Chapter 7
Partial differential equations

In Chapter 4, we discussed numerical methods for solving initial-value and boundary-value problems given in the form of differential equations with one independent variable, that is, ordinary differential equations. Many of those methods can be generalized to study differential equations involving more than one independent variable, that is, partial differential equations. Many physics problems are given in the form of a second-order partial differential equation, elliptic, parabolic, or hyperbolic.

7.1 Partial differential equations in physics

In this chapter, we discuss numerical schemes for solving partial differential equations. There are several types of partial differential equations in physics. The Poisson equation

$$\nabla^2\phi(\mathbf{r}) = -\rho(\mathbf{r})/\epsilon_0 \tag{7.1}$$

for the electrostatic potential $\phi(\mathbf{r})$ at the position $\mathbf{r}$ under the given charge distribution $\rho(\mathbf{r})$ is a typical elliptic equation. The diffusion equation

$$\frac{\partial n(\mathbf{r}, t)}{\partial t} - \boldsymbol{\nabla} \cdot D(\mathbf{r})\boldsymbol{\nabla} n(\mathbf{r}, t) = S(\mathbf{r}, t) \tag{7.2}$$

for the concentration $n(\mathbf{r}, t)$ at the position $\mathbf{r}$ and time t under the given source $S(\mathbf{r}, t)$ is a typical parabolic equation. Here $D(\mathbf{r})$ is the diffusion coefficient at the position $\mathbf{r}$. The wave equation

$$\frac{1}{c^2}\frac{\partial^2 u(\mathbf{r}, t)}{\partial t^2} - \nabla^2 u(\mathbf{r}, t) = R(\mathbf{r}, t) \tag{7.3}$$

for the generalized displacement $u(\mathbf{r}, t)$ at the position $\mathbf{r}$ and time t under the given source $R(\mathbf{r}, t)$ is a typical hyperbolic equation. The time-dependent Schrödinger equation

$$-\frac{\hbar}{i}\frac{\partial \Psi(\mathbf{r}, t)}{\partial t} = \mathcal{H}\Psi(\mathbf{r}, t) \tag{7.4}$$

for the wavefunction $\Psi(\mathbf{r}, t)$ of a quantum system defined by the Hamiltonian $\mathcal{H}$, can be viewed as a diffusion equation under imaginary time.

All the above equations are linear if the sources and other quantities in the equations are not related to the solutions. There are also nonlinear equations in physics that rely heavily on numerical solutions. For example, the equations for fluid dynamics,

$$\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} + \frac{1}{\rho} \nabla P - \eta \nabla^2 \mathbf{v} = 0, \tag{7.5}$$

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \mathbf{v} = 0, \tag{7.6}$$

$$f(P, \rho) = 0, \tag{7.7}$$

require numerical solutions under most conditions. Here the first equation is the Navier–Stokes equation, in which $\mathbf{v}$ is the velocity, ρ the density, P the pressure, and η the kinetic viscosity of the fluid. The Navier–Stokes equation can be derived from the Newton equation for a small element in the fluid. The second equation is the continuity equation, which is the result of mass conservation. The third equation is the equation of state, which can also involve temperature as an additional variable in many cases. We will cover numerical solutions of the hydrodynamical equations in Chapter 9.

In this chapter, we will first outline an analytic scheme that can simplify numerical tasks drastically in many cases and is commonly known as the separation of variables. The basic idea of the separation of variables is to reduce a partial differential equation to several ordinary differential equations. The separation of variables is the standard method in analytic solutions of most partial differential equations. Even in the numerical study of partial differential equations, due to the limitation of computing resources, the separation of variables can serve the purpose of simplifying a problem to the point where it can be solved with the available computing resources. Later in the chapter, we will introduce several numerical schemes used mainly in solving linear partial differential equations.

7.2 Separation of variables

Before we introduce numerical schemes for partial differential equations, we now discuss an analytic method, that is, the *separation of variables*, for solving partial differential equations. In many cases, using a combination of the separation of variables and a numerical scheme increases the speed of computing and the accuracy of the solution. The combination of analytic and numerical methods becomes critical in cases where the memory and speed of the available computing resources are limited. This section is far from being a complete discussion; interested readers should consult a standard textbook, such as Courant and Hilbert (1989).

Here we will just illustrate how the method works. Each variable (time or one of the spatial coordinates) is isolated from the rest, and the solutions of the resulting ordinary differential equations for all the variables are obtained before they are combined into the general solution of the original partial differential equation. Boundary and initial conditions are then used to determine the unknown parameters left in the solution, which usually appear as the coefficients or eigenvalues.

