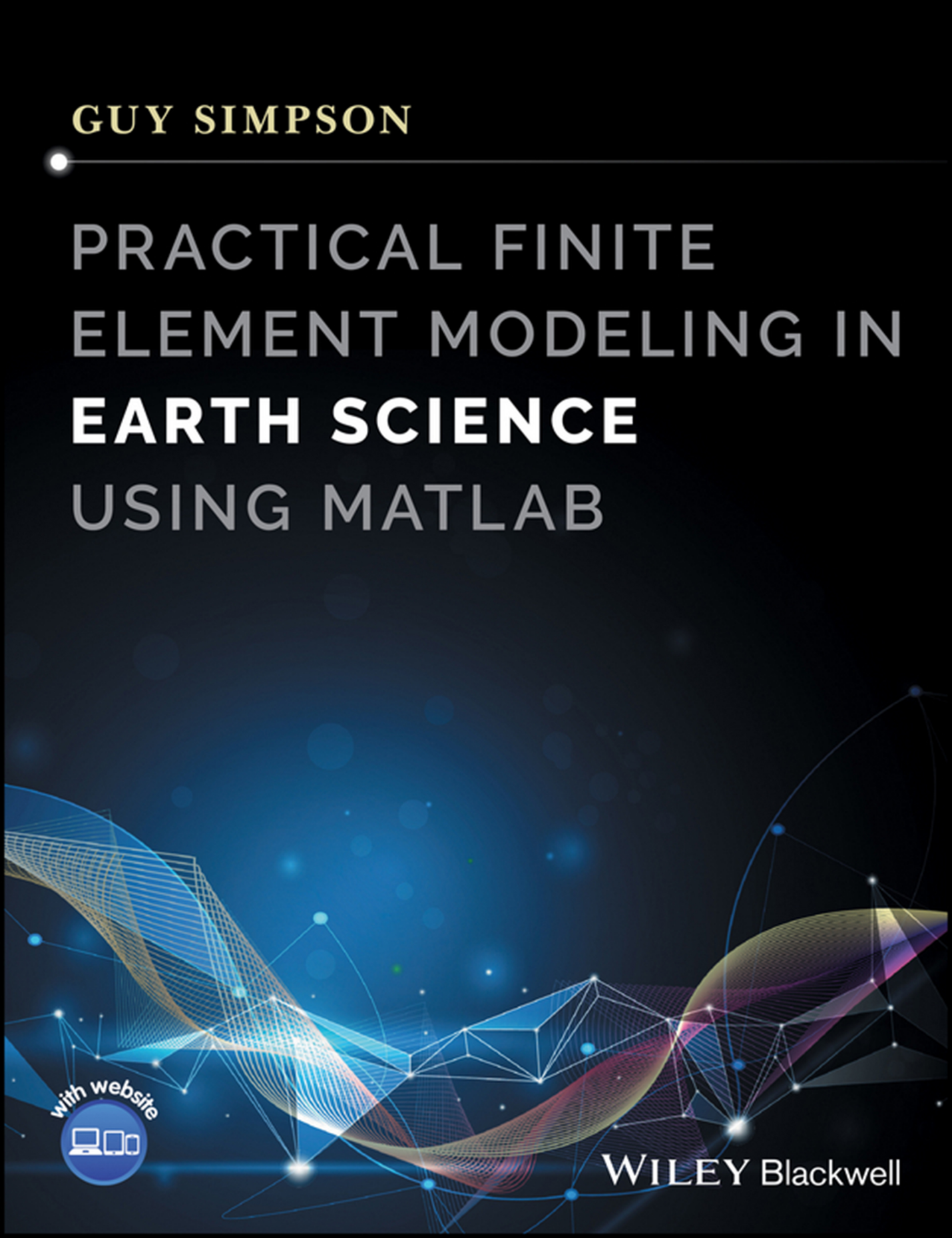
GUY SIMPSON

PRACTICAL FINITE ELEMENT MODELING IN EARTH SCIENCE USING MATLAB



WILEY Blackwell

Practical Finite Element Modeling in Earth Science Using Matlab

Practical Finite Element Modeling in Earth Science Using Matlab

Guy Simpson
Department of Earth Science
University of Geneva
Switzerland

WILEY Blackwell

Registered Office
John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, UK

Editorial Offices
111 River Street, Hoboken, NJ 07030, USA
9600 Garsington Road, Oxford, OX4 2DQ, UK
The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, UK

For details of our global editorial offices, customer services, and more information about Wiley products visit us at www.wiley.com.

