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,

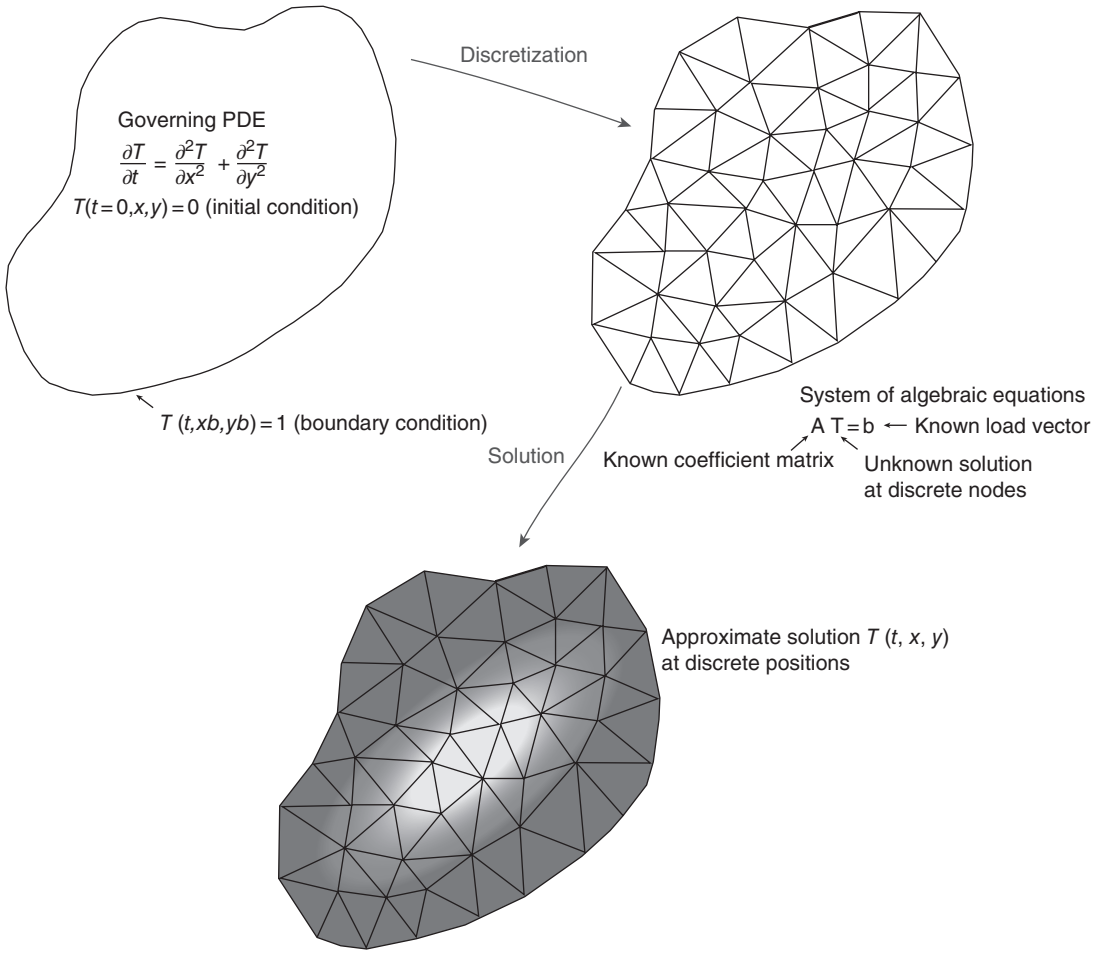

Figure 1.1 Major steps involved in obtaining a numerical solution to a PDE.

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.

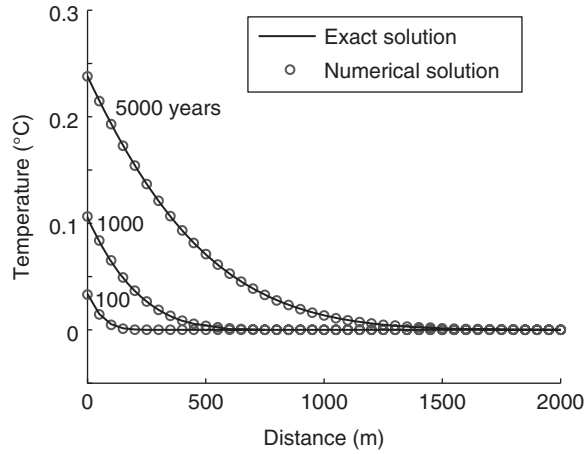

Figure 1.2 Comparison between the numerical (circles) and analytical solution (line, see Equation 1.7) for the temperature around a creeping fault after 100, 1000, and 5000 years (see Equations 1.1 and 1.6). The fault (located at $x = 0$) creep generates frictional heat that conducts outward into the 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.