Consider first the standing waves on a light, uniform string that has both ends ($x = 0$ and $x = L$) fixed. The wave equation is

$$\frac{\partial^2 u(x,t)}{\partial t^2} - c^2 \frac{\partial^2 u(x,t)}{\partial x^2} = 0, \tag{7.8}$$

where $u(x,t)$ is the displacement of the string at the location x and time t, and $c = \sqrt{T/\rho}$ is the phase speed of the wave, with T being the tension on the string and ρ the linear mass density of the string. The boundary condition for fixed ends is $u(0,t) = u(L,t) = 0$ in this case. Now if we assume that the solution is given by

$$u(x,t) = X(x)\Theta(t), \tag{7.9}$$

where $X(x)$ is a function of x only and $\Theta(t)$ is a function of t only, we have

$$\frac{\Theta''(t)}{\Theta(t)} = c^2 \frac{X''(x)}{X(x)} = -\omega^2, \tag{7.10}$$

after we substitute Eq. (7.9) into Eq. (7.8). Note that the introduced parameter (eigenvalue) ω must be independent of the position from the first ratio and independent of the time from the second ratio. Thus we must have

$$X''(x) = -\frac{\omega^2}{c^2} X(x) = -k^2 X(x), \tag{7.11}$$

where $k = \omega/c$ is determined from the boundary condition $X(0) = 0$ and $X(L) = 0$. We can view either ω or k as being the eigenvalue sought for the eigenvalue problem defined in Eq. (7.11) under the given boundary condition. The two independent, analytical solutions of Eq. (7.11) are $\sin kx$ and $\cos kx$. Then we have

$$X(x) = A \sin kx + B \cos kx. \tag{7.12}$$

From the boundary condition $X(0) = 0$, we must have $B = 0$; from $X(L) = 0$, we must have

$$k_n = \frac{n\pi}{L}, \tag{7.13}$$

with $n = 1, 2, \ldots, \infty$. Using this k_n in the equation for $\Theta(t)$, we obtain

$$\Theta(t) = C \sin \omega_n x + D \cos \omega_n x, \tag{7.14}$$

with

$$\omega_n = ck_n = \frac{n\pi c}{L}. \tag{7.15}$$

Combining the solutions for $X(x)$ and $\Theta(t)$, we obtain the general solution

$$u(x,t) = \sum_{n=1}^{\infty} (a_n \sin \omega_n t + b_n \cos \omega_n t) \sin k_n x, \tag{7.16}$$

where a_n and b_n are two sets of parameters that are determined by the initial displacement $u(x, 0) = u_0(x)$ and the initial velocity $\partial u(x, t)/\partial t|_{t=0} = v_0(x)$. Using $t = 0$ for $u(x, t)$ given in Eq. (7.16) and its partial derivative of time, we have

$$u_0(x) = \sum_{n=1}^{\infty} b_n \sin k_n x, \tag{7.17}$$

$$v_0(x) = \sum_{n=1}^{\infty} a_n \omega_n \sin k_n x. \tag{7.18}$$

Then we have

$$b_l = \frac{2}{L} \int_0^L u_0(x) \sin k_l x \, dx, \tag{7.19}$$

$$a_l = \frac{2}{\omega_l L} \int_0^L v_0(x) \sin k_l x \, dx, \tag{7.20}$$

after multiplying Eqs. (7.17) and (7.18) by $\sin k_l x$ and integrating over $[0, L]$. We have also used the orthogonal properties of $\sin k_n x$. This completes the solution of the problem.

What happens if the string is not light, and/or carries a mass density $\rho(x)$ that is not a constant? The separation of variables still works, but we need to solve the eigenvalue problem numerically. Let us take a closer look at the problem. The equation now becomes

$$\frac{\partial^2 u(x, t)}{\partial t^2} = \frac{T}{\rho(x)} \frac{\partial^2 u(x, t)}{\partial x^2} - g, \tag{7.21}$$

where $g = 9.80$ m/s^2 is the magnitude of the gravitational acceleration. This is an inhomogeneous equation. We can solve it with a general solution of the homogeneous equation $u_{\mathrm{h}}(x, t)$ (with $g = 0$) and a particular solution $u_{\mathrm{p}}(x, t)$ of the inhomogeneous equation.

First let us consider the solution of the homogeneous equation. We can still separate t from x by taking $u_{\mathrm{h}}(x, t) = X(x)\Theta(t)$; then we have

$$X''(x) = -\frac{\omega^2 \rho(x)}{T} X(x), \tag{7.22}$$

which can be solved as an eigenvalue problem with using either the scheme developed in Chapter 4 or that in Chapter 5. We will leave it as an exercise for the reader to find the eigenvalues ω_n for the loaded string.