Wiley also publishes its books in a variety of electronic formats and by print-on-demand. Some content that appears in standard print versions of this book may not be available in other formats.

Library of Congress Cataloging-in-Publication data applied for

ISBN: 9781119248620

Cover design: Wiley
Cover image: ©Nongkran_ch/iStockphoto

Set in 10/12pt Warnock by SPi Global, Pondicherry, India

10 9 8 7 6 5 4 3 2 1

All difficult things have their origin in that which is easy, and great things in that which is small.

—Lao Tzu

Brief Contents

Contents

Preface

Over the past few decades, mathematical models have become an increasingly important tool for Earth scientists to understand and make predictions about how our planet functions and evolves through time and space. These models often consist of partial differential equations (PDEs) that are discretized with a numerical method and solved on a computer. The most commonly used discretization methods are the finite difference method (FDM), the finite volume method, the finite element method (FEM), the discrete element method, the boundary element method, and various spectral methods. In theory, each method provides the same solution to the original PDEs. However, in practice, certain methods are better suited to certain problems than others. Often, one method dominates within any given discipline and in the Earth sciences, the FDM is the most prevalent, due to its simplicity. Although the FEM is arguably better suited to many Earth science problems—especially those with complicated geometry and/or material behavior—Earth scientists have been hesitant to wholeheartedly embrace this technique because it is regarded as being complicated to implement compared to other schemes. However, this perceived difficulty largely reflects the fact that most textbooks on this method are written by engineers or mathematicians for engineers who have a different educational background as compared to Earth scientists and who are interested in different applications. This is unfortunate because the FEM is a remarkably flexible and powerful tool with enormous potential in the Earth sciences that is no more difficult (or even easier) to implement than other numerical schemes.

The text is intended for students and researchers in Earth science, attempting their first steps with the FEM. It provides a practical guide on how the FEM can easily be used to solve various Earth science problems using Matlab. For the most part, I assume that the equations governing the processes of interest are known. Emphasis is on how one actually computes the solution using the FEM. The text does not deal in detail with benchmarking and interpretation of model results or with application of the model results to specific case studies. To guide readers, many sample finite element Matlab scripts are presented. These scripts are written with an emphasis on simplicity and clarity, not on modularity and efficiency. However, once the underlying concepts are clear, these standalone codes could easily be modularized, optimized, and transported to other more efficient languages such as Fortran or C/C++. It is assumed that the reader is familiar with linear algebra and PDEs and has basic programming experience. Some of these aspects are covered briefly in Chapter 1 and Appendix B. The text is directed toward graduate students, advanced undergraduates, and Earth science researchers. While the text is intended to show how finite element programs can be written from scratch, it should also be of interest to researchers who use existing FEM software (e.g., ABAQUS and COMSOL) but who want to know more about what goes on within the "black box". Because the level of the material presented is quite basic compared to other finite element texts, readers are strongly advised to consult other more advanced books once they understand the basics and see how programs are constructed

in practice. The following titles are good starting points: *The Finite Element Method* (three volumes, by Zienkiewicz and Taylor, 2000a, b, c) and *The Finite Element Method* (by Hughes, 2000).

This book is structured in two parts. Part I begins with a general introduction to numerical modeling before passing to a series of chapters that show how an archetypical mathematical model (i.e., the diffusion equation) is discretized with the FEM, programmed in Matlab, and solved on a computer. Each chapter builds on the previous one and introduces one (or more) key aspect of the FEM. Chapter 7 generalizes the concepts introduced in Chapters 1–6 by showing how the FEM can extended from single parabolic equations to systems of equations and also to elliptic and hyperbolic equations. By the end of Part I, the reader should understand the essentials of the FEM and be able to write their own Matlab scripts from scratch to solve the most commonly encountered PDEs in one dimension (1D), two dimension (2Ds), and three dimension (3Ds). This material can be taught as a one semester course on numerical modeling in Earth science for master and PhD students. Part II comprises a series of independent chapters, each of which focuses on how the FEM can be applied in different contexts in Earth science. The problems investigated are heat transfer in the crust, landscape evolution modeling, fluid flow in porous media, flexure, and deformation of Earth's crust. Although readers can choose to read only the chapter(s) that fall closest to their topic of interest, each chapter introduces a different aspect of the FEM, and so every chapter should be studied by readers interested in eventually mastering the technique.

