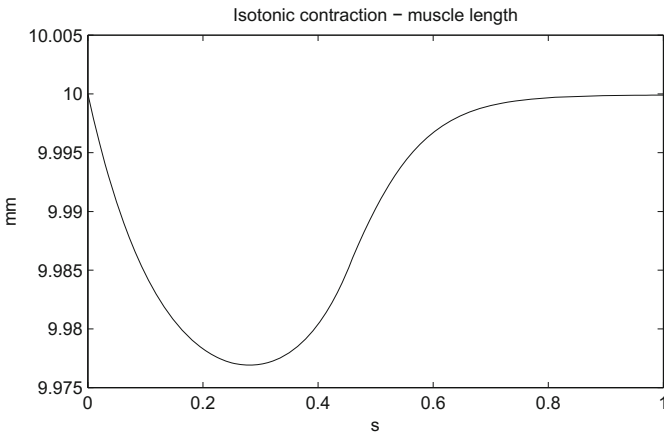
```
function y_prime = isotonic_prime(t,y)
global L0 alpha beta a S0 gamma t_0 t_ip;
y_prime = zeros(2,1);

if (t < 2*t_ip + t_0)
    f = sin((pi/2)*(t+t_0)/(t_ip+t_0));
else
    f = 0;
end

L = y(1);
```

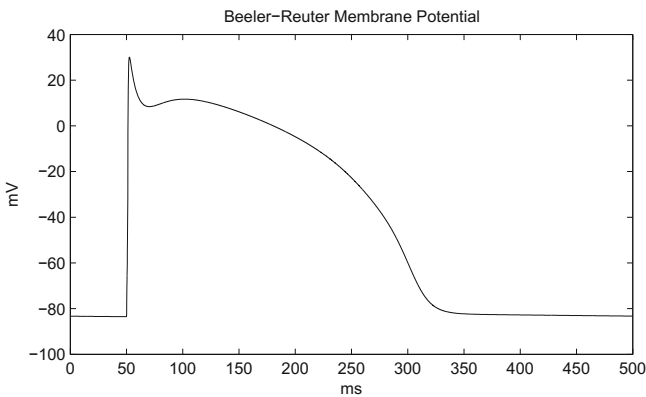
```
Lc = y(2);  
Ls = L-Lc;  
Ts = beta*(exp(alpha*Ls)-1);  
  
y_prime(1) = (exp(alpha*Lc)/(exp(alpha*L0)+exp(alpha*Lc)))*...  
    a*(Ts-S0*f)/(Ts+gamma*S0);  
y_prime(2) = a*(Ts-S0*f)/(Ts+gamma*S0);
```

to produce the plot



Problems of Chap. 3

3.1 (a)



This plot was produced by the following Matlab code (BR_solve.m and BR_prime.m). Note that the maximum step for `ode15s` was conservatively set to 0.1 ms to ensure the applied stimulus at $t = 50$ ms was not bypassed:

BR_solve.m

```

% solves the Beeler-Reuter (1977) model
global Cm r_Ca Ca_SR k_up A_K1 A_x1 g_Na;
global g_NaC V_Na g_s A_s t_on t_dur;

Cm = 1;           % uC/cm^2
r_Ca = 1e-7;     % M*cm^2/nC
Ca_SR = 1e-7;   % M
k_up = 0.07;     % 1/ms
A_K1 = 0.35;     % uA/cm^2
A_x1 = 0.8;      % uA/cm^2
V_Na = 50;       % mV
g_Na = 4;        % mS/cm^2
g_NaC = 0.003;  % mS/cm^2
g_s = 0.09;     % mS/cm^2
A_s = 40;       % uA/cm^2
t_on = 50;      % ms
t_dur = 1;      % ms

Y_init = [-83.3, 1.87e-7, 0.1644, 0.01, 0.9814, 0.9673, 0.0033, 0.9884];
options = odeset('MaxStep', 0.1);
[time, Y_out] = ode15s('BR_prime', [0 500], Y_init, options);
plot(time, Y_out(:,1), 'k'), xlabel('ms'), ylabel('mV'), ...
     title('Beeler-Reuter Membrane Potential');

```

BR_prime.m

```

function y_prime = BR_prime(t,y)
global Cm r_Ca Ca_SR k_up A_K1 A_x1 g_Na;
global g_NaC V_Na g_s A_s t_on t_dur;
y_prime = zeros(8,1);

V = y(1);
Ca = y(2);
x1 = y(3);
m = y(4);
h = y(5);
j = y(6);
d = y(7);
f = y(8);
alpha_x1 = 0.0005*exp(0.083*(V+50))/(exp(0.057*(V+50))+1);
beta_x1 = 0.0013*exp(-0.06*(V+20))/(exp(-0.04*(V+20))+1);
alpha_m = -(V+47)/(exp(-0.1*(V+47))-1);
beta_m = 40*exp(-0.056*(V+72));
alpha_h = 0.126*exp(-0.25*(V+77));
beta_h = 1.7/(exp(-0.082*(V+22.5))+1);
alpha_j = 0.055*exp(-0.25*(V+78))/(exp(-0.2*(V+78))+1);
beta_j = 0.3/(exp(-0.1*(V+32))+1);
alpha_d = 0.095*exp(-0.01*(V-5))/(exp(-0.072*(V-5))+1);
beta_d = 0.07*exp(-0.017*(V+44))/(exp(0.05*(V+44))+1);
alpha_f = 0.012*exp(-0.008*(V+28))/(exp(0.15*(V+28))+1);
beta_f = 0.0065*exp(-0.02*(V+30))/(exp(-0.2*(V+30))+1);

V_Ca = -82.3-13.0287*log(Ca);

```

```

i_K1 = A_K1*(4*(exp(0.04*(V+85))-1)/(exp(0.08*(V+53))+exp(0.04*(V+53))) + ...
      0.2*(V+23)/(1-exp(-0.04*(V+23)));
i_x1 = A_x1*x1*(exp(0.04*(V+77))-1)/exp(0.04*(V+35));
i_Na = (g_Na*m^3*h*j + g_NaC)*(V-V_Na);
i_s = g_s*d*f*(V-V_Ca);

if ((t >= t_on)&&(t<t_on+t_dur))
    i_stim = A_s;
else
    i_stim = 0;
end;

y_prime(1) = -(i_K1+i_x1+i_Na+i_s-i_stim)/Cm;
y_prime(2) = -r_Ca*i_s+k_up*(Ca_SR-Ca);
y_prime(3) = alpha_x1*(1-x1)-beta_x1*x1;
y_prime(4) = alpha_m*(1-m)-beta_m*m;
y_prime(5) = alpha_h*(1-h)-beta_h*h;
y_prime(6) = alpha_j*(1-j)-beta_j*j;
y_prime(7) = alpha_d*(1-d)-beta_d*d;
y_prime(8) = alpha_f*(1-f)-beta_f*f;

```

(b) To solve the Beeler–Reuter model using the forward-Euler method, the following code was implemented (`BR_forward_Euler_solve.m` and `BR_forward_Euler.m`). Note that `BR_forward_Euler.m` makes use of the `BR_prime.m` function defined above:

BR_forward_Euler_solve.m

```

% solves the Beeler-Reuter (1977) model with
% the forward-Euler and ode15s algorithms
global Cm r_Ca Ca_SR k_up A_K1 A_x1 g_Na;
global g_NaC V_Na g_s A_s t_on t_dur;
global Y_init;

Cm = 1;           % uC/cm^2
r_Ca = 1e-7;     % M*cm^2/nC
Ca_SR = 1e-7;   % M
k_up = 0.07;    % 1/ms
A_K1 = 0.35;    % uA/cm^2
A_x1 = 0.8;     % uA/cm^2
V_Na = 50;     % mV
g_Na = 4;       % mS/cm^2
g_NaC = 0.003; % mS/cm^2
g_s = 0.09;    % mS/cm^2
A_s = 40;      % uA/cm^2
t_on = 50;     % ms
t_dur = 1;     % ms

Y_init = [-83.3, 1.87e-7, 0.1644, 0.01, 0.9814, 0.9673, 0.0033, 0.9884];

% solve model using forward-Euler with step-size 0.01 ms
[time_fE, Y_out_fE] = BR_forward_Euler(0.01);

% solve using ode15s

```

```

options = odeset('MaxStep', 0.1);
[time_15s, Y_out_15s] = ode15s('BR_prime', [0 500], Y_init, options);

% plot results
plot(time_15s, Y_out_15s(:,1), 'k', time_fE, Y_out_fE(:,1), 'k--'), ...
     xlabel('ms'), ylabel('mV'), ...
     title('Beeler-Reuter Membrane Potential'), ...
     legend('ode15s', 'forward-Euler: 0.01ms');

```

BR_forward_Euler.m

```

function [time, Y_out] = BR_forward_Euler(h)
% solves the Beeler-Reuter (1977) model
% from t = 0 to 500ms using the forward-Euler
% method with fixed step-size h (in ms)

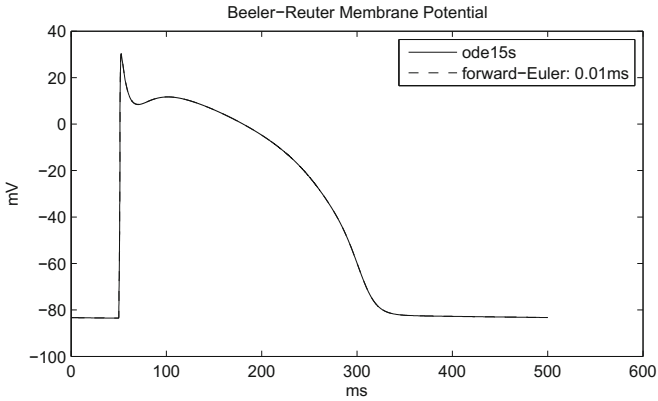
global Y_init;
S = round(500/h)+1;          % output time length
time = zeros(S,1);
Y_out = zeros(S,8);

t = 0;
Y = Y_init;
Y_out(1,:) = Y_init;

% main loop
for ii = 1:S
    Y = Y + h*BR_prime(t,Y)';
    t = t+h;
    time(ii+1) = t;
    Y_out(ii+1,:) = Y;
end;

```

producing the following plot:



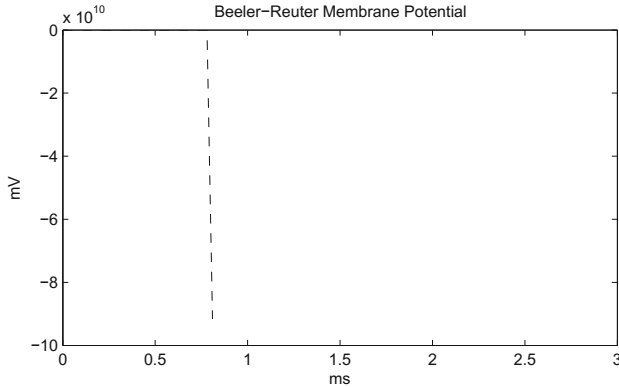
(c) The following code was used to plot the forward-Euler solution for V_m over the first few milliseconds using a step size of 0.03 ms:

```
% solves the Beeler-Reuter (1977) model using
% forward-Euler with step-size 0.03 ms
global Cm r_Ca Ca_SR k_up A_K1 A_x1 g_Na;
global g_NaC V_Na g_s A_s t_on t_dur;
global Y_init;
Cm = 1; % uC/cm^2
r_Ca = 1e-7; % M*cm^2/nC
Ca_SR = 1e-7; % M
k_up = 0.07; % 1/ms
A_K1 = 0.35; % uA/cm^2
A_x1 = 0.8; % uA/cm^2
V_Na = 50; % mV
g_Na = 4; % mS/cm^2
g_NaC = 0.003; % mS/cm^2
g_s = 0.09; % mS/cm^2
A_s = 40; % uA/cm^2
t_on = 50; % ms
t_dur = 1; % ms
Y_init = [-83.3, 1.87e-7, 0.1644, 0.01, 0.9814, 0.9673, 0.0033, 0.9884];

[time_fE, Y_out_fE] = BR_forward_Euler(0.03);

% plot first 100 points only
plot(time_fE(1:100), Y_out_fE(1:100,1), 'k--'), ...
    xlabel('ms'), ylabel('mV'), ...
    title('Beeler-Reuter Membrane Potential');
```

producing the result:



where it is apparent the method has become unstable.

(d) To solve the Beeler–Reuter model using the backward-Euler method, we rewrite Eq. 3.11 in the form:

$$\mathbf{y}_n - \mathbf{y}_{n-1} - h\mathbf{f}(t_n, \mathbf{y}_n) = \mathbf{0}$$

and use Newton’s method (Eq. 3.12) to solve for \mathbf{y}_n at each step. This technique has been utilized in the below Matlab code (BR_backward_Euler_solve.m and BR_backward_Euler.m):

BR_backward_Euler_solve.m

```
% solves the Beeler-Reuter (1977) model with
% the backward-Euler and ode15s algorithms
global Cm r_Ca Ca_SR k_up A_K1 A_x1 g_Na;
global g_NaC V_Na g_s A_s t_on t_dur;
global Y_init;

Cm = 1;           % uC/cm^2
r_Ca = 1e-7;     % M*cm^2/nC
Ca_SR = 1e-7;    % M
k_up = 0.07;     % 1/ms
A_K1 = 0.35;     % uA/cm^2
A_x1 = 0.8;      % uA/cm^2
V_Na = 50;       % mV
g_Na = 4;        % mS/cm^2
g_NaC = 0.003;  % mS/cm^2
g_s = 0.09;     % mS/cm^2
A_s = 40;        % uA/cm^2
t_on = 50;       % ms
t_dur = 1;       % ms
```

```

Y_init = [-83.3, 1.87e-7, 0.1644, 0.01, 0.9814, 0.9673, ...
          0.0033, 0.9884];

% solve model using backward-Euler with step-sizes of
% 0.01 ms and 0.1 ms
h1 = 0.01;
[time_bE_h1, Y_out_bE_h1] = BR_backward_Euler(h1);
h2 = 0.1;
[time_bE_h2, Y_out_bE_h2] = BR_backward_Euler(h2);

% solve using ode15s
options = odeset('MaxStep', 0.1);
[time_15s, Y_out_15s] = ode15s('BR_prime', [0 500], ...
    Y_init, options);

% plot results
plot(time_15s, Y_out_15s(:,1), 'k', ...
     time_bE_h1, Y_out_bE_h1(:,1), 'k--', ...
     time_bE_h2, Y_out_bE_h2(:,1), 'k:'), ...
     xlabel('ms'), ylabel('mV'), ...
     title('Beeler-Reuter Membrane Potential'), ...
     legend('ode15s', 'backward-Euler: 0.01ms', ...
           'backward-Euler: 0.1ms'));

```

BR_backward_Euler.m

```

function [time, Y_out] = BR_backward_Euler(t_step)
% solves the Beeler-Reuter (1977) model
% from t = 0 to 500ms using the backward-Euler
% method with fixed step-size t_step (in ms)

global Y_init Y_prev h_step;

S = round(500/t_step)+1;      % output time length
time = zeros(S,1);
Y_out = zeros(S,8);
t = 0;
Y_prev = Y_init;
h_step = t_step;
Y_out(1,:) = Y_init;

% main loop
for ii = 1:S
    t = t+h_step;
    % use Newton's method to determine Y at the next step
    Y_iter = Y_prev;
    bE_res_0 = Y_iter' - Y_prev' - h_step*BR_prime(t,Y_iter);
    while max(abs(bE_res_0)) > 1e-6
        % determine Jacobian

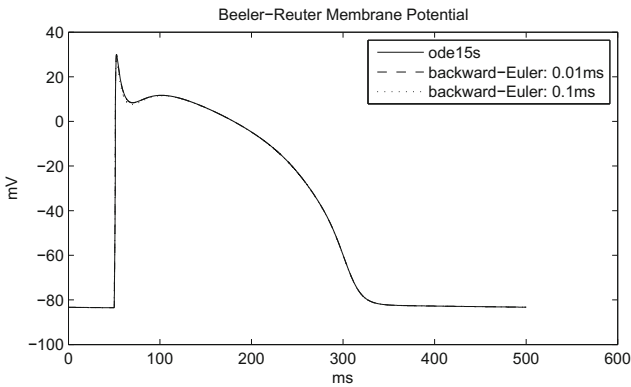
```

```

Jac = zeros(8,8);
for jj = 1:8
    Y_test = Y_iter;
    step_j = 1e-6*max(abs(Y_iter(jj)),1e-3);
    Y_test(jj) = Y_iter(jj) + step_j;
    bE_res = Y_test' - Y_prev' - h_step*BR_prime(t,Y_test);
    Jac(:,jj) = (bE_res-bE_res_0)/step_j;
end;
Y_iter = Y_iter - (Jac\bE_res_0)';
bE_res_0 = Y_iter' - Y_prev' - h_step*BR_prime(t,Y_iter);
end;
time(ii+1) = t;
Y_out(ii+1,:) = Y_iter;
Y_prev = Y_iter;
end;

```

which produces the following plot:



Note that unlike the forward-Euler method which was unstable for the step-size of 0.03 ms, the backward-Euler algorithm is stable at the even larger step-size of 0.1 ms. However, too large a step may lead to convergence issues with Newton’s method.

(e) The following Matlab code was used to solve the Beeler–Reuter model for a range of Matlab in-built ODE solvers, determining the computational time taken for each solver (using Matlab’s tic and toc timing commands):

```

% solves the Beeler-Reuter (1977) model with
% several in-built Matlab ODE algorithms
global Cm r_Ca Ca_SR k_up A_K1 A_x1 g_Na;
global g_NaC V_Na g_s A_s t_on t_dur;

Cm = 1;           % uC/cm^2
r_Ca = 1e-7;     % M*cm^2/nC
Ca_SR = 1e-7;   % M

```

```

k_up = 0.07;      % 1/ms
A_K1 = 0.35;     % uA/cm^2
A_x1 = 0.8;      % uA/cm^2
V_Na = 50;       % mV
g_Na = 4;        % mS/cm^2
g_NaC = 0.003;  % mS/cm^2
g_s = 0.09;     % mS/cm^2
A_s = 40;       % uA/cm^2
t_on = 50;      % ms
t_dur = 1;      % ms

Y_init = [-83.3, 1.87e-7, 0.1644, 0.01, 0.9814, 0.9673, ...
          0.0033, 0.9884];

% solve using ode15s
options = odeset('MaxStep', 0.1);
tic
[time_15s, Y_out_15s] = ode15s('BR_prime', [0 500], ...
    Y_init, options);
ode15s_time_taken = toc;

% solve using ode23s
tic
[time_23s, Y_out_23s] = ode23s('BR_prime', [0 500], ...
    Y_init, options);
ode23s_time_taken = toc;

% solve using ode23t
tic
[time_23t, Y_out_23t] = ode23t('BR_prime', [0 500], ...
    Y_init, options);
ode23t_time_taken = toc;

% solve using ode23tb
tic
[time_23tb, Y_out_23tb] = ode23tb('BR_prime', [0 500], ...
    Y_init, options);
ode23tb_time_taken = toc;

% solve using ode45
tic
[time_45, Y_out_45] = ode45('BR_prime', [0 500], ...
    Y_init, options);
ode45_time_taken = toc;

```



```

% solve using ode23
tic
[time_23, Y_out_23] = ode23('BR_prime', [0 500], ...
    Y_init, options);
ode23_time_taken = toc;

% solve using ode113
tic
[time_113, Y_out_113] = ode113('BR_prime', [0 500], ...
    Y_init, options);
ode113_time_taken = toc;

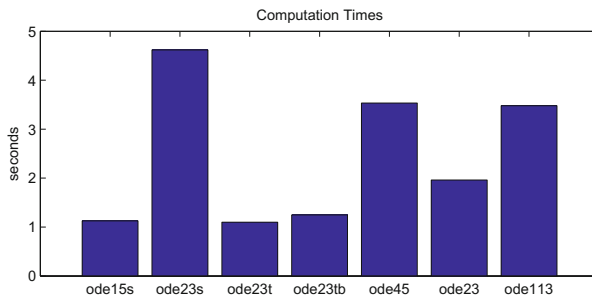
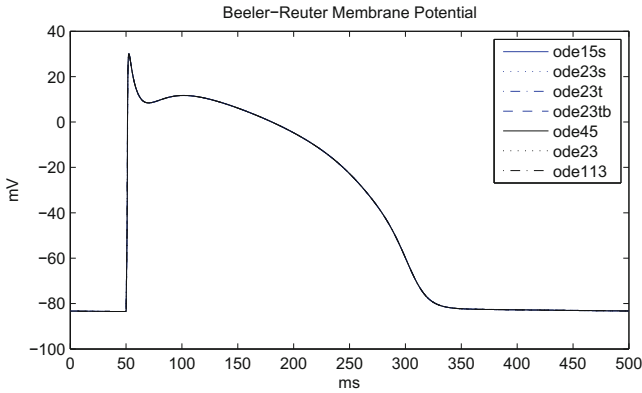
time_taken = [ode15s_time_taken, ode23s_time_taken, ...
    ode23t_time_taken, ode23tb_time_taken, ...
    ode45_time_taken, ode23_time_taken, ...
    ode113_time_taken];

% plot results
figure(1), plot(time_15s, Y_out_15s(:,1), 'b', ...
    time_23s, Y_out_23s(:,1), 'b:', ...
    time_23t, Y_out_23t(:,1), 'b-.', ...
    time_23tb, Y_out_23tb(:,1), 'b--', ...
    time_45, Y_out_45(:,1), 'k', ...
    time_23, Y_out_23(:,1), 'k:', ...
    time_113, Y_out_113(:,1), 'k-.'), ...
    xlabel('ms'), ylabel('mV'), ...
    title('Beeler-Reuter Membrane Potential'), ...
    legend('ode15s', 'ode23s', 'ode23t', 'ode23tb', ...
    'ode45', 'ode23', 'ode113');

figure(2), bar(time_taken), title('Computation Times'), ...
    set(gca, 'XTickLabel', {'ode15s', 'ode23s', ...
    'ode23t', 'ode23tb', 'ode45', 'ode23', 'ode113'}), ...
    ylabel('seconds');

```

producing the following plots:



From these plots, we can see that all of Matlab’s ODE solvers were able to accurately compute the Beeler–Reuter equations, however `ode15s`, `ode23t`, and `ode23tb` exhibited the fastest computational times.⁵ Of these, `ode23t` was slightly more efficient than `ode15s`. In most cases however, `ode15s` remains the general-purpose ODE solver of choice for most biological systems, although some other solvers may be slightly more computationally efficient for specific systems.

3.2 (a) The $I_{Na,p} + I_K$ model can be represented by the ODE system

$$\frac{dy}{dt} = \mathbf{f}(t, \mathbf{y})$$

To solve such a system, the third-order Runge–Kutta algorithm with coefficients

$$\begin{array}{c|c} 0 & \\ \hline \frac{1}{2} & \frac{1}{2} \\ \frac{3}{4} & 0 \quad \frac{3}{4} \\ \hline \frac{2}{9} & \frac{1}{3} \quad \frac{4}{9} \end{array}$$

is equivalent to the algorithm

⁵These particular computation times were obtained on a modest 1.3 GHz Intel i5 MacBook Air with 8 GB RAM running OS X 10.8.5.

$$\begin{aligned}\mathbf{K}_1 &= \mathbf{f}(t_{n-1}, \mathbf{y}_{n-1}) \\ \mathbf{K}_2 &= \mathbf{f}\left(t_{n-1} + \frac{h}{2}, \mathbf{y}_{n-1} + \frac{h}{2}\mathbf{K}_1\right) \\ \mathbf{K}_3 &= \mathbf{f}\left(t_{n-1} + \frac{3h}{4}, \mathbf{y}_{n-1} + \frac{3h}{4}\mathbf{K}_2\right) \\ \mathbf{y}_n &= \mathbf{y}_{n-1} + \frac{h}{9}(2\mathbf{K}_1 + 3\mathbf{K}_2 + 4\mathbf{K}_3)\end{aligned}$$

which is implemented below in the code INapK_solve.m, INapK_Runge_Kutta.m and INapK_prime.m:

INapK_solve.m

```
% solves the I_Na,p + I_K model
% using a third-order Runge-Kutta algorithm
% and ode15s

global C g_Na E_Na g_K E_K tau_n g_L E_L I;
global Y_init;

C = 1;           % uF/cm^2
g_Na = 20;       % mS/cm^2
E_Na = 60;       % mV
g_K = 10;        % mS/cm^2
E_K = -90;       % mV
tau_n = 1;       % ms
g_L = 8;         % mS/cm^2
E_L = -80;       % mV
I = 40;          % uA/cm^2

Y_init = [-72.9 0.36];
h1 = 0.01;
h2 = 0.1;

% solve model with all steps and methods
[time_h1, Y_out_h1] = INapK_Runge_Kutta(h1);
[time_h2, Y_out_h2] = INapK_Runge_Kutta(h2);
[time_15s, Y_out_15s] = ode15s('INapK_prime', [0 30], Y_init);

% plot solutions
plot(time_15s, Y_out_15s(:,1), 'k', ...
      time_h1, Y_out_h1(:,1), 'k--', ...
      time_h2, Y_out_h2(:,1), 'k:', ...
      xlabel('ms'), ylabel('mV'), ...
```

```

title('I_{Na,p}+I_K Model'), ...
legend('ode15s', 'RK: 0.01 ms', 'RK: 0.1 ms');

```

INapK_Runge_Kutta.m

```

function [time, Y_out] = INapK_Runge_Kutta(h)
% solves the I_Na,p + I_K model
% from t = 0 to 30 ms using a
% third-order Runge-Kutta algorithm
% with fixed step-size h (in ms)

global Y_init;

S = round(30/h)+1;      % output time length
time = zeros(S,1);
Y_out = zeros(S,2);

t = 0;
Y = Y_init;
Y_out(1,:) = Y_init;

% main loop
for ii = 1:S
    K1 = INapK_prime(t,Y);
    K2 = INapK_prime(t+h/2,Y+K1'*h/2);
    K3 = INapK_prime(t+3*h/4,Y+K2'*3*h/4);

    Y = Y + (h/9)*(2*K1'+3*K2'+4*K3');
    t = t+h;
    time(ii+1) = t;
    Y_out(ii+1,:) = Y;
end;

```

iNapK_prime.m

```

function y_prime = INapK_prime(t,y)
% solves the I_Na,p + I_K model
global C g_Na E_Na g_K E_K tau_n g_L E_L I;
y_prime = zeros(2,1);

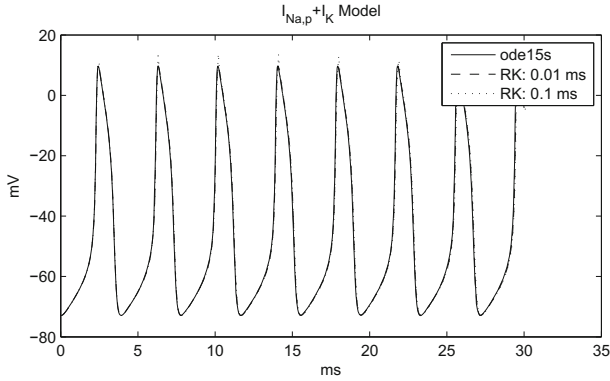
V = y(1);
n = y(2);

```

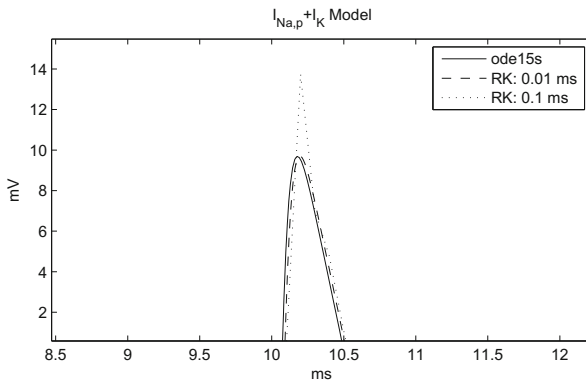
```

m_inf = 1/(1+exp(-(V+20)/15));
n_inf = 1/(1+exp(-(V+25)/5));
y_prime(1) = -(1/C)*(g_Na*m_inf*(V-E_Na) + ...
    g_K*n*(V-E_K) + g_L*(V-E_L) - I);
y_prime(2) = (n_inf-n)/tau_n;
    
```

Executing `INapK_solve` from the Matlab command line produces the following plot of solutions:



The differences between these three methods can be highlighted more clearly when zooming in on the plot near $t = 10$ ms:



(b) To solve the above model using the generalized- α method, we first re-write the implicit algorithm of Eq. 3.26 for use with Newton’s method: i.e. in the form $\mathbf{g}(\mathbf{x}) = \mathbf{0}$. For the general first-order ODE system

$$\frac{dy}{dt} = \mathbf{f}(t, \mathbf{y})$$

the generalized- α method can be expressed as:

$$\begin{aligned} \dot{\mathbf{y}}_{n-1} + \alpha_m (\dot{\mathbf{y}}_n - \dot{\mathbf{y}}_{n-1}) - \mathbf{f}(t_{n-1} + h\alpha_f, \mathbf{y}_{n-1} + \alpha_f (\mathbf{y}_n - \mathbf{y}_{n-1})) &= \mathbf{0} \\ \mathbf{y}_n - \mathbf{y}_{n-1} - h\dot{\mathbf{y}}_{n-1} - h\gamma (\dot{\mathbf{y}}_n - \dot{\mathbf{y}}_{n-1}) &= \mathbf{0} \end{aligned}$$

which is in the required form $\mathbf{g}(\mathbf{x}) = \mathbf{0}$ with $\mathbf{x} = [\dot{\mathbf{y}}_n^T, \mathbf{y}_n^T]^T$. The method parameters are

$$\alpha_f = \frac{1}{1 + \rho_\infty}, \quad \alpha_m = \frac{1}{2} \left(\frac{3 - \rho_\infty}{1 + \rho_\infty} \right), \quad \gamma = \frac{1}{1 + \rho_\infty}$$

where ρ_∞ specifies the required level of high-frequency damping from 0 to 1, with $\rho_\infty = 0$ denoting maximal damping and $\rho_\infty = 1$ no damping. The algorithm is implemented below in the code `INapK_generalized_alpha_solve.m` and `INapK_generalized_alpha.m`:

INapK_generalized_alpha_solve.m

```
% solves the I_Na,p + I_K model
% using the generalized-alpha and
% ode15s methods.
global C g_Na E_Na g_K E_K tau_n g_L E_L I;
global Y_init;

C = 1;           % uF/cm^2
g_Na = 20;       % mS/cm^2
E_Na = 60;       % mV
g_K = 10;        % mS/cm^2
E_K = -90;       % mV
tau_n = 1;       % ms
g_L = 8;         % mS/cm^2
E_L = -80;       % mV
I = 40;          % uA/cm^2

Y_init = [-72.9 0.36];

% solve model with ode15s and generalized_alpha method
% using three high-frequency damping factors
[time_15s, Y_out_15s] = ode15s('INapK_prime', [0 30], Y_init);
[time_1, Y_out_1] = INapK_generalized_alpha(0.1,1);
[time_2, Y_out_2] = INapK_generalized_alpha(0.1,0.5);
[time_3, Y_out_3] = INapK_generalized_alpha(0.1,0);

% plot solutions
plot(time_15s, Y_out_15s(:,1), 'k', ...
```

```

time_1, Y_out_1(:,1), 'k--', ...
time_2, Y_out_2(:,1), 'k-.', ...
time_3, Y_out_3(:,1), 'k:'), ...
xlabel('ms'), ylabel('mV'), ...
title('I_{Na,p}+I_K Model'), ...
legend('ode15s', 'gen. alpha: \rho_{\infty} = 1',...
'gen. alpha: \rho_{\infty} = 0.5',...
'gen. alpha: \rho_{\infty} = 0');