For the particular solution of the inhomogeneous equation, we can simplify the problem by considering the static case, that is, $u_{\mathrm{p}}(x, t) = u_{\mathrm{p}}(x)$. Then we have

$$\frac{d^2 u_{\mathrm{p}}(x)}{dx^2} = \frac{g\rho(x)}{T}, \tag{7.23}$$

which is a linear equation set if we discretize the second-order derivative involved. This equation determines the shape of the string when it is in equilibrium. We will see the similar example of a loaded bench in Section 7.4.

After we obtain the general solution $u_{\rm h}(x, t)$ of the homogeneous equation and a particular solution of the inhomogeneous equation, for example, $u_{\rm p}(x)$ from the above static equation, we have the general solution of the inhomogeneous equation as

$$u(x, t) = u_{\rm h}(x, t) + u_{\rm p}(x, t). \tag{7.24}$$

The initial displacement $u_0(x)$ and initial velocity $v_0(x)$ are still needed to determined the remaining parameters in the general solution above.

This procedure of separating variables can also be applied to higher-dimensional systems. Let us consider the diffusion equation in an isotropic, three-dimensional, and infinite space, with a time-dependent source and a zero initial value, as an example. The diffusion equation under a time-dependent source and a constant diffusion coefficient is given by

$$\frac{\partial n(\mathbf{r}, t)}{\partial t} - D\nabla^2 n(\mathbf{r}, t) = S(\mathbf{r}, t). \tag{7.25}$$

The initial value $n(\mathbf{r}, 0) = 0$ will be considered first. Note that $t = 0$ in this case can be viewed as the moment of the introduction of the source. Then we can view the source as an infinite number of impulsive sources, each within a time interval $d\tau$, given by $S(\mathbf{r}, \tau)\delta(t - \tau)d\tau$. This way of treating the source is called the *impulse method*. The solution $n(\mathbf{r}, t)$ is then the superposition of the solution in each time interval $d\tau$, with

$$n(\mathbf{r}, t) = \int_0^t \eta(\mathbf{r}, t; \tau)\, d\tau, \tag{7.26}$$

where $\eta(\mathbf{r}, t; \tau)$ satisfies

$$\frac{\partial \eta(\mathbf{r}, t; \tau)}{\partial t} - D\nabla^2 \eta(\mathbf{r}, t; \tau) = S(\mathbf{r}, \tau)\delta(t - \tau), \tag{7.27}$$

with $\eta(\mathbf{r}, 0; \tau) = 0$. The impulse equation above is equivalent to a homogeneous equation

$$\frac{\partial \eta(\mathbf{r}, t; \tau)}{\partial t} - D\nabla^2 \eta(\mathbf{r}, t; \tau) = 0, \tag{7.28}$$

with the initial condition $\eta(\mathbf{r}, \tau; \tau) = S(\mathbf{r}, \tau)$. Thus we have transformed the original inhomogeneous equation with a zero initial value into an integral of a function that is a solution of a homogeneous equation with a nonzero initial value. We can generally write the solution of the inhomogeneous equation as the sum of a solution of the corresponding homogeneous equation and a particular solution of the inhomogeneous equation with the given initial condition satisfied. So if we find a way to solve the corresponding homogeneous equation with a nonzero initial value, we find the solution of the general problem.

Now let us turn to the separation of variables for a homogeneous diffusion equation with a nonzero initial value. We will assume that the space is an isotropic, three-dimensional, and infinite space, in order to simplify our discussion. Other finite boundary situations can be solved in a similar manner. We will also suppress the parameter τ to simplify our notation. The solution of $\eta(\mathbf{r}, t) = \eta(\mathbf{r}, t; \tau)$ can be assumed to be

$$\eta(\mathbf{r}, t) = X(\mathbf{r})\Theta(t), \tag{7.29}$$

where $X(\mathbf{r})$ is a function of $\mathbf{r}$ only and $\Theta(t)$ is a function of t only. If we substitute this assumed solution into the equation, we have

$$X(\mathbf{r})\Theta'(t) - D\Theta(t)\nabla^2 X(\mathbf{r}) = 0, \tag{7.30}$$

or

$$\frac{\Theta'(t)}{D\Theta(t)} = \frac{\nabla^2 X(\mathbf{r})}{X(\mathbf{r})} = -k^2, \tag{7.31}$$

where $k = |\mathbf{k}|$ is an introduced parameter (eigenvalue) that will be determined later. Now we have two separate equations,

$$\nabla^2 X(\mathbf{r}) + k^2 X(\mathbf{r}) = 0, \tag{7.32}$$

and

$$\Theta'(t) + \omega\Theta(t) = 0, \tag{7.33}$$

which are related by $\omega = Dk^2$. The spatial equation is now a standard eigenvalue problem with k^2 being the eigenvalue to be determined by the specific boundary condition. Because we have assumed that the space is isotropic, three-dimensional, and infinite, the solution is given by plane waves with