I am very much indebted to Yuri Podladchikov who initially inspired my interest in the subject presented and who contributed in a major way to my understanding of the FEM. During the same period, I also benefited enormously from discussions and interaction with many other colleagues from the ETH in Zurich, including, in particular, Alan Thompson, Jamie Connolly, Neil Mancktelow, Jean-Pierre Burg, Steve Miller, Luigi Burlini, Katja Petrini, Taras Gerya, Stefan Schmalholz, Daniel Schmid, Boris Kaus, and Dave May. Line Probst from the University of Geneva is thanked for many comments and corrections on the text. I gratefully acknowledge financial support for my research from the Swiss National Science Foundation, the Department of Earth Science at the University of Geneva, and the Schmidheiny Foundation. Finally, I thank the encouragement and support from my parents, my wife Katja, and my children Luca, Lara, and Fabio.

Guy Simpson
Geneva

Symbols

Matlab variable	Dimensions	Description
b	[sdof, 1]	Global right-hand-side vector
bee	[nst, ntot]	Kinematic strain—displacement matrix
bcdof	[1 ndn]	Array containing list of equations where Dirichlet boundary conditions are imposed
ndn	scalar	Number of equations to which Dirichlet conditions are applied
bcval	[1 ndn]	Array containing fixed boundary values
bx0	[1 ny] in 2D	Arrays containing list of nodes on $x = 0$ boundary
bxn	[1 ny] in 2D	Arrays containing list of nodes on $x = lx$ boundary
coord	[nod, ndim]	Node coordinates for one element
dee	[nst, nst]	Material deformation matrix
der	[ndim nod]	Derivatives of shape functions in local coordinates, evaluated at an integration point
der_s	[ndim nod nip]	Derivatives of shape functions saved for all integration points
deriv	[ndim nod]	Derivatives of shape functions in global coordinates, evaluated at an integration point
detjac	scalar	Determinant of the Jacobian matrix
displ	[sdof, 1]	Global solution vector
displ0	[sdof, 1]	Global solution vector from previous time step
dt	scalar	Time increment
dx	scalar	Element dimension in the x-direction
F	[ntot, 1]	Element load vector
ff	[sdof, 1]	Global right-hand-side load vector
fun	[1 nod]	Shape functions, evaluated at an integration point
fun_s	[nip nod]	Shape functions saved for all integration points
g	[ntot 1]	Equation list for one element
g_coord	[ndim nn]	Node coordinates for entire mesh
g_g	[ntot nels]	Equation numbers of each element for entire mesh
g_num	[nod nels]	Node numbers of each element for entire mesh
invjac	[ndim, ndim]	Inverse of the Jacobian matrix
jac	[ndim, ndim]	Jacobian matrix
KM	[ntot, ntot]	Element stiffness matrix

Matlab variable	Dimensions	Description
`lhs`	`[sdof, sdof]`	Global stiffness matrix (sparse)
`lx`	scalar	Total domain length in the x-direction
`MM`	`[ntot, ntot]`	Element mass matrix
`ndim`	scalar	Number of spatial dimensions
`ndof`	scalar	Number of degrees of freedom (unknowns) per node
`nels`	scalar	Total number of elements in mesh
`nf`	`[ndof nn]`	Equation numbers on each node for entire mesh
`nip`	scalar	Number of Gauss–Legendre integration points for an element
`nod`	scalar	Number of nodes in one element
`nn`	scalar	Total number of nodes in mesh
`nst`	scalar	Number of stress (strain) components
`ntot`	scalar	Number of degrees of freedom per element (`=ndof*nod`)
`nx`	scalar	Number of mesh nodes in the x-direction
`num`	`[nod 1]`	Node list for one element
`phase`	`[1 nels]`	Phase index for each element
`points`	`[nip ndim]`	Positions of Gauss–Legendre integration points (in local coordinates)
`rhs`	`[sdof, sdof]`	Global right-hand-side matrix (sparse)
`sdof`	scalar	Total number of unknowns (equations) in the global system
`stress`	`[nst, 1]`	Stress components (e.g., σ_{xx}, σ_{zz}, and σ_{xz}) for a single integration point
`tensor`	`[nst nip nels]`	Stresses at each integration point for all elements
`wts`	`[1 nip]`	Weights of Gauss–Legendre integration points