```

INapK_generalized_alpha.m

```

function [time, Y_out] = INapK_generalized_alpha(h,rho_inf)
% solves the I_Na,p + I_K model
% from t = 0 to 30 ms using the
% generalized-alpha method
% with fixed step-size h (in ms)
% and high-frequency damping factor rho_inf between 0 and 1

global Y_init;

S = round(30/h)+1;      % output time length
time = zeros(S,1);
Y_out = zeros(S,2);

t = 0;
Y_prev = Y_init;
Y_out(1,:) = Y_init;
Y_prime_prev = INapK_prime(0,Y_init)';

a_f = 1/(1+rho_inf);    % alpha_f
a_m = 0.5*(3-rho_inf)/(1+rho_inf); % alpha_m
gamma = 1/(1+rho_inf);  % gamma

% main loop
for ii = 1:S
    % Use Newton's method to determine [Y_prime, Y] at the next step
    % First, begin with a constant predictor
    Y_iter = [Y_prime_prev, Y_prev];
    gen_alpha_res_Yprime = (1-a_m)*Y_prime_prev+a_m*Y_iter(1:2)...
        - INapK_prime(t+a_f*h, (1-a_f)*Y_prev+a_f*Y_iter(3:4))';
    gen_alpha_res_Y = Y_iter(3:4)-Y_prev...
        - h*(Y_prime_prev + gamma*(Y_iter(1:2)-Y_prime_prev));
    gen_alpha_res_0 = [gen_alpha_res_Yprime,gen_alpha_res_Y];
    while max(abs(gen_alpha_res_0)) > 1e-6
        % determine Jacobian
        Jac = zeros(4,4);
        for jj = 1:4
            Y_test = Y_iter;
            step_j = 1e-6*max(abs(Y_iter(jj)),1e-3);
            Y_test(jj) = Y_iter(jj) + step_j;
            gen_alpha_res_Yprime = (1-a_m)*Y_prime_prev+a_m*Y_test(1:2)...
                - INapK_prime(t+a_f*h, (1-a_f)*Y_prev+a_f*Y_test(3:4))';
            gen_alpha_res_Y = Y_test(3:4)-Y_prev...
                - h*(Y_prime_prev + gamma*(Y_test(1:2)-Y_prime_prev));
            gen_alpha_res = [gen_alpha_res_Yprime,gen_alpha_res_Y];

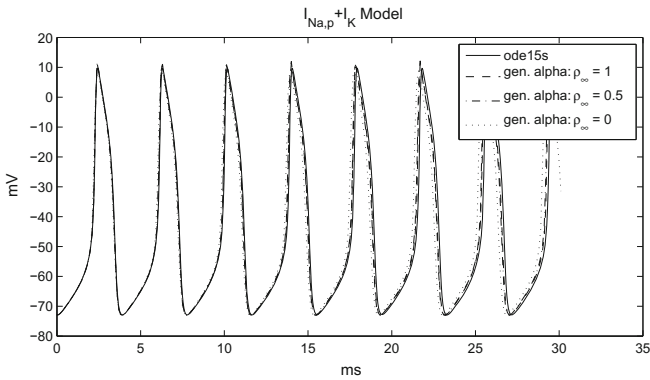
```

```

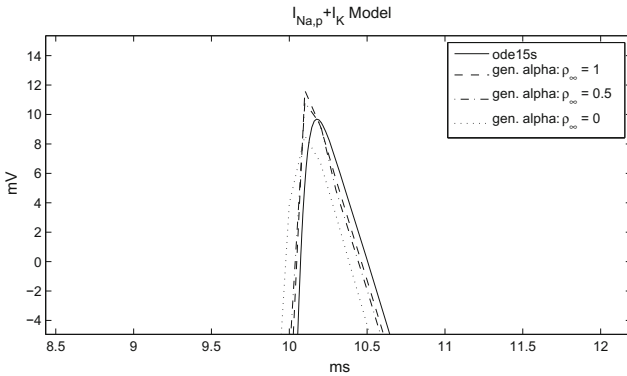
        Jac(:,jj) = (gen_alpha_res-gen_alpha_res_0)/step_j;
    end;
    Y_iter = Y_iter - (Jac\gen_alpha_res_0)';
    gen_alpha_res_Yprime = (1-a_m)*Y_prime_prev+a_m*Y_iter(1:2)...
        - INapK_prime(t+a_f*h, (1-a_f)*Y_prev+a_f*Y_iter(3:4))';
    gen_alpha_res_Y = Y_iter(3:4)-Y_prev...
        - h*(Y_prime_prev + gamma*(Y_iter(1:2)-Y_prime_prev));
    gen_alpha_res_0 = [gen_alpha_res_Yprime,gen_alpha_res_Y];
end;
t = t+h;
time(ii+1) = t;
Y_out(ii+1,:) = Y_iter(3:4);
Y_prime_prev = Y_iter(1:2);
Y_prev = Y_iter(3:4);
end;

```

Executing `INapK_generalized_alpha_solve` from the Matlab command line produces the following plot:



The differences between the three levels of damping can be seen more clearly when zooming in on the plot near $t = 10$ ms:



where it can be seen that membrane potential peak is progressively reduced with increased levels of damping.

3.3 Using Newton's backward difference formula (Eq. 3.42), the interpolating polynomial passing through the $k + 1$ step solutions $y_{n-1}, y_{n-1}, \dots, y_{n-k-1}$ is:

$$\begin{aligned} \phi(t) = & \nabla^0 y_{n-1} + \frac{(t - t_{n-1})}{h} \nabla^1 y_{n-1} + \frac{(t - t_{n-1})(t - t_{n-2})}{2! h^2} \nabla^2 y_{n-1} + \dots \\ & + \frac{(t - t_{n-1})(t - t_{n-2}) \dots (t - t_{n-k})}{k! h^k} \nabla^k y_{n-1} \end{aligned}$$

Substituting $t = t_n$, we obtain

$$\begin{aligned} \phi(t_n) = & \nabla^0 y_{n-1} + \frac{(t_n - t_{n-1})}{h} \nabla^1 y_{n-1} + \frac{(t_n - t_{n-1})(t_n - t_{n-2})}{2! h^2} \nabla^2 y_{n-1} + \dots \\ & + \frac{(t_n - t_{n-1})(t_n - t_{n-2}) \dots (t_n - t_{n-k})}{k! h^k} \nabla^k y_{n-1} \\ = & \nabla^0 y_{n-1} + \frac{h}{h} \nabla^1 y_{n-1} + \frac{(h)(2h)}{2! h^2} \nabla^2 y_{n-1} + \dots + \frac{(h)(2h) \dots (kh)}{k! h^k} \nabla^k y_{n-1} \\ = & \nabla^0 y_{n-1} + \nabla^1 y_{n-1} + \nabla^2 y_{n-1} + \dots + \frac{k! h^k}{k! h^k} \nabla^k y_{n-1} \\ = & \sum_{i=0}^k \nabla^i y_{n-1} \\ = & y_n^p \quad \text{as required.} \end{aligned}$$

3.4 The k -step BDF formula is given by Eq. 3.58:

$$\sum_{i=1}^k \frac{1}{i} \nabla^i y_n = hf(t_n, y_n)$$

For $k = 7$, we form the backward difference expressions:

$$\begin{aligned} \nabla^1 y_n &= y_n - y_{n-1} \\ \nabla^2 y_n &= y_n - 2y_{n-1} + y_{n-2} \\ \nabla^3 y_n &= y_n - 3y_{n-1} + 3y_{n-2} - y_{n-3} \\ \nabla^4 y_n &= y_n - 4y_{n-1} + 6y_{n-2} - 4y_{n-3} + y_{n-4} \\ \nabla^5 y_n &= y_n - 5y_{n-1} + 10y_{n-2} - 10y_{n-3} + 5y_{n-4} - y_{n-5} \\ \nabla^6 y_n &= y_n - 6y_{n-1} + 15y_{n-2} - 20y_{n-3} + 15y_{n-4} - 6y_{n-5} + y_{n-6} \\ \nabla^7 y_n &= y_n - 7y_{n-1} + 21y_{n-2} - 35y_{n-3} + 35y_{n-4} - 21y_{n-5} + 7y_{n-6} - y_{n-7} \end{aligned}$$

Substituting these into the above k -step formula, and after re-arranging, we obtain the 7-step BDF formula:

$$y_n - \frac{980}{363}y_{n-1} + \frac{490}{121}y_{n-2} - \frac{4900}{1089}y_{n-3} + \frac{1225}{363}y_{n-4} \\ - \frac{196}{121}y_{n-5} + \frac{490}{1089}y_{n-6} - \frac{20}{363}y_{n-7} = \frac{140}{363}hf(t_n, y_n)$$

For the test ODE

$$\frac{dy}{dt} = \lambda y$$

with $\lambda < 0$, we take the initial value at $t = 0$ to be y_0 . The exact solution is then given by $y_0 e^{\lambda t}$. For a fixed step-size of h , we can write the exact solutions to the first seven steps up to $t = 6h$ as

$$\begin{aligned} t = 0 &= t_{n-7}, & y_{n-7} &= y_0 \\ t = h &= t_{n-6}, & y_{n-6} &= y_0 e^{\lambda h} \\ t = 2h &= t_{n-5}, & y_{n-5} &= y_0 e^{2\lambda h} \\ t = 3h &= t_{n-4}, & y_{n-4} &= y_0 e^{3\lambda h} \\ t = 4h &= t_{n-3}, & y_{n-3} &= y_0 e^{4\lambda h} \\ t = 5h &= t_{n-2}, & y_{n-2} &= y_0 e^{5\lambda h} \\ t = 6h &= t_{n-1}, & y_{n-1} &= y_0 e^{6\lambda h} \end{aligned}$$

Substituting these values into the above 7-step BDF formula, and noting that $f(t_n, y_n) = \lambda y_n$, we obtain:

$$\begin{aligned} y_n - \frac{980}{363}y_0 e^{6\lambda h} + \frac{490}{121}y_0 e^{5\lambda h} - \frac{4900}{1089}y_0 e^{4\lambda h} + \frac{1225}{363}y_0 e^{3\lambda h} \\ - \frac{196}{121}y_0 e^{2\lambda h} + \frac{490}{1089}y_0 e^{\lambda h} - \frac{20}{363}y_0 = \frac{140}{363}h\lambda y_n \\ \therefore y_n \left(\frac{140}{363}h\lambda - 1 \right) = y_0 e^{6\lambda h} \left(-\frac{980}{363} + \frac{490}{121}e^{-\lambda h} - \frac{4900}{1089}e^{-2\lambda h} + \frac{1225}{363}e^{-3\lambda h} \right. \\ \left. - \frac{196}{121}e^{-4\lambda h} + \frac{490}{1089}e^{-5\lambda h} - \frac{20}{363}e^{-6\lambda h} \right) \\ y_n (420h\lambda - 1089) = y_{n-1} (-2940 + 4410e^{-\lambda h} - 4900e^{-2\lambda h} + 3675e^{-3\lambda h} \\ - 1764e^{-4\lambda h} + 490e^{-5\lambda h} - 60e^{-6\lambda h}) \end{aligned}$$

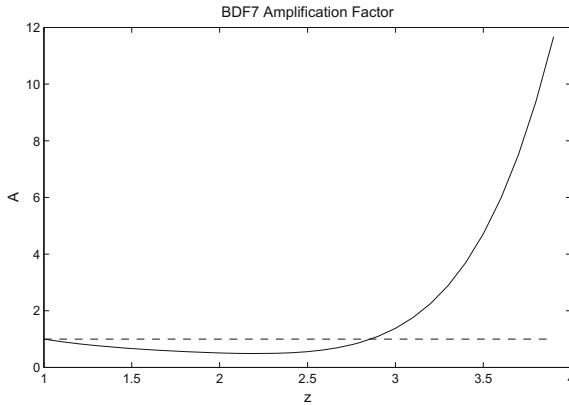
Substituting $z = e^{-\lambda h}$ and re-arranging, we obtain:

$$\begin{aligned} y_n &= y_{n-1} \left(\frac{-2940 + 4410z - 4900z^2 + 3675z^3 - 1764z^4 + 490z^5 - 60z^6}{-420 \ln z - 1089} \right) \\ &= y_{n-1} \left(\frac{60z^6 - 490z^5 + 1764z^4 - 3675z^3 + 4900z^2 - 4410z + 2940}{420 \ln z + 1089} \right) \\ &= y_{n-1} A \end{aligned}$$

where A is a multiplicative factor that depends on z . For $\lambda < 0$ and h positive, z will be > 1 . In the limit as the step-size h becomes infinitely large,

$$\lim_{z \rightarrow \infty} A = \frac{60z^6}{420 \ln z} = \frac{360z^5}{420/z} = \frac{6}{7}z^6 = \infty$$

where we have used L'Hôpital's rule⁶ to evaluate the limit. Hence, the method is unstable for large step-sizes. The plot of A against z is shown below:



where the dashed line represents $A = 1$. Clearly the method is unstable when $A > 1$, corresponding to a value of z near 2.8 on the plot. To numerically determine this value of z , we can use the Matlab `fzero` command, which solves the one-variable equation $g(x) = 0$, to solve the equation $A = 1$. Namely,

$$\frac{60z^6 - 490z^5 + 1764z^4 - 3675z^3 + 4900z^2 - 4410z + 2940}{420 \ln z + 1089} = 1$$

which can be re-arranged to

$$60z^6 - 490z^5 + 1764z^4 - 3675z^3 + 4900z^2 - 4410z + 2940 - 420 \ln z - 1089 = 0$$

This equation can be solved for z using the Matlab command:

```
z = fzero('BDF_zero', 2.8);
```

where 2.8 is the initial estimate for z , and the function `BDF_zero` is defined as

```
function f_out = BDF_zero(z)
```

⁶Pronounced lopi-tahl, the rule is $\lim_{x \rightarrow \infty} \frac{f(x)}{g(x)} = \frac{f'(x)}{g'(x)}$ provided (i) the limit exists, (ii) $g'(x) \neq 0$, and (iii) the corresponding limits of $f(x)$ and $g(x)$ are identical and both equal to 0 or $\pm\infty$.

$$\begin{aligned} f_{\text{out}} &= 60*z^6 - 490*z^5 + 1764*z^4 \dots \\ &- 3675*z^3 + 4900*z^2 - 4410*z + 2940 \dots \\ &- 420*\log(z) - 1089; \end{aligned}$$

Executing the above `fzero` command yields a value of $z \approx 2.8596$. For stability, we therefore require

$$\begin{aligned} z &= e^{-\lambda h} < 2.8596 \\ -\lambda h &< \ln(2.8596) \approx 1.0507 \end{aligned}$$

Hence for absolute stability, we require $h < \frac{1.0507}{|\lambda|}$.

Problems of Chap. 4

$$4.1 \quad (a) \nabla f = \begin{pmatrix} 2 \\ -1 \\ 3 \end{pmatrix} \quad (b) \nabla g = \begin{pmatrix} 2x \\ 2y \\ 2z \end{pmatrix} \quad (c) \nabla h = \begin{pmatrix} 1/y \\ -x/y^2 \\ 0 \end{pmatrix}$$

$$4.2 \quad (a) \nabla \cdot \mathbf{u} = 3, \nabla \times \mathbf{u} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad (b) \nabla \cdot \mathbf{v} = 0, \nabla \times \mathbf{v} = \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix}$$

$$(c) \nabla \cdot \mathbf{w} = 4, \nabla \times \mathbf{w} = \begin{pmatrix} -3 \\ -2 \\ 2 \end{pmatrix}$$

4.3 Writing $\mathbf{u} = (u_x, u_y, u_z)^T$ and $\mathbf{v} = (v_x, v_y, v_z)^T$, we have:

(a)

$$\begin{aligned} \nabla \cdot (\nabla \times \mathbf{u}) &= \nabla \cdot \begin{pmatrix} \frac{\partial u_y}{\partial z} - \frac{\partial u_z}{\partial y} \\ \frac{\partial u_z}{\partial x} - \frac{\partial u_x}{\partial z} \\ \frac{\partial u_x}{\partial y} - \frac{\partial u_y}{\partial x} \end{pmatrix} \\ &= \frac{\partial}{\partial x} \left(\frac{\partial u_y}{\partial z} - \frac{\partial u_z}{\partial y} \right) + \frac{\partial}{\partial y} \left(\frac{\partial u_z}{\partial x} - \frac{\partial u_x}{\partial z} \right) + \frac{\partial}{\partial z} \left(\frac{\partial u_x}{\partial y} - \frac{\partial u_y}{\partial x} \right) \\ &= \frac{\partial^2 u_y}{\partial x \partial z} - \frac{\partial^2 u_z}{\partial x \partial y} + \frac{\partial^2 u_z}{\partial x \partial y} - \frac{\partial^2 u_x}{\partial y \partial z} + \frac{\partial^2 u_x}{\partial y \partial z} - \frac{\partial^2 u_y}{\partial x \partial z} \\ &= 0 \end{aligned}$$

(b)

$$\begin{aligned} \nabla \cdot (\mathbf{u} \times \mathbf{v}) &= \nabla \cdot \begin{pmatrix} u_y v_z - u_z v_y \\ u_z v_x - u_x v_z \\ u_x v_y - u_y v_x \end{pmatrix} \\ &= \frac{\partial}{\partial x} (u_y v_z - u_z v_y) + \frac{\partial}{\partial y} (u_z v_x - u_x v_z) + \frac{\partial}{\partial z} (u_x v_y - u_y v_x) \end{aligned}$$

$$\begin{aligned}
&= v_z \frac{\partial u_y}{\partial x} + u_y \frac{\partial v_z}{\partial x} - v_y \frac{\partial u_z}{\partial x} - u_z \frac{\partial v_y}{\partial x} + v_x \frac{\partial u_z}{\partial y} + u_z \frac{\partial v_x}{\partial y} - v_z \frac{\partial u_x}{\partial y} - u_x \frac{\partial v_z}{\partial y} \\
&\quad + v_y \frac{\partial u_x}{\partial z} + u_x \frac{\partial v_y}{\partial z} - v_x \frac{\partial u_y}{\partial z} - u_y \frac{\partial v_x}{\partial z} \\
&= v_x \left(\frac{\partial u_z}{\partial y} - \frac{\partial u_y}{\partial z} \right) + v_y \left(-\frac{\partial u_z}{\partial x} + \frac{\partial u_x}{\partial z} \right) + v_z \left(\frac{\partial u_y}{\partial x} - \frac{\partial u_x}{\partial y} \right) \\
&\quad + u_x \left(-\frac{\partial v_z}{\partial y} + \frac{\partial v_y}{\partial z} \right) + u_y \left(\frac{\partial v_z}{\partial x} - \frac{\partial v_x}{\partial z} \right) + u_z \left(-\frac{\partial v_y}{\partial x} + \frac{\partial v_x}{\partial y} \right) \\
&= v_x \left(\frac{\partial u_z}{\partial y} - \frac{\partial u_y}{\partial z} \right) + v_y \left(\frac{\partial u_x}{\partial z} - \frac{\partial u_z}{\partial x} \right) + v_z \left(\frac{\partial u_y}{\partial x} - \frac{\partial u_x}{\partial y} \right) \\
&\quad + u_x \left(\frac{\partial v_y}{\partial z} - \frac{\partial v_z}{\partial y} \right) + u_y \left(\frac{\partial v_z}{\partial x} - \frac{\partial v_x}{\partial z} \right) + u_z \left(\frac{\partial v_x}{\partial y} - \frac{\partial v_y}{\partial x} \right) \\
&= \mathbf{v} \cdot (\nabla \times \mathbf{u}) + \mathbf{u} \cdot (-\nabla \times \mathbf{v}) \\
&= \mathbf{v} \cdot (\nabla \times \mathbf{u}) - \mathbf{u} \cdot (\nabla \times \mathbf{v})
\end{aligned}$$

(c) Starting from the right-hand side, we have:

$$\begin{aligned}
\nabla(\nabla \cdot \mathbf{u}) &= \nabla \left(\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} + \frac{\partial u_z}{\partial z} \right) \\
&= \begin{pmatrix} \frac{\partial}{\partial x} \left(\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} + \frac{\partial u_z}{\partial z} \right) \\ \frac{\partial}{\partial y} \left(\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} + \frac{\partial u_z}{\partial z} \right) \\ \frac{\partial}{\partial z} \left(\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} + \frac{\partial u_z}{\partial z} \right) \end{pmatrix} \\
&= \begin{pmatrix} \frac{\partial^2 u_x}{\partial x^2} + \frac{\partial^2 u_y}{\partial x \partial y} + \frac{\partial^2 u_z}{\partial x \partial z} \\ \frac{\partial^2 u_x}{\partial x \partial y} + \frac{\partial^2 u_y}{\partial y^2} + \frac{\partial^2 u_z}{\partial y \partial z} \\ \frac{\partial^2 u_x}{\partial x \partial z} + \frac{\partial^2 u_y}{\partial y \partial z} + \frac{\partial^2 u_z}{\partial z^2} \end{pmatrix} \\
\nabla^2 \mathbf{u} &= \frac{\partial^2 \mathbf{u}}{\partial x^2} + \frac{\partial^2 \mathbf{u}}{\partial y^2} + \frac{\partial^2 \mathbf{u}}{\partial z^2} \\
&= \begin{pmatrix} \frac{\partial^2 u_x}{\partial x^2} + \frac{\partial^2 u_x}{\partial y^2} + \frac{\partial^2 u_x}{\partial z^2} \\ \frac{\partial^2 u_y}{\partial x^2} + \frac{\partial^2 u_y}{\partial y^2} + \frac{\partial^2 u_y}{\partial z^2} \\ \frac{\partial^2 u_z}{\partial x^2} + \frac{\partial^2 u_z}{\partial y^2} + \frac{\partial^2 u_z}{\partial z^2} \end{pmatrix} \\
\therefore \nabla(\nabla \cdot \mathbf{u}) - \nabla^2 \mathbf{u} &= \begin{pmatrix} \frac{\partial^2 u_y}{\partial x \partial y} + \frac{\partial^2 u_z}{\partial x \partial z} - \frac{\partial^2 u_x}{\partial y^2} - \frac{\partial^2 u_x}{\partial z^2} \\ \frac{\partial^2 u_x}{\partial x \partial y} + \frac{\partial^2 u_z}{\partial y \partial z} - \frac{\partial^2 u_y}{\partial x^2} - \frac{\partial^2 u_y}{\partial z^2} \\ \frac{\partial^2 u_x}{\partial x \partial z} + \frac{\partial^2 u_y}{\partial y \partial z} - \frac{\partial^2 u_z}{\partial x^2} - \frac{\partial^2 u_z}{\partial y^2} \end{pmatrix}
\end{aligned}$$

On the left-hand side, we have:

$$\begin{aligned}
 \nabla \times (\nabla \times \mathbf{u}) &= \nabla \times \begin{pmatrix} \frac{\partial u_y}{\partial z} - \frac{\partial u_z}{\partial y} \\ \frac{\partial u_x}{\partial z} - \frac{\partial u_z}{\partial x} \\ \frac{\partial u_x}{\partial y} - \frac{\partial u_y}{\partial x} \end{pmatrix} \\
 &= \begin{pmatrix} \frac{\partial}{\partial z} \left(\frac{\partial u_x}{\partial x} - \frac{\partial u_x}{\partial z} \right) - \frac{\partial}{\partial y} \left(\frac{\partial u_x}{\partial y} - \frac{\partial u_y}{\partial x} \right) \\ \frac{\partial}{\partial x} \left(\frac{\partial u_x}{\partial y} - \frac{\partial u_y}{\partial x} \right) - \frac{\partial}{\partial z} \left(\frac{\partial u_y}{\partial z} - \frac{\partial u_z}{\partial y} \right) \\ \frac{\partial}{\partial y} \left(\frac{\partial u_y}{\partial z} - \frac{\partial u_z}{\partial y} \right) - \frac{\partial}{\partial x} \left(\frac{\partial u_z}{\partial x} - \frac{\partial u_x}{\partial z} \right) \end{pmatrix} \\
 &= \begin{pmatrix} \frac{\partial^2 u_x}{\partial x \partial z} - \frac{\partial^2 u_x}{\partial z^2} - \frac{\partial^2 u_x}{\partial y^2} + \frac{\partial^2 u_y}{\partial x \partial y} \\ \frac{\partial^2 u_x}{\partial x \partial y} - \frac{\partial^2 u_y}{\partial x^2} - \frac{\partial^2 u_y}{\partial z^2} + \frac{\partial^2 u_z}{\partial y \partial z} \\ \frac{\partial^2 u_y}{\partial y \partial z} - \frac{\partial^2 u_z}{\partial y^2} - \frac{\partial^2 u_z}{\partial x^2} + \frac{\partial^2 u_x}{\partial x \partial z} \end{pmatrix} \\
 &= \begin{pmatrix} \frac{\partial^2 u_y}{\partial x \partial y} + \frac{\partial^2 u_z}{\partial x \partial z} - \frac{\partial^2 u_x}{\partial y^2} - \frac{\partial^2 u_x}{\partial z^2} \\ \frac{\partial^2 u_x}{\partial x \partial y} + \frac{\partial^2 u_z}{\partial y \partial z} - \frac{\partial^2 u_y}{\partial x^2} - \frac{\partial^2 u_y}{\partial z^2} \\ \frac{\partial^2 u_x}{\partial x \partial z} + \frac{\partial^2 u_y}{\partial y \partial z} - \frac{\partial^2 u_z}{\partial x^2} - \frac{\partial^2 u_z}{\partial y^2} \end{pmatrix}
 \end{aligned}$$

which is equal to the right-hand side. Hence, we have shown:

$$\nabla \times (\nabla \times \mathbf{u}) = \nabla(\nabla \cdot \mathbf{u}) - \nabla^2 \mathbf{u}$$

4.4 From the divergence theorem (Eq. 4.5), we have

$$\int_V (\nabla \cdot \mathbf{F}) dV = \int_S \mathbf{F} \cdot d\mathbf{S}$$

Choosing $\mathbf{F} = \begin{pmatrix} x \\ 0 \\ 0 \end{pmatrix}$, we obtain $\nabla \cdot \mathbf{F} = 1$ and $\mathbf{F} \cdot d\mathbf{S} = \mathbf{F} \cdot \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix} dS = xn_x dS$.

Hence, substituting these into the above divergence theorem, we obtain

$$\int_V dV = V = \int_S xn_x dS$$

Similarly, we can choose $\mathbf{F} = \begin{pmatrix} 0 \\ y \\ 0 \end{pmatrix}$, to obtain $\nabla \cdot \mathbf{F} = 1$ and $\mathbf{F} \cdot d\mathbf{S} = yn_y dS$.

Therefore,

$$\int_V dV = V = \int_S yn_y dS$$

Finally, choosing $\mathbf{F} = \begin{pmatrix} 0 \\ 0 \\ z \end{pmatrix}$, we have $\nabla \cdot \mathbf{F} = 1$ and $\mathbf{F} \cdot d\mathbf{S} = zn_z dS$. Hence,

$$\int_V dV = V = \int_S zn_z dS$$

Thus, for an arbitrary closed surface, we have shown its volume is given by

$$V = \int_S xn_x dS = \int_S yn_y dS = \int_S zn_z dS$$

as required.