$$X(\mathbf{r}) = Ce^{i\mathbf{k}\cdot\mathbf{r}}, \tag{7.34}$$

where C is a general constant. Similarly, the time-dependent equation for $\Theta(t)$ is a standard initial-value problem with the solution

$$\Theta(t) = Be^{-\omega(t-\tau)}, \tag{7.35}$$

where $\omega = Dk^2$ and B is a coefficient to be determined by the initial condition of the problem. If we combine the spatial solution and the time solution, we have

$$\eta(\mathbf{r}, t) = \frac{1}{(2\pi)^{3/2}} \int A_{\mathbf{k}} e^{i\mathbf{k}\cdot\mathbf{r} - \omega(t-\tau)}\, d\mathbf{k}, \tag{7.36}$$

and we can obtain the coefficient $A_{\mathbf{k}}$ from the initial value of $\eta(\mathbf{r}, \tau) = S(\mathbf{r}, \tau)$ with the Fourier transform of the above equation as

$$A_{\mathbf{k}} = \frac{1}{(2\pi)^{3/2}} \int S(\mathbf{r}, \tau) e^{-i\mathbf{k}\cdot\mathbf{r}}\, d\mathbf{r}. \tag{7.37}$$

If we substitute the above result for $A_{\mathbf{k}}$ into the integral for $\eta(\mathbf{r}, t)$ and complete the integration over $\mathbf{k}$, we obtain

$$\eta(\mathbf{r}, t) = \frac{1}{\sqrt{4\pi Dt}} \int S(\mathbf{r}', \tau) e^{-(\mathbf{r}-\mathbf{r}')^2/4Dt}\, d\mathbf{r}', \tag{7.38}$$

which can be substituted into the integral expression of $n(\mathbf{r}, t)$. We then reach the final solution of the original problem,

$$n(\mathbf{r}, t) = \int_0^t \frac{d\tau}{\sqrt{4\pi D(t-\tau)}} \int S(\mathbf{r}', \tau) e^{-(\mathbf{r}-\mathbf{r}')^2/4D(t-\tau)}\, d\mathbf{r}'. \tag{7.39}$$

This integral can be evaluated numerically if the form of $S(\mathbf{r}, t)$ is given. We should realize that if the space is not infinite but confined by some boundary, the above closed form of the solution will be in a different but similar form, because the idea of the Fourier transform is quite general in the sense of the spatial eigenstates. In most cases, the spatial eigenstates form an orthogonal basis set which can be used for a general Fourier transform, as discussed in Chapter 6. It is worth pointing out that the above procedure is also similar to the Green's function method, that is, transforming the problem into a Green's function problem and expressing the solution as a convolution integral of Green's function and the source. For more discussions on the Green's function method, see Courant and Hilbert (1989).

For the wave equation with a source that varies over time and space, we can follow more or less the same steps to complete the separation of variables. The general solution for a nonzero initial displacement and/or velocity is a linear combination of the solution $u_{\mathrm{h}}(\mathbf{r}, t)$ of the homogeneous equation and a particular solution $u_{\mathrm{p}}(\mathbf{r}, t)$ of the inhomogeneous equation. For the homogeneous equation

$$\nabla^2 u_{\mathrm{h}}(\mathbf{r}, t) = \frac{1}{c^2}\frac{\partial^2 u_{\mathrm{h}}(\mathbf{r}, t)}{\partial t^2}, \tag{7.40}$$

we can assume that $u_{\mathrm{h}}(\mathbf{r}, t) = X(\mathbf{r})\Theta(t)$, and then we have $\Theta''(t) = -\omega_{\mathbf{k}}^2\Theta(t)$ and $\nabla^2 X(\mathbf{r}) = -k^2 X(\mathbf{r})$, with $\omega_{\mathbf{k}}^2 = c^2k^2$. If we solve the eigenvalue problem for the spatial part, we obtain the general solution

$$u_{\mathrm{h}}(\mathbf{r}, t) = \sum_{\mathbf{k}} \left(A_{\mathbf{k}} e^{-i\omega_{\mathbf{k}} t} + B_{\mathbf{k}} e^{i\omega_{\mathbf{k}} t}\right) u_{\mathbf{k}}(\mathbf{r}), \tag{7.41}$$

where $u_{\mathbf{k}}(\mathbf{r})$ is the eigenstate of the equation for $X(\mathbf{r})$ with the eigenvalue $\mathbf{k}$. The particular solution $u_{\mathrm{p}}(\mathbf{r}, t)$ can be obtained by performing a Fourier transform of time on both sides of Eq. (7.3). Then we have

$$\nabla^2 \tilde{u}_{\mathrm{p}}(\mathbf{r}, \omega) + \frac{\omega^2}{c^2}\tilde{u}_{\mathrm{p}}(\mathbf{r}, \omega) = -\tilde{R}(\mathbf{r}, \omega), \tag{7.42}$$