The text focuses entirely on the FEM that is widely regarded as being one of the most powerful, flexible, and robust techniques, while also being mathematically sound (Hughes, 2000; Zienkiewicz and Taylor, 2000a). The technique is slightly harder to learn than, for example, the finite difference technique. However, as you will see later, the effort invested in initially learning the FEM pays off later in the wide range of problems that the method is capable of solving. The FEM is especially well suited (though not restricted) to solving mechanical problems and problems that involve complex shapes. Another advantage of programming with the FEM is that the main structure of the code remains the same even for very different physical problems. Thus, once you learn this basic structure, you can easily modify it to solve various problems with minimal effort.

1.6 Matlab Script

The following is a Matlab script used to compute the numerical and analytical solution for Equations 1.1 and 1.6 presented in Figure 1.2. The reader is advised to reinspect the script after reading Chapters 2 and 3. Details of the time-stepping scheme are presented in Appendix E.

```
%--------------------------------------------
% Program diffn1d.m
% 1-D FEM solution of diffusion equation
% and comparison with analytical solution
%--------------------------------------------

clear % clear memory from current workspace
seconds_per_yr = 60*60*24*365; % number of seconds in 1 year

% physical parameters
lx    = 2000 ;      % length of spatial domain
Cp    = 1e3  ;      % rock heat capacity J/kg/K
rho   = 2700 ;      % rock density
K     = 3.3  ;      % bulk thermal conductivity W/m/K
kappa = K/(Cp*rho); % thermal diffusivity
tau   = 10e6 ;      % shear stress resolved on fault (Pa)
udot  = 10e-3/seconds_per_yr ;          % fault slip rate (m/s)
dTdx  = (1/2)*tau*udot/(rho*Cp*kappa) ; % T gradient at fault

% numerical parameters
dt      = seconds_per_yr ;  % time step (s)
theta   = 1                 % time stepping parameter [0,1]
ntime   = 5000 ;            % number of time steps
nels    = 200 ;             % total number of elements
nod     = 2 ;               % number of nodes per element
nn      = nels+1            % total number of nodes
dx      = lx/nels ;         % element size
g_coord = [0:dx:lx] ;       % spatial domain (1-D mesh)

% explicit time stepping options
lumped_explicit = 'N';
if theta==0 % if fully explicit
    lumped_explicit = input('Would you like to lump the mass matrix? Y/N [N]:','s');
    if isempty(lumped_explicit)
      lumped_explicit = 'N';
    end
end
```

```
%  define boundary conditions
 bcdof = [  nn ]  ; % boundary nodes
 bcval = [  0 ]   ; % boundary values

% define connectivity and equation numbering
g_num      = zeros(nod,nels) ;
g_num(1,:) = [1:nn-1]  ;
g_num(2,:) = [2:nn]    ;

% initialise matrices and vectors
b      = zeros(nn,1);         % system rhs vector
lhs    = sparse(nn,nn);       % system lhs matrix
rhs    = sparse(nn,nn);       % system rhs matrix
displ  = zeros(nn,1);         % initial temperature (°C)
lumped_diag = zeros(nn,1) ;   % storage for lumped diagonal
%-------------------------------------------
% matrix assembly
%-------------------------------------------
 for iel=1:nels % loop over all elements
   num = g_num(:,iel)  ;           % retrieve equation number
   dx  = abs(diff(g_coord(num))) ; % length of element
   MM  = dx*[1/3  1/6 ; 1/6 1/3 ] ;% mass matrix
   KM  = [kappa/dx -kappa/dx ; -kappa/dx kappa/dx ];% diffn matrix
   if lumped_explicit=='N'
     lhs(num,num) = lhs(num,num) + MM/dt + theta*KM  ; % assemble lhs
     rhs(num,num) = rhs(num,num) + MM/dt - (1-theta)*KM  ; % assemble rhs
   else
     lumped_diag(num) = lumped_diag(num) + sum(MM)'/dt ; % lumped diagonal
     rhs(num,num) = rhs(num,num) + diag(sum(MM))/dt - (1-theta)*KM  ; % assemble rhs
    end
 end    % end of element loop
%-------------------------------------------

% time loop
t  = 0 ; % time
k  = 1 ; % counter
ii = [100 1000 5000]; % array used for plotting
for n=1:ntime
   n
   t  = t + dt ;                    % compute time
   b  = rhs*displ  ;                % form rhs load vector

   % impose boundary conditions
   lhs(bcdof,:) = 0 ;               % zero the relevent equations
   tmp = spdiags(lhs,0) ;           % store diagonal
   tmp(bcdof)=1 ;                   % place 1 on stored-diagonal
   lhs=spdiags(tmp,0,lhs);          % reinsert diagonal
   b(bcdof) = bcval ;               % set rhs

   b(1) = b(1) + dTdx*kappa ;       % add heat flux at left boundary

   if lumped_explicit=='N'
       displ = lhs \ b ; % solve system of equations