4.5 (a) Assume we have a spherical shell region V centred on the microsphere, having inner radius r and outer radius $r + \Delta r$. The total amount of drug C within the spherical shell is the volume integral of concentration c , namely:

$$C = \int_V c dV = \int_r^{r+\Delta r} 4\pi\rho^2 c d\rho = 4\pi r^2 c \Delta r \quad (\text{if } \Delta r \text{ is sufficiently small})$$

Since C is a conserved quantity, its time rate of change within the spherical shell must equal the rate of amount of drug entering plus the rate at which it is produced (or lost) within the shell. We can express this statement as a differential equation:

$$\frac{\partial C}{\partial t} = - \int_S \mathbf{\Gamma} \cdot d\mathbf{S} + \int_V f dV$$

where $\mathbf{\Gamma}$ is the outward flux (i.e. amount of drug diffusing out of the boundaries S per unit time per unit area), and f is the rate of drug production per unit volume, which equals $-k_{up}c$. To find $\mathbf{\Gamma}$, we use Fick's Law of diffusion:

$$\mathbf{\Gamma} = -D\nabla c$$

and since the system is spherically symmetric, the concentration gradient will be along the radial direction only. Hence,

$$\mathbf{\Gamma} = -D \frac{\partial c}{\partial r} \mathbf{n}_r$$

where \mathbf{n}_r is the unit vector along the local radial axis. For our spherical shell,

$$\begin{aligned}
-\int_S \mathbf{F} \cdot d\mathbf{S} &= \int_S D \frac{\partial c}{\partial r} \mathbf{n}_r \cdot d\mathbf{S} \\
&= \left[D (4\pi \rho^2) \frac{\partial c}{\partial \rho} \right]_{\rho=r+\Delta r} - \left[D (4\pi \rho^2) \frac{\partial c}{\partial \rho} \right]_{\rho=r} \\
&= 4\pi \frac{\left(\left[D \rho^2 \frac{\partial c}{\partial \rho} \right]_{\rho=r+\Delta r} - \left[D \rho^2 \frac{\partial c}{\partial \rho} \right]_{\rho=r} \right)}{\Delta r} \Delta r \\
&= 4\pi \left(\frac{\partial}{\partial r} \left[D r^2 \frac{\partial c}{\partial r} \right] \right) \Delta r \quad (\text{for sufficiently small } \Delta r)
\end{aligned}$$

The rate of drug production within the shell is given by

$$\begin{aligned}
\int_V f \, dV &= \int_V -k_{up} c \, dV \\
&= \int_r^{r+\Delta r} - (4\pi \rho^2 k_{up} c) \, d\rho \\
&= -4\pi r^2 k_{up} c \Delta r \quad (\text{if } \Delta r \text{ is sufficiently small})
\end{aligned}$$

Substituting all these expressions into our earlier conservation law, we have:

$$\begin{aligned}
\frac{\partial C}{\partial t} &= - \int_S \mathbf{F} \cdot d\mathbf{S} + \int_V f \, dV \\
4\pi r^2 \Delta r \frac{\partial c}{\partial t} &= 4\pi \Delta r \frac{\partial}{\partial r} \left[D r^2 \frac{\partial c}{\partial r} \right] - 4\pi r^2 k_{up} c \Delta r
\end{aligned}$$

Dividing all terms by $4\pi r^2 \Delta r$, we obtain the required PDE:

$$\frac{\partial c}{\partial t} = \frac{1}{r^2} \frac{\partial}{\partial r} \left[D r^2 \frac{\partial c}{\partial r} \right] - k_{up} c$$

(b) Substituting $c = u/r$ into the above PDE, we obtain

$$\begin{aligned}
\frac{1}{r} \frac{\partial u}{\partial t} &= \frac{1}{r^2} \frac{\partial}{\partial r} \left[D r^2 \left(\frac{1}{r} \frac{\partial u}{\partial r} - \frac{u}{r^2} \right) \right] - k_{up} \left(\frac{u}{r} \right) \\
&= \frac{1}{r^2} \frac{\partial}{\partial r} \left[D r \frac{\partial u}{\partial r} - D u \right] - k_{up} \left(\frac{u}{r} \right) \\
&= \frac{1}{r^2} \left[D \frac{\partial u}{\partial r} + D r \frac{\partial^2 u}{\partial r^2} - D \frac{\partial u}{\partial r} \right] - k_{up} \left(\frac{u}{r} \right) \\
&= \frac{1}{r^2} \left[D r \frac{\partial^2 u}{\partial r^2} \right] - k_{up} \left(\frac{u}{r} \right) \\
&= \left(\frac{D}{r} \right) \frac{\partial^2 u}{\partial r^2} - k_{up} \left(\frac{u}{r} \right)
\end{aligned}$$

Multiplying all terms by r , leads to the following simplified PDE in u :

$$\frac{\partial u}{\partial t} = D \frac{\partial^2 u}{\partial r^2} - k_{up}u$$

To find the steady-state concentration, we set $\partial u/\partial t = 0$. The PDE is then transformed into the following ODE:

$$D \frac{d^2 u}{dr^2} - k_{up}u = 0$$

which has the characteristic equation

$$Dm^2 - k_{up} = 0$$

with roots $m = \pm \sqrt{k_{up}/D}$. Hence, its general solution is

$$u = C_1 e^{r\sqrt{\frac{k_{up}}{D}}} + C_2 e^{-r\sqrt{\frac{k_{up}}{D}}}$$

where C_1, C_2 are constants of integration. Since u cannot increase without bound as $r \rightarrow \infty$, we must have $C_1 = 0$. Furthermore, since the concentration c at $r = R$ (i.e. on the surface of the microsphere) equals c_0 , u must therefore equal to $c_0 R$ at $r = R$. Hence,

$$\begin{aligned} u(R) &= c_0 R = C_2 e^{-R\sqrt{\frac{k_{up}}{D}}} \\ \therefore C_2 &= c_0 R e^{R\sqrt{\frac{k_{up}}{D}}} \end{aligned}$$

Hence,

$$\begin{aligned} u &= c_0 R e^{R\sqrt{\frac{k_{up}}{D}}} e^{-r\sqrt{\frac{k_{up}}{D}}} \\ &= c_0 R e^{-(r-R)\sqrt{\frac{k_{up}}{D}}} \end{aligned}$$

and since $c = u/r$, the steady-state concentration of c is given by:

$$c_\infty(r) = c_0 \left(\frac{R}{r} \right) e^{-(r-R)\sqrt{\frac{k_{up}}{D}}} \quad (r > R)$$

Including the region inside the microsphere, the complete solution for the steady-state concentration is

$$c_\infty(r) = \begin{cases} c_0 \left(\frac{R}{r} \right) e^{-(r-R)\sqrt{\frac{k_{up}}{D}}} & r > R \\ c_0 & r \leq R \end{cases}$$

4.6 Let $c_{i,j}^n$ denote the amount of coins held by person (i, j) at time frame n . Each of the three rules for distribution yield an expression for the amount of coins held by person (i, j) at time frame $n + 1$, as follows:

(a) Denote the fixed fraction of coins given to each neighbour as α . Then,

$$\begin{aligned}
 c_{i,j}^{n+1} &= c_{i,j}^n - \overbrace{4\alpha c_{i,j}^n}^{\text{giving to 4 neighbours}} + \overbrace{\alpha c_{i-1,j}^n + \alpha c_{i+1,j}^n + \alpha c_{i,j-1}^n + \alpha c_{i,j+1}^n}^{\text{receiving from 4 neighbours}} \\
 &= c_{i,j}^n + \alpha c_{i-1,j}^n - 2\alpha c_{i,j}^n + \alpha c_{i+1,j}^n + \alpha c_{i,j-1}^n - 2\alpha c_{i,j}^n + \alpha c_{i,j+1}^n \\
 c_{i,j}^{n+1} - c_{i,j}^n &= \alpha (c_{i-1,j}^n - 2c_{i,j}^n + c_{i+1,j}^n + c_{i,j-1}^n - 2c_{i,j}^n + c_{i,j+1}^n) \\
 \Delta t \frac{c_{i,j}^{n+1} - c_{i,j}^n}{\Delta t} &= h^2 \alpha \left(\frac{c_{i-1,j}^n - 2c_{i,j}^n + c_{i+1,j}^n}{h^2} + \frac{c_{i,j-1}^n - 2c_{i,j}^n + c_{i,j+1}^n}{h^2} \right) \\
 \frac{c_{i,j}^{n+1} - c_{i,j}^n}{\Delta t} &= \frac{\alpha}{\mu} \left(\frac{c_{i-1,j}^n - 2c_{i,j}^n + c_{i+1,j}^n}{h^2} + \frac{c_{i,j-1}^n - 2c_{i,j}^n + c_{i,j+1}^n}{h^2} \right)
 \end{aligned}$$

where $\mu = \Delta t/h^2$. The terms in this expression represent finite-difference approximations to first-order derivatives in time and second-order derivatives space. In the limit as $h, \Delta t \rightarrow 0$, keeping μ fixed, the expression reduces to the familiar diffusion PDE:

$$\frac{\partial c}{\partial t} = D \left(\frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2} \right)$$

where the diffusion coefficient $D = \alpha/\mu$.

(b) Denote the fixed fraction of coins given to each neighbour by α , the income received per unit time by β . Then,

$$\begin{aligned}
 c_{i,j}^{n+1} &= c_{i,j}^n - \overbrace{4\alpha c_{i,j}^n}^{\text{giving to 4 neighbours}} + \overbrace{\alpha c_{i-1,j}^n + \alpha c_{i+1,j}^n + \alpha c_{i,j-1}^n + \alpha c_{i,j+1}^n}^{\text{receiving from 4 neighbours}} + \overbrace{\beta \Delta t}^{\text{income}} \\
 c_{i,j}^{n+1} - c_{i,j}^n &= \alpha c_{i-1,j}^n - 2\alpha c_{i,j}^n + \alpha c_{i+1,j}^n + \alpha c_{i,j-1}^n - 2\alpha c_{i,j}^n + \alpha c_{i,j+1}^n + \beta \Delta t \\
 \Delta t \frac{c_{i,j}^{n+1} - c_{i,j}^n}{\Delta t} &= h^2 \alpha \left(\frac{c_{i-1,j}^n - 2c_{i,j}^n + c_{i+1,j}^n}{h^2} + \frac{c_{i,j-1}^n - 2c_{i,j}^n + c_{i,j+1}^n}{h^2} \right) + \beta \Delta t \\
 \frac{c_{i,j}^{n+1} - c_{i,j}^n}{\Delta t} &= \frac{\alpha}{\mu} \left(\frac{c_{i-1,j}^n - 2c_{i,j}^n + c_{i+1,j}^n}{h^2} + \frac{c_{i,j-1}^n - 2c_{i,j}^n + c_{i,j+1}^n}{h^2} \right) + \beta
 \end{aligned}$$

where $\mu = \Delta t/h^2$. In the limit as $h, \Delta t \rightarrow 0$, keeping μ fixed, the expression reduces to the PDE:

$$\frac{\partial c}{\partial t} = D \left(\frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2} \right) + \beta$$

where $D = \alpha/\mu$.

(c) Denote the fraction of coins received from each neighbour by α , and the proportion given to charity per unit time by β . Then,

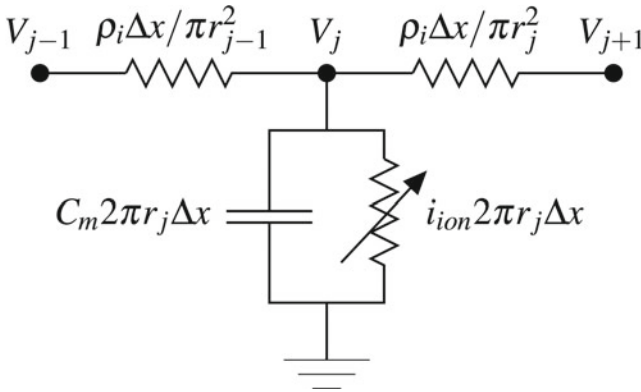
$$\begin{aligned}
 c_{i,j}^{n+1} &= c_{i,j}^n - \overbrace{4\alpha c_{i,j}^n}^{\text{giving to 4 neighbours}} + \overbrace{\alpha c_{i-1,j}^n + \alpha c_{i+1,j}^n + \alpha c_{i,j-1}^n + \alpha c_{i,j+1}^n}^{\text{receiving from 4 neighbours}} - \overbrace{\beta \Delta t c_{i,j}^n}^{\text{giving to charity}} \\
 c_{i,j}^{n+1} - c_{i,j}^n &= \alpha c_{i-1,j}^n - 2\alpha c_{i,j}^n + \alpha c_{i+1,j}^n + \alpha c_{i,j-1}^n - 2\alpha c_{i,j}^n + \alpha c_{i,j+1}^n - \beta \Delta t c_{i,j}^n \\
 \Delta t \frac{c_{i,j}^{n+1} - c_{i,j}^n}{\Delta t} &= h^2 \alpha \left(\frac{c_{i-1,j}^n - 2c_{i,j}^n + c_{i+1,j}^n}{h^2} + \frac{c_{i,j-1}^n - 2c_{i,j}^n + c_{i,j+1}^n}{h^2} \right) - \beta \Delta t c_{i,j}^n \\
 \frac{c_{i,j}^{n+1} - c_{i,j}^n}{\Delta t} &= \frac{\alpha}{\mu} \left(\frac{c_{i-1,j}^n - 2c_{i,j}^n + c_{i+1,j}^n}{h^2} + \frac{c_{i,j-1}^n - 2c_{i,j}^n + c_{i,j+1}^n}{h^2} \right) - \beta c_{i,j}^n
 \end{aligned}$$

where $\mu = \Delta t/h^2$. In the limit as $h, \Delta t \rightarrow 0$, keeping μ fixed, the expression reduces to the PDE:

$$\frac{\partial c}{\partial t} = D \left(\frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2} \right) - \beta c$$

where $D = \alpha/\mu$.

4.7 Discretizing the nerve fibre into elements of length Δx , we obtain the following electrical equivalent circuit for one node, where r_j denotes the radius at node j :



From Kirchhoff's current law, the current flowing into node j must equal the current leaving it. Namely:

$$\begin{aligned}
 \overbrace{\frac{V_{j+1} - V_j}{\rho_i \Delta x / \pi r_j^2} + \frac{V_{j-1} - V_j}{\rho_i \Delta x / \pi r_{j-1}^2}}^{\text{current entering}} &= \overbrace{C_m 2\pi r_j \Delta x \frac{dV_j}{dt} + i_{ion} 2\pi r_j \Delta x}_{\text{current leaving}} \\
 \frac{\pi}{\rho_i} \left[r_j^2 \frac{V_{j+1} - V_j}{\Delta x} + r_{j-1}^2 \frac{V_{j-1} - V_j}{\Delta x} \right] &= 2\pi r_j \Delta x \left(C_m \frac{dV_j}{dt} + i_{ion} \right) \\
 \frac{1}{\Delta x} \left[\frac{r_j^2}{2\rho_i} \frac{V_{j+1} - V_j}{\Delta x} - \frac{r_{j-1}^2}{2\rho_i} \frac{V_j - V_{j-1}}{\Delta x} \right] &= r_j \left(C_m \frac{dV_j}{dt} + i_{ion} \right)
 \end{aligned}$$

In the limit as $\Delta x \rightarrow 0$, the above finite difference expressions approximate to the following PDE:

$$\frac{\partial}{\partial x} \left[\frac{r^2}{2\rho_i} \frac{\partial V}{\partial x} \right] = r \left(C_m \frac{\partial V}{\partial t} + i_{ion} \right)$$

or to be compatible with COMSOL's general PDE form:

$$r C_m \frac{\partial V}{\partial t} + \frac{\partial}{\partial x} \left[-\frac{r^2}{2\rho_i} \frac{\partial V}{\partial x} \right] = -r i_{ion}$$

4.8 To derive the underlying PDE for the muscle displacement, we discretize the muscle strand into N sub-units, each of length $\Delta x = L/N$. We can therefore characterise $N + 1$ nodes, each having displacement u_i ($i = 0 \dots N$). Denoting the change in length of the i th sub-unit by l_i , and its corresponding l_1, l_2 values by l_{1i}, l_{2i} respectively. Then the following relationships hold:

$$\begin{aligned} l_i &= l_{1i} + l_{2i} = u_i - u_{i-1} \\ F_i &= \left(\frac{k_1}{\Delta x} \right) l_{1i} && \text{for the series element} \\ F_i &= \left(\frac{k_2}{\Delta x} \right) l_{2i} + \left(\frac{b}{\Delta x} \right) \frac{dl_{2i}}{dt} && \text{for the parallel spring/dashpot} \end{aligned}$$

where F_i is the passive force generated by the i th sub-unit, which acts on the point masses to oppose the changes in length. F_i is the same for both the series spring and the parallel spring/dashpot combination in the i th sub-unit, since these elements are in series. Re-arranging the last two of the above relationships, we obtain:

$$\begin{aligned} l_{1i} &= \left(\frac{F_i}{k_1} \right) \Delta x \\ l_{2i} &= \left(\frac{F_i}{k_2} \right) \Delta x - \left(\frac{b}{k_2} \right) \frac{dl_{2i}}{dt} \end{aligned}$$

Adding these together yields:

$$\begin{aligned} l_{1i} + l_{2i} &= \left(\frac{1}{k_1} + \frac{1}{k_2} \right) F_i \Delta x - \left(\frac{b}{k_2} \right) \frac{dl_{2i}}{dt} \\ l_i &= \left(\frac{1}{k_1} + \frac{1}{k_2} \right) F_i \Delta x - \left(\frac{b}{k_2} \right) \frac{dl_{2i}}{dt} \\ &= \left(\frac{1}{k_1} + \frac{1}{k_2} \right) F_i \Delta x - \left(\frac{b}{k_2} \right) \left[\frac{dl_i}{dt} - \frac{dl_{1i}}{dt} \right] \end{aligned}$$

and using the series element relationship $F_i = (k_1/\Delta x)l_{1i}$, we obtain $dF_i/dt = (k_1/\Delta x)dl_{1i}/dt$ and hence $dl_{1i}/dt = (\Delta x/k_1)dF_i/dt$. Substituting the latter into the above expression, we obtain:

$$l_i = \left(\frac{1}{k_1} + \frac{1}{k_2}\right) F_i \Delta x - \left(\frac{b}{k_2}\right) \left[\frac{dl_i}{dt} - \frac{\Delta x}{k_1} \frac{dF}{dt}\right]$$

$$l_i + \left(\frac{b}{k_2}\right) \frac{dl_i}{dt} = \left(\frac{1}{k_1} + \frac{1}{k_2}\right) F_i \Delta x + \left(\frac{b}{k_1 k_2}\right) \frac{dF}{dt} \Delta x$$

Multiplying throughout by $k_1 k_1 / \Delta x$ and re-arranging terms, yields:

$$\left(\frac{1}{\Delta x}\right) \left[bk_1 \frac{dl_i}{dt} + k_1 k_2 l_i \right] = b \frac{dF_i}{dt} + (k_1 + k_2) F_i$$

Finally, substituting $l_i = u_i - u_{i-1}$, we obtain:

$$\left(\frac{1}{\Delta x}\right) \left[bk_1 \left(\frac{du_i}{dt} - \frac{du_{i-1}}{dt}\right) + k_1 k_2 (u_i - u_{i-1}) \right] = b \frac{dF_i}{dt} + (k_1 + k_2) F_i \quad (\text{B.1})$$

A similar relation holds also for the $(i + 1)$ th sub-unit:

$$\left(\frac{1}{\Delta x}\right) \left[bk_1 \left(\frac{du_{i+1}}{dt} - \frac{du_i}{dt}\right) + k_1 k_2 (u_{i+1} - u_i) \right] = b \frac{dF_{i+1}}{dt} + (k_1 + k_2) F_{i+1} \quad (\text{B.2})$$

The total force acting on point mass i is given by

$$F_{i+1} - F_i = (M \Delta x) \frac{d^2 u_i}{dt^2}$$

Therefore, subtracting Eq. B.1 from Eq. B.2, we obtain:

$$\begin{aligned} \left(\frac{1}{\Delta x}\right) \left[bk_1 \left(\frac{du_{i+1}}{dt} - \frac{du_i}{dt} + \frac{du_{i-1}}{dt}\right) + k_1 k_2 (u_{i+1} - 2u_i + u_{i-1}) \right] \\ = b \left(\frac{dF_{i+1}}{dt} - \frac{dF_i}{dt}\right) + (k_1 + k_2) [F_{i+1} - F_i] \\ = b M \Delta x \frac{d}{dt} \left[\frac{d^2 u_i}{dt^2}\right] + (k_1 + k_2) M \Delta x \frac{d^2 u_i}{dt^2} \end{aligned}$$

Dividing both sides by Δx :

$$\begin{aligned} bk_1 \frac{d}{dt} \left[\frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta^2 x}\right] + k_1 k_2 \left(\frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta^2 x}\right) \\ = b M \frac{d}{dt} \left[\frac{d^2 u_i}{dt^2}\right] + (k_1 + k_2) M \frac{d^2 u_i}{dt^2} \end{aligned}$$

In the limit as $\Delta x \rightarrow 0$, this expression reduces to the PDE:

$$bk_1 \frac{\partial}{\partial t} \left[\frac{\partial^2 u}{\partial x^2}\right] + k_1 k_2 \frac{\partial^2 u}{\partial x^2} = b M \frac{\partial^3 u}{\partial t^3} + (k_1 + k_2) M \frac{\partial^2 u}{\partial t^2}$$

where x is the spatial coordinate along the length of the muscle. Re-arranging terms, the above PDE can also be re-written as:

$$b \frac{\partial}{\partial t} \left[k_1 \frac{\partial^2 u}{\partial x^2} - M \frac{\partial^2 u}{\partial t^2} \right] + k_2 \left(k_1 \frac{\partial^2 u}{\partial x^2} - M \frac{\partial^2 u}{\partial t^2} \right) = k_1 M \frac{\partial^2 u}{\partial t^2}$$

4.9 (a) To solve the atrial tissue PDE system using the method of lines, we first discretize the 2D domain into a square grid of size $N \times N$. There will therefore be a total of N^2 individual nodes, with $2N^2$ variables to be solved for (i.e. V_m and u at each node). The following PDE must be discretized:

$$\begin{aligned} \beta C_m \left(\frac{\partial V_m}{\partial t} \right) &= \nabla \cdot (\sigma \nabla V_m) - \beta i_{ion} + i_{stim} \\ &= \sigma \left(\frac{\partial^2 V_m}{\partial x^2} + \frac{\partial^2 V_m}{\partial y^2} \right) - \beta i_{ion} + i_{stim} \quad (\text{since } \sigma \text{ is constant}) \\ \therefore \frac{\partial V_m}{\partial t} &= \frac{\sigma}{\beta C_m} \left(\frac{\partial^2 V_m}{\partial x^2} + \frac{\partial^2 V_m}{\partial y^2} \right) - \frac{i_{ion}}{C_m} + \frac{i_{stim}}{\beta C_m} \end{aligned}$$

Denoting V_m and u for the (i, j) th node as $V_{i,j}$ and $u_{i,j}$ respectively, this PDE can be discretized as:

$$\begin{aligned} \frac{dV_{i,j}}{dt} &= \frac{\sigma}{\beta C_m} \left[\frac{V_{i,j+1} - 2V_{i,j} + V_{i,j-1}}{h^2} + \frac{V_{i+1,j} - 2V_{i,j} + V_{i-1,j}}{h^2} \right] - \frac{i_{ion,i,j}}{C_m} + \frac{i_{stim,i,j}}{\beta C_m} \\ &= \frac{\sigma}{\beta C_m} \left[\frac{V_{i,j+1} + V_{i,j-1} + V_{i+1,j} + V_{i-1,j} - 4V_{i,j}}{h^2} \right] - \frac{i_{ion,i,j}}{C_m} + \frac{i_{stim,i,j}}{\beta C_m} \end{aligned}$$

where $i_{ion,i,j}$ and $i_{stim,i,j}$ are i_{ion} and i_{stim} evaluated at node (i, j) . To setup this system in Matlab, its in-built ode solvers require the independent variables to be in a single-column array format. Denoting this array by \mathbf{Y} , we can assign the first N^2 elements to the V_m variables and the final N^2 elements for the u , according the following ‘‘map’’:

$$\begin{aligned} V_{i,j} &= Y_{(i-1)N+j} \\ u_{i,j} &= Y_{N^2+(i-1)N+j} \end{aligned}$$

with $i, j = 1 \dots N$. To implement the zero-flux boundary conditions on the external boundaries, we set the V_m value of points exterior to the boundary to its value at the adjacent boundary point. In Matlab, this can be achieved by padding the 2D V_m -array with extra rows and columns whose V_m values are equal to the adjacent boundary.

The Matlab code below⁷ solves this PDE system using a spatial discretization of 51×51 nodes, plotting the solution for V_m at times 0.3, 0.35, 0.4, and 0.45 s, where it can be seen that a self-perpetuating reentrant spiral wave of activation is initiated by the stimulus protocol delivered.

atrial_prime.m:

```
function Y_prime = atrial_prime(t,Y)
global beta sigma Cm a b c1 c2 A B d e N h
V = reshape(Y(1:N^2),N,N)'; % membrane potentials
U = reshape(Y(N^2+1:2*N^2),N,N)'; % u values
Y_prime = zeros(2*N^2,1);
% Next,"pad" the V array to implement zero-flux b.c.'s
VV = [V(1,1), V(1,1:N), V(1,N); ...
      V(1:N,1), V, V(1:N,N); ...
      V(N,1), V(N,1:N), V(N,N)];
% calculate derivatives
for i = 1:N
    for j=1:N
        % obtain Vm value of current node
        % and its four neighbours
        Vm = VV(i+1,j+1); % padded indices
        Vm_left = VV(i+1,j);
        Vm_right = VV(i+1,j+2);
        Vm_below = VV(i,j+1);
        Vm_above = VV(i+2,j+1);
        % obtain remaining nodal variables
        u = U(i,j);
        x = (j-1)*h; % x value
        y = (i-1)*h; % y value
        i_ion = c1*(Vm-a)*(Vm-A)*(Vm-B)+c2*u*(Vm-B);
        if (t >= 0.01)&&(t<=0.011)
            if (y < 0.01)
                i_stim = 50;
            else
                i_stim = 0;
            end;
        elseif (t >= 0.15)&&(t<=0.151)
            if (x < 0.01)
                i_stim = 50;
            else
                i_stim = 0;
            end;
        else
            i_stim = 0;
        end;
        % determine derivative of Vm
        Y_prime((i-1)*N+j) = ...
            sigma/(beta*Cm)*(Vm_left+Vm_right+Vm_below+Vm_above-4*Vm)/(h^2)
        ... - i_ion/Cm + i_stim/(beta*Cm);
        % determine derivative of u
        Y_prime(N^2+(i-1)*N+j) = e*(Vm-d*u-b);
    end;
end;
```

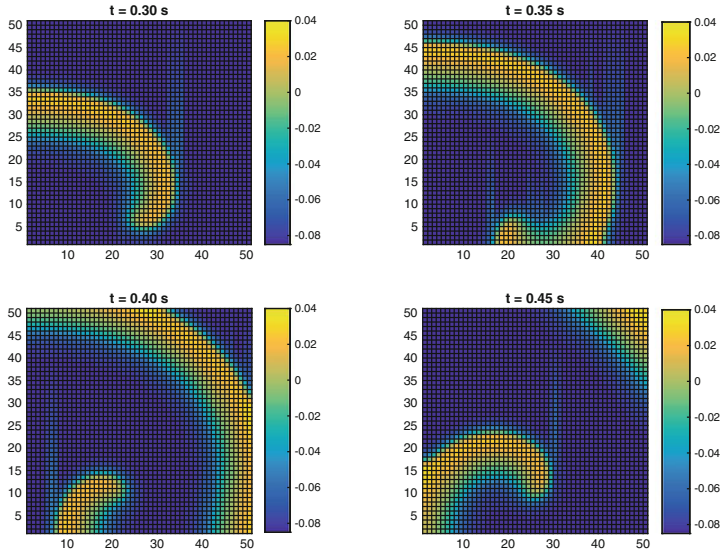
⁷This code took just over 12min to solve using Matlab (R2014b) on a MacBook Air (2013) with 8 GB RAM.

atrial_solve_MOL.m

```

% solves atrial tissue electrical model
% using the method of lines
global beta sigma Cm a b c1 c2 A B d e N h
beta = 100;           % 1/m
sigma = 0.001;       % S/m
Cm = 0.01;           % F/m^2
a = -0.0668;         % V
b = -0.085;          % V
c1 = 530;            % S/(V*m)^2
c2 = 4;              % S/m^2
A = 0.055;           % V
B = -0.085;          % V
d = 0.14;            % V
e = 285.7;           % (1/(V*s))
N = 51;
h = 0.1/(N-1);
Y_init = zeros(2*N^2,1);
Y_init(1:N^2) = -0.085;
options = odeset('MaxStep',0.001);
[t_out, Y_out] = ode15s('atrial_prime', 0:0.001:0.5, Y_init, options);
% plot result
V_array_1 = Y_out(300,1:N^2);
V_out_1 = reshape(V_array_1,N,N);
V_array_2 = Y_out(350,1:N^2);
V_out_2 = reshape(V_array_2,N,N);
V_array_3 = Y_out(400,1:N^2);
V_out_3 = reshape(V_array_3,N,N);
V_array_4 = Y_out(450,1:N^2);
V_out_4 = reshape(V_array_4,N,N);
subplot(2,2,1), pcolor(V_out_1), colorbar, axis('square'), ...
    caxis([-0.085,0.04]), title('t = 0.30 s');
subplot(2,2,2), pcolor(V_out_2), colorbar, axis('square'), ...
    caxis([-0.085,0.04]), title('t = 0.35 s');
subplot(2,2,3), pcolor(V_out_3), colorbar, axis('square'), ...
    caxis([-0.085,0.04]), title('t = 0.40 s');
subplot(2,2,4), pcolor(V_out_4), colorbar, axis('square'), ...
    caxis([-0.085,0.04]), title('t = 0.45 s');

```

(b) To solve the above PDE system using an explicit finite difference scheme, we denote V_m and u for the (i, j) th node and n th time step by $V_{i,j}^n$ and $u_{i,j}^n$ respectively. The PDE can therefore be discretized as:

$$\frac{V_{i,j}^{n+1} - V_{i,j}^n}{\Delta t} = \frac{\sigma}{\beta C_m} \left[\frac{V_{i,j+1}^n + V_{i,j-1}^n + V_{i+1,j}^n + V_{i-1,j}^n - 4V_{i,j}^n}{h^2} \right] - \frac{i_{ion,i,j}^n}{C_m} + \frac{i_{stim,i,j}^n}{\beta C_m}$$

where h and Δt are the spacial resolution and time step respectively, and $i_{ion,i,j}^n$, $i_{stim,i,j}^n$ are i_{ion} and i_{stim} evaluated at node (i, j) and time step n . Re-arranging the above, we have:

$$V_{i,j}^{n+1} = V_{i,j}^n + \frac{\sigma \Delta t}{\beta C_m h^2} \left[V_{i,j+1}^n + V_{i,j-1}^n + V_{i+1,j}^n + V_{i-1,j}^n - 4V_{i,j}^n \right] - \frac{i_{ion,i,j}^n}{C_m} + \frac{i_{stim,i,j}^n}{\beta C_m}$$

Similarly for the u variables, we have

$$\begin{aligned} \frac{U_{i,j}^{n+1} - U_{i,j}^n}{\Delta t} &= e (V_{i,j}^n - dU_{i,j}^n - b) \\ \therefore U_{i,j}^{n+1} &= U_{i,j}^n + e (V_{i,j}^n - dU_{i,j}^n - b) \Delta t \end{aligned}$$

The Matlab code below implements the above explicit FD scheme for V_m and u , plotting the solution for V_m at $t = 0.3, 0.35, 0.4,$ and 0.45 s. As in part a), a ‘padded’ V_m -array was used to implement the zero-flux boundary conditions. The code takes

advantage of Matlab's array processing abilities, and is much more rapid to solve for than the method of lines approach above.⁸

```

% solves atrial tissue electrical model
% using an explicit FD method
beta = 100;           % 1/m
sigma = 0.001;       % S/m
Cm = 0.01;           % F/m^2
a = -0.0668;         % V
b = -0.085;          % V
c1 = 530;            % S/(V*m)^2
c2 = 4;              % S/m^2
A = 0.055;           % V
B = -0.085;          % V
d = 0.14;            % V
e = 285.7;           % (1/(V*s))
N = 51;
h = 0.1/(N-1);      % m
Dt = 1e-5;           % s
% initialise Vm and u
V = ones(N)*-0.085;
U = zeros(N);
% Next, "pad" the V array to implement zero-flux b.c.'s
VV = [V(1,1), V(1,1:N), V(1,N); ...
      V(1:N,1), V, V(1:N,N); ...
      V(N,1), V(N,1:N), V(N,N)];
% initialise spatial coordinates
x = zeros(N);
y = zeros(N);
for j=2:N
    x(:,j) = x(:,j-1)+h;
end; for i=2:N
    y(i,:) = y(i-1,)+h;
end;
% explicit FD time stepping loop
for t = 0:Dt:0.5
    i_ion = c1*(V-a).*(V-A).*(V-B)+c2*U.*(V-B);
    if (t >= 0.01)&&(t<=0.011)
        i_stim = 50*(y<0.01);
    elseif (t >= 0.15)&&(t<=0.151)
        i_stim = 50*(y<0.01);
    else
        i_stim = zeros(N);
    end;
    V = V + sigma*Dt/(beta*Cm*h^2)*(VV(2:N+1,1:N)+VV(2:N+1,3:N+2) + ...
        VV(1:N,2:N+1)+VV(3:N+2,2:N+1)-4*VV(2:N+1,2:N+1) ...
        - i_ion/Cm + i_stim/(beta*Cm));
    U = U + e*(VV(2:N+1,2:N+1)-d*U-b)*Dt;
    if (t == 0.3)
        V_array_1 = V;
    elseif (t == 0.35)
        V_array_2 = V;

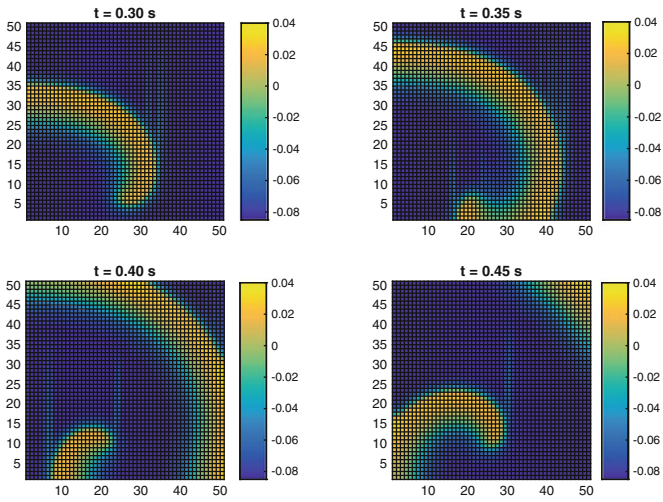
```

⁸This FD scheme took just over 4 s to solve for and plot using Matlab (R2014b) on a MacBook Air (2013) with 8 GB RAM.

```

elseif (t == 0.4)
    V_array_3 = V;
elseif (t == 0.45)
    V_array_4 = V;
end;
end;
% plot result
subplot(2,2,1), pcolor(V_out_1), colorbar, axis('square'), ...
    caxis([-0.085,0.04]), title('t = 0.30 s');
subplot(2,2,2), pcolor(V_out_2), colorbar, axis('square'), ...
    caxis([-0.085,0.04]), title('t = 0.35 s');
subplot(2,2,3), pcolor(V_out_3), colorbar, axis('square'), ...
    caxis([-0.085,0.04]), title('t = 0.40 s');
subplot(2,2,4), pcolor(V_out_4), colorbar, axis('square'), ...
    caxis([-0.085,0.04]), title('t = 0.45 s');