where $\tilde{u}_{\mathrm{p}}(\mathbf{r}, \omega)$ and $\tilde{R}(\mathbf{r}, \omega)$ are the Fourier transforms of $u_{\mathrm{p}}(\mathbf{r}, t)$ and $R(\mathbf{r}, t)$, respectively. We can expand $\tilde{u}_{\mathrm{p}}(\mathbf{r}, \omega)$ and $\tilde{R}(\mathbf{r}, \omega)$ in terms of the eigenstates $u_{\mathbf{k}}(\mathbf{r})$ as

$$\tilde{u}_{\mathrm{p}}(\mathbf{r}, \omega) = \sum_{\mathbf{k}} c_{\mathbf{k}} u_{\mathbf{k}}(\mathbf{r}) \tag{7.43}$$

and

$$\tilde{R}(\mathbf{r}, \omega) = \sum_{\mathbf{k}} d_{\mathbf{k}} u_{\mathbf{k}}(\mathbf{r}), \tag{7.44}$$

which give

$$c_{\mathbf{k}} = \frac{c^2 d_{\mathbf{k}}}{\omega_{\mathbf{k}}^2 - \omega^2} \tag{7.45}$$

from Eq. (7.42). If we put all these together, we obtain the general solution for the inhomogeneous equation with

$$u(\mathbf{r}, t) = u_{\mathrm{h}}(\mathbf{r}, t) + \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \left(\sum_{\mathbf{k}} \frac{d_{\mathbf{k}}}{k^2 - \omega^2/c^2} \right) e^{-i\omega t}\, d\omega, \tag{7.46}$$

where $d_{\mathbf{k}}$ is given by

$$d_{\mathbf{k}} = \int \tilde{R}(\mathbf{r}, \omega) u_{\mathbf{k}}^*(\mathbf{r})\, d\mathbf{r}, \tag{7.47}$$

which is a function of ω. We have assumed that $u_{\mathbf{k}}(\mathbf{r})$ are normalized and form an orthogonal basis set. Here $u_{\mathbf{k}}^*(\mathbf{r})$ is the complex conjugate of $u_{\mathbf{k}}(\mathbf{r})$. The initial condition can now be used to determine $A_{\mathbf{k}}$ and $B_{\mathbf{k}}$.

A useful result of the separation of variables is that we may then need to deal with an equation or equations involving only a single variable, so that all the methods we discussed in Chapter 4 become applicable.

7.3 Discretization of the equation

The essence of all numerical schemes for solving differential equations lies in the discretization of the continuous variables, that is, the spatial coordinates and time. If we use the rectangular coordinate system, we have to discretize $\partial A(\mathbf{r}, t)/\partial r_i$, $\partial^2 A(\mathbf{r}, t)/\partial r_i \partial r_j$, $\partial A(\mathbf{r}, t)/\partial t$, and $\partial^2 A(\mathbf{r}, t)/\partial^2 t$, where r_i or r_j is either x, y, or z. Sometimes we also need to deal with a situation similar to having a spatially dependent diffusion constant, for example, to discretize $\nabla \cdot D(\mathbf{r}) \nabla A(\mathbf{r}, t)$. Here $A(\mathbf{r}, t)$ is a generic function from one of the equations discussed at the beginning of this chapter. We can apply the numerical schemes developed in Chapter 3 for first-order and second-order derivatives for all the partial derivatives. Typically, we use the two-point or three-point formula for the first-order partial derivative, that is,

$$\frac{\partial A(\mathbf{r}, t)}{\partial t} = \frac{A(\mathbf{r}, t_{k+1}) - A(\mathbf{r}, t_k)}{\tau} \tag{7.48}$$

or

$$\frac{\partial A(\mathbf{r}, t)}{\partial t} = \frac{A(\mathbf{r}, t_{k+1}) - A(\mathbf{r}, t_{k-1})}{2\tau}. \tag{7.49}$$

We can also use the three-point formula

$$\frac{\partial^2 A(\mathbf{r}, t)}{\partial^2 t} = \frac{A(\mathbf{r}, t_{k+1}) - 2A(\mathbf{r}, t_k) + A(\mathbf{r}, t_{k-1})}{\tau^2} \tag{7.50}$$

for the second-order derivative. Here $t_k = t$ and $\tau = t_{k+1} - t_k = t_k - t_{k-1}$. The same formulas can be applied to the spatial variables as well, for example,

$$\frac{\partial A(\mathbf{r},t)}{\partial x} = \frac{A(x_{k+1},y,z,t) - A(x_{k-1},y,z,t)}{2h_x}, \tag{7.51}$$

with $h_x = x_{k+1} - x_k = x_k - x_{k-1}$, and

$$\frac{\partial^2 A(\mathbf{r},t)}{\partial^2 x} = \frac{A(x_{k+1},y,z,t) - 2A(x_k,y,z,t) + A(x_{k-1},y,z,t)}{h_x^2}. \tag{7.52}$$

The partial differential equations can then be solved at discrete points.