About the Companion Website

This book is accompanied by a companion website:

www.wiley.com/go/simpson

This website includes:

- (.mat) files

Part I

The Finite Element Method with Matlab

Part I has two main purposes. The first purpose is to introduce readers to the Galerkin form of the finite element method (FEM), which is a numerical technique for discretizing partial differential equations (PDEs). The second purpose is to show practically how the resulting equations are programmed and solved on a computer using Matlab. Each chapter builds on the previous one and introduces one or more key concept. We will consider how the FEM is applied in one dimension (1D), two dimension (2D), and three dimension (3D), using a parabolic (diffusion) equation as an example. Chapter 7 generalizes the concepts and extends application of the FEM to systems of equations and to elliptic and hyperbolic problems.

Practical Finite Element Modeling in Earth Science Using Matlab, First Edition. Guy Simpson.

Companion website: www.wiley.com/go/simpson

1

Preliminaries

This chapter provides a short introduction to mathematical models consisting of systems of partial differential equations (PDEs) along with auxiliary (boundary and initial) conditions. We discuss how these equations can be solved, either exactly or using numerical methods. We also briefly consider the important issues of precision and stability of a numerical solution. A Matlab script is provided at the end of the chapter to enable readers to compare an analytical solution with its corresponding numerical approximation.

1.1 Mathematical Models

The application of the principles of conservation of mass, momentum, and energy combined with experimentally derived laws produces sets of PDEs that describe variations in velocity (or displacement), pressure, and temperature in space and time. When combined with boundary and initial conditions, these equations constitute mathematical models that can be solved and studied in a way somewhat similar to performing experiments in a laboratory. Whether a model is mathematical or analogue, both are simplified abstractions of reality. However, such models are useful because they can help isolate the influence of certain parameters or scenarios, study complex system interactions, and make predictions.

An example of a mathematical model that has important application in Earth science is the heat conduction equation, often more generally referred to as the diffusion equation. A complete derivation of the heat conduction equation is given in Appendix A. In one dimension (1D), the heat conduction equation can be written as follows:

$$\rho\, c\, \frac{\partial T}{\partial t} = k \frac{\partial^2 T}{\partial x^2} + A \tag{1.1}$$

Here, T is the temperature (K), x is the distance (m), t is the time (s), ρ is the rock density (kg m^{-3}), c is the specific heat capacity (J kg^{-1} K^{-1}), k is the thermal conductivity (W m^{-1} K^{-1}), and A is the rate of internal heat production per unit volume (J s^{-1} m^{-3}). In Equation 1.1, the temperature (the unknown) is referred to as the dependent variable, while t and x are known as independent variables. This type of equation is called a "partial differential equation" since the dependent variable depends on more than one independent variable. The physical parameters ρ, c, k, and A are assumed to be known. Obtaining a solution to the equation means finding the function $T(x, t)$ (i.e., T as a function of x and t) that satisfies the PDE.

Practical Finite Element Modeling in Earth Science Using Matlab, First Edition. Guy Simpson.