```



4.10 (a) We can write the voltage distribution $V(x, y)$ in the Petri dish in terms of polar coordinates $V(r, \theta)$, to take advantage of the circular geometry of the problem, where $r = \sqrt{x^2 + y^2}$ and $\theta = \cos^{-1}(x/r)$. The relevant PDE is:

$$\nabla \cdot (\sigma \nabla V) = 0$$

and since the conductivity σ is constant throughout the domain, we can divide through by σ to obtain:

$$\nabla \cdot (\nabla V) = \frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} = 0$$

To express this PDE in terms of r and θ polar coordinates, we first note that

$$\begin{aligned}
\frac{\partial r}{\partial x} &= \frac{\partial}{\partial x} \left[\frac{x}{\sqrt{x^2 + y^2}} \right] \\
&= \frac{x}{\sqrt{x^2 + y^2}} \\
&= \frac{x}{r} \\
\frac{\partial \theta}{\partial x} &= \frac{\partial}{\partial x} \left[\cos^{-1} \left(\frac{x}{r} \right) \right] \\
&= \frac{-1}{\sqrt{1 - \frac{x^2}{r^2}}} \frac{\partial}{\partial x} \left[\frac{x}{r} \right] \\
&= \frac{-1}{\sqrt{\frac{r^2}{r^2} - \frac{x^2}{r^2}}} \left[\frac{1}{r} - \frac{x}{r^2} \frac{\partial r}{\partial x} \right] \\
&= \frac{-1}{\sqrt{\frac{y^2}{r^2}}} \left[\frac{1}{r} - \left(\frac{x}{r^2} \right) \left(\frac{x}{r} \right) \right] \\
&= -\frac{r}{y} \left[\frac{1}{r} - \frac{x^2}{r^3} \right] \\
&= -\frac{1}{y} \left[1 - \frac{x^2}{r^2} \right] \\
&= -\frac{1}{y} \left[\frac{r^2}{r^2} - \frac{x^2}{r^2} \right] \\
&= -\frac{1}{y} \left[\frac{y^2}{r^2} \right] \\
&= -\frac{y}{r^2}
\end{aligned}$$

Using a similar evaluation, we find that

$$\frac{\partial r}{\partial y} = \frac{y}{r} \quad \frac{\partial \theta}{\partial y} = \frac{x}{r^2}$$

Using the above expressions, we can now evaluate the first term of the PDE, $\partial^2 V / \partial x^2$. We begin with:

$$\begin{aligned}
\frac{\partial V}{\partial x} &= \frac{\partial V}{\partial r} \frac{\partial r}{\partial x} + \frac{\partial V}{\partial \theta} \frac{\partial \theta}{\partial x} \quad (\text{using the chain rule}) \\
&= \frac{x}{r} \frac{\partial V}{\partial r} - \frac{y}{r^2} \frac{\partial V}{\partial \theta} \quad (\text{using the previous expressions})
\end{aligned}$$

And taking one more derivative, we have:

$$\frac{\partial^2 V}{\partial x^2} = \frac{\partial}{\partial x} \left[\frac{x}{r} \frac{\partial V}{\partial r} \right] - \frac{\partial}{\partial x} \left[\frac{y}{r^2} \frac{\partial V}{\partial \theta} \right]$$

Evaluating each of the terms on the right-hand side separately, we have:

$$\begin{aligned} \frac{\partial}{\partial x} \left[\frac{x}{r} \frac{\partial V}{\partial r} \right] &= \frac{\partial}{\partial r} \left[\frac{x}{r} \frac{\partial V}{\partial r} \right] \frac{\partial r}{\partial x} + \frac{\partial}{\partial \theta} \left[\frac{x}{r} \frac{\partial V}{\partial r} \right] \frac{\partial \theta}{\partial x} \\ &= \frac{x}{r} \left\{ \left[-\frac{x}{r^2} + \frac{1}{r} \frac{\partial x}{\partial r} \right] \frac{\partial V}{\partial r} + \frac{x}{r} \frac{\partial^2 V}{\partial r^2} \right\} - \frac{y}{r^2} \left\{ \left[\frac{1}{r} \frac{\partial x}{\partial \theta} \right] \frac{\partial V}{\partial r} + \frac{x}{r} \frac{\partial^2 V}{\partial r \partial \theta} \right\} \\ &= \frac{x}{r} \left\{ \left[-\frac{x}{r^2} + \frac{1}{r} \cos \theta \right] \frac{\partial V}{\partial r} + \frac{x}{r} \frac{\partial^2 V}{\partial r^2} \right\} - \frac{y}{r^2} \left\{ \left[-\frac{1}{r} r \sin \theta \right] \frac{\partial V}{\partial r} + \frac{x}{r} \frac{\partial^2 V}{\partial r \partial \theta} \right\} \\ &= \frac{x}{r} \left\{ \left[-\frac{x}{r^2} + \frac{1}{r} \frac{x}{r} \right] \frac{\partial V}{\partial r} + \frac{x}{r} \frac{\partial^2 V}{\partial r^2} \right\} - \frac{y}{r^2} \left\{ \left[-\frac{1}{r} r \frac{y}{r} \right] \frac{\partial V}{\partial r} + \frac{x}{r} \frac{\partial^2 V}{\partial r \partial \theta} \right\} \\ &= \frac{y^2}{r^3} \frac{\partial V}{\partial r} + \frac{x^2}{r^2} \frac{\partial^2 V}{\partial r^2} - \frac{xy}{r^3} \frac{\partial^2 V}{\partial r \partial \theta} \\ \frac{\partial}{\partial x} \left[\frac{y}{r^2} \frac{\partial V}{\partial \theta} \right] &= \frac{\partial}{\partial r} \left[\frac{y}{r^2} \frac{\partial V}{\partial \theta} \right] \frac{\partial r}{\partial x} + \frac{\partial}{\partial \theta} \left[\frac{y}{r^2} \frac{\partial V}{\partial \theta} \right] \frac{\partial \theta}{\partial x} \\ &= \frac{x}{r} \left\{ \left[\frac{1}{r^2} \frac{\partial y}{\partial r} - \frac{2y}{r^3} \right] \frac{\partial V}{\partial \theta} + \frac{y}{r^2} \frac{\partial^2 V}{\partial r \partial \theta} \right\} - \frac{y}{r^2} \left\{ \left[\frac{1}{r^2} \frac{\partial y}{\partial \theta} \right] \frac{\partial V}{\partial \theta} + \frac{y}{r^2} \frac{\partial^2 V}{\partial \theta^2} \right\} \\ &= \frac{x}{r} \left\{ \left[\frac{1}{r^2} \sin \theta - \frac{2y}{r^3} \right] \frac{\partial V}{\partial \theta} + \frac{y}{r^2} \frac{\partial^2 V}{\partial r \partial \theta} \right\} - \frac{y}{r^2} \left\{ \left[\frac{1}{r^2} r \cos \theta \right] \frac{\partial V}{\partial \theta} + \frac{y}{r^2} \frac{\partial^2 V}{\partial \theta^2} \right\} \\ &= \frac{x}{r} \left\{ \left[\frac{1}{r^2} \frac{y}{r} - \frac{2y}{r^3} \right] \frac{\partial V}{\partial \theta} + \frac{y}{r^2} \frac{\partial^2 V}{\partial r \partial \theta} \right\} - \frac{y}{r^2} \left\{ \left[\frac{1}{r^2} r \frac{x}{r} \right] \frac{\partial V}{\partial \theta} + \frac{y}{r^2} \frac{\partial^2 V}{\partial \theta^2} \right\} \\ &= \frac{x}{r} \left\{ -\frac{y}{r^3} \frac{\partial V}{\partial \theta} + \frac{y}{r^2} \frac{\partial^2 V}{\partial r \partial \theta} \right\} - \frac{y}{r^2} \left\{ \frac{x}{r^2} \frac{\partial V}{\partial \theta} + \frac{y}{r^2} \frac{\partial^2 V}{\partial \theta^2} \right\} \\ &= -\frac{xy}{r^4} \frac{\partial V}{\partial \theta} + \frac{xy}{r^3} \frac{\partial^2 V}{\partial r \partial \theta} - \frac{xy}{r^4} \frac{\partial V}{\partial \theta} - \frac{y^2}{r^4} \frac{\partial^2 V}{\partial \theta^2} \\ &= -\frac{2xy}{r^4} \frac{\partial V}{\partial \theta} + \frac{xy}{r^3} \frac{\partial^2 V}{\partial r \partial \theta} - \frac{y^2}{r^4} \frac{\partial^2 V}{\partial \theta^2} \end{aligned}$$

Combining both terms together, we obtain

$$\frac{\partial^2 V}{\partial x^2} = \frac{y^2}{r^3} \frac{\partial V}{\partial r} + \frac{2xy}{r^4} \frac{\partial V}{\partial \theta} - \frac{2xy}{r^3} \frac{\partial^2 V}{\partial r \partial \theta} + \frac{x^2}{r^2} \frac{\partial^2 V}{\partial r^2} + \frac{y^2}{r^4} \frac{\partial^2 V}{\partial \theta^2}$$

After a similar lengthy derivation, we also obtain for $\partial^2 V / \partial y^2$:

$$\frac{\partial^2 V}{\partial y^2} = \frac{x^2}{r^3} \frac{\partial V}{\partial r} - \frac{2xy}{r^4} \frac{\partial V}{\partial \theta} + \frac{2xy}{r^3} \frac{\partial^2 V}{\partial r \partial \theta} + \frac{y^2}{r^2} \frac{\partial^2 V}{\partial r^2} + \frac{x^2}{r^4} \frac{\partial^2 V}{\partial \theta^2}$$

Adding both of the above second-order partial derivatives, we obtain:

$$\begin{aligned}\frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} &= \frac{y^2 + x^2}{r^3} \frac{\partial V}{\partial r} + \frac{x^2 + y^2}{r^2} \frac{\partial^2 V}{\partial r^2} + \frac{y^2 + x^2}{r^4} \frac{\partial^2 V}{\partial \theta^2} \\ &= \frac{1}{r} \frac{\partial V}{\partial r} + \frac{\partial^2 V}{\partial r^2} + \frac{1}{r^2} \frac{\partial^2 V}{\partial \theta^2}\end{aligned}$$

Substituting this into the PDE, we obtain

$$\frac{1}{r} \frac{\partial V}{\partial r} + \frac{\partial^2 V}{\partial r^2} + \frac{1}{r^2} \frac{\partial^2 V}{\partial \theta^2} = 0$$

(b) We can discretize the above PDE using a regular grid over (r, θ) coordinate space. Furthermore, due to symmetry of the problem about the x -axis, we will utilise only the half-domain $\theta \in [0, \pi]$, $r \in [0, R]$, such that for the (i, j) th node, we have:

$$\begin{aligned}r_i &= i \Delta r & 0 &= 1 \dots N \\ \theta_j &= j \Delta \theta & 0 &= 1 \dots M\end{aligned}$$

where $\Delta r, \Delta \theta$ are the spatial resolutions in r and θ respectively, with the total number of grid points in r and θ given by $N + 1, M + 1$ respectively, such that

$$\Delta r = \frac{R}{N} \quad \text{and} \quad \Delta \theta = \frac{\pi}{M}$$

Denoting the value of V at the (i, j) th node by $V_{i,j}$, the above PDE can be discretized using finite difference approximations of the derivatives as follows:

$$\frac{1}{r_i} \left(\frac{V_{i+1,j} - V_{i,j}}{\Delta r} \right) + \left(\frac{V_{i+1,j} - 2V_{i,j} + V_{i-1,j}}{\Delta^2 r} \right) + \frac{1}{r_i^2} \left(\frac{V_{i,j+1} - 2V_{i,j} + V_{i,j-1}}{\Delta^2 \theta} \right) = 0$$

This approximation, however, cannot be used at the centre of the domain, since evaluation at $r_i = 0$ (i.e. $i = 0$) will lead to a singular value in the expression.⁹ However, for $i \geq 1$ we can continue to expand the above as follows:

$$\frac{1}{i \Delta r} \left(\frac{V_{i+1,j} - V_{i,j}}{\Delta r} \right) + \left(\frac{V_{i+1,j} - 2V_{i,j} + V_{i-1,j}}{\Delta^2 r} \right) + \frac{1}{i^2 \Delta^2 r} \left(\frac{V_{i,j+1} - 2V_{i,j} + V_{i,j-1}}{\Delta^2 \theta} \right) = 0$$

Multiplying all terms by $i^2 \Delta^2 r \Delta^2 \theta$, we obtain:

$$i \Delta^2 \theta (V_{i+1,j} - V_{i,j}) + i^2 \Delta^2 \theta (V_{i+1,j} - 2V_{i,j} + V_{i-1,j}) + (V_{i,j+1} - 2V_{i,j} + V_{i,j-1}) = 0$$

or

$$(i^2 \Delta^2 \theta + i \Delta^2 \theta) V_{i+1,j} + (i^2 \Delta^2 \theta) V_{i-1,j} + V_{i,j+1} + V_{i,j-1} - (2i^2 \Delta^2 \theta + i \Delta^2 \theta + 2) V_{i,j} = 0$$

⁹This property is a consequence of the choice of polar coordinates, rather than any actual discontinuity in the solution for V at the centre of the domain.

At the centre of the Petri dish, where $i = 0$, we can assign V to equal the mean value of all surrounding nodes, namely

$$V_{0,j} = \frac{1}{M+1} \sum_{j=0}^M V_{1,j} \quad j = 0 \cdots M$$

or more simply,

$$(M+1)V_{0,j} - \sum_{j=0}^M V_{1,j} = 0 \quad j = 0 \cdots M$$

Furthermore, along the walls of the dish (i.e. at $i = N$), the following boundary conditions are enforced:

$$V_{N,j} = \begin{cases} V_0 & j\Delta\theta \leq \frac{\theta_0}{2} & \text{(active electrode)} \\ 0 & j\Delta\theta \geq \pi - \frac{\theta_0}{2} & \text{(ground electrode)} \\ V_{N-1,j} & \frac{\theta_0}{2} < j\Delta\theta < \pi - \frac{\theta_0}{2} & \text{(zero - flux boundaries)} \end{cases}$$

Finally, due to the symmetry of the problem, there is a zero-flux boundary condition for V on the lower edge of the half-domain, corresponding the x-axis, i.e. when $j = 0$ and M . Hence,

$$V_{i,0} = V_{i,1} \quad \text{and} \quad V_{i,M} = V_{i,M-1}$$

To solve for the individual $V_{i,j}$ using all the above relationships, we must set up a matrix system of the form

$$\mathbf{A}\mathbf{y} = \mathbf{b}$$

where \mathbf{y} is a column vector of length $(N+1)(M+1)$ containing the unknown $V_{i,j}$, \mathbf{A} is an $(N+1)(M+1) \times (N+1)(M+1)$ matrix, and \mathbf{b} is also a column vector of length $(N+1)(M+1)$. To map the $V_{i,j}$ to the elements of \mathbf{y} , we can use:

$$y_{1+j+i(M+1)} = V_{i,j}$$

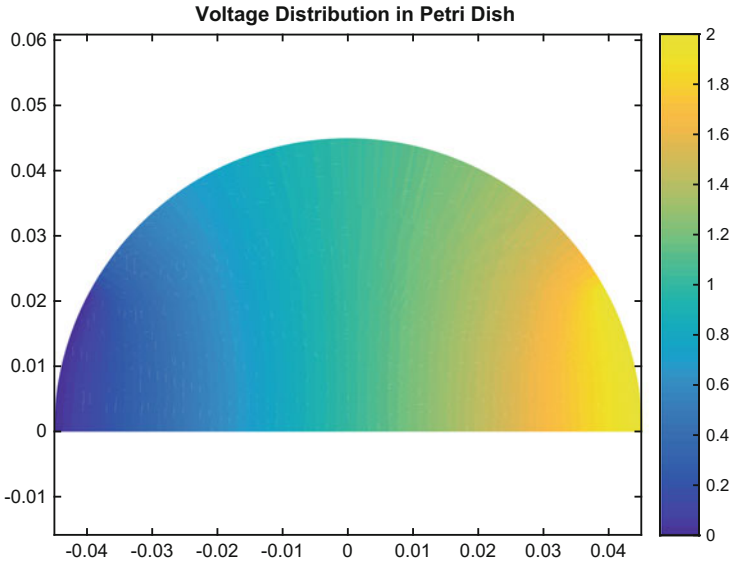
The Matlab code below implements the above equations for $N, M = 100$, solves the resulting matrix system, and plots the voltage distribution in the symmetric half-domain:

```
% solves for the Petri dish voltage distribution using
% finite differences on a polar coordinate grid
R = 0.045; % m
theta_0 = pi/3; % radians
V_0 = 2; % V
N = 100;
M = 100;
```

```

D_r = R/N;
D_theta = pi/M;
B = zeros((N+1)*(M+1),1);
A = sparse((N+1)*(M+1), (N+1)*(M+1));
for i = 0:N
    for j = 0:M
        index = 1+j+i*(M+1);
        if (i == 0) % i.e. at centre of dish
            A(index,index) = M+1;
            A(index,1+(0:M)+M+1) = -1;
        elseif (i == N) % i.e. on circular boundary
            theta = j*D_theta;
            if (theta <= theta_0/2)
                % active electrode
                A(index,index) = 1;
                B(index) = V_0;
            elseif (theta >= pi-theta_0/2)
                % ground electrode
                A(index,index) = 1;
            else
                % zero-flux boundaries
                A(index,index) = 1;
                A(index,1+j+(N-1)*(M+1)) = -1;
            end;
        elseif (j == 0) % i.e. on symmetric boundary
            A(index,index) = 1;
            A(index,1+1+i*(M+1)) = -1;
        elseif (j == M) % i.e. also on symmetric boundary
            A(index,index) = 1;
            A(index,1+M-1+i*(M+1)) = -1;
        else % everywhere else
            A(index,index) = -(2*i^2*D_theta^2+i*D_theta^2+2);
            A(index,1+j+(i+1)*(M+1)) = i^2*D_theta^2+i*D_theta^2;
            A(index,1+j+(i-1)*(M+1)) = i^2*D_theta^2;
            A(index,1+j+1+i*(M+1)) = 1;
            A(index,1+j-1+i*(M+1)) = 1;
        end;
    end;
end;
% solve matrix system
Y = A\B; VV = reshape(Y,N+1,M+1);
% plot solution
[RR, WW] = meshgrid(0:D_r:R, 0:D_theta:pi);
XX = RR.*cos(WW);
YY = RR.*sin(WW);
pcolor(XX,YY,VV), colorbar, axis('equal'),...
    shading('interp'),
    title('Voltage Distribution in Petri Dish');

```

(c) The electric field at all points within the dish is given by

$$\mathbf{E} = -\sigma \nabla V = -\sigma \begin{pmatrix} \frac{\partial V}{\partial x} \\ \frac{\partial V}{\partial y} \end{pmatrix}$$

In particular, the electric field magnitude E is given by

$$E = \sigma \sqrt{\left(\frac{\partial V}{\partial x}\right)^2 + \left(\frac{\partial V}{\partial y}\right)^2}$$

From part (a), these derivatives can be expressed in polar coordinates as

$$\begin{aligned} \frac{\partial V}{\partial x} &= \frac{x}{r} \frac{\partial V}{\partial r} - \frac{y}{r^2} \frac{\partial V}{\partial \theta} = \cos \theta \frac{\partial V}{\partial r} - \frac{\sin \theta}{r} \frac{\partial V}{\partial \theta} \\ \frac{\partial V}{\partial y} &= \frac{y}{r} \frac{\partial V}{\partial r} + \frac{x}{r^2} \frac{\partial V}{\partial \theta} = \sin \theta \frac{\partial V}{\partial r} + \frac{\cos \theta}{r} \frac{\partial V}{\partial \theta} \end{aligned}$$

Therefore,

$$\begin{aligned} E &= \sigma \sqrt{\left(\cos \theta \frac{\partial V}{\partial r} - \frac{\sin \theta}{r} \frac{\partial V}{\partial \theta}\right)^2 + \left(\sin \theta \frac{\partial V}{\partial r} + \frac{\cos \theta}{r} \frac{\partial V}{\partial \theta}\right)^2} \\ &= \sigma \sqrt{\cos^2 \theta \left(\frac{\partial V}{\partial r}\right)^2 - \frac{2 \sin \theta \cos \theta}{r} \left(\frac{\partial V}{\partial r}\right) \left(\frac{\partial V}{\partial \theta}\right) + \frac{\sin^2 \theta}{r^2} \left(\frac{\partial V}{\partial \theta}\right)^2} \\ &\quad + \sin^2 \theta \left(\frac{\partial V}{\partial r}\right)^2 - \frac{2 \sin \theta \cos \theta}{r} \left(\frac{\partial V}{\partial r}\right) \left(\frac{\partial V}{\partial \theta}\right) + \frac{\cos^2 \theta}{r^2} \left(\frac{\partial V}{\partial \theta}\right)^2} \end{aligned}$$

$$= \sigma \sqrt{\left(\frac{\partial V}{\partial r}\right)^2 - \frac{2 \sin 2\theta}{r} \left(\frac{\partial V}{\partial r}\right) \left(\frac{\partial V}{\partial \theta}\right) + \frac{1}{r^2} \left(\frac{\partial V}{\partial \theta}\right)^2}$$

Due to symmetry of the voltage distribution about the x-axis, the gradient of the voltage at the centre of the dish (and hence the electric field) must be parallel to the x-axis itself. Hence, at the centre of the dish, the electric field is given by:

$$\mathbf{E}|_{r=0} = -\sigma \left(\frac{\partial V}{\partial x} \right) \Big|_{x=0, y=0}$$

and its magnitude, E_c , is equal to

$$E_c = \sigma \left| \frac{\partial V}{\partial r} \right|_{r=0, \theta=0}$$

since the x-axis is aligned with the $\theta = 0$ axis at $r = 0$. Hence, the electric field magnitude in the dish relative to the centre is given by

$$\frac{E}{E_c} = \frac{\sqrt{\left(\frac{\partial V}{\partial r}\right)^2 - \frac{2 \sin 2\theta}{r} \left(\frac{\partial V}{\partial r}\right) \left(\frac{\partial V}{\partial \theta}\right) + \frac{1}{r^2} \left(\frac{\partial V}{\partial \theta}\right)^2}}{\left| \frac{\partial V}{\partial r} \right|_{r=0, \theta=0}}$$

Defining a scalar function $\Gamma(r, \theta)$ such that

$$\Gamma(r, \theta) = \begin{cases} 1 & 0.9 \leq \frac{E}{E_c} \leq 1.1 \\ 0 & \text{otherwise} \end{cases}$$

then the area of the dish for which the electric field magnitude is $\pm 10\%$ of E_c is given by

$$\text{Area} = \int_0^\pi \int_0^R \Gamma(r, \theta) r \, dr \, d\theta$$

The Matlab code below approximates this integral by summing the integrand over all elements of the 2D voltage polar-grid array \mathbb{V} , plotting the area against the electrode angle θ_0 :

```
% Determines the area for which electric field
% magnitude is with +/- of its value at the
% the centre of the Petri dish, for a range of
% electrode angles.

R = 0.045; % m
V_0 = 2; % V
N = 100;
M = 100;
D_r = R/N;
```

```

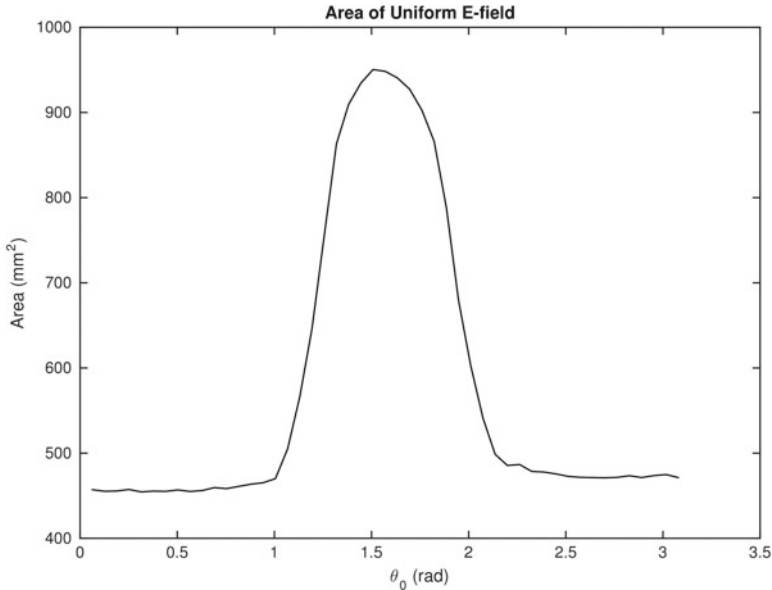
D_theta = pi/M;
Area_array = [];
for theta_0 = pi/50:pi/50:49*pi/50
    B = zeros((N+1)*(M+1),1);
    A = sparse((N+1)*(M+1), (N+1)*(M+1));
    for i = 0:N
        for j = 0:M
            index = 1+j+i*(M+1);
            if (i == 0) % i.e. at centre of dish
                A(index,index) = M+1;
                A(index,1+(0:M)+M+1) = -1;
            elseif (i == N) % i.e. on circular boundary
                theta = j*D_theta;
                if (theta <= theta_0/2)
                    % active electrode
                    A(index,index) = 1;
                    B(index) = V_0;
                elseif (theta >= pi-theta_0/2)
                    % ground electrode
                    A(index,index) = 1;
                else
                    % zero-flux boundaries
                    A(index,index) = 1;
                    A(index,1+j+(N-1)*(M+1)) = -1;
                end;
            elseif (j == 0) % i.e. on symmetric boundary
                A(index,index) = 1;
                A(index,1+1+i*(M+1)) = -1;
            elseif (j == M) % i.e. also on symmetric boundary
                A(index,index) = 1;
                A(index,1+M-1+i*(M+1)) = -1;
            else
                % everywhere else
                A(index,index) = -(2*i^2*D_theta^2+i*D_theta^2+2);
                A(index,1+j+(i+1)*(M+1)) = i^2*D_theta^2+i*D_theta^2;
                A(index,1+j+(i-1)*(M+1)) = i^2*D_theta^2;
                A(index,1+j+1+i*(M+1)) = 1;
                A(index,1+j-1+i*(M+1)) = 1;
            end;
        end;
    end;
end;
% solve matrix system
Y = A\B;
VV = reshape(Y,N+1,M+1)';
% Determine E-field magnitude at centre of disk using
% finite difference approximation (normalised to sigma = 1)
Ec = abs(VV(2,1)-VV(1,1))/D_r;
% Determine Gamma coefficient
E_ratio = ones(N+1,M+1);
Gamma = zeros(N+1,M+1);
for i = 2:N+1
    r = (i-1)*D_r;
    for j = 1:M+1
        theta = (j-1)*D_theta;
        dVdr = (VV(i,j)-VV(i-1,j))/D_r;
        if (j==1)
            dVdtheta = 0;
        else
            dVdtheta = (VV(i,j)-VV(i,j-1))/D_theta;
        end;
    end;
end;

```

```

E_ratio(i,j) = sqrt(dVdr^2-2*sin(2*theta)/r*dVdr*dVdtheta + ...
    dVdtheta^2/r^2)/Ec;
if ((E_ratio(i,j)>=0.9)&&(E_ratio(i,j)<=1.1))
    Gamma(i,j) = 1;
end;
end;
end;
% determine area of uniform E-field
Area = 0;
for i = 1:N+1
    r = i*D_r;
    for j = 1:M+1
        Area = Area + Gamma(i,j)*r*D_r*D_theta;
    end;
end;
Area_array = [Area_array, Area];
end;
% plot solution
plot(pi/50:pi/50:49*pi/50, Area_array*1e6, 'k'), xlabel('\theta_0 (rad)'),
... ylabel('Area (mm^2)'), title('Area of Uniform E-field');

```



From the plot produced by this code (shown above), a maximum uniform electric field area of approximately 950 mm^2 is attained for an electrode angle of around $24\pi/50$ radians, corresponding to $\theta_0 \approx 86^\circ$.

Problems of Chap. 5

5.1 To solve the PDE, we can make use of the system matrices of Eqs. 5.23 and 5.24, determined at the local element level, namely:

$$K_{e,ij} = \int_{\Omega} (\kappa \nabla \varphi_i) \cdot (\nabla \varphi_j) \, dV$$

$$f_{e,j} = \int_{\Omega} f \varphi_j \, dV$$

where $i, j = 1, 2$, $\kappa = 1$, and $f = 2$. Using 1D Lagrange shape functions, we have $\varphi_1 = (1 - \xi)$ and $\varphi_2 = \xi$. The (1,1) component of the element stiffness matrix can be determined from

$$K_{e,11} = h \int_0^1 \left(\frac{1}{h}\right) \frac{\partial \varphi_1(\xi)}{\partial \xi} \left(\frac{1}{h}\right) \frac{\partial \varphi_1(\xi)}{\partial \xi} \, d\xi$$

where h denotes the element size, and the various h factors convert the shape function derivatives and integral from the local element to the spatial frame. Hence,

$$K_{e,11} = h \int_0^1 \left(\frac{1}{h}\right) (-1) \left(\frac{1}{h}\right) (-1) \, d\xi$$

$$= \frac{1}{h}$$

Similarly for the other components,

$$K_{e,12} = h \int_0^1 \left(\frac{1}{h}\right) \frac{\partial \varphi_1(\xi)}{\partial \xi} \left(\frac{1}{h}\right) \frac{\partial \varphi_2(\xi)}{\partial \xi} \, d\xi$$

$$= h \int_0^1 \left(\frac{1}{h}\right) (-1) \left(\frac{1}{h}\right) (1) \, d\xi$$

$$= -\frac{1}{h}$$

$$K_{e,21} = h \int_0^1 \left(\frac{1}{h}\right) \frac{\partial \varphi_2(\xi)}{\partial \xi} \left(\frac{1}{h}\right) \frac{\partial \varphi_1(\xi)}{\partial \xi} \, d\xi$$

$$= h \int_0^1 \left(\frac{1}{h}\right) (1) \left(\frac{1}{h}\right) (-1) \, d\xi$$

$$= -\frac{1}{h}$$

$$K_{e,22} = h \int_0^1 \left(\frac{1}{h}\right) \frac{\partial \varphi_2(\xi)}{\partial \xi} \left(\frac{1}{h}\right) \frac{\partial \varphi_2(\xi)}{\partial \xi} \, d\xi$$

$$= h \int_0^1 \left(\frac{1}{h}\right) (1) \left(\frac{1}{h}\right) (1) \, d\xi$$

$$= \frac{1}{h}$$

For the element load vector calculations, we have

$$\begin{aligned}
 f_{e,1} &= h \int_0^1 2\varphi_1(\xi) \, d\xi \\
 &= h \int_0^1 2(1-\xi) \, d\xi \\
 &= h [2\xi - \xi^2]_0^1 \\
 &= h \\
 f_{e,2} &= h \int_0^1 2\varphi_2(\xi) \, d\xi \\
 &= h \int_0^1 2\xi \, d\xi \\
 &= h [\xi^2]_0^1 \\
 &= h
 \end{aligned}$$

Hence, the element stiffness matrix and load vector are

$$\mathbf{K}_e = \frac{1}{h} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \quad \mathbf{f}_e = h \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Assembling the individual matrices into the global system matrices, noting that $h = 0.25$ and that matrix components are added wherever the element matrices overlap, we obtain:

$$\mathbf{K} = 4 \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix} \quad \mathbf{f} = 0.25 \begin{bmatrix} 1 \\ 2 \\ 2 \\ 2 \\ 1 \end{bmatrix}$$

The matrix system may then be written as:

$$\begin{bmatrix} 4 & -4 & 0 & 0 & 0 \\ -4 & 8 & -4 & 0 & 0 \\ 0 & -4 & 8 & -4 & 0 \\ 0 & 0 & -4 & 8 & -4 \\ 0 & 0 & 0 & -4 & 4 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \end{bmatrix} = \begin{bmatrix} 0.25 \\ 0.5 \\ 0.5 \\ 0.5 \\ 0.25 \end{bmatrix}$$

To enforce the Dirichlet boundary conditions $u_1 = 1$ and $u_5 = -1$, we add two rows to the load vector and two additional rows and columns to the stiffness matrix, corresponding to two dummy Lagrange multiplier variables λ_1 and λ_2 , to obtain the full matrix system as follows:

$$\begin{bmatrix} 4 & -4 & 0 & 0 & 0 & 1 & 0 \\ -4 & 8 & -4 & 0 & 0 & 0 & 0 \\ 0 & -4 & 8 & -4 & 0 & 0 & 0 \\ 0 & 0 & -4 & 8 & -4 & 0 & 0 \\ 0 & 0 & 0 & -4 & 4 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ \lambda_1 \\ \lambda_2 \end{bmatrix} = \begin{bmatrix} 0.25 \\ 0.5 \\ 0.5 \\ 0.5 \\ 0.25 \\ 1 \\ -1 \end{bmatrix}$$

where we have preserved symmetry in the square matrix by appending symmetric columns to its end. Inverting this matrix system in Matlab, we obtain (correct to 4 decimal places) $u_1 = 1.0000$, $u_2 = 0.6875$, $u_3 = 0.2500$, $u_4 = -0.3125$, $u_5 = -1.0000$, $\lambda_1 = -1$, $\lambda_2 = 3$.