However, when we need to deal with a discrete mesh that is not uniform or with an inhomogeneity such as $\nabla \cdot D(\mathbf{r})\nabla A(\mathbf{r},t)$, we may need to introduce some other discretization scheme. Typically, we can construct a functional from the field of the solution in an integral form. The differential equation is recovered from functional variation. This is in the same spirit as deriving the Lagrange equation from the optimization of the action integral. The advantage here is that we can discretize the integral first and then carry out the functional variation at the lattice points. The corresponding difference equation results. Let us take the one-dimensional Poisson equation

$$\frac{d}{dx}\epsilon(x)\frac{d\phi(x)}{dx} = -\rho(x), \tag{7.53}$$

for $x \in [0, L]$, as an illustrative example. Here $\epsilon(x)$ is the electric permittivity that has a spatial dependence.

For simplicity, let us take the homogeneous Dirichlet boundary condition $\phi(0) = \phi(L) = 0$. We can construct a functional

$$U = \int_0^L \left\{ \frac{1}{2}\epsilon(x)\left[\frac{d\phi(x)}{dx}\right]^2 - \rho(x)\phi(x) \right\} dx, \tag{7.54}$$

which leads to the Poisson equation if we take

$$\frac{\delta U}{\delta \phi} = 0. \tag{7.55}$$

If we use the three-point formula for the first-order derivative in the integral and the trapezoid rule to convert the integral into a summation, we have

$$U \simeq \frac{1}{2h}\sum_k \epsilon_{k-1/2}(\phi_k - \phi_{k-1})^2 - h\sum_k \rho_k\phi_k \tag{7.56}$$

as the discrete representation of the functional. We have used $\phi_k = \phi(x_k)$ for notational convenience and have used $\epsilon_{k-1/2} = \epsilon(x_{k-1/2})$ as the value of $\epsilon(x)$ in the middle of the interval $[x_{k-1}, x_k]$. Now if we treat each discrete variable ϕ_k as an independent variable, the functional variation in Eq. (7.55) becomes a partial derivative

$$\frac{\partial U}{\partial \phi_k} = 0. \tag{7.57}$$

We then obtain the desired difference equation

$$\epsilon_{k+1/2}\phi_{k+1} - (\epsilon_{k+1/2} + \epsilon_{k-1/2})\phi_k + \epsilon_{k-1/2}\phi_{k-1} = -h^2\rho_k, \tag{7.58}$$

which is a discrete representation of the original Poisson equation. This scheme is extremely useful when the geometry of the system is not rectangular or when the coefficients in the equation are not constants. For example, we may want to study the Poisson equation in a cylindrically or spherically symmetric case, or diffusion of some contaminated materials underground where we have a spatially dependent diffusion coefficient. The stationary one-dimensional diffusion equation is equivalent to the one-dimensional Poisson equation if we replace $\epsilon(x)$ with $D(x)$ and $\rho(x)$ with $S(x)$ in Eqs. (7.53)–(7.58). The scheme can also be generalized for the time-dependent diffusion equation by replacing the source term $h^2 S_k$ with $h^2\{S_k(t_i) - [n_k(t_{i+1}) - n_k(t_i)]/\tau\}$. For higher spatial dimensions, the integral is a multidimensional integral. The aspects of time dependence and higher dimensions will be discussed in more detail later in this chapter.

Sometimes it is more convenient to deal with the physical quantities only at the lattice points. Equation (7.58) can be modified to such a form, with the quantity midway between two neighboring points replaced by the average of two lattice points, for example, $\epsilon_{k+1/2} \simeq (\epsilon_k + \epsilon_{k+1})/2$. The difference equation given in Eq. (7.58) can then be modified to

$$(\epsilon_{k+1} + \epsilon_k)\phi_{k+1} - 4\epsilon_k\phi_k + (\epsilon_{k-1} + \epsilon_k)\phi_{k-1} = -2h^2\rho_k, \tag{7.59}$$

which does not sacrifice too much accuracy if the permittivity $\epsilon(x)$ is a slowly-varying function of x.