Companion website: www.wiley.com/go/simpson

More generally, the heat equation just introduced is also referred to mathematically as a parabolic (initial value) problem, which are typically of the form

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \tag{1.2}$$

Parabolic equations involve time-dependent behavior (term 1) and dissipation (terms 2 and 3), together which tend to smooth the solution with increasing time (at least for linear problems). Note that the signs in front of the second-order spatial derivatives on the right-hand side of 1.2 are necessarily positive; otherwise, the solutions grow rather than decay in time. Note also that the solution to parabolic equations depends on the initial value of the solution at $t = 0$ (hence the name initial value problems). The other two major classes of PDEs are elliptic (boundary value) problems and hyperbolic. Elliptic equations are typically associated with steady-state problems. Examples of elliptic equations are Poisson's equation,

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f \tag{1.3}$$

and Laplace's equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \tag{1.4}$$

which govern incompressible potential flow and steady heat transfer. Note that these equations don't involve any time derivatives and so their solutions depend only on the boundary conditions (hence the name boundary value problems) and any source (if present). Hyperbolic (initial value) PDEs involve time-dependent wave-like solutions. An example of a hyperbolic equation is the first-order wave equation

$$\frac{\partial u}{\partial t} = \frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} \tag{1.5}$$

Here, the first term accounts for time-dependent behavior, while the second and third terms translate the solution laterally without any dissipation. Hyperbolic equations are common in problems involving flowing fluids.

1.2 Boundary and Initial Conditions

The solution to a PDE is not unique until boundary conditions are imposed. Boundary conditions essentially "ground" the solution to some specific physical scenario. There are four types of boundary conditions commonly encountered in the solution of PDEs:

1) Dirichlet, where the value of the solution is imposed on the boundary
2) Neumann boundary conditions, where the derivative of the solution is imposed on the boundary
3) Robin boundary conditions, where one specifies some linear combination of the solution and its derivative
4) Periodic (or repeating) boundary conditions, where one assumes that the solution at one end of the model domain is equal to the solution at the other end

The number of boundary conditions necessary to determine a solution to a differential equation matches the order of the highest spatial derivative in the differential equation. For example, Equation 1.1 contains a second-order spatial derivative and so two boundary conditions must be

specified, one at each end of the domain. The equation also contains a first-order time derivative, so we must also provide an initial condition. This means we must define the value of T everywhere (over the entire domain) at $t = 0$. Equation 1.5 has only first-order spatial derivatives and so requires only one boundary condition in each direction. In this case, the boundary condition should be imposed at the end of the domain from where flow arrives, whereas the downstream end should be left unconstrained so that the flow can exit uninhibited.

1.3 Analytical Solutions

For relatively simple PDEs and for certain boundary conditions and initial conditions, it may be possible to find an exact (also known as a closed-form or analytical) solution. As an example, consider 1D heat transfer about a steadily creeping, narrow, planar, vertical fault. In this case, Equation 1.1 needs to be solved with A given by (e.g., see McKenzie and Brune, 1972)

$$A = \delta(x_0)\tau v \tag{1.6}$$

where τ is the (constant) shear stress (Pa) resolved on the fault plane, v is the fault slip rate (m s^{-1}), and $\delta(x_0)$ is the Dirac function, that is, ∞ when $x_0 = 0$, 0 when $x \neq 0$, and $\int_{-\infty}^{\infty} \delta(x_0)dx_0 = 1$. The initial temperature at $t = 0$ is assumed to be 0°C everywhere. The spatial domain extends horizontally from $-\infty$ to $+\infty$ on either side of the fault located at $x = 0$. The boundary conditions are that the first derivative of the temperature vanishes at $\pm\infty$. The exact solution to Equation 1.1 combined with 1.6 can be written down directly using the Green's function for this equation (Morse and Feshbach ,1953, p. 981). The solution is

$$T(x,t) = \frac{\tau v}{\kappa\sqrt{\pi}\rho c}\left(|x|\sqrt{\pi}\,\mathrm{erf}\left(\frac{|x|}{2t\sqrt{\kappa t}}\right) + 2t\exp\left(\frac{-|x|^2}{4t\kappa}\right)\sqrt{\frac{\kappa}{t}} - |x|\sqrt{\pi}\right) \tag{1.7}$$

where κ $(= k/(\rho c))$ is the thermal diffusivity (m^2 s^{-1}) and erf is the error function ($\mathrm{erf}(x) = 2/\pi \int_0^x \exp(-t^2)dt$). This solution can easily be evaluated exactly at any desired x and t once the values for the various physical parameters are specified (as done in the following).