To obtain the exact solution to the PDE, we rewrite it as

$$-\frac{\partial^2 u}{\partial x^2} = 2$$

Integrating twice, we obtain:

$$u = -x^2 + c_1 x + c_2$$

where c_1 , c_2 are constants of integration. Their value can be determined from the boundary conditions, namely

$$\begin{aligned} u(0) &= 1 = c_2 \\ u(1) &= -1 = -1^2 + c_1 + c_2 \end{aligned} \quad (.3)$$

yielding $c_1 = -1$, $c_2 = 1$. Hence the exact PDE solution is

$$u = -x^2 - x + 1$$

Evaluating this solution at the node positions $x = 0, 0.25, 0.5, 0.75$ and 1 , we obtain: $u(0) = 1$, $u(0.25) = 0.6875$, $u(0.5) = 0.25$, $u(0.75) = -0.3125$, and $u(1) = -1$. These values correspond to the nodal values u_1, u_2, u_3, u_4 , and u_5 obtained by solving the matrix system earlier. Therefore, the FEM solution agrees with the exact solution.

5.2 To solve this PDE using FEM, we begin by first formulating the strong PDE form of the problem into its equivalent weak form. Rewriting the PDE as

$$\begin{aligned} -\frac{\partial^2 u}{\partial x^2} &= x \\ u(0) &= 1 \\ \frac{\partial u}{\partial x}(1) &= -1 \end{aligned}$$

we first multiply by our test function u_{test} , where we impose $u_{test}(0) = 0$, namely, at the Dirichlet boundary.¹⁰ Integrating across the domain, we obtain:

$$\int_0^1 -u_{test} \frac{\partial^2 u}{\partial x^2} dx = \int_0^1 x u_{test} dx$$

Integrating the left-hand side by parts, yields:

$$\begin{aligned} \left[-u_{test} \frac{\partial u}{\partial x} \right]_0^1 - \int_0^1 -\frac{\partial u_{test}}{\partial x} \frac{\partial u}{\partial x} dx &= \int_0^1 x u_{test} dx \\ -u_{test}(1) \frac{\partial u}{\partial x}(1) + u_{test}(0) \frac{\partial u}{\partial x}(0) - \int_0^1 -\frac{\partial u_{test}}{\partial x} \frac{\partial u}{\partial x} dx &= \int_0^1 x u_{test} dx \\ u_{test}(1) + \int_0^1 \frac{\partial u_{test}}{\partial x} \frac{\partial u}{\partial x} dx &= \int_0^1 x u_{test} dx \end{aligned}$$

where we have used $\frac{\partial u}{\partial x}(1) = -1$ and $u_{test}(0) = 0$. Utilising our 1D Lagrange basis functions $\varphi_i(x)$, we employ Galerkin's method to substitute $u_{test} = \varphi_j$ and $u = \sum_i u_i \varphi_i$, to obtain:

$$\begin{aligned} \varphi_j(1) + \sum_{i=1}^N \int_0^1 \frac{\partial \varphi_j}{\partial x} \frac{\partial \varphi_i}{\partial x} dx &= \int_0^1 x \varphi_j dx \\ \sum_{i=1}^N \int_0^1 \frac{\partial \varphi_j}{\partial x} \frac{\partial \varphi_i}{\partial x} dx &= -\varphi_j(1) + \int_0^1 x \varphi_j dx \end{aligned}$$

For our 1D Lagrange functions, $\varphi_j(1)$ will equal 1 only for $j = N$, where N is the number of basis functions (and global nodes). For all other values of j , it will be 0. The above is equivalent to the following matrix system:

$$\begin{aligned} \mathbf{Ku} &= \mathbf{f} \\ K_{ij} &= \int_0^1 \frac{\partial \varphi_j}{\partial x} \frac{\partial \varphi_i}{\partial x} dx \\ f_i &= -\delta_i^N + \int_0^1 x \varphi_i dx \end{aligned}$$

where δ_i^N is the *Kronecker delta*, defined by

$$\delta_i^j = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

¹⁰Later, the actual Dirichlet condition $x(0) = 1$ will be enforced through the method of Lagrange multipliers, but for now, this constraint on u_{test} will allow us to conveniently obtain the PDE weak form.

To evaluate these components, it is convenient to work with local element versions of these matrices, namely:

$$\begin{aligned} K_{e,ij} &= h \int_0^1 \left(\frac{1}{h}\right) \frac{\partial \varphi_j}{\partial \xi} \left(\frac{1}{h}\right) \frac{\partial \varphi_i}{\partial \xi} d\xi \\ &= \frac{1}{h} \int_0^1 \frac{\partial \varphi_j}{\partial \xi} \frac{\partial \varphi_i}{\partial \xi} d\xi \\ f_{e,i} &= -\delta_x(1) + h \int_0^1 x \phi_i d\xi \end{aligned}$$

where $i, j = 1, 2$, h is the element size, and $\delta_x(1) = 1$ if $x = 1$, 0 otherwise. Using the 1D Lagrange shape functions, $\varphi_1 = (1 - \xi)$ and $\varphi_2 = \xi$, the four components of the element stiffness matrix can be determined using:

$$\begin{aligned} K_{e,11} &= \frac{1}{h} \int_0^1 \frac{\partial \varphi_1(\xi)}{\partial \xi} \frac{\partial \varphi_1(\xi)}{\partial \xi} d\xi \\ &= \frac{1}{h} \int_0^1 (-1)(-1) d\xi \\ &= \frac{1}{h} \\ K_{e,12} &= \frac{1}{h} \int_0^1 \frac{\partial \varphi_1(\xi)}{\partial \xi} \frac{\partial \varphi_2(\xi)}{\partial \xi} d\xi \\ &= \frac{1}{h} \int_0^1 (-1)(1) d\xi \\ &= -\frac{1}{h} \\ K_{e,21} &= \frac{1}{h} \int_0^1 \frac{\partial \varphi_2(\xi)}{\partial \xi} \frac{\partial \varphi_1(\xi)}{\partial \xi} d\xi \\ &= \frac{1}{h} \int_0^1 (1)(-1) d\xi \\ &= -\frac{1}{h} \\ K_{e,22} &= \frac{1}{h} \int_0^1 \frac{\partial \varphi_2(\xi)}{\partial \xi} \frac{\partial \varphi_2(\xi)}{\partial \xi} d\xi \\ &= \frac{1}{h} \int_0^1 (1)(1) d\xi \\ &= \frac{1}{h} \end{aligned}$$

For the element load vector, we must evaluate the integral of $x\varphi_j$ with respect to the local element coordinate ξ . To do this, we first express x in local element coordinates.

Since the element is isoparametric, the value of x within the element is simply the weighted sum of its shape functions, namely:

$$\begin{aligned} x &= x_1\varphi_1(\xi) + x_2\varphi_2(\xi) \\ &= x_1(1 - \xi) + x_2\xi \end{aligned}$$

where x_1, x_2 are the values of x at nodes 1 and 2 of the element. The components of the load vector are therefore

$$\begin{aligned} f_{e,1} &= -\delta_x(1) + h \int_0^1 x\varphi_1(\xi) \, d\xi \\ &= -\delta_x(1) + h \int_0^1 [x_1(1 - \xi) + x_2\xi] (1 - \xi) \, d\xi \\ &= -\delta_x(1) + h \int_0^1 [x_1(1 - \xi)^2 + x_2\xi(1 - \xi)] \, d\xi \\ &= -\delta_x(1) + h \left[-\frac{x_1(1 - \xi)^3}{3} + x_2 \left(\frac{\xi^2}{2} - \frac{\xi^3}{3} \right) \right]_0^1 \\ &= -\delta_x(1) + h \left[\frac{1}{3}x_1 + \frac{1}{6}x_2 \right] \\ f_{e,2} &= -\delta_x(1) + h \int_0^1 x\varphi_2(\xi) \, d\xi \\ &= -\delta_x(1) + h \int_0^1 [x_1(1 - \xi) + x_2\xi] \xi \, d\xi \\ &= -\delta_x(1) + h \int_0^1 [x_1(1 - \xi)\xi + x_2\xi^2] \, d\xi \\ &= -\delta_x(1) + h \left[x_1 \left(\frac{\xi^2}{2} - \frac{\xi^3}{3} \right) + x_2 \frac{\xi^3}{3} \right]_0^1 \\ &= -\delta_x(1) + h \left[\frac{1}{6}x_1 + \frac{1}{3}x_2 \right] \end{aligned}$$

Using $h = 0.25$, we can write the stiffness matrix of each element as

$$\mathbf{K}_e = \frac{1}{h} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 4 & -4 \\ -4 & 4 \end{bmatrix}$$

Furthermore, the element load vectors \mathbf{f}_e are given by

$$\begin{aligned} \mathbf{f}_e &= -\delta_x(1) + \frac{h}{6} \begin{bmatrix} 2x_1 + x_2 \\ x_1 + 2x_2 \end{bmatrix} \\ &= -\delta_x(1) + \frac{1}{24} \begin{bmatrix} 2x_1 + x_2 \\ x_1 + 2x_2 \end{bmatrix} \end{aligned}$$

These element load vectors will be different for each element, and are given as follows:

$$\text{Element 1 : } x_1 = 0, x_2 = 0.25 \quad \mathbf{f}_e = \frac{1}{24} \begin{bmatrix} 0.25 \\ 0.5 \end{bmatrix}$$

$$\text{Element 2 : } x_1 = 0.25, x_2 = 0.5 \quad \mathbf{f}_e = \frac{1}{24} \begin{bmatrix} 1 \\ 1.25 \end{bmatrix}$$

$$\text{Element 3 : } x_1 = 0.5, x_2 = 0.75 \quad \mathbf{f}_e = \frac{1}{24} \begin{bmatrix} 1.75 \\ 2 \end{bmatrix}$$

$$\text{Element 4 : } x_1 = 0.75, x_2 = 1 \quad \mathbf{f}_e = \begin{bmatrix} 0 \\ -1 \end{bmatrix} + \frac{1}{24} \begin{bmatrix} 2.5 \\ 2.75 \end{bmatrix}$$

Assembling the individual element matrices into the global system matrices, noting that matrix components are added wherever the element matrices overlap, we obtain:

$$\mathbf{K} = \begin{bmatrix} 4 & -4 & 0 & 0 & 0 \\ -4 & 8 & -4 & 0 & 0 \\ 0 & -4 & 8 & -4 & 0 \\ 0 & 0 & -4 & 8 & -4 \\ 0 & 0 & 0 & -4 & 4 \end{bmatrix} \quad \mathbf{f} = \frac{1}{24} \begin{bmatrix} 0.25 \\ 1.5 \\ 3 \\ 4.5 \\ -21.25 \end{bmatrix}$$

The matrix system may then be written as:

$$\begin{bmatrix} 4 & -4 & 0 & 0 & 0 \\ -4 & 8 & -4 & 0 & 0 \\ 0 & -4 & 8 & -4 & 0 \\ 0 & 0 & -4 & 8 & -4 \\ 0 & 0 & 0 & -4 & 4 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \end{bmatrix} = \begin{bmatrix} 0.0104 \\ 0.0625 \\ 0.1250 \\ 0.1875 \\ -0.8854 \end{bmatrix}$$

To enforce the Dirichlet boundary condition at $x = 0$, namely $u_1 = 1$, we add a row to the load vector and one additional row and column to the stiffness matrix, corresponding to a dummy Lagrange multiplier variable λ_1 , to obtain the full matrix system as follows:

$$\begin{bmatrix} 4 & -4 & 0 & 0 & 0 & 1 \\ -4 & 8 & -4 & 0 & 0 & 0 \\ 0 & -4 & 8 & -4 & 0 & 0 \\ 0 & 0 & -4 & 8 & -4 & 0 \\ 0 & 0 & 0 & -4 & 4 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ \lambda_1 \end{bmatrix} = \begin{bmatrix} 0.0104 \\ 0.0625 \\ 0.1250 \\ 0.1875 \\ -0.8854 \\ 1 \end{bmatrix}$$

Inverting this matrix system in Matlab, we obtain (correct to 4 decimal places) $u_1 = 1.0000$, $u_2 = 0.8724$, $u_3 = 0.7292$, $u_4 = 0.5547$, $u_5 = 0.3333$, $\lambda_1 = -0.5000$.

To obtain the exact solution to the PDE, we have

$$-\frac{\partial^2 u}{\partial x^2} = x$$

Integrating twice, we obtain:

$$\begin{aligned}\frac{\partial u}{\partial x} &= -\frac{x^2}{2} + c_1 \\ u &= -\frac{x^3}{6} + c_1 x + c_2\end{aligned}$$

where c_1, c_2 are constants of integration, whose value can be determined from the boundary conditions, namely

$$\begin{aligned}u(0) &= 1 = c_2 \\ \frac{\partial u}{\partial x}(1) &= -1 = -\frac{1}{2} + c_1\end{aligned}\tag{.4}$$

yielding $c_1 = -\frac{1}{2}, c_2 = 1$. Hence the exact PDE solution is

$$u = -\frac{x^3}{6} - \frac{x}{2} + 1$$

Evaluating this solution at the node positions $x = 0, 0.25, 0.5, 0.75$ and 1 , we obtain: $u(0) = 1, u(0.25) = 0.8724, u(0.5) = 0.7292, u(0.75) = 0.5547$, and $u(1) = 0.3333$. These values correspond to the nodal values u_1, u_2, u_3, u_4 , and u_5 obtained by solving the matrix system earlier. Therefore, the FEM solution agrees with the exact solution.

5.3 For the cubic Lagrange 1D element, there are four nodes located at $\xi = 0, \xi = 1/3, \xi = 2/3$ and $\xi = 1$. The four cubic Lagrange functions can be determined as follows:

For $\varphi_1(\xi)$, we require $\varphi_1(0) = 1, \varphi_1(1/3) = 0, \varphi_1(2/3) = 0$ and $\varphi_1(1) = 0$. The cubic polynomial must therefore satisfy the following form:

$$\varphi_1(\xi) = c_1 \left(\xi - \frac{1}{3}\right) \left(\xi - \frac{2}{3}\right) (\xi - 1)$$

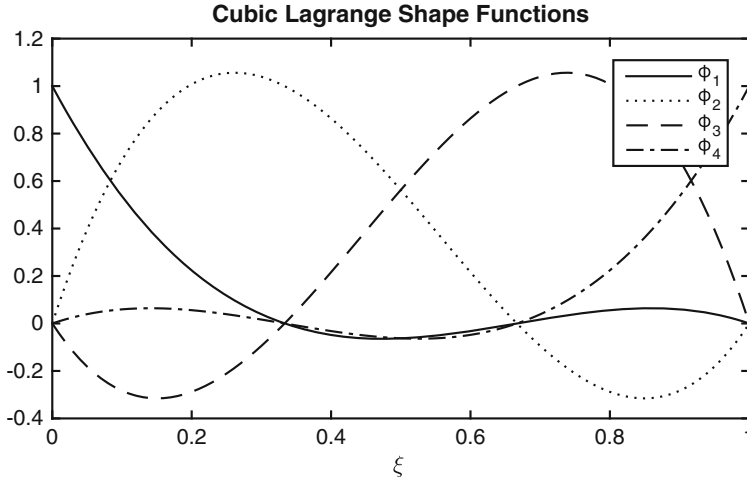
where c_1 is a constant. Its value can be determined from the requirement that $\varphi_1(0) = 1$. Hence, $c_1 = -9/2$, and we obtain

$$\varphi_1(\xi) = \frac{9}{2} \left(\xi - \frac{1}{3}\right) \left(\xi - \frac{2}{3}\right) (1 - \xi)$$

A similar analysis can be applied to the other three shape functions, requiring these to equal 1 at one node and 0 at all others. This yields the remaining shape functions as follows:

$$\begin{aligned} \varphi_2(\xi) &= \frac{27}{2}\xi \left(\xi - \frac{2}{3}\right) (\xi - 1) \\ \varphi_3(\xi) &= \frac{27}{2}\xi \left(\xi - \frac{1}{3}\right) (1 - \xi) \\ \varphi_4(\xi) &= \frac{9}{2}\xi \left(\xi - \frac{1}{3}\right) \left(\xi - \frac{2}{3}\right) \end{aligned}$$

A plot of these shape functions is shown below:



5.4 This diffusion PDE is similar to Example 5.1, in which we computed the global damping, stiffness and load matrices/vectors using Eqs. 5.5–5.7

$$\begin{aligned} D_{ij} &= \int_0^1 \varphi_i(x)\varphi_j(x) \, dx && \text{(damping matrix)} \\ K_{ij} &= \int_0^1 D \frac{d\varphi_i(x)}{dx} \frac{d\varphi_j(x)}{dx} \, dx && \text{(stiffness matrix)} \\ f_i &= q(t)\varphi_i(1) = q(t)\delta_i^N && \text{(load vector)} \end{aligned}$$

where D is the diffusion coefficient ($= 1$ for this problem), δ_i^N is the Kronecker delta, and $q(t) = \left[D \frac{\partial c}{\partial x} \right]_{x=1}$, which represents the specified flux at $x = 1$. As in Example 5.1, since zero-flux boundary conditions have been specified at both ends of the domain, all elements of the load vector will be 0.

To determine the damping and stiffness matrices, is convenient to express these at the local element level, namely:

$$\begin{aligned} D_{e,ij} &= h \int_0^1 \varphi_i(\xi)\varphi_j(\xi) \, d\xi \\ K_{e,ij} &= h \int_0^1 \left(\frac{1}{h}\right) \frac{d\varphi_i(\xi)}{d\xi} \left(\frac{1}{h}\right) \frac{d\varphi_j(\xi)}{d\xi} \, d\xi \\ &= \frac{1}{h} \int_0^1 \frac{d\varphi_i(\xi)}{d\xi} \frac{d\varphi_j(\xi)}{d\xi} \, d\xi \end{aligned}$$

We can then substitute the three quadratic Lagrange shape functions,

$$\varphi_1(\xi) = 2(0.5 - \xi)(1 - \xi)$$

$$\varphi_2(\xi) = 4\xi(1 - \xi)$$

$$\varphi_3(\xi) = 2\xi(\xi - 0.5)$$

into these expressions to determine the above system matrices. For the damping matrix, we have:

$$\begin{aligned} D_{e,11} &= h \int_0^1 \varphi_1(\xi)\varphi_1(\xi) \, d\xi \\ &= 4h \int_0^1 (0.5 - \xi)^2(1 - \xi)^2 \, d\xi \\ &= 4h \int_0^1 (0.25 - \xi + \xi^2)(1 - 2\xi + \xi^2) \, d\xi \\ &= 4h \int_0^1 [0.25 - \xi + \xi^2 - 0.5\xi + 2\xi^2 - 2\xi^3 + 0.25\xi^2 - \xi^3 + \xi^4] \, d\xi \\ &= 4h \int_0^1 [0.25 - 1.5\xi + 3.25\xi^2 - 3\xi^3 + \xi^4] \, d\xi \\ &= 4h \left[0.25\xi - 0.75\xi^2 + \frac{3.25}{3}\xi^3 - 0.75\xi^4 + 0.2\xi^5 \right]_0^1 \\ &= \frac{2h}{15} \end{aligned}$$

$$\begin{aligned} D_{e,12} &= h \int_0^1 \varphi_1(\xi)\varphi_2(\xi) \, d\xi \\ &= 8h \int_0^1 (0.5 - \xi)(1 - \xi)\xi(1 - \xi) \, d\xi \\ &= 8h \int_0^1 (0.5\xi - \xi^2)(1 - 2\xi + \xi^2) \, d\xi \\ &= 8h \int_0^1 [0.5\xi - \xi^2 - \xi^2 + 2\xi^3 + 0.5\xi^3 - \xi^4] \, d\xi \\ &= 8h \int_0^1 [0.5\xi - 2\xi^2 + 2.5\xi^3 - \xi^4] \, d\xi \\ &= 8h \left[0.25\xi^2 - \frac{2}{3}\xi^3 + \frac{2.5}{4}\xi^4 - 0.2\xi^5 \right]_0^1 \\ &= \frac{h}{15} \end{aligned}$$

$$D_{e,13} = h \int_0^1 \varphi_1(\xi)\varphi_3(\xi) \, d\xi$$

$$\begin{aligned}
&= 4h \int_0^1 (0.5 - \xi)(1 - \xi)\xi(\xi - 0.5) \, d\xi \\
&= 4h \int_0^1 (\xi - 0.5)^2 \xi(\xi - 1) \, d\xi \\
&= 4h \int_0^1 (\xi^2 - \xi + 0.25)(\xi^2 - \xi) \, d\xi \\
&= 4h \int_0^1 [\xi^4 - \xi^3 + 0.25\xi^2 - \xi^3 + \xi^2 - 0.25\xi] \, d\xi \\
&= 4h \int_0^1 [\xi^4 - 2\xi^3 + 1.25\xi^2 - 0.25\xi] \, d\xi \\
&= 4h \left[0.2\xi^5 - 0.5\xi^4 + \frac{1.25}{3}\xi^3 - 0.125\xi^2 \right]_0^1 \\
&= -\frac{h}{30} \\
D_{e,22} &= h \int_0^1 \varphi_2(\xi)\varphi_2(\xi) \, d\xi \\
&= 16h \int_0^1 \xi^2(1 - \xi)^2 \, d\xi \\
&= 16h \int_0^1 \xi^2(1 - 2\xi + \xi^2) \, d\xi \\
&= 16h \int_0^1 [\xi^2 - 2\xi^3 + \xi^4] \, d\xi \\
&= 16h \left[\frac{1}{3}\xi^3 - 0.5\xi^4 + 0.2\xi^5 \right]_0^1 \\
&= \frac{8h}{15} \\
D_{e,23} &= h \int_0^1 \varphi_2(\xi)\varphi_3(\xi) \, d\xi \\
&= 8h \int_0^1 \xi(1 - \xi)\xi(\xi - 0.5) \, d\xi \\
&= 8h \int_0^1 \xi^2(-0.5 + 1.5\xi - \xi^2) \, d\xi \\
&= 8h \int_0^1 [-0.5\xi^2 + 1.5\xi^3 - \xi^4] \, d\xi \\
&= 8h \left[-\frac{0.5}{3}\xi^3 + \frac{1.5}{4}\xi^4 - 0.2\xi^5 \right]_0^1 \\
&= \frac{h}{15}
\end{aligned}$$

$$\begin{aligned}
 D_{e,33} &= h \int_0^1 \varphi_3(\xi) \varphi_3(\xi) \, d\xi \\
 &= 4h \int_0^1 \xi^2 (\xi - 0.5)^2 \, d\xi \\
 &= 4h \int_0^1 \xi^2 (\xi^2 - \xi + 0.25) \, d\xi \\
 &= 4h \int_0^1 [\xi^4 - \xi^3 + 0.25\xi^2] \, d\xi \\
 &= 4h \left[0.2\xi^5 - 0.25\xi^4 + \frac{0.25}{3}\xi^3 \right]_0^1 \\
 &= \frac{2h}{15}
 \end{aligned}$$

with remaining terms being symmetric, that is: $D_{e21} = D_{e12}$, $D_{e31} = D_{e13}$, $D_{e32} = D_{e23}$. Hence, the 3×3 element damping matrix is given by:

$$\mathbf{D}_e = \frac{h}{30} \begin{bmatrix} 4 & 2 & -1 \\ 2 & 16 & 2 \\ -1 & 2 & 4 \end{bmatrix}$$

For the element stiffness matrix, we must first calculate the derivatives of our shape functions:

$$\begin{aligned}
 \frac{d\varphi_1(\xi)}{d\xi} &= \frac{d}{d\xi} [2(0.5 - \xi)(1 - \xi)] \\
 &= \frac{d}{d\xi} [2\xi^2 - 3\xi + 1] \\
 &= 4\xi - 3 \\
 \frac{d\varphi_2(\xi)}{d\xi} &= \frac{d}{d\xi} [4\xi(1 - \xi)] \\
 &= \frac{d}{d\xi} [4\xi - 4\xi^2] \\
 &= 4 - 8\xi \\
 \frac{d\varphi_3(\xi)}{d\xi} &= \frac{d}{d\xi} [2\xi(\xi - 0.5)] \\
 &= \frac{d}{d\xi} [2\xi^2 - \xi] \\
 &= 4\xi - 1
 \end{aligned}$$

Hence, the components of the element stiffness matrix can be determined as follows:

$$\begin{aligned}
 K_{e,11} &= \frac{1}{h} \int_0^1 \frac{d\varphi_1(\xi)}{d\xi} \frac{d\varphi_1(\xi)}{d\xi} d\xi \\
 &= \frac{1}{h} \int_0^1 (4\xi - 3)^2 d\xi \\
 &= \frac{1}{h} \int_0^1 (16\xi^2 - 24\xi + 9) d\xi \\
 &= \frac{1}{h} \left[\frac{16}{3} \xi^3 - 12\xi^2 + 9\xi \right]_0^1 \\
 &= \frac{7}{3h} \\
 K_{e,12} &= \frac{1}{h} \int_0^1 \frac{d\varphi_1(\xi)}{d\xi} \frac{d\varphi_2(\xi)}{d\xi} d\xi \\
 &= \frac{1}{h} \int_0^1 (4\xi - 3)(4 - 8\xi) d\xi \\
 &= \frac{1}{h} \int_0^1 (-12 + 40\xi - 32\xi^2) d\xi \\
 &= \frac{1}{h} \left[-12\xi + 20\xi^2 - \frac{32}{3}\xi^3 \right]_0^1 \\
 &= -\frac{8}{3h} \\
 K_{e,13} &= \frac{1}{h} \int_0^1 \frac{d\varphi_1(\xi)}{d\xi} \frac{d\varphi_3(\xi)}{d\xi} d\xi \\
 &= \frac{1}{h} \int_0^1 (4\xi - 3)(4\xi - 1) d\xi \\
 &= \frac{1}{h} \int_0^1 (16\xi^2 - 16\xi + 3) d\xi \\
 &= \frac{1}{h} \left[\frac{16}{3}\xi^3 - 8\xi^2 + 3\xi \right]_0^1 \\
 &= \frac{1}{3h} \\
 K_{e,22} &= \frac{1}{h} \int_0^1 \frac{d\varphi_2(\xi)}{d\xi} \frac{d\varphi_2(\xi)}{d\xi} d\xi \\
 &= \frac{1}{h} \int_0^1 (4 - 8\xi)^2 d\xi \\
 &= \frac{1}{h} \int_0^1 (16 - 64\xi + 64\xi^2) d\xi
 \end{aligned}$$

$$\begin{aligned}
&= \frac{1}{h} \left[16\xi - 32\xi^2 + \frac{64}{3}\xi^3 \right]_0^1 \\
&= \frac{16}{3h} \\
K_{e,23} &= \frac{1}{h} \int_0^1 \frac{d\varphi_2(\xi)}{d\xi} \frac{d\varphi_3(\xi)}{d\xi} d\xi \\
&= \frac{1}{h} \int_0^1 (4 - 8\xi)(4\xi - 1) d\xi \\
&= \frac{1}{h} \int_0^1 (-4 + 24\xi - 32\xi^2) d\xi \\
&= \frac{1}{h} \left[-4\xi + 12\xi^2 - \frac{32}{3}\xi^3 \right]_0^1 \\
&= -\frac{8}{3h} \\
K_{e,33} &= \frac{1}{h} \int_0^1 \frac{d\varphi_3(\xi)}{d\xi} \frac{d\varphi_3(\xi)}{d\xi} d\xi \\
&= \frac{1}{h} \int_0^1 (4\xi - 1)^2 d\xi \\
&= \frac{1}{h} \int_0^1 (16\xi^2 - 8\xi + 1) d\xi \\
&= \frac{1}{h} \left[\frac{16}{3}\xi^3 - 4\xi^2 + \xi \right]_0^1 \\
&= \frac{7}{3h}
\end{aligned}$$

with the remaining components being symmetric. Thus, the element stiffness matrix is given by

$$\mathbf{K}_e = \frac{1}{3h} \begin{bmatrix} 7 & -8 & 1 \\ -8 & 16 & -8 \\ 1 & -8 & 7 \end{bmatrix}$$

Assembling the four individual element matrices into the global system matrices, using $h = 0.25$ and noting that matrix components are added wherever the element matrices overlap, we obtain the 9×9 global system matrices:

$$\mathbf{D} = \frac{1}{120} \begin{bmatrix} 4 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 16 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & 8 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 16 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & 8 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 16 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 2 & 8 & 2 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 16 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & 4 \end{bmatrix} \quad (\text{damping matrix})$$

$$\approx \begin{bmatrix} 3.33 & 1.67 & -0.83 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1.67 & 13.33 & 1.67 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.83 & 1.67 & 6.67 & 1.67 & -0.83 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.67 & 13.33 & 1.67 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.83 & 1.67 & 6.67 & 1.67 & -0.83 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.67 & 13.33 & 1.67 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.83 & 1.67 & 6.67 & 1.67 & -0.83 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1.67 & 13.33 & 1.67 \\ 0 & 0 & 0 & 0 & 0 & 0 & -0.83 & 1.67 & 3.33 \end{bmatrix} \times 10^{-2}$$

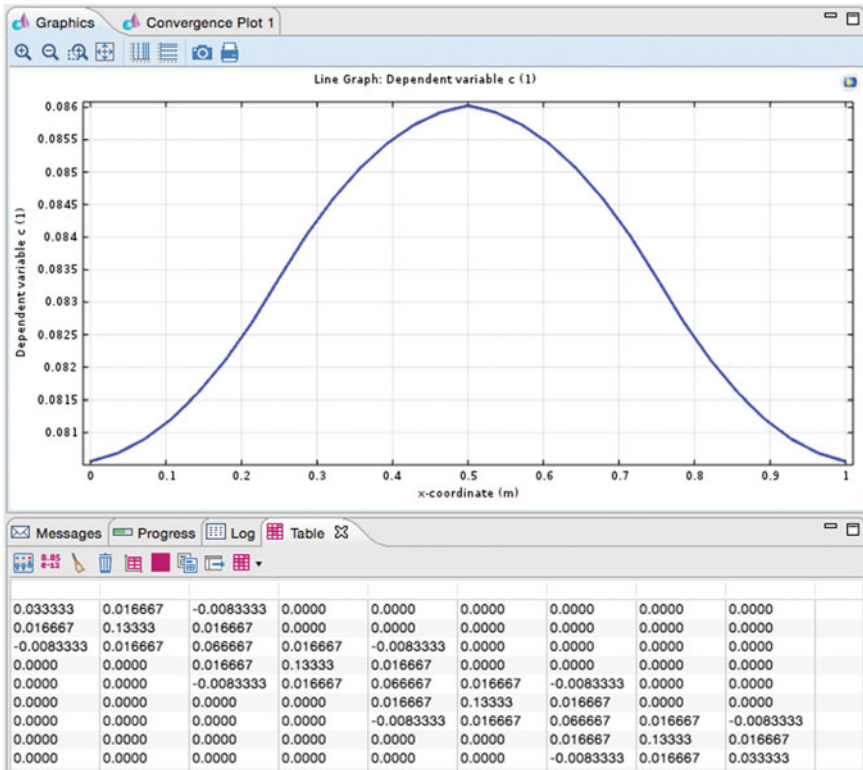
and

$$\mathbf{K} = \frac{4}{3} \begin{bmatrix} 7 & -8 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -8 & 16 & -8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -8 & 14 & -8 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -8 & 16 & -8 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -8 & 14 & -8 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -8 & 16 & -8 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -8 & 14 & -8 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -8 & 16 & -8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -8 & 7 \end{bmatrix} \quad (\text{stiffness matrix})$$

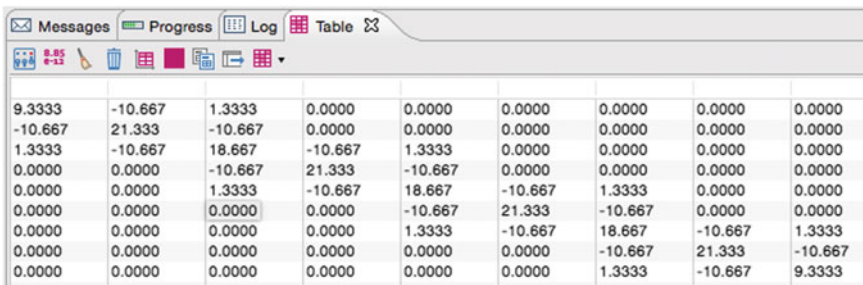
$$\approx \begin{bmatrix} 9.33 & -10.67 & 1.33 & 0 & 0 & 0 & 0 & 0 & 0 \\ -10.67 & 21.33 & -10.67 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1.33 & -10.67 & 18.67 & -10.67 & 1.33 & 0 & 0 & 0 & 0 \\ 0 & 0 & -10.67 & 21.33 & -10.67 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.33 & -10.67 & 18.67 & -10.67 & 1.33 & 0 & 0 \\ 0 & 0 & 0 & 0 & -10.67 & 21.33 & -10.67 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.33 & -10.67 & 18.67 & -10.67 & 1.33 \\ 0 & 0 & 0 & 0 & 0 & 0 & -10.67 & 21.33 & -10.67 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1.33 & -10.67 & 9.33 \end{bmatrix}$$

To solve the PDE in COMSOL and inspect the resulting damping and stiffness matrices, we can implement the same steps as Example 5.1, however this time specifying 4 elements under the mesh distribution settings, and ‘Quadratic’ for the Lagrange element order under the Discretization tab of the General Form PDE node in the model

tree. The solution at $t = 0.1$ is shown plotted below, along with the COMSOL-generated damping matrix.