7.4 The matrix method for difference equations

When a partial differential equation is discretized and given in a difference equation form, we can generally solve it with the matrix methods that we introduced in Chapter 5. Recall that we outlined how to solve the Hartree–Fock equation for atomic systems with the matrix method in Section 5.8 as a special example. In general, when we have a differential equation with the form

$$\mathcal{L}u(\mathbf{r}, t) = f(\mathbf{r}, t), \tag{7.60}$$

where $\mathcal{L}$ is a linear differential operator of the spatial and time variables, $u(\mathbf{r}, t)$ is the physical quantity to be solved, and $f(\mathbf{r}, t)$ is the source, we can always discretize the equation and put it into the matrix form

$$\mathbf{A}\mathbf{u} = \mathbf{b}, \tag{7.61}$$

where the coefficient matrix $\mathbf{A}$ is from the discretization of the operator $\mathcal{L}$, the column vector $\mathbf{u}$ is from the discrete values of $u(\mathbf{r}, t)$ excluding the boundary and initial points, and $\mathbf{b}$ is the known vector constructed from the discrete values of $f(\mathbf{r}, t)$ and the boundary and initial points of $u(\mathbf{r}, t)$. The time variable is usually separated with the Fourier transform first unless it has to be dealt with at different spatial points. Situations like this will be discussed in Section 7.7.

For example, the difference equations for the one-dimensional Poisson equation obtained in the preceding section can be cast into such a form. For the difference equation given in Eq. (7.59), we have

$$A_{ij} = \begin{cases} -4\epsilon_i & \text{for } i = j, \\ \epsilon_i + \epsilon_{i+1} & \text{for } i = j - 1, \\ \epsilon_i + \epsilon_{i-1} & \text{for } i = j + 1, \\ 0 & \text{otherwise,} \end{cases} \tag{7.62}$$

and

$$b_i = -2h^2 \rho_i. \tag{7.63}$$

This matrix representation of the difference equation resulting from the discretization of a differential equation is very general.

Let us illustrate this method further by studying an interesting example. Consider a situation in which a person is sitting at the middle of a long bench supported at both ends. Assume that the length of the bench is L. If we want to know the displacement of the bench at every point, we can establish the equation for the curvature at different locations from Newton's equation for each tiny slice of the bench. Then we obtain

$$YI \frac{d^2 u(x)}{dx^2} = f(x), \tag{7.64}$$

where $u(x)$ is the curvature (that is, the inverse of the effective radius of the curve at x), Y is Young's modulus of the bench, $I = t^3 w/3$ is a geometric constant, with t being the thickness and w the width of the bench, and $f(x)$ is the force density on the bench. Because both ends are fixed, the curvatures there are taken to be zero: that is, $u(0) = u(L) = 0$. If we discretize the equation with evenly spaced intervals, that is, $x_0 = 0, x_1 = h, \ldots, x_{n+1} = L$, via the three-point formula for the second-order derivative, we have

$$u_{i+1} - 2u_i + u_{i-1} = \frac{h^2 f_i}{YI} \tag{7.65}$$

for $i = 0, 1, \ldots, n + 1$, which is equivalent to

$$\begin{pmatrix} -2 & 1 & \cdots & \cdots & 0 \\ 1 & -2 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & 1 & -2 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}, \tag{7.66}$$

with $u_0 = u_{n+1} = 0$, as required by the boundary condition, and $b_i = h^2 f_i / YI$. We can easily solve the problem with the Gaussian elimination scheme or the LU decomposition scheme, as discussed in Section 2.4. For the specific problem discussed here, we have $e_i = c_i = 1$ and $d_i = -2$. Assume that the force distribution on the bench is given by

$$f(x) = \begin{cases} -f_0[e^{-(x-L/2)^2/x_0^2} - e^{-1}] - \rho g & \text{for } |x - L/2| \leq x_0, \\ -\rho g & \text{otherwise,} \end{cases} \tag{7.67}$$

with $f_0 = 200$ N/m and $x_0 = 0.25$ m, and that the bench has a length of 3.0 m, a width of 0.20 m, a thickness of 0.030 m, a linear density of $\rho = 3.0$ kg/m, and a Young's modulus of 1.0×10^9 N/m^2. Here $g = 9.80$ m/s^2 is the magnitude of the gravitational acceleration. Note that we have taken a truncated Gaussian form for the weight of the person distributed over the bench.