1.4 Numerical Solutions

Although it is normally always desirable to obtain exact solutions to the PDE(s) being investigated, in practice this is often not possible. A closed-form solution may either not exist, or it may be too complicated to be of practical use. This may be due a number of factors, including nonlinearities in the governing equation, variable material properties, complicated geometries or boundary conditions, and so on. In such cases, one must resort to numerical methods that provide an approximate solution to the governing differential equation(s). Today, with powerful computers, many complicated problems can be solved quickly using numerical techniques.

The process of obtaining a computational solution consists of two stages shown schematically in Figure 1.1. The first stage converts the continuous PDE and auxiliary conditions (boundary and initial conditions) into a discrete system of algebraic equations. This first stage is called "discretization" and may be performed using various methods (one of which is the finite element method or FEM). The second stage involves solving the system of algebraic equations (normally performed on a computer,

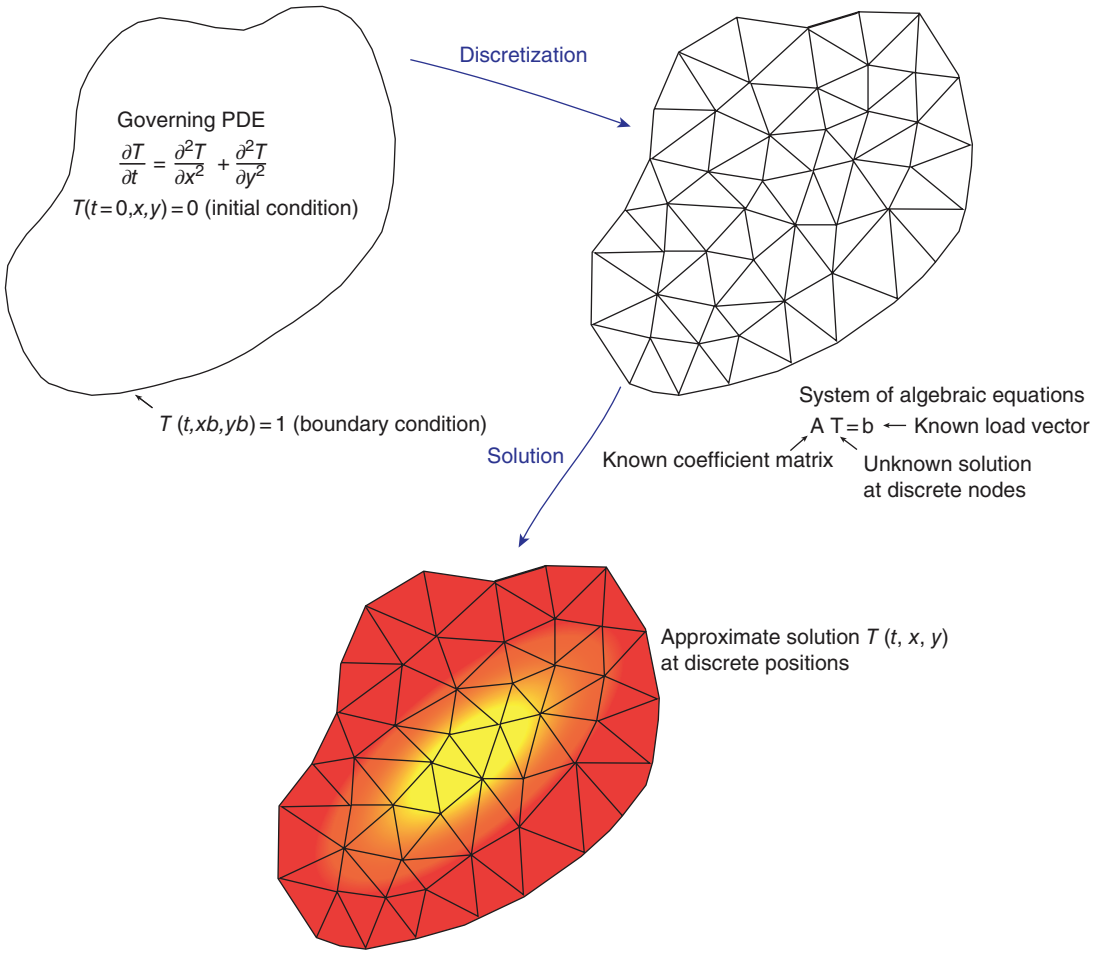


Figure 1.1 Major steps involved in obtaining a numerical solution to a PDE.

see Appendix B) to obtain an approximate solution to the original PDE. This second stage typically will involve some standard mathematical method such as Gaussian elimination.