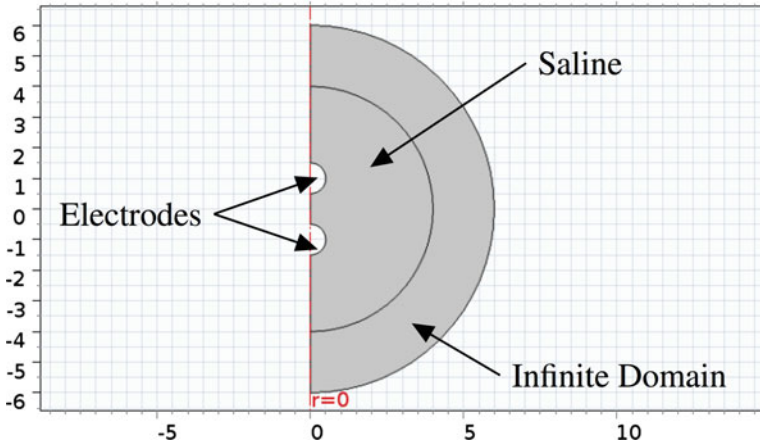
The COMSOL-generated stiffness matrix is shown below:



Both of these COMSOL matrices agree with those we obtained analytically.

Problems of Chap. 6

6.1 To solve this problem in COMSOL, we can utilise the 2D axisymmetric geometry shown below, where all dimensions are in mm. Note that the electrodes are defined as semi-circular boundaries of the saline domain.



Using the electric currents physics mode of the AC/DC module, we set the conductivity of the saline as 1 S m^{-1} , the boundaries one electrode at ground, and the boundaries of the other electrode to a potential of 1 V. The surrounding hemispherical domain is assigned an infinite element with zero-flux boundary. Defining an integration coupling operator along the active electrode boundaries, we can determine the total current i flowing into the saline domain as

$$i = \int_{A_e} 2\pi r J_n \, ds$$

where A_e is the active electrode boundary, s is the arc-length along the boundary, and J_n is the normal component of the inward current density, given by the COMSOL variable $ec.nJ$. The resistance R between the electrodes can then be determined using

$$R = \frac{1}{i}$$

Setting the mesh distribution along the boundaries of each electrode to be 100 elements, COMSOL yields a value of $R = 237.23 \, \Omega$.

6.2 Using Eq. 6.16 for the 2D case, we can determine the conductivity tensor in the tissue slab using

$$\sigma = \sigma_1 \mathbf{n}_1 \mathbf{n}_1^T + \sigma_2 \mathbf{n}_2 \mathbf{n}_2^T$$

where σ_1, σ_2 are the conductivities in the fibre and transverse-fibre directions, and $\mathbf{n}_1, \mathbf{n}_2$ are the corresponding orthogonal unit vectors in the fibre and transverse-fibre directions. For a fibre angle of θ , these directions are given by

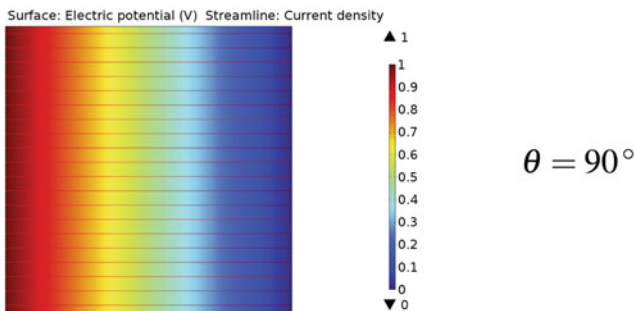
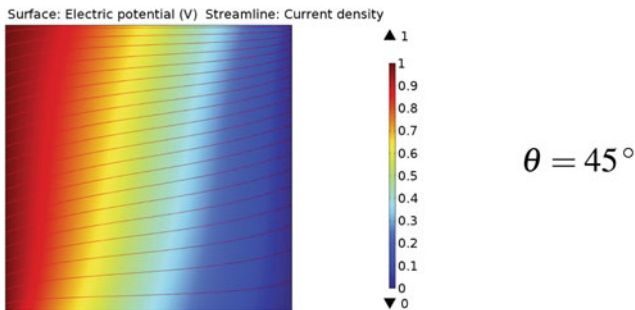
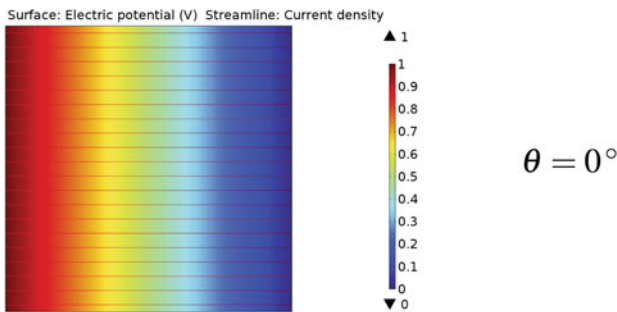
$$\mathbf{n}_1 = \begin{pmatrix} \cos \theta \\ \sin \theta \end{pmatrix} \quad \mathbf{n}_2 = \begin{pmatrix} -\sin \theta \\ \cos \theta \end{pmatrix}$$

with resulting conductivity tensor

$$\sigma = \sigma_1 \begin{pmatrix} \cos^2 \theta & \cos \theta \sin \theta \\ \cos \theta \sin \theta & \sin^2 \theta \end{pmatrix} + \sigma_2 \begin{pmatrix} \sin^2 \theta & -\sin \theta \cos \theta \\ -\sin \theta \cos \theta & \cos^2 \theta \end{pmatrix}$$

$$= \begin{pmatrix} \sigma_1 \cos^2 \theta + \sigma_2 \sin^2 \theta & (\sigma_1 - \sigma_2) \sin \theta \cos \theta \\ (\sigma_1 - \sigma_2) \sin \theta \cos \theta & \sigma_1 \sin^2 \theta + \sigma_2 \cos^2 \theta \end{pmatrix}$$

Using $\sigma_1 = 0.2 \text{ mS cm}^{-1}$, $\sigma_2 = 0.1 \text{ mS cm}^{-1}$, along with values of $\theta = 0^\circ$, 45° , 90° , we obtain the following COMSOL plots of voltage distributions and current streamlines (using the 2D streamline plot type):



6.3 This problem is similar to that of Sect. 6.1.5, with the exception that the electric potential on the electrode disc is replaced with a normal current density boundary

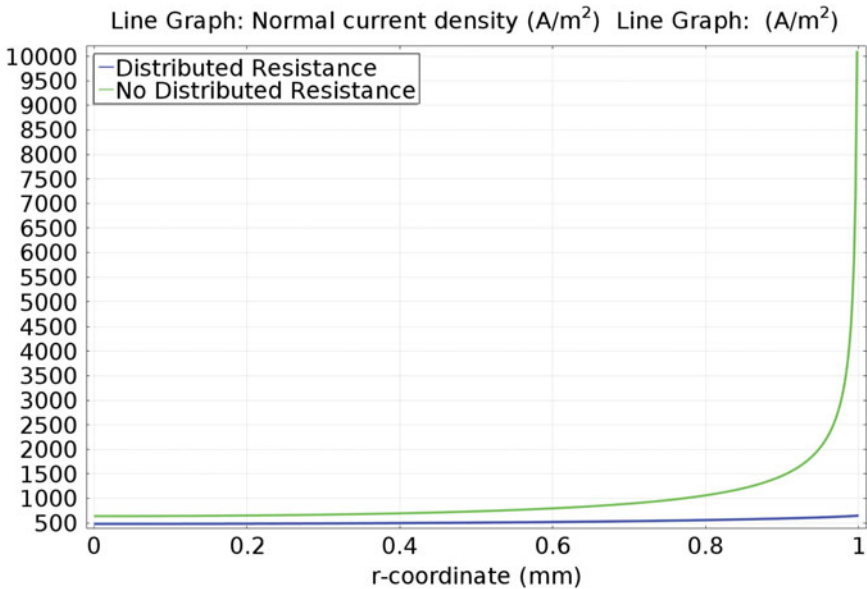
condition, with inward normal current density J_n given by:

$$J_n = \frac{V_s - V}{R}$$

where V_s is the supply voltage (1 V), V is the potential in the saline medium adjacent to the electrode, and R is the distributed resistance ($0.001 \Omega \text{ m}^2$). The COMSOL-generated plot of current density as a function of radial position along the disc electrode is shown below, where the theoretical plot with no distributed resistance has been generated using

$$J_n = \frac{2\sigma V_s}{\pi\sqrt{R_e^2 - r^2}}$$

where r is the radial position along the disc, σ is the conductivity of the saline medium (1 S m^{-1}), V_s is the supply voltage (1 V) and R_e is the electrode radius (1 mm). Note that the effect of the distributed resistance is to smooth out the variations in current density, particularly at the edge of the disc, resulting in a near-constant current across the disc electrode.



Problems of Chap. 7

7.1 At steady-state, $\partial c / \partial t = 0$ and the PDE reduces to the ODE

$$D \frac{d^2 c}{dr^2} - k_{up} c = 0$$

which can be solved for using the methods of Chap. 2. The characteristic equation of this ODE is

$$Dm^2 - k_{up} = 0$$

$$\therefore m = \pm \sqrt{\frac{k_{up}}{D}}$$

Hence the solution is of the form

$$c(x) = C_1 e^{x\sqrt{\frac{k_{up}}{D}}} + C_2 e^{-x\sqrt{\frac{k_{up}}{D}}}$$

where C_1, C_2 are constants which can be determined from the boundary conditions. Specifically, when $x = 0, c = C_0$. Hence

$$C_0 = C_1 + C_2$$

Also, when $x = d_c, c = 0$. Hence

$$0 = C_1 e^{d_c \sqrt{\frac{k_{up}}{D}}} + C_2 e^{-d_c \sqrt{\frac{k_{up}}{D}}}$$

Substituting $C_2 = C_0 - C_1$ into the above, we have

$$C_1 e^{d_c \sqrt{\frac{k_{up}}{D}}} + (C_0 - C_1) e^{-d_c \sqrt{\frac{k_{up}}{D}}} = 0$$

$$C_1 e^{d_c \sqrt{\frac{k_{up}}{D}}} - C_1 e^{-d_c \sqrt{\frac{k_{up}}{D}}} = -C_0 e^{-d_c \sqrt{\frac{k_{up}}{D}}}$$

$$C_1 \left[e^{d_c \sqrt{\frac{k_{up}}{D}}} - e^{-d_c \sqrt{\frac{k_{up}}{D}}} \right] = -C_0 e^{-d_c \sqrt{\frac{k_{up}}{D}}}$$

$$\therefore C_1 = \frac{C_0 e^{-d_c \sqrt{\frac{k_{up}}{D}}}}{e^{-d_c \sqrt{\frac{k_{up}}{D}}} - e^{d_c \sqrt{\frac{k_{up}}{D}}}}$$

and using $C_2 = C_0 - C_1$, we also obtain

$$C_2 = C_0 - \frac{C_0 e^{-d_c \sqrt{\frac{k_{up}}{D}}}}{e^{-d_c \sqrt{\frac{k_{up}}{D}}} - e^{d_c \sqrt{\frac{k_{up}}{D}}}}$$

$$= C_0 \left[\frac{e^{-d_c \sqrt{\frac{k_{up}}{D}}} - e^{d_c \sqrt{\frac{k_{up}}{D}}}}{e^{-d_c \sqrt{\frac{k_{up}}{D}}} - e^{d_c \sqrt{\frac{k_{up}}{D}}}} \right] - \frac{C_0 e^{-d_c \sqrt{\frac{k_{up}}{D}}}}{e^{-d_c \sqrt{\frac{k_{up}}{D}}} - e^{d_c \sqrt{\frac{k_{up}}{D}}}}$$

$$= - \frac{C_0 e^{d_c \sqrt{\frac{k_{up}}{D}}}}{e^{-d_c \sqrt{\frac{k_{up}}{D}}} - e^{d_c \sqrt{\frac{k_{up}}{D}}}}$$

Substituting these values of C_1, C_2 into the general solution form, we obtain

$$c(x) = \frac{C_1}{\left[\frac{C_0 e^{-d_c \sqrt{\frac{k_{up}}{D}}}}{e^{-d_c \sqrt{\frac{k_{up}}{D}}} - e^{d_c \sqrt{\frac{k_{up}}{D}}}} \right]} e^{x \sqrt{\frac{k_{up}}{D}}} + \frac{C_2}{\left[\frac{-C_0 e^{d_c \sqrt{\frac{k_{up}}{D}}}}{e^{-d_c \sqrt{\frac{k_{up}}{D}}} - e^{d_c \sqrt{\frac{k_{up}}{D}}}} \right]} e^{-x \sqrt{\frac{k_{up}}{D}}}$$

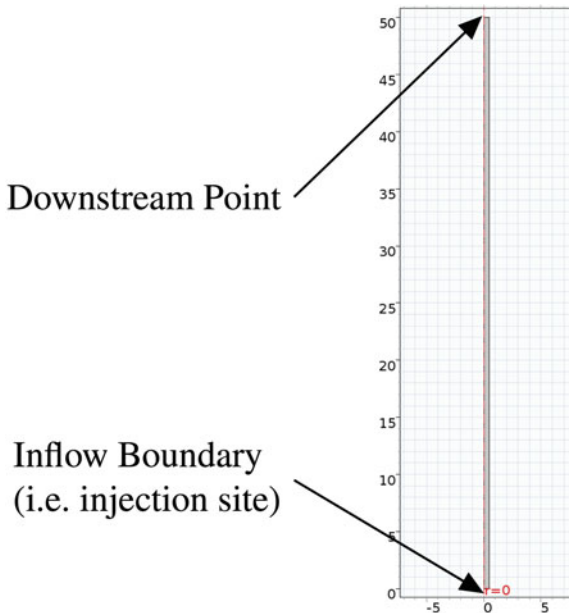
$$= \frac{C_0}{e^{-d_c \sqrt{\frac{k_{up}}{D}}} - e^{d_c \sqrt{\frac{k_{up}}{D}}}} \left[e^{(x-d_c) \sqrt{\frac{k_{up}}{D}}} - e^{-(x-d_c) \sqrt{\frac{k_{up}}{D}}} \right]$$

which may also be written as

$$c(x) = C_0 \left[\frac{\sinh \left((x - d_c) \sqrt{\frac{k_{up}}{D}} \right)}{\sinh \left(-d_c \sqrt{\frac{k_{up}}{D}} \right)} \right]$$

where sinh is the hyperbolic sine function.

7.2 To simulate this system in COMSOL, we can utilise the 2D axisymmetric geometry shown below (all dimensions are in cm), corresponding to a single rectangular domain with axisymmetric axis coinciding with the left long edge of the rectangle. Note that the lower boundary is the site of indicator injection, and the upper left corner of the rectangular domain is the downstream site of concentration measurement.



We can also use two physics to implement the model: (1) Transport of Diluted Species and (2) Global ODEs and DAEs, and define the following parameters: Q (volume

flow rate), R (radius of vessel), and $M0$ (total amount of indicator injected). Under the Transport of Diluted Species node, the diffusion coefficient of the indicator species is set to a user-defined value $1 \times 10^{-9} \text{ m}^2 \text{ s}^{-1}$. The components of the velocity field are also specified as a function of the total volumetric flow, Q , to correspond to a parabolic velocity profile. To determine this velocity field, we have from Eq. 7.7:

$$u = \frac{u_{\max}}{R^2} (R^2 - r^2)$$

where u_{\max} is the maximum velocity at the axis of the vessel, R is its radius, and r is the radial coordinate. The total flow through the vessel is then determined by:

$$\begin{aligned} Q &= \int_0^R 2\pi r u \, dr \\ &= \frac{2\pi u_{\max}}{R^2} \int_0^R r (R^2 - r^2) \\ &= \frac{2\pi u_{\max}}{R^2} \left[\frac{r^2 R^2}{2} - \frac{r^4}{4} \right]_0^R \\ &= \frac{2\pi u_{\max}}{R^2} \left[\frac{R^4}{2} - \frac{R^4}{4} \right] \\ &= \frac{\pi R^2 u_{\max}}{2} \end{aligned}$$

Hence,

$$u_{\max} = \frac{2Q}{\pi R^2}$$

Substituting this expression for u_{\max} into the expression for the parabolic velocity profile, we obtain

$$u = \frac{2Q}{\pi R^4} (R^2 - r^2)$$

This expression for u is entered into the velocity field for the z-component, and a value of 0 entered for the r-component. We also enter a flux boundary condition on the lower boundary (i.e. the site of injection), with flux given by

$$M0 / (\pi * R^2 * (100 \text{ [ms]})) * \text{rect1}(t \text{ [1/s]})$$

where `rect1` is a user-defined rectangular function having lower limit of 0.005, upper limit 0.105, and smoothing factor 0.01. This defines a rectangular pulse of duration 0.1 that begins at $t = 0$, taking the smooth onset of the pulse into account. We also specify an outflow boundary condition for the upper boundary. Finally for this physics node, we specify 'Isotropic diffusion' under the Inconsistent stabilization tab (make sure the Stabilization option is checked under the view menu).

Under the Component definitions, we define a point integration operator (`intop1`) for the upper left-hand corner of the rectangular domain. In the variables sub-node,

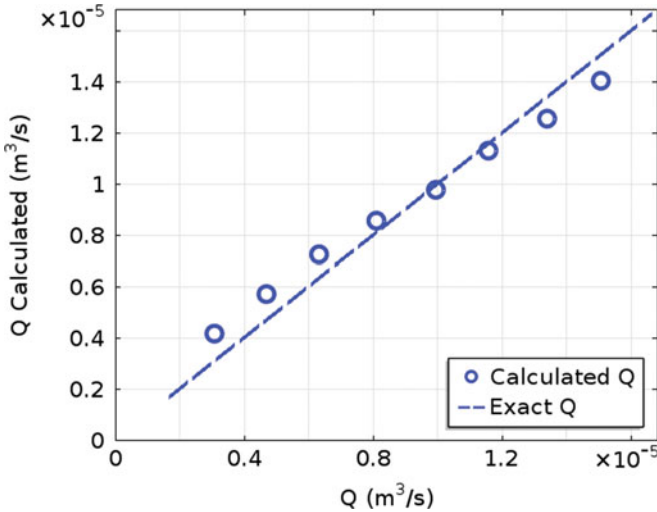
we can then define a global variable c_d with expression $\text{intop1}(c)$ to define the indicator concentration at the downstream site. Under the Global ODEs and DAEs physics node, we can then specify a global ODE variable c_int satisfying the equation

$$c_intt - c_d = 0$$

which states that the time-derivative of c_int equals c_d . This is equivalent to the integral

$$c_int = \int_0^t c_d dt$$

We can then define a variable c_id with expression equal to M_0/c_int . Finally, we mesh the model geometry using the mapped mesh option consisting of 5×500 quadrilateral elements over the rectangular domain. For the time-dependent solver, specify the times from 0 to 20 s in steps of 0.1 s, using strict time stepping. Performing a parameter sweep on parameter Q generates the following result for Q_id , evaluated at the final output time of $t = 20$ s.

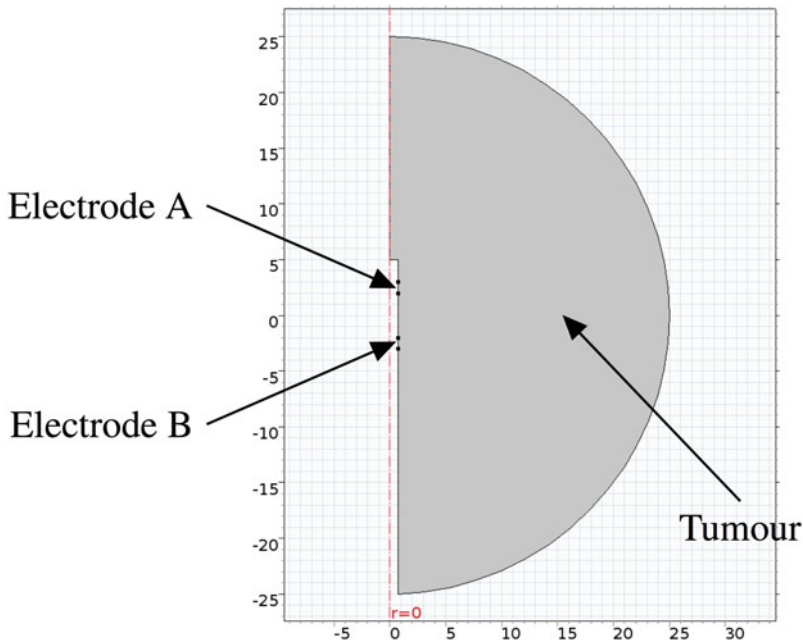


This result verifies that the method of indicator dilution is able to approximate flow rate from the expression $Q = M_0 [\int_0^\infty c dt]^{-1}$.

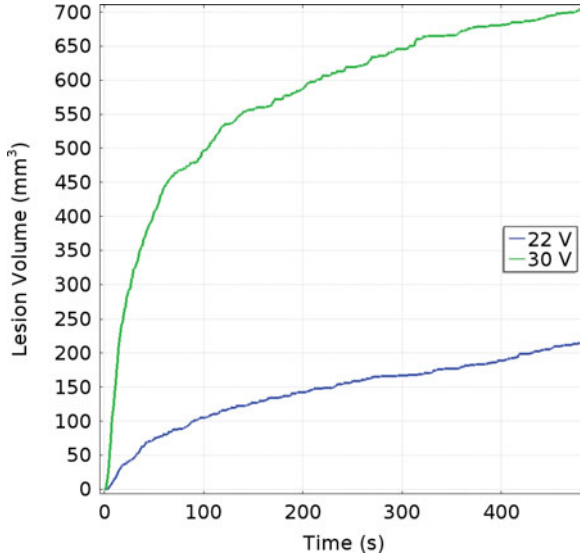
7.3 To simulate this model in COMSOL, we utilise a 2D axisymmetric geometry as shown below (all dimensions are in cm), and then follow a similar implementation to that of Example 7.2.3, using the three physics nodes (1) Electric Currents, (2) Heat Transfer in Solids and (3) the General Form PDE, with the following boundary conditions:

- An electric ground for the boundary of Electrode B

- An electric potential condition for Electrode boundary A (22 and 30 V, implemented as a parametric sweep on a defined applied voltage parameter).
- Electric insulation on all other boundaries (except the axisymmetric axis, which employs a similar axisymmetric condition).
- A temperature boundary condition (37 °C for the outer boundaries of the tumour).
- Thermal insulation on all other boundaries.



For the mesh, we set the general element size as 'Extra fine', and for the time-dependent solver, we use output times from 0 to 480 s in steps of 1 s. Defining variables and integration operators as in Example 7.2.3, we obtain the following result for the lesion volume for both 22 and 30 V probe voltages.



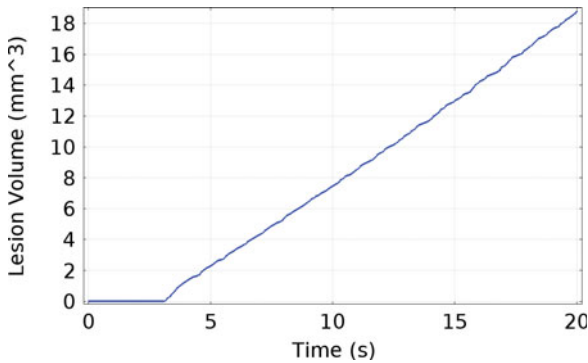
7.4 To determine the negative complex part of the permittivity of heart tissue, we can employ parameter definitions within COMSOL to enter the permittivity parameters of Eq. 7.19 and undertake the necessary complex number calculations, as shown below:

| Name | Expression | Value |
|---------------------|--|---------------------|
| e_inf | 4 | 4 |
| delta_e1 | 50 | 50 |
| tau_1 | 7.96 [ps] | 7.9600E-12 s |
| alpha_1 | 0.1 | 0.1 |
| delta_e2 | 1200 | 1200 |
| tau_2 | 159.15 [ns] | 1.5915E-7 s |
| alpha_2 | 0.05 | 0.05 |
| delta_e3 | 4.5e5 | 4.5000E5 |
| tau_3 | 72.34 [us] | 7.2340E-5 s |
| alpha_3 | 0.22 | 0.22 |
| delta_e4 | 2.5e7 | 2.5000E7 |
| tau_4 | 4.547 [ms] | 0.004547 s |
| alpha_4 | 0 | 0 |
| e1 | delta_e1/(1+(j*omega*tau_1)^(1-alpha_1)) | 49.999 - 0.0035632i |
| e2 | delta_e2/(1+(j*omega*tau_2)^(1-alpha_2)) | 925.57 - 458.97i |
| e3 | delta_e3/(1+(j*omega*tau_3)^(1-alpha_3)) | 2284.9 - 6085.7i |
| e4 | delta_e4/(1+(j*omega*tau_4)^(1-alpha_4)) | 0.12252 - 1750.1i |
| e_r | e_inf+e1+e2+e3+e4 | 3264.6 - 8294.8i |
| epsilon_prime_prime | -epsilon0_const*imag(e_r) | 7.3444E-8 F/m |

Note that the last column of the above table represents the numerical values calculated by COMSOL. As can be seen from the last row of the table, COMSOL yields a value of $\epsilon'' = 7.3444 \times 10^{-8} \text{ F m}^{-1}$. This parameter can then be used in the user-defined heat source as follows:

$$\begin{aligned} & \sigma \cdot \epsilon_c \cdot \text{norm}E^2 + \\ & \epsilon_{\text{prime_prime}} \cdot \omega \cdot \epsilon_c \cdot \text{norm}E^2 + \\ & \rho_b \cdot C_b \cdot \omega_b \cdot (T_b - T) \end{aligned}$$

where the first term denotes the conductive (i.e. Joule) heating component, the second term denotes dielectric heating, and the third term denotes blood perfusion heating. All other model settings are as given in Example 7.2.3. Solving this model yields the following lesion volume against time plot:



Comparing this result with that of Example 7.2.3 (Fig. 7.13), we see that addition of the dielectric heating component at 500 kHz doubles the lesion volume. At this frequency, dielectric heating is therefore significant and should not be neglected when simulating RF atrial ablation.