   else
       displ = b./lumped_diag ; % fully explicit, diagonalised solution
   end

   % evaluate analytical solution
```

```
    x      = g_coord ;
    term1  = abs(x).*sqrt(pi).*erf((1/2)*abs(x)./(sqrt(kappa./t).*t));
    term2  = 2*t.*exp(-(1/4)*abs(x).^2./(kappa*t)).*sqrt(kappa./t);
    term3  = abs(x)*sqrt(pi) ;
    term4  = (tau*udot)/(2*kappa*sqrt(pi)*rho*Cp);
    Texact = term4*(term1+term2-term3);

   % plotting
   if mod(n,ii(k))==0
     k = k+1;
     hold on
     figure(1)
     plot(g_coord,displ,'o-',g_coord,Texact,'r')
     title(['Time (kyr) = ', num2str(t/seconds_per_yr/1e3)])
     xlabel('Distance away from fault (m)')
     ylabel('Temperature (°C)')
     drawnow
     t/seconds_per_yr
     pause
   end
   hold off
end % end of time loop

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

1.7 Exercises

1) Depending on the values of the constant coefficients A, B, C, D, E, F, and G, the following PDE

$$A\frac{\partial^2 u}{\partial x^2} + B\frac{\partial^2 u}{\partial x \partial y} + C\frac{\partial^2 u}{\partial y^2} + D\frac{\partial u}{\partial x} + E\frac{\partial u}{\partial y} + Fu + G = 0$$

can be classified according to the following scheme (Garabedian, 1964):

Elliptic:	$B^2 - 4AC < 0$
Parabolic:	$B^2 - 4AC = 0$
Hyperbolic:	$B^2 - 4AC > 0$

Using this scheme, classify the following PDEs:

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

$$\frac{1}{c^2}\frac{\partial^2 T}{\partial t^2} = \frac{\partial^2 T}{\partial x^2}$$

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0$$

$$\frac{\partial T}{\partial t} + u\frac{\partial T}{\partial x} = 0$$

$$\frac{\partial T}{\partial t} + u\frac{\partial T}{\partial x} = \kappa \frac{\partial^2 T}{\partial x^2}$$

2) Match the following exact solutions

$$\phi = \sin(2\pi x - t)$$
$$\phi = \sin \pi x \cos \pi t$$
$$\phi = \sin(\pi x) e^{-\pi y}$$
$$\phi = \sin(\pi x) e^{-\pi^2 t}$$

with their corresponding PDEs:

$$\frac{\partial^2 \phi}{\partial t^2} = \frac{\partial^2 \phi}{\partial x^2}$$
$$\frac{\partial \phi}{\partial t} = \frac{\partial^2 \phi}{\partial x^2}$$
$$\frac{\partial \phi}{\partial t} + \frac{\partial \phi}{\partial x} = 0$$
$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 0$$

In each case, explain your reasoning. Write a matlab script to evaluate and compare the analyical solutions. In each case, assume the independent variables extend from 0 to 1. Make a list of the most important characteristics of each.

3) In 1D, the equation governing diffusion-advection of a passive scalar $T(x, t)$ is

$$\frac{\partial T}{\partial t} + u \frac{\partial T}{\partial x} = \alpha \frac{\partial^2 T}{\partial x^2}$$

where u is the flow velocity and α is the diffusivity. For the conditions

$$T(x < 0, t = 0) = 1$$
$$T(x > 0, t = 0) = 0$$
$$T(x = -2, t) = 1$$
$$T(x = 2, t) = 0$$

the exact solution (valid before the influence of the step reaches the boundary) is

$$T(x,t) = \frac{1}{2} - \frac{2}{\pi} \sum_{k=1}^{N} \sin\left((2k-1) \frac{\pi(x - ut)}{L} \right) \frac{\exp\left(-\alpha(2k-1)^2 \pi^2 t / L^2\right)}{2k-1}$$

Write a Matlab script to evaluate the exact solution for $L = 4$, $u = 1$, $\alpha = 0.1$, and t between 0 and 1. Study the effect of changing α and u.

4) Use the script listed in Section 1.6 to study the accuracy and stability of the numerical solution (with respect to the exact solution). Try a range of different values for `dx` (modified by varying the number of finite elements, `nels`) and `dt`. The parameter θ can be used to study different time-stepping schemes, ranging from fully implicit ($\theta = 1$) to fully explicit ($\theta = 0$) (see Appendix E). In addition, the explicit case can be solved more efficiently by diagonalizing the system stiffness matrix. Warning: For the boundary conditions considered, the finite element solution can only match the analytical solution until the temperature perturbation influences the lateral boundary. To study longer times, one would need to move the lateral boundary further away (i.e., increase `lx`) to avoid boundary effects.

Suggested Reading

E. A. Bender, *An Introduction to Mathematical Modeling,* Dover Publications Inc., New York, 2000.
C. A. J. Fletcher, *Computational Techniques for Fluid Dynamics,* Springer, Berlin, 2000.
A. Ismail-Zadeh and P. Tackley, *Computational Methods for Geodynamics*, Cambridge University Press, Cambridge, 2000.
R. Slinderland and L. Kump, *Mathematical Modeling of Earth's Dynamical Systems,* Princeton University Press, Princeton, NJ, 2011.
G. Strang, *Introduction to Applied Mathematics*, Wellesley-Cambridge Press, Cambridge, MA, 1986.