Two important issues that must be considered when obtaining a numerical solution to PDEs are *error* and *stability*. All numerical methods introduce discretization errors, which in principle can be reduced by increasing the spatial and temporal resolution. This can be achieved by increasing the number of nodes (in time or space) where the solution is computed, or equivalently, by decreasing the spacing between nodes. It both cases, this should be performed without changing the total spatial or temporal extent of the model domain. Ideally, a numerical solution will converge to the exact solution as the resolution is increased. Even if an exact solution doesn't exist, one should always check that the numerical solution doesn't change significantly as the numerical resolution is changed, indicating that convergence has been achieved. Other errors may also arise (e.g., round-off errors produced during the solution of systems of linear algebraic equations), though these are usually small in comparison to discretization errors.

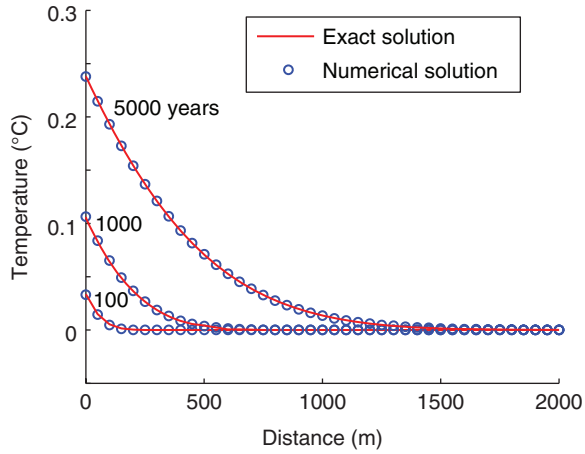


Figure 1.2 Comparison between the numerical (circles) and analytical solution (line, see Equation 1.7) for the temperature around a creeping fault after 100, 1000, and 5000 years (see Equations 1.1 and 1.6). The fault (located at $x = 0$) creep generates frictional heat that conducts outward into the surrounding rocks. Only the domain to the right of the fault is shown (the temperature is symmetrical about $x = 0$). The numerical solution is computed using the FEM. The Matlab script used to compute these results is provided at the end of the chapter.

The issue of stability concerns whether numerical errors, which are always present, decay or grow with time. A stable solution is one where the errors decay with time. An unstable solution is one where the errors grow with time, something that will eventually lead to large oscillations that have no physical meaning (i.e., they are simply numerical errors). Numerical methods are typically referred to as being either stable, unstable, or conditionally stable (meaning it can exhibit both behaviors depending on certain conditions). A stable method is an essential property of any numerical scheme. However, it is important to emphasize that a stable method can still be inaccurate. Thus, it is also important to assess the precision of a numerical solution. The best way this can be achieved is by directly comparing the numerical solution with an exact solution (as done in Figure 1.2). This approach is desirable because a numerical solution may look correct and may display the expected behavior but may be completely wrong (e.g., due to a simple erroneous factor in the numerical code). When an exact solution is not available, one should attempt to compare the numerical solution with other published numerical results.

Figure 1.2 shows a comparison between a numerical solution (computed using the FEM) to Equations 1.1 and 1.6, along with the analytical solution to the same equations (i.e., Equation 1.7). The Matlab code used to generate the figure is reproduced in Section 1.6. In this example, one sees that the agreement between the approximate and exact solutions is very good, indicating that the numerical solution is indeed a faithful representation of the original governing PDE. This comparison illustrates the importance of exact analytical solutions, since they provide a means of verifying the accuracy of a numerical solution.

1.5 Numerical Solution Methods

There are many different numerical methods available for solving PDEs, including the FEM, finite different method, finite volume method, boundary element method, discrete element method, and spectral methods. A comparison between three of these methods for a simple problem is given in Appendix C. In theory, each numerical method should provide the same (correct) solution to the original differential equation. However, in practice, some methods are better suited to certain types of equations and model geometries than others. Often the best approach is to choose the method that best suits the problem being investigated. This approach, however, requires considerable experience.