Problems of Chap. 8

8.1 (a) Using the definitions of scalar dot and vector cross products given by Eqs. 8.1 and 8.2, we have:

$$\begin{aligned} (\mathbf{a} \times \mathbf{b}) \cdot \mathbf{c} &= \begin{vmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3 \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix} \cdot \mathbf{c} \\ &= \left\{ \mathbf{e}_1 \begin{vmatrix} a_2 & a_3 \\ b_2 & b_3 \end{vmatrix} - \mathbf{e}_2 \begin{vmatrix} a_1 & a_3 \\ b_1 & b_3 \end{vmatrix} + \mathbf{e}_3 \begin{vmatrix} a_1 & a_2 \\ b_1 & b_2 \end{vmatrix} \right\} \cdot \mathbf{c} \\ &= (\mathbf{e}_1 \cdot \mathbf{c}) \begin{vmatrix} a_2 & a_3 \\ b_2 & b_3 \end{vmatrix} - (\mathbf{e}_2 \cdot \mathbf{c}) \begin{vmatrix} a_1 & a_3 \\ b_1 & b_3 \end{vmatrix} + (\mathbf{e}_3 \cdot \mathbf{c}) \begin{vmatrix} a_1 & a_2 \\ b_1 & b_2 \end{vmatrix} \\ &= c_1 \begin{vmatrix} a_2 & a_3 \\ b_2 & b_3 \end{vmatrix} - c_2 \begin{vmatrix} a_1 & a_3 \\ b_1 & b_3 \end{vmatrix} + c_3 \begin{vmatrix} a_1 & a_2 \\ b_1 & b_2 \end{vmatrix} \\ &= \begin{vmatrix} c_1 & c_2 & c_3 \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix} \\ &= \begin{vmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{vmatrix} \end{aligned}$$

Expanding this determinant, we obtain:

$$\begin{aligned} \begin{vmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{vmatrix} &= a_1 \begin{vmatrix} b_2 & b_3 \\ c_2 & c_3 \end{vmatrix} - a_2 \begin{vmatrix} b_1 & b_3 \\ c_1 & c_3 \end{vmatrix} + a_3 \begin{vmatrix} b_1 & b_2 \\ c_1 & c_2 \end{vmatrix} \\ &= a_1 (b_2 c_3 - b_3 c_2) - a_2 (b_1 c_3 - b_3 c_1) + a_3 (b_1 c_2 - b_2 c_1) \\ &= a_1 b_2 c_3 - a_1 b_3 c_2 - a_2 b_1 c_3 + a_2 b_3 c_1 + a_3 b_1 c_2 - a_3 b_2 c_1 \\ &= \varepsilon_{ijk} a_i b_j c_k \end{aligned}$$

Hence,

$$(\mathbf{a} \times \mathbf{b}) \cdot \mathbf{c} = \begin{vmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{vmatrix} = \varepsilon_{ijk} a_i b_j c_k$$

(b) We recall that

$$\sigma_{ij}^H = \frac{1}{3} \sigma_{\alpha\alpha} \delta_{ij} \quad \varepsilon_{ij}^H = \frac{1}{3} \varepsilon_{\alpha\alpha} \delta_{ij}$$

and writing these hydrostatic tensors in terms of their components, we have:

$$\begin{aligned} \boldsymbol{\sigma}^H &= \begin{pmatrix} \frac{1}{3}(\sigma_{11} + \sigma_{22} + \sigma_{33}) & 0 & 0 \\ 0 & \frac{1}{3}(\sigma_{11} + \sigma_{22} + \sigma_{33}) & 0 \\ 0 & 0 & \frac{1}{3}(\sigma_{11} + \sigma_{22} + \sigma_{33}) \end{pmatrix} \\ \boldsymbol{\varepsilon}^H &= \begin{pmatrix} \frac{1}{3}(\varepsilon_{11} + \varepsilon_{22} + \varepsilon_{33}) & 0 & 0 \\ 0 & \frac{1}{3}(\varepsilon_{11} + \varepsilon_{22} + \varepsilon_{33}) & 0 \\ 0 & 0 & \frac{1}{3}(\varepsilon_{11} + \varepsilon_{22} + \varepsilon_{33}) \end{pmatrix} \end{aligned}$$

For the deviatoric tensors, we have

$$\boldsymbol{\sigma}^D = \boldsymbol{\sigma} - \boldsymbol{\sigma}^H \quad \boldsymbol{\varepsilon}^D = \boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}^H$$

and therefore

$$\begin{aligned} \boldsymbol{\sigma}^D &= \begin{pmatrix} \frac{1}{3}(2\sigma_{11} - \sigma_{22} - \sigma_{33}) & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \frac{1}{3}(-\sigma_{11} + 2\sigma_{22} - \sigma_{33}) & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \frac{1}{3}(-\sigma_{11} - \sigma_{22} + 2\sigma_{33}) \end{pmatrix} \\ \boldsymbol{\varepsilon}^D &= \begin{pmatrix} \frac{1}{3}(2\varepsilon_{11} - \varepsilon_{22} - \varepsilon_{33}) & \varepsilon_{12} & \varepsilon_{13} \\ \varepsilon_{21} & \frac{1}{3}(-\varepsilon_{11} + 2\varepsilon_{22} - \varepsilon_{33}) & \varepsilon_{23} \\ \varepsilon_{31} & \varepsilon_{32} & \frac{1}{3}(-\varepsilon_{11} - \varepsilon_{22} + 2\varepsilon_{33}) \end{pmatrix} \end{aligned}$$

In particular, we note that

$$\begin{aligned} \text{trace}(\boldsymbol{\sigma}^D) &= \frac{1}{3} \{2\sigma_{11} - \sigma_{22} - \sigma_{33} - \sigma_{11} + 2\sigma_{22} - \sigma_{33} - \sigma_{11} - \sigma_{22} + 2\sigma_{33}\} = 0 \\ \text{trace}(\boldsymbol{\varepsilon}^D) &= \frac{1}{3} \{2\varepsilon_{11} - \varepsilon_{22} - \varepsilon_{33} - \varepsilon_{11} + 2\varepsilon_{22} - \varepsilon_{33} - \varepsilon_{11} - \varepsilon_{22} + 2\varepsilon_{33}\} = 0 \end{aligned}$$

Writing

$$\boldsymbol{\sigma}^H = \begin{pmatrix} \sigma_H & 0 & 0 \\ 0 & \sigma_H & 0 \\ 0 & 0 & \sigma_H \end{pmatrix}, \quad \boldsymbol{\varepsilon}^H = \begin{pmatrix} \varepsilon_H & 0 & 0 \\ 0 & \varepsilon_H & 0 \\ 0 & 0 & \varepsilon_H \end{pmatrix}$$

with

$$\sigma_H = \frac{1}{3} (\sigma_{11} + \sigma_{22} + \sigma_{33}), \quad \varepsilon_H = \frac{1}{3} (\varepsilon_{11} + \varepsilon_{22} + \varepsilon_{33})$$

we have:

$$\sigma_{ij}^H \varepsilon_{ij}^D = \sigma_{11}^H \varepsilon_{11}^D + \sigma_{22}^H \varepsilon_{22}^D + \sigma_{33}^H \varepsilon_{33}^D = \sigma_H \text{trace}(\boldsymbol{\varepsilon}^D) = 0$$

and

$$\sigma_{ij}^D \varepsilon_{ij}^H = \sigma_{11}^D \varepsilon_{11}^H + \sigma_{22}^D \varepsilon_{22}^H + \sigma_{33}^D \varepsilon_{33}^H = \varepsilon_H \text{trace}(\boldsymbol{\sigma}^D) = 0$$

Hence we have verified that $\sigma_{ij}^H \varepsilon_{ij}^D = \sigma_{ij}^D \varepsilon_{ij}^H = 0$.

8.2 (a) For the simple shear deformation given, we have

$$u_1 = \lambda x_2, \quad u_2 = 0$$

Therefore,

$$\varepsilon_{11} = \frac{\partial u_1}{\partial x_1} = 0$$

$$\varepsilon_{12} = \varepsilon_{21} = \frac{1}{2} \left(\frac{\partial u_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1} \right) = \frac{1}{2} (\lambda + 0) = \frac{\lambda}{2}$$

$$\varepsilon_{22} = \frac{\partial u_2}{\partial x_2} = 0$$

and

$$E_{11} = \frac{1}{2} \left(\frac{\partial u_1}{\partial x_1} + \frac{\partial u_1}{\partial x_1} + \frac{\partial u_1}{\partial x_1} \frac{\partial u_1}{\partial x_1} \right) = 0$$

$$E_{12} = E_{21} = \frac{1}{2} \left(\frac{\partial u_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1} + \frac{\partial u_1}{\partial x_2} \frac{\partial u_2}{\partial x_1} \right) = \frac{1}{2} (\lambda + 0 + \lambda \cdot 0) = \frac{\lambda}{2}$$

$$E_{22} = \frac{1}{2} \left(\frac{\partial u_2}{\partial x_2} + \frac{\partial u_2}{\partial x_2} + \frac{\partial u_2}{\partial x_2} \frac{\partial u_2}{\partial x_2} \right) = 0$$

In matrix form, these strain tensors are

$$\boldsymbol{\varepsilon} = \begin{pmatrix} 0 & \frac{\lambda}{2} \\ \frac{\lambda}{2} & 0 \end{pmatrix}, \quad \mathbf{E} = \begin{pmatrix} 0 & \frac{\lambda}{2} \\ \frac{\lambda}{2} & 0 \end{pmatrix}$$

(b) For the uniform inflation deformation, we have

$$u_1 = (R - 1)x_1, \quad u_2 = (R - 1)x_2$$

Therefore,

$$\begin{aligned} \varepsilon_{11} &= \frac{\partial u_1}{\partial x_1} = R - 1 \\ \varepsilon_{12} = \varepsilon_{21} &= \frac{1}{2} \left(\frac{\partial u_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1} \right) = 0 \\ \varepsilon_{22} &= \frac{\partial u_2}{\partial x_2} = R - 1 \end{aligned}$$

and

$$\begin{aligned} E_{11} &= \frac{1}{2} \left(\frac{\partial u_1}{\partial x_1} + \frac{\partial u_1}{\partial x_1} + \frac{\partial u_1}{\partial x_1} \frac{\partial u_1}{\partial x_1} \right) = \frac{1}{2} (2R - 2 + (R - 1)^2) = \frac{1}{2} ((R - 1)(R + 1)) \\ &= \frac{1}{2} (R^2 - 1) \\ E_{12} = E_{21} &= \frac{1}{2} \left(\frac{\partial u_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1} + \frac{\partial u_1}{\partial x_2} \frac{\partial u_2}{\partial x_1} \right) = 0 \\ E_{22} &= \frac{1}{2} \left(\frac{\partial u_2}{\partial x_2} + \frac{\partial u_2}{\partial x_2} + \frac{\partial u_2}{\partial x_2} \frac{\partial u_2}{\partial x_2} \right) = \frac{1}{2} (2R - 2 + (R - 1)^2) = \frac{1}{2} ((R - 1)(R + 1)) \\ &= \frac{1}{2} (R^2 - 1) \end{aligned}$$

In matrix form, these strain tensors are

$$\boldsymbol{\varepsilon} = \begin{pmatrix} R - 1 & 0 \\ 0 & R - 1 \end{pmatrix}, \quad \mathbf{E} = \begin{pmatrix} \frac{1}{2}(R^2 - 1) & 0 \\ 0 & \frac{1}{2}(R^2 - 1) \end{pmatrix}$$

(c) In the case of rotation, a point originally at (x_1, x_2) is rotated to the new point (\bar{x}_1, \bar{x}_2) according to the rotation transformation:

$$\begin{pmatrix} \bar{x}_1 \\ \bar{x}_2 \end{pmatrix} = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

Hence the displacements are given by

$$\begin{aligned} u_1 &= x_1 \cos \theta - x_2 \sin \theta - x_1 = x_2 (\cos \theta - 1) - x_2 \sin \theta \\ u_2 &= x_2 \cos \theta + x_1 \sin \theta - x_2 = x_1 (\cos \theta - 1) + x_1 \sin \theta \end{aligned}$$

Therefore,

$$\begin{aligned}\varepsilon_{11} &= \frac{\partial u_1}{\partial x_1} = \cos \theta - 1 \\ \varepsilon_{12} = \varepsilon_{21} &= \frac{1}{2} \left(\frac{\partial u_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1} \right) = \frac{1}{2} (-\sin \theta + \sin \theta) = 0 \\ \varepsilon_{22} &= \frac{\partial u_2}{\partial x_2} = \cos \theta - 1\end{aligned}$$

and

$$\begin{aligned}E_{11} &= \frac{1}{2} \left(\frac{\partial u_1}{\partial x_1} + \frac{\partial u_1}{\partial x_1} + \frac{\partial u_1}{\partial x_1} \frac{\partial u_1}{\partial x_1} \right) = \frac{1}{2} (2(\cos \theta - 1) + (\cos \theta - 1)^2) \\ &= \frac{1}{2} ((\cos \theta - 1)(\cos \theta + 1)) = \frac{1}{2} (\cos^2 \theta - 1) = -\frac{1}{2} \sin^2 \theta \\ E_{12} = E_{21} &= \frac{1}{2} \left(\frac{\partial u_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1} + \frac{\partial u_1}{\partial x_2} \frac{\partial u_2}{\partial x_1} \right) = \frac{1}{2} (-\sin \theta + \sin \theta + -\sin \theta \cdot \sin \theta) \\ &= -\frac{1}{2} \sin^2 \theta \\ E_{22} &= \frac{1}{2} \left(\frac{\partial u_2}{\partial x_2} + \frac{\partial u_2}{\partial x_2} + \frac{\partial u_2}{\partial x_2} \frac{\partial u_2}{\partial x_2} \right) = \frac{1}{2} (2(\cos \theta - 1) + (\cos \theta - 1)^2) \\ &= \frac{1}{2} ((\cos \theta - 1)(\cos \theta + 1)) = \frac{1}{2} (\cos^2 \theta - 1) = -\frac{1}{2} \sin^2 \theta\end{aligned}$$

In matrix form, these strain tensors are therefore given by

$$\boldsymbol{\varepsilon} = \begin{pmatrix} \cos \theta - 1 & 0 \\ 0 & \cos \theta - 1 \end{pmatrix}, \quad \mathbf{E} = \frac{1}{2} \begin{pmatrix} -\sin^2 \theta & -\sin^2 \theta \\ -\sin^2 \theta & -\sin^2 \theta \end{pmatrix}$$

8.3 (a) For an incompressible deformation, the volume of the spherical shell must be preserved. That is:

$$\frac{4}{3}\pi (b^3 - a^3) = \frac{4}{3}\pi (B^3 - A^3)$$

Therefore,

$$\begin{aligned}b^3 &= a^3 + B^3 - A^3 \\ b &= \sqrt[3]{a^3 + B^3 - A^3}\end{aligned}$$

(b) For a particle in the myocardial wall initially located at a radial position of R , we can use a similar analysis as part a) to show that the volume of the shell between radial distances of A and R must be preserved. Hence, if this particle moves to a new radial position of r , then we must have

$$r = \sqrt[3]{a^3 + R^3 - A^3}$$

which corresponds to a radial displacement $u(R) = r - R$, with u_1, u_2, u_3 components of

$$\begin{aligned}
 u_1 &= \left(\frac{x_1}{R}\right) u(R) = \frac{x_1}{R} \left(\sqrt[3]{a^3 + R^3 - A^3} - R\right) = x_1 \left(\sqrt[3]{1 + \frac{a^3 - A^3}{R^3}} - 1\right) \\
 u_2 &= \left(\frac{x_2}{R}\right) u(R) = \frac{x_2}{R} \left(\sqrt[3]{a^3 + R^3 - A^3} - R\right) = x_2 \left(\sqrt[3]{1 + \frac{a^3 - A^3}{R^3}} - 1\right) \\
 u_3 &= \left(\frac{x_3}{R}\right) u(R) = \frac{x_3}{R} \left(\sqrt[3]{a^3 + R^3 - A^3} - R\right) = x_3 \left(\sqrt[3]{1 + \frac{a^3 - A^3}{R^3}} - 1\right)
 \end{aligned}$$

where $R = \sqrt{x_1^2 + x_2^2 + x_3^2}$. For convenience, we let $\gamma = \sqrt[3]{1 + \frac{a^3 - A^3}{R^3}} - 1$ and write these displacement components as

$$u_1 = \gamma x_1, \quad u_2 = \gamma x_2, \quad u_3 = \gamma x_3$$

To determine the components of Cauchy strain, we need to determine the partial derivatives of these displacements with respect to x_1, x_2 and x_3 . These in turn require the following derivatives:

$$\begin{aligned}
 \frac{d\gamma}{dR} &= \frac{1}{3} \left(1 + \frac{a^3 - A^3}{R^3}\right)^{-\frac{2}{3}} \left[\frac{-3(a^3 - A^3)}{R^4}\right] \\
 &= -\left[\frac{a^3 - A^3}{R^4}\right] \left(1 + \frac{a^3 - A^3}{R^3}\right)^{-\frac{2}{3}} \\
 &= -\left[\frac{(\gamma + 1)^3 - 1}{R}\right] (\gamma + 1)^{-2} \\
 &= -\frac{1}{R} \left[\gamma + 1 - \frac{1}{(\gamma + 1)^2}\right] \\
 \frac{\partial R}{\partial x_1} &= \frac{1}{2} (x_1^2 + x_2^2 + x_3^2)^{-\frac{1}{2}} 2x_1 = \frac{x_1}{R} \\
 \frac{\partial R}{\partial x_2} &= \frac{1}{2} (x_1^2 + x_2^2 + x_3^2)^{-\frac{1}{2}} 2x_2 = \frac{x_2}{R} \\
 \frac{\partial R}{\partial x_3} &= \frac{1}{2} (x_1^2 + x_2^2 + x_3^2)^{-\frac{1}{2}} 2x_3 = \frac{x_3}{R}
 \end{aligned}$$

Hence, the derivatives of displacement u_1 with respect to each of x_1, x_2 , and x_3 are:

$$\begin{aligned}
 \frac{\partial u_1}{\partial x_1} &= \gamma + x_1 \frac{\partial \gamma}{\partial x_1} = \gamma + x_1 \frac{d\gamma}{dR} \frac{\partial R}{\partial x_1} = \gamma - \frac{x_1^2}{R^2} \left[\gamma + 1 - \frac{1}{(\gamma + 1)^2}\right] \\
 \frac{\partial u_1}{\partial x_2} &= x_1 \frac{\partial \gamma}{\partial x_2} = x_1 \frac{d\gamma}{dR} \frac{\partial R}{\partial x_2} = -\frac{x_1 x_2}{R^2} \left[\gamma + 1 - \frac{1}{(\gamma + 1)^2}\right]
 \end{aligned}$$

$$\frac{\partial u_1}{\partial x_3} = x_1 \frac{\partial \gamma}{\partial x_3} = x_1 \frac{d\gamma}{dR} \frac{\partial R}{\partial x_3} = -\frac{x_1 x_3}{R^2} \left[\gamma + 1 - \frac{1}{(\gamma + 1)^2} \right]$$

Again for convenience, we let $\beta = \gamma + 1 - \frac{1}{(\gamma + 1)^2}$. The above derivatives then become:

$$\frac{\partial u_1}{\partial x_1} = \gamma - \frac{\beta x_1^2}{R^2}, \quad \frac{\partial u_1}{\partial x_2} = -\frac{\beta x_1 x_2}{R^2}, \quad \frac{\partial u_1}{\partial x_3} = -\frac{\beta x_1 x_3}{R^2}$$

Similarly, we obtain the remaining derivatives of the other displacement terms as

$$\begin{aligned} \frac{\partial u_2}{\partial x_1} &= -\frac{\beta x_1 x_2}{R^2}, & \frac{\partial u_2}{\partial x_2} &= \gamma - \frac{\beta x_2^2}{R^2}, & \frac{\partial u_2}{\partial x_3} &= -\frac{\beta x_2 x_3}{R^2} \\ \frac{\partial u_3}{\partial x_1} &= -\frac{\beta x_1 x_3}{R^2}, & \frac{\partial u_3}{\partial x_2} &= -\frac{\beta x_2 x_3}{R^2}, & \frac{\partial u_3}{\partial x_3} &= \gamma - \frac{\beta x_3^2}{R^2} \end{aligned}$$

The Cauchy strain is given by $\varepsilon_{ij} = \frac{1}{2} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right)$. Inserting the above derivatives, we obtain its following matrix form:

$$\boldsymbol{\varepsilon} = \begin{pmatrix} \gamma - \frac{\beta x_1^2}{R^2} & -\frac{\beta x_1 x_2}{R^2} & -\frac{\beta x_1 x_3}{R^2} \\ -\frac{\beta x_1 x_2}{R^2} & \gamma - \frac{\beta x_2^2}{R^2} & -\frac{\beta x_2 x_3}{R^2} \\ -\frac{\beta x_1 x_3}{R^2} & -\frac{\beta x_2 x_3}{R^2} & \gamma - \frac{\beta x_3^2}{R^2} \end{pmatrix}$$

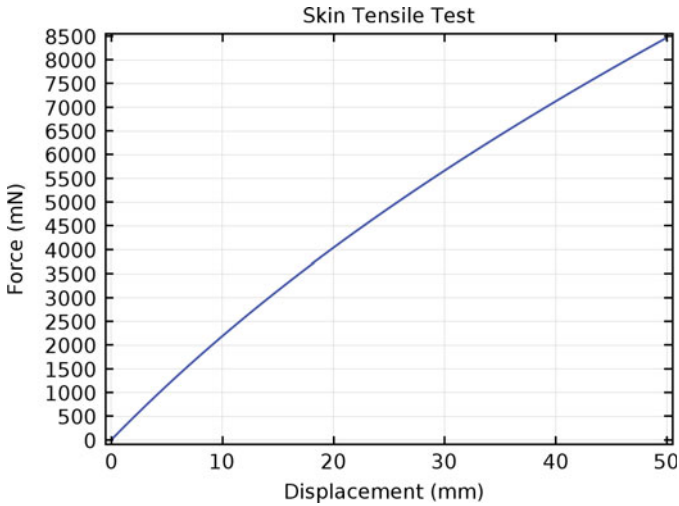
where, as indicated earlier, $\gamma = \sqrt[3]{1 + \frac{a^3 - A^3}{R^3}} - 1$ and $\beta = \gamma + 1 - \frac{1}{(\gamma + 1)^2}$.

8.4 (a) Since the skin thickness of the sample is small compared to its other dimensions, a 2D implementation in COMSOL is preferable, using the plane stress condition (i.e. there are no components of stress perpendicular to the plane). Since the model is symmetric, we can implement only half of the geometry, employing a symmetric boundary condition on the lower face. The following are key points regarding this COMSOL implementation

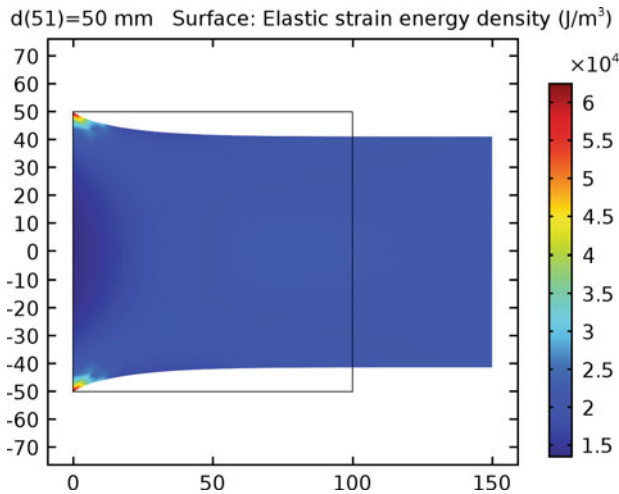
- In the Solid Mechanics settings, specify a thickness of 1 mm and the plane stress 2D approximation mode.
- Specify a hyperelastic material employing the Mooney-Rivlin, two parameters material model. Enter all user-defined material values as given.
- Specify a prescribed displacement boundary condition on the right boundary with only a prescribed x-displacement of \bar{d} , where \bar{d} is a user-defined global model parameter. Specify a symmetry boundary condition for the lower boundary a fixed constraint boundary condition for the left boundary.
- Define an integration boundary operator, `intop1`, acting over the right boundary. Define a model variable `F` representing the applied force, given by the expression `2*intop1(solid.Tax)*(1 [mm])`. In this expression, `Tax` denotes a COMSOL in-built variable for the x-component of traction, `(1 [mm])` denotes

the skin thickness, and the factor 2 takes into account that only half the geometry is implemented.

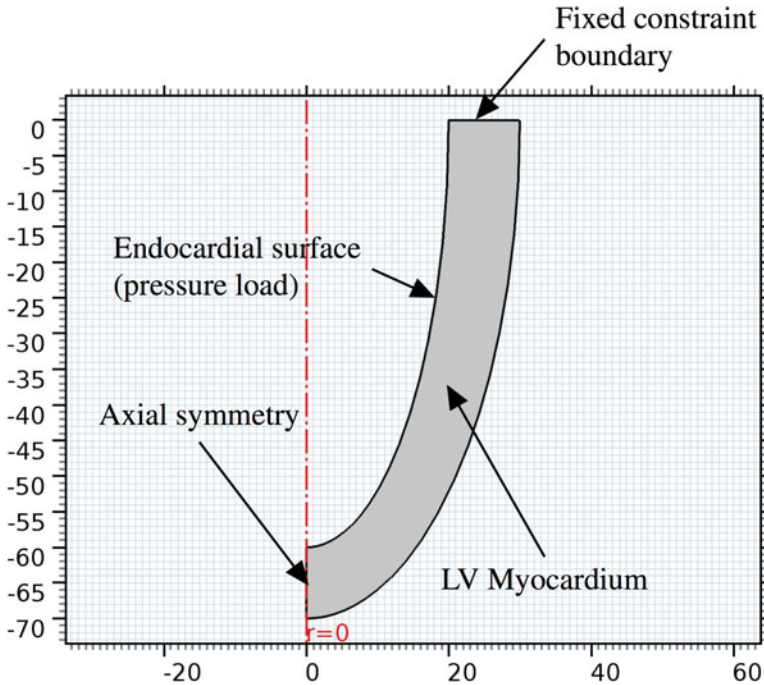
- Perform a parameter sweep of parameter d from 0 to 50 mm.
- Setup a 1D plot group (global plot) to plot the resulting applied force vs displacement, as shown below:



(b) To plot the elastic strain energy, first create a 2D mirror dataset so that the entire geometry of the skin sample can be visualised. Setup a new 2D plot group (surface plot) and plot COMSOL's in-built elastic strain energy variable `solid.ws`. Right-click the surface plot-sub-node and specify deformation with a scale factor of 1. The resulting plot of strain energy in the deformed sample is shown below:



8.5 To implement this model in COMSOL, setup the 2D axisymmetric geometry shown below, where all dimensions are shown in mm:



Additional points regarding model implementation are as follows:

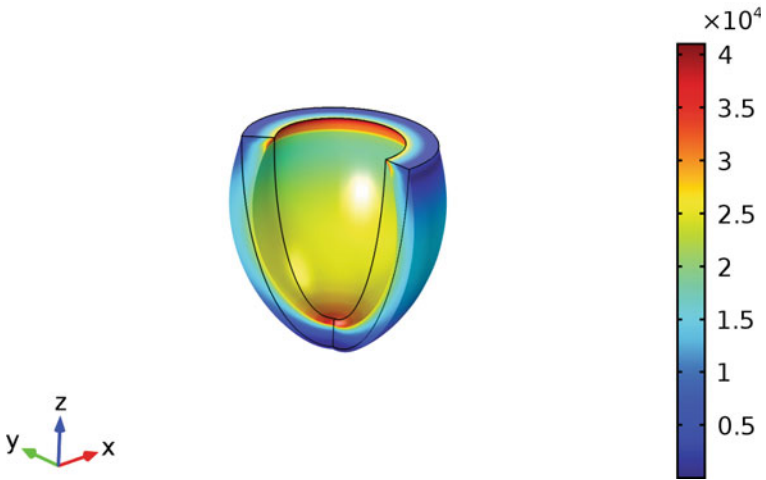
- Define a model parameter `press` for the endocardial pressure with a default value of 50 [mmHg].
- For the model geometry, use two ellipses for the epi and endocardial surfaces, perform a boolean subtraction of one from the other, and subtract a rectangle covering their upper half from both.
- Define an integration boundary operator, `intop1`, acting over the endocardial edge. This operator is then used in a variable expression to calculate the volume of the LV cavity according to `intop1(-pi*nr*r^2)`. Note the negative sign for `nr` is to specify the outward normal of the LV *cavity* as opposed to the LV wall. To understand this expression, we note that the volume of a solid of revolution is given by

$$\begin{aligned}
 V &= \int_A 2\pi r dr dz \\
 &= \int_A \pi \nabla \cdot \begin{pmatrix} r^2 \\ 0 \end{pmatrix} dA \quad (\text{where } dA = dr dz) \\
 &= \int_L \pi n_r r^2 dL \quad (\text{which follows from the divergence theorem})
 \end{aligned}$$

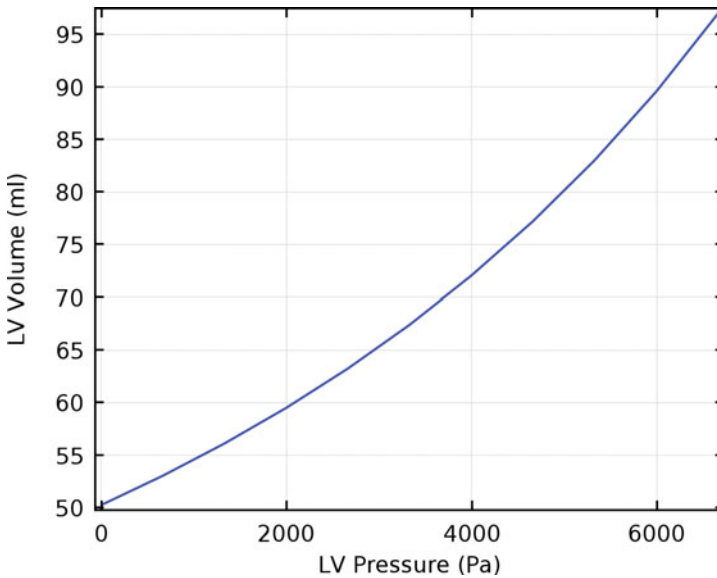
where L is the edge boundary of the axisymmetric solid of revolution, and n_r is the r-component of the outward normal along the boundary.

- Select the hyperelastic/Mooney-Rivlin, two parameters material model and enter all material coefficients as described.
- For the base boundary, select the Fixed Constraint boundary condition. For the endocardial edge, select the Boundary Load boundary condition, and specify the load type as 'pressure'. Enter the value as the pressure parameter `press`.
- For the parameter sweep, specify parameter `press` ranging from 0 to 50 mmHg in steps of 5 mmHg.
- For the mesh, select the 'Fine' mesh setting.
- Solving the model produces the following von Mises stress distribution in the deformed (i.e. inflated) state at 50 mmHg, where we have used the default 3D stress plot:

`press(11)=50 mmHg` Surface: von Mises stress (N/m²)



- Specifying a new 1D Plot Group (Global plot), produces the following plot of variable ∇ against `press`:



Problems of Chap. 9

9.1 Following the same principles as Sect. 9.1.1, we can represent the fluid motion in the circular tube using a series of sliding tubes. Denoting the radius of one such tube as r , the forces acting on its surface boundaries will be:

1. the force on the upstream end due to the upstream pressure, given by $F_{up} = \pi r^2 P$.
2. the force on the downstream end due to the downstream pressure, given by $F_{down} = 0$, since in this case there is no downstream pressure.
3. the traction acting on the curved surface of the tube, due to the viscous force from the relative velocities of layers sliding past each other. This viscous force equals the viscous stress τ multiplied by the curved surface area, or

$$F_{viscous} = 2\pi r L \tau = 2\pi r L \mu \frac{\partial v}{\partial r}$$

where v is the fluid velocity.

Adding the above three forces together yields the total force on the inner tube, which is the mass of the tube multiplied by its acceleration. This total force is given by

$$F = \int_0^r 2\pi r L \rho \frac{\partial v}{\partial t} dr$$

Hence,

$$\pi r^2 P + \mu \frac{\partial v}{\partial r} 2\pi r L = F = \int_0^r 2\pi r L \rho \frac{\partial v}{\partial t} dr$$

Differentiating both sides with respect to r , we obtain:

$$2\pi r P + \frac{\partial^2 v}{\partial r^2} 2\pi r \mu L + 2\pi \mu L \frac{\partial v}{\partial r} = 2\pi r L \rho \frac{\partial v}{\partial t}$$

Dividing throughout by $2\pi r \mu L$ and re-arranging:

$$\frac{\rho}{\mu} \frac{\partial v}{\partial t} - \frac{\partial^2 v}{\partial r^2} - \frac{1}{r} \frac{\partial v}{\partial r} = \frac{P}{\mu L}$$

or

$$\frac{\rho}{\mu} \frac{\partial v}{\partial t} + \frac{1}{r} \frac{\partial}{\partial r} \left[-r \frac{\partial v}{\partial r} \right] = \frac{P}{\mu L}$$

Multiplying throughout by r , we obtain the resulting PDE:

$$\frac{\rho r}{\mu} \frac{\partial v}{\partial t} + \frac{\partial}{\partial r} \left[-r \frac{\partial v}{\partial r} \right] = \frac{Pr}{\mu L}$$

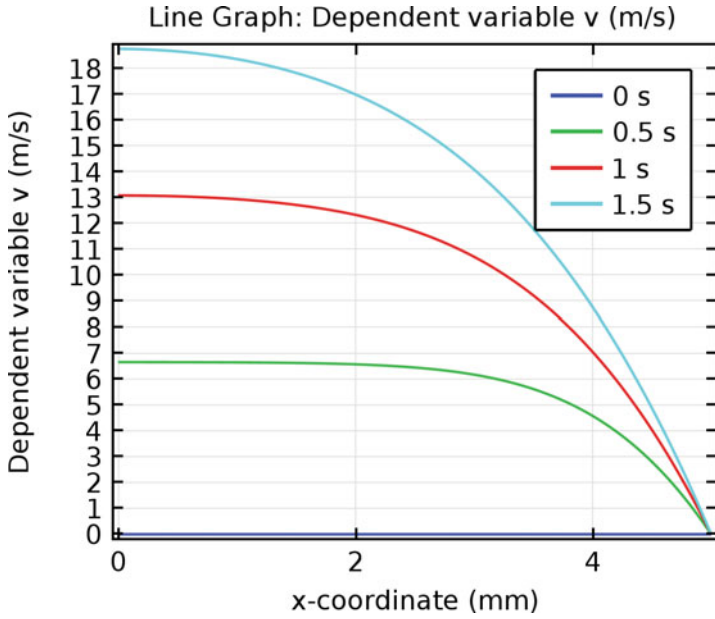
with initial and boundary conditions on $v(r, t)$ given by

$$\begin{aligned} v(r, 0) &= 0 && \text{(initial value)} \\ v(D/2, t) &= 0 && \text{(no - slip wall)} \\ \partial v(0, t) / \partial r &= 0 && \text{(radial symmetry at } r = 0) \end{aligned}$$

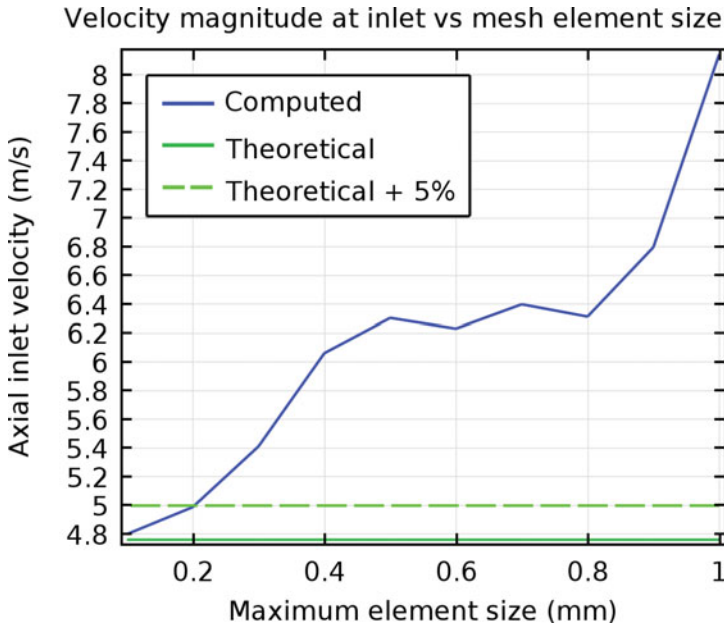
To solve this PDE, we can use COMSOL's mathematics PDE interface (General Form PDE) over a 1D spatial dimension, using the following settings:

$$\begin{aligned} \text{flux : } \Gamma &= -r \partial v / \partial r \\ \text{damping coefficient : } d_a &= \rho r / \mu \\ \text{source term : } f &= Pr / \mu L \end{aligned}$$

and using $r \equiv x$ for COMSOL's 1D PDE General form, the above quantities are written as $-x*v_x$, $rho*x/mu$, and $P*x / (mu*L)$ respectively. Implementing this PDE in COMSOL, using a Dirichlet boundary condition of $v = 0$ at $x = D/2$ and zero-flux boundary condition at $x = 0$, produces the following solution for v at $t = 0, 0.5, 1,$ and 1.5 s:

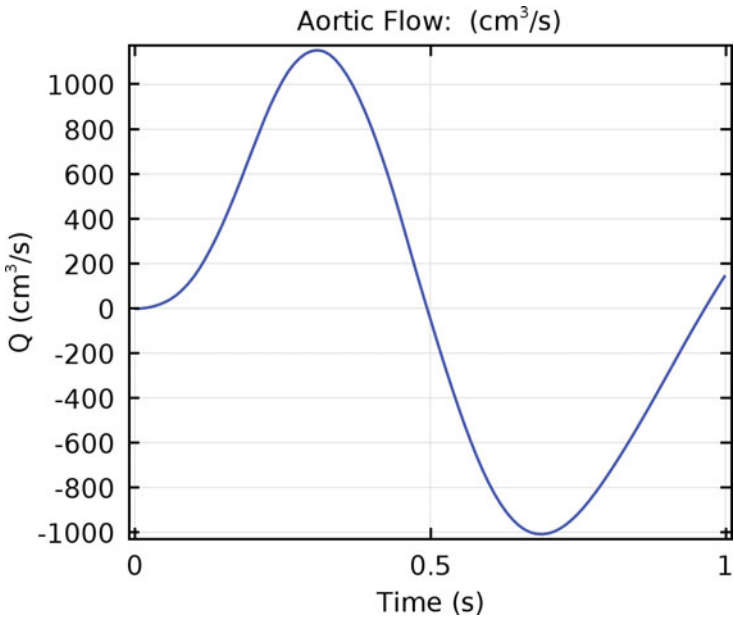


9.2 We can implement the axisymmetric model of Sect. 9.1.1 in COMSOL using an additional parameter `mesh` to denote the maximum element size. Specifying a custom mesh size with maximum element size of `mesh`, and performing a parameter sweep, results in the following plot of axial velocity at the inlet of the tube as a function of element size:



from which it is evident that a maximum element size of 0.2 mm achieves an error of 5% of theoretical axial velocity of $V_{th} = \frac{\Delta P D^2}{16\mu L} \approx 4.76 \text{ m s}^{-1}$.

9.3 We can utilize the COMSOL model of Sect. 9.4.2, this time defining an interpolation function (under Global Definitions | Functions | Interpolation) and entering a table of the pressure values and times specified. In the Units tab of the interpolation function settings, specify the units of the function argument as ‘ms’ and the function output as ‘mmHg’. Under the Interpolation and Extrapolation tab, specify the interpolation type as ‘Cubic Spline’. This will create an interpolation function with default name ‘int1’. Under the component 1 variable definitions, we can then specify the variable P_{in} as the expression $\text{int1}(\tau)$. Specify the mesh element size as ‘Finer’, and specify the global ODE variable P_{out} to have the initial value 72 [mmHg]. All other settings are the same as per the COMSOL model of Sect. 9.4.2. Using these settings, the resulting plot of aortic flow is shown below:



Index

A

- Action Potential, 36, 42, 49, 151, 155, 217, 234, 379, 406
- Algebraic Equation (AE), 29–30
- Arrhenius Equation, 251
- Arrhenius, Svante, 251
- Axial Streaming, 330
 - model of, 330–339
- Axisymmetric Models, 128, 211–215, 239–241, 244–247, 251–258, 302, 308–310, 314–321, 325–329, 340, 357, 470, 475–477, 488–490, 492–493

B

- Backward Difference
 - backward difference operator, 83
 - Newton's backward difference formula, 83
- Backward Differentiation Formula (BDF), 93–96, 98, 103, 427–430
 - coefficients, 95
 - order selection, 95
- Bacterial Growth, models of, 6, 23, 29–30
- Basis Functions, 159, 164–179, 183–190, 196, *see also* Shape Functions
 - 1D linear (witch's hat), 164, 165, 167, 170, 171, 174, 176
- Beeler–Reuter Model, 100–102, 410–420
- Bidomain Equations, 216–217
 - COMSOL example (cardiac reentry), 217–225
- Bioheat Equation, 250–251, *see also* Heat Transfer
- Blood Flow, 324–339
 - hydraulic circuits, 37–42, 324–329, 340–341, 493

- in a cylindrical vessel, 19–21, 306–310, 340–341, 490–493
 - models of, 242–247, 314–321, 325–339
 - non-Newtonian properties of, 329–330
- Body Force, 274, 312, 323
 - Boundary Conditions, 4–6, 120–123, 133, 160, 161, 191, 203, 217, 258, 324, 340, 449, 457, 462, 474, 477, 491
 - COMSOL, 356, 362–364
 - Dirichlet, 120–123, 159, 163, 171, 456
 - essential, 123, 163
 - initial value, 32, 159, 163, 171, 196
 - mixed (Robin), 122
 - natural, 163, 171
 - Neumann, 121, 122, 160, 163, 170
 - ODEs, 32
 - zero-flux, 123, 124, 142, 145, 147, 148, 150, 155, 171, 196, 218, 440, 443, 463
- Bulk Modulus, 293–295, 301, 302
 - artificial (deformed geometry), 335

C

- Cable Models, 25–26, 148–152, 154, 226–233, 386
- Cardiac
 - cellular automata model, 26, 386–390
 - defibrillation model, 369–379
 - elastance model, 37–42
 - ionic model, 100–102, 410–420
 - mechanical constitutive law, 293
 - muscle contraction model, 51–53, 406–410
 - passive inflation model, 301–302, 488–490
 - passive shear model, 294–299

- passive stretch model, 154–155, 438–440
- spiral wave reentry model, 155–156, 217–225, 440–444
- Cauchy, Augustin-Louis, 273
- Compliance, 37, 38, 49, 324, 325, 327
- Computer-Aided Design (CAD), 359
- COMSOL, 355–379
 - AC/DC interface, 203
 - chemical reaction engineering module, 238
 - coefficient form PDE, 363
 - component couplings, 364
 - component definitions, 208, 212, 221, 231, 254, 286, 295, 309, 316, 326, 333, 374
 - Computational Fluid Dynamics (CFD) module, 324
 - deformed geometry, 331, 335–336
 - discretization, 174
 - electric currents, 209, 213, 232, 255, 375
 - example models
 - 1D diffusion, 171–175
 - aortic blood flow, 325–329
 - axial streaming of blood cell, 330–339
 - axonal stimulation, 226–233
 - cardiac defibrillation, 369–379
 - cardiac spiral wave reentry, 217–225
 - cell culture electric field stimulator, 207–210
 - diffusion and uptake into a spherical cell, 238–241
 - drug delivery in a coronary stent revisited, 314–321
 - drug delivery in coronary stent, 242–247
 - electrode disc access resistance, 210–215
 - fluid flow in cylindrical tube, 308–310
 - logistic growth ODE, 79–81
 - myocardial shear, 294–299
 - respirator strap tension device, 284–290
 - RF atrial ablation, 251–258
 - functions, 360
 - general form edge PDE, 232
 - general form PDE, 173, 222, 223, 256, 363, 364, 375, 376
 - geometric entity level, 361
 - geometry, 173, 208, 212, 220, 230, 239, 244, 254, 285, 308, 316, 326, 332, 358–359, 371
 - global definitions, 80, 173, 208, 211, 219, 230, 244, 253, 285, 295, 308, 316, 325, 331, 359–360, 374
 - global ODEs and DAEs, 80, 318, 327, 334
 - heat transfer in solids, 255
 - heat transfer module, 253
 - laminar flow, 309, 318, 327, 336
 - materials, 287, 362
 - mesh, 173, 213, 233, 240, 256, 297, 309, 337–338, 366, 377
 - model tree, 356
 - model wizard, 80, 173, 207, 211, 219, 229, 239, 244, 253, 284, 295, 308, 315, 325, 331, 357–358, 370
 - moving mesh interface, 330
 - nonlinear structural materials module, 295
 - parametric sweep, 81, 209, 259, 287, 290, 297, 367, 368, 477, 487, 489, 492
 - PDE/ODEs on boundaries, edges and points, 225–226
 - results, 81, 174, 209, 213, 223, 233, 241, 246, 257, 288, 298, 309, 320, 328, 367–368, 377
 - solid mechanics, 287, 297
 - study, 80, 174, 209, 213, 223, 233, 240, 245, 256, 287, 289, 297, 309, 318, 320, 327, 338, 367, 377
 - system matrices, 174
 - tangential derivatives, 226
 - transport of diluted species, 239, 244, 319
 - user interface, 355
- Conductivity, Electrical, 119, 204–206
 - anisotropic conductivity tensor, 205
- Conservation Law Formulation, 117–119
- Constitutive Law, 281, *see also* Solid Mechanics
- Constitutive Relation, electrical, 202
- Constraint Matrix, 171
- Continuous Models, macroscopic form, 9–12
- Contour, 106
- Convection, 241–242, 248–249
- Convergence, 6, 65, *see also* Newton's Method
- Conway, John, 12
- Coronary Stent, models of drug delivery, 242–317

- Curl, 112–430
 curl-free field, 203
- D**
- Damping
 algorithmic, 73
 force, 31
 high frequency, 73, 74, 77–79, 81, 82, 424–426
- Damping Matrix, 97, 166–168, 170, 175, 463, 464, 466, 469, 470
- Dashpot, 46, 154, 155, 291, 438
- Defibrillation, model of, 369–379
- Deformed Geometry, 331, 335–336
- Del Operator, 106, 119, 322, *see also* Nabla Operator
- Deterministic Models, 7–9
- Differential-Algebraic Equation (DAE), 30, 98, 99, 171
- Diffusion, 237–241
 coefficient, 7, 23, 118
 equation, 7, 23, 119–120, 140, 159, 160, 381
 analytical solution, 120–127, 153, 433–435
 numerical solution, 140–147, 171–175, 196, 463–470
 Fick's laws, 118, 237–238
 models of, 238–247
- Dimensional Analysis, 16–21, 24, 382–383
 Buckingham π -theorem, 19–21
 fundamental dimensions, 16
- Direct Current (DC), 251
- Discrete Models, 9–12, 25, 386
- Distributed Systems Models, 105, 148–150, 154–156, 436–444
- Divergence, 108–112, 153, 430
- Divergence Theorem, 113–117
- Drug Delivery
 from coronary stent, 242–247, 314–321
 from microsphere, 153, 433–435
- Dynamic Models, 7, *see also* Time-Dependent
- E**
- Eigenvalues, 62–63, 75–79, 269, 271, 362
- Eigenvectors, 62, 63, 205
- Electrical Stimulation of Tissues, 201–235, 369–379
 cardiac spiral wave reentry, 155–156, 217–225, 440–445
 cell culture electric field stimulator, 156, 207–210, 445–454
 continuum models of excitable tissues, 215–217
 electrode disc stimulation, 128–139, 210–215, 235, 472–473
 nerve axon extracellular stimulation, 226–233
- Electrocardiogram (ECG), 49, 217, 402–403
- Electrode Disc (Isopotential)
 analytical solution, 128–139
 COMSOL implementation, 210–215, 235, 472–473
- Electromagnetic Fields, 201–202, *see also* Maxwell's Equations
- Electrostatics, 203
- Eulerian Framework, 311, 314
- Euler, Leonhard, 34
- Euler Method
 backward, 63–65, 99, 102, 415–417
 forward, 60–63, 102, 412–415
 modified, 65–66, *see also* Trapezoidal Method
- Euler's Formula, 34
- F**
- Fåhræus-Lindqvist Effect, 330
- Fåhræus, Robert (Robin) Sanno, 330
- Fick, Adolf, 237
- Fick's Laws, 118, 121, 237–238, *see also* Diffusion
- Finite Difference Method, 139–147
 cardiac reentrant arrhythmia example, 155–156, 443–445
 derivative approximations, 140
 diffusion example, 142–147
 electric currents example, 156, 445–454
 error analysis, 141, 144
 explicit scheme, 141–144
 implicit scheme, 144–147
 polar coordinates, 156, 445–454
 stability, 141–142, 144
- Finite Element Method, 159–197
 1D basis functions, 159, 164–171, 177–179, 196–197, 454–470
 2D/3D basis functions, 185–190
 assembly of system matrices, 191, 456, 461, 468
 degrees of freedom, 179
 diffusion example, 159–175
 COMSOL implementation, 171–175
 elements, 159, 164, 165, 171, 183–190

- Galerkin method, 165, 183–184, 458
 Gaussian Quadrature, 192–194
 isoparametric elements, 184–185
 local element coordinates, 175–177, 184, 459
 mesh, 183
 nodes, 164, 167, 177, 178, 183–185, 188–191
 non-linear systems, 179, 194–195
 shape functions, 176–179, 185–190
 strong PDE form, 160, 179, 457
 test functions, 160, 161, 163, 165, 166, 180, 181, 183
 weak PDE form, 160–182, 457
- Fluid Mechanics, 305–341
 constitutive law for incompressible, isotropic fluid, 306
 equation of continuity for incompressible flow, 117, 313
 Eulerian framework, 311, 314
 Lagrangian framework, 311, 314
 laminar flow in a circular tube, 306–310, 340–341, 490–493
 momentum balance, 24, 311–312
 Navier-Stokes equations, 313
 non-laminar flow, 321–324
 parabolic velocity, 243, 258, 308, 310, 476
 Reynolds number, 323
 Stokes number, 323
 turbulence, 323
- Flux, 109, 110, 113, 117, 118, 121, 122, 160, 172, 226, 228, 237, 241, 242, 248, 249, 256, 258, 363
- Fourier, Jean-Baptiste Joseph, 248
- Frankenhaeuser–Huxley Neural Model, 49–51, 403–406
- G**
- Galerkin, Boris Grigoryevich, 165
 Galerkin Method, 165, 183–184, 458
 Game of Life, 12
 Generalized Minimal Residual Method (GMRES), 195
 Generalized- α Method, 73–82, 103, 423–426
 amplification matrix, 75
 COMSOL implementation, 79–81, 98
 high frequency damping factor, 78, 79, 81, 82
 Gradient, 105–108, 113, 118, 119, 153, 203, 226, 237, 314, 430
- Green's Identity, 180–181
 Green, George, 276
- H**
- Heat Transfer, 247–258
 bioheat equation, 250–251
 conduction and convection, 248–249
 RF atrial ablation, 251–258, 260, 479–480
 specific heat capacity, 248
 tissue damage, 251
 tumour ablation, 259–260, 477–478
- Hermite Shape Functions, 178–179
- Hodgkin–Huxley Model, 24, 42–45, 148–149
- Hodgkin, Sir Alan, 42
- Hookean Elastic Solid, 281
- Hooke, Robert, 281
- Hooke's Law, 281
- Huxley, Sir Andrew, 42
- Hydrogel Sensor, 25
- Hyperelastic Materials, 291–294, *see also* Solid Mechanics
 Holzapfel constitutive law, 293
 Mooney–Rivlin constitutive law, 293
- I**
- Indicator-Dilution Model, 258–259, 475–477
- Indicial Notation, 265–266
- Initial Values
 as boundary conditions, 32
 COMSOL implementation, 80, 99, 173, 223, 232, 239, 256, 318, 327, 376
 consistent, 99
 Matlab ODE solver implementation, 36, 352
- Integration
 by parts, 161, 163, 180
 constants, 29, 32, 121, 122, 124, 125, 127, 307
 numerical, 192–194, *see also* Quadrature, numerical
 ODEs (analytical and numerical), 53, 55–103
 operator (COMSOL), 208, 212, 254, 286, 295, 316, 326, 333, 365, 366
- Ion Channels
 gating formulations, 25, 43, 46, 50, 101, 148
 stochastic model of, 7–9

J

Jacobian, 62, 63, 65, 417, 426

K

Kronecker Delta, 266, 281, 306

L

Lagrange Multiplier, 171, 456, 461

Lagrange Shape Functions, 177–178, 196, 455, 458, 459, 462–463

Lagrangian Framework, 311, 314

Lamé, Gabriel Léon Jean Baptiste, 281

Lamé's Constants, 281

Laplace Equation, 120

Laplacian, 119–120

Left Ventricle, 37–42, 294, 399–402, 484–486

passive inflation, model of, 301–302, 488–490

Length Constant, 26

Level Surface, 106

Lindqvist, Johan Torsten, 330

Linear Models, 6

Load Vector, 166, 167, 175, 184, 191, 195, 455, 456, 459–461, 463

Logistic Equation, 29–30, 79–81

Lumped Parameter Models, 21, 29, 37–53

M

Magnetostatics, 203

Mass Matrix, 97–99

singular mass matrix (COMSOL), 99, 223, 377

Matlab, 343–352

\, 350

linspace, 345

logninv, 11

ode113, 35, 98, 102

ode15i, 98

ode15s, 35, 36, 40, 45, 55, 97, 98, 102, 151, 352

ode23s, 35, 98, 102

ode23tb, 70, 98, 102

ode23t, 35, 98, 102

ode23, 35, 70, 72, 98, 102

ode45, 35, 55, 70, 72, 98, 102

odeset, 36, 97

sparse, 449, 452

tic, 102

toc, 102

example code

Beeler–Reuter model, 100–102, 410–420

cardiac cellular automata model, 388–390

cardiac elastance model, 39–41

cardiac muscle contraction, 51–53, 406–410

cardiac reentry, 155–156, 441–445

cardiac windkessel model, 48–49, 399–402

cell culture electric field stimulator, 156, 448–454

diffusion equation, explicit finite difference scheme, 143

diffusion equation, implicit finite difference scheme, 146

diffusion equation series solution, 127

ECG model, 49, 402–403

forward Euler method, 61

Frankenhaeuser–Huxley neural model, 49–51, 403–406

glucose–insulin kinetics, 48, 397–399

Hodgkin–Huxley model, 44–45

Hodgkin–Huxley nerve cable, 151

neural spiking model, 102–103, 420–426

neuronal branching model, 14

passive muscle spring model, 11

single ion channel gate, 8

Van der Pol oscillator, 36–37

solving ODEs, 35–36, 97–100, 352

symbolic math toolbox, 90

Maxwell, James Clerk, 201

Maxwell's Equations, 201–202

Ampère's law, 201, 250

charge density, 201

current density, 201

displacement current, 202

electric displacement, 201

electric field, 201, 203

electric potential, 203

Faraday's law of induction, 201

Gauss' law, 202

Gauss' law for magnetism, 202

magnetic field, 201

magnetization, 201

permeability, 202

permittivity, 202, 250

Mesh (COMSOL), 173, 213–214, 233, 240, 256, 297, 309

boundary layers, 240

- convergence analysis, 310, 340, 367, 492–493
 - specifying size, 256
- Method of Lines, 139, 147
 - cardiac spiral wave reentry, 155–156, 440–443
 - Hodgkin-Huxley nerve cable, 148–152
- Model
 - coding, 4
 - formulation, 4
 - scaling, 21–25, 383
 - types, 5–17
 - validation, 5
 - verification, 5
- Modelling
 - bioengineering, 3–4
 - definition, 3
 - process, 4–5
- Monodomain Equation, 217
- Mooney, Melvin, 293
- Mooney–Rivlin Constitutive Law, 293, 301, 302, 486–490
- Moving Mesh, *see* Deformed Geometry
- Muscle
 - cardiac, active model, 51–53, 406–410
 - cardiac, passive model, 154–155, 438–440
 - skeletal, passive model, 10–12, 46–47, 391–393
- Myocardial Shear, model of, 294–299

- N**
- Nabla Operator, 106, *see also* Del Operator
- Navier, Claude-Louis, 314
- Navier-Stokes Equations, 311–314, 321–324, *see also* Fluid Mechanics
- Neuron
 - $I_{Na,p} + I_K$ model, 102
 - action potential, 36, 42
 - branching model, 13–14
 - cable model, 148–152, 154, 227, 437–438
 - chronaxie, 47
 - electrical stimulation, 226–233
 - Frankenhaeuser–Huxley model, 49–51
 - Hodgkin–Huxley model, 24–25, 42–45
 - rheobase, 47
- Newton Interpolating Polynomial, 83–85
 - error in, 85–86
- Newton Method, 64, 74, 94, 100, 102, 103, 195, 415–417, 423
 - COMSOL, 298, 338
 - convergence, 65, 195, 417
 - damping factor, 65, 195
- Newton, Isaac, 3, 306
- Newton’s Second Law of Motion, 273, 274, 311
- Newtonian Fluid, 306, 311–313, 321, 329
- Non-Linear Models, 6
- Numerical Differentiation Formula (NDF), 96–98
 - coefficients, 97

- O**
- Ohm’s Law, 37, 119, 204, 205
- Ohm, Georg Simon, 204
- Ordinary Differential Equations (ODEs), 53
 - analytical solution methods, 29–34
 - boundary conditions, 32
 - characteristic equation, 32
 - homogeneous, 32
 - initial value, 32
 - linear, 31–34
 - non-homogeneous, 32
 - numerical solution methods, 35–36, 55–103
 - system of, 35
- Oscillator
 - coupled, 47–48, 395–397
 - damped, 30–31
 - Van der Pol, 36–37

- P**
- Parameters
 - defining in COMSOL, 80, 208, 211, 219, 230, 244, 253, 285, 295, 308, 316, 325, 331
 - scaling of, 21–23
- Parametric Sweep (COMSOL), 81, 209, 259, 287, 290, 297, 367, 368, 477, 487, 489, 492
- Partial Differential Equations (PDEs), 105–156
 - analytical solution methods, 123–139
 - boundary conditions, 120–123
 - COMSOL general form, 226
 - conservation law formulation, 117–119
 - diffusion, 118–120, 238
 - electric potential, 119, 206
- Pennes Bioheat Equation, 251
- Pennes, Harry, 251
- Pharmacokinetic Models
 - glucose–insulin interaction, 21–23, 48, 397–399

- Plato, 3
- Poisson, Denis Siméon, 281
- Poisson Equation, 120
- Poisson Ratio, 281
- Pouillet, Claude, 204
- Pouillet's Law, 204, 205, 227
- Predictor-Corrector Methods, 86–93
 - Adams-Bashforth-Moulton scheme, 86–93
- Principal Values, 271
- Q**
- Quadrature, numerical, 67, 192–194
 - Gaussian, 192–194
 - midpoint rule, 67
 - Simpson's rule, 67
- R**
- Resistance
 - access, 24, 128, 139, 211
 - electrical, 24, 204, 205, 284
 - hydraulic, 37, 324
- Resistivity, 24, 25, 148, 149, 204, 205, 227
- Respirator Strap Tension, 284–290
- Reynolds Number, 323, *see also* Fluid Mechanics
- Reynolds, Osborne, 323
- RF Ablation, 251–258, 260, 479–480
- Rivlin, Ronald Samuel, 293
- Root Mean Square (RMS), 250, 251
- Rule-Based Models, 12–14
 - cellular automata, 26–27, 386–390
- Runge-Kutta Methods, 66–73
 - classical fourth-order, 70
 - Dormand-Prince pair, 72
 - embedded methods, 72–73
 - local extrapolation, 72
 - variable step size, Matlab implementation, 70–72
- S**
- Scaling (of Models), 21–23
- Separation of Variables, 29, 123–139
- Shape Functions, 175–179, 185–190, 455, 458–460, 462–464, 466
 - bilinear, 185–186
 - biquadratic, 189–190
 - cubic, 196, 462–463
 - Hermite, 178–179
 - Lagrange, 177–178
 - linear, 176–177
 - linear triangular, 185–188
 - linear trilinear, 188
 - quadratic, 177–178
- Shear Modulus, 281
- SI units, 16, 19, 360
 - base quantities, 16
- Solid Mechanics, 263–302
 - Cauchy infinitesimal strain tensor, 278
 - Cauchy momentum equation, 275
 - Cauchy stress, 273–274
 - constitutive law, 281
 - elastostatics PDE, 275
 - Green's Strain Tensor, 277
 - hyperelasticity, 291–299
 - linear elasticity, 281
 - viscoelasticity, 290–291
- Solvers
 - ODE, 35–36, 55, 97–100, 102, 417–420
 - parametric sweep, 209, 287, 297, 487, 489
 - stationary, 285, 289, 301, 302
 - time-dependent, 80, 96, 99, 173, 223, 233, 240, 245, 256, 318, 320, 327, 338
- Source/Sink Terms, 80, 118–120, 206, 216, 217, 219, 222, 223, 228, 229, 232, 233, 248–251, 253, 256, 260, 318, 327, 334, 363, 364, 370, 376, 377, 479, 491
- Source Vector, 166
- Sparse Matrix, 170, 195, 449, 452
- Spring, 10–12, 30–31, 46–48, 51–53, 154–155, 291, 391–393, 395–397, 406–410, 438–440
- Stability
 - finite difference methods, 141–142, 144
 - ODE numerical methods, 61–64, 66, 70, 75, 77–78, 94, 96, 102, 103, 412–417, 429–430
- Stabilization (COMSOL), 245, 320
- Static Models, 7, 120, *see also* Stationary
- Stationary, 7, *see also* Static Models
- Stereolithography (STL), 359
- Stiffness Matrix, 97, 166–168, 170–172, 175, 176, 184, 191, 195, 455, 456, 459–461, 463, 466–470
- Stiff Systems, 62, 94, 98
- Stochastic Models, 7–9
- Stokes, Sir George Gabriel, 314, 323
- Strain, 275–281
 - bulk modulus, 293
 - deformation tensor, 276
 - deviatoric, 282–284, 291
 - displacement field, 277

- hydrostatic, 282–284
 - invariants, 292
 - isochoric right Cauchy–Green deformation tensor, 293
 - left Cauchy–Green deformation tensor, 276
 - principal stretch ratios, 292–293
 - pure extension, 280
 - right Cauchy–Green deformation tensor, 276
 - shear, 280–281, 294–299
 - strain energy, 283, 291–294
 - volumetric strain, 278
 - Strain Gauge, modelling of, 284–290
 - Strain Rate, 305–306, 312
 - shear rate, 329
 - Stress, 271–275
 - Cauchy stress, 273
 - deviatoric, 282–284, 291
 - hydrostatic, 282–284
 - shear, 18
 - state of stress, 273
 - symmetric tensor, 273
 - traction (or stress vector), 271–272
 - viscous, 306, 307
 - von Mises, 283, 284, 288, 289
 - von Mises Stress, 298
 - Symmetric Matrix, 170, 171
 - Systems Biology, 3, 343
- T**
- Taylor’s Theorem, 55–59
 - multivariate form, 58–59
 - univariate form, 55
 - Tensors, 263–265
 - conductivity tensor, 205, 218
 - dyadic components, 264
 - invariants, 268–271
 - principal axes, 271
 - principal values, 271
 - strain tensor, 277, 278
 - stress tensor, 272
 - symmetric, 205
 - transformation law, 266–268
- Tetrodotoxin (TTX), 51
 - Time-Dependent, 7, *see also* Dynamic Models
 - Torque, 333–335
 - Trapezoidal Method, 65–66
- U**
- Units
 - dimensional analysis, 16–19, 23–24, 382–383
 - use in COMSOL, 80, 81, 219, 224, 229, 233, 253, 257, 287, 288, 290, 299, 318, 327, 328, 334, 360–361, 364, 370, 378, 379
- V**
- Validation of Models, 4, 5
 - Variables, 7, 10, 21, 29, 30
 - COMSOL units of, 219, 229, 333, 334, 370
 - global (Matlab), 40
 - state, 35, 36, 38, 39, 41
 - Vector
 - cross product, 112, 114
 - dot product, 106
 - field, 106, 108
 - Verification of Models, 4, 5
 - Virtual Reality Modelling Language (VRML), 359
 - Viscoelasticity, 290–291, *see also* Solid Mechanics
 - Viscosity, 18–21, 24, 306, 313, 329–330
 - Volume Conductor, 204–207
 - Von Mises, Richard Edler, 283
- W**
- Windkessel Models, 48, 399–402
- Y**
- Young’s Modulus, 281
 - Young, Thomas, 281