The following program uses the LU decomposition method introduced in Section 2.4 to solve the bench problem with the force distribution and parameters given above using the SI units.

```
// A program to solve the problem of a person sitting
// on a bench as described in the text.

import java.lang.*;
public class Bench {
  final static int n = 99, m = 2;
  public static void main(String argv[]) {
  double d[] = new double[n];
  double b[] = new double[n];
  double c[] = new double[n];
  double l = 3, l2 = l/2, h = l/(n+1), h2 = h*h;
  double x0 = 0.25, x2 = x0*x0, e0 = 1/Math.E;
  double rho = 3, g = 9.8, f0 = 200;
  double y = 1e9*Math.pow(0.03,3)*0.2/3;

 // Evaluate the coefficient matrix elements
    for (int i=0; i<n; ++i) {
      d[i] = -2;
      c[i] =  1;
      b[i] = -rho*g;
      double x = h*(i+1)-l2;
      if (Math.abs(x) < x0)
        b[i] -= f0*(Math.exp(-x*x/x2)-e0);
      b[i] *= h2/y;
    }

 // Obtain the solution of the curverture of the bench
    double u[] = tridiagonalLinearEq(d, c, c, b);

 // Output the result in every m time steps
    double x = h;
    double mh = m*h;
    for (int i=0; i<n; i+=m) {
      System.out.println(x + " " + 100*u[i]);
      x += mh;
    }
  }

// Method to solve the tridiagonal linear equation set.

  public static double[] tridiagonalLinearEq(double d[],
    double e[], double c[], double b[]) {...}
}
```

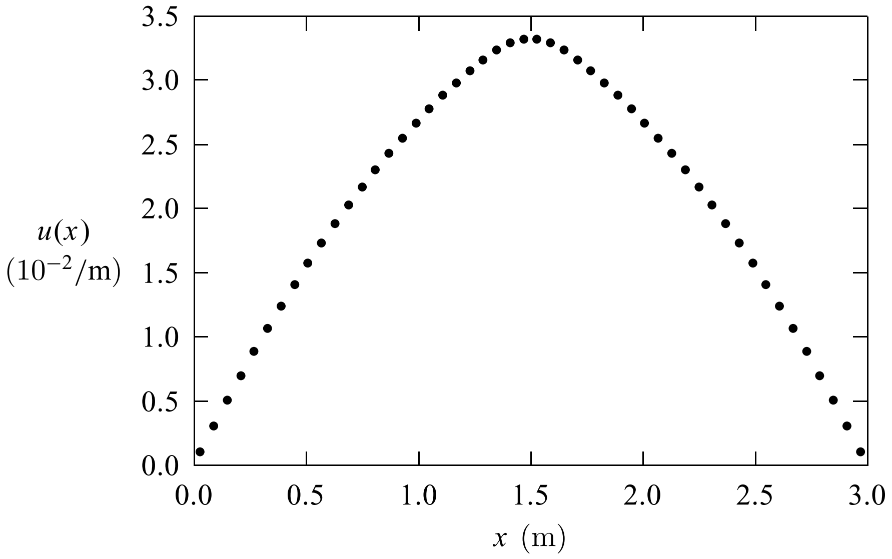

Fig. 7.1 The curvature of the bench, as evaluated with the program given.

The numerical result for the curvature of the bench calculated with the above program is illustrated in Fig. 7.1. Under these realistic parameters, the bench does not deviate from the original position very much. Even at the middle of the bench, the effective radius for the curve is still about 30 m.

A noticeable feature of this problem is that when the equation is discretized into a difference equation, it becomes extremely simple with a symmetric, tridiagonal coefficient matrix. In general, all one-dimensional problems with the same mathematical structure can be solved in the same fashion, such as the one-dimensional Poisson equation or the stationary one-dimensional diffusion equation. We will show later that equations with higher spatial dimensions, or with time dependence, can be solved in a similar manner. We need to note that the boundary condition as well as the coefficient matrix can become more complicated. However, if we split the coefficient matrix among the coordinates correctly, we can still preserve its tridiagonal nature, at least, in each step along each coordinate direction. As long as the coefficient matrix is tridiagonal, we can use the simple LU decomposition outlined in Section 2.4 to obtain the solution of the linear equation set. Sometimes we may also need to solve the linear problem with the full matrix that is no longer tridiagonal. Then the Gaussian elimination or the LU decomposition introduced in Chapter 5 can be used.

7.5 The relaxation method

As we have discussed, a functional can be constructed for the purpose of discretizing a differential equation. The procedure for reaching the differential equation is to optimize the functional. In most cases, the optimization is equivalent to the minimization of the functional. A numerical scheme can therefore be devised to find the numerical solution iteratively, as long as the functional associated with the equation does not increase with each new iteration. The solution of the differential equation is obtained when the functional no longer changes. This numerical scheme, formally known as the *relaxation method*, is extremely powerful for solving elliptic equations, including the Poisson equation, the stationary diffusion equation, and the spatial part of the wave equation after the separation of variables.

Let us first examine the stationary one-dimensional diffusion equation

$$-\frac{d}{dx}\left[D(x)\frac{dn(x)}{dx}\right] = S(x), \tag{7.68}$$

which can be written in a discrete form as

$$n_i = \frac{1}{D_{i+1/2} + D_{i-1/2}}\left(D_{i+1/2}n_{i+1} + D_{i-1/2}n_{i-1} + h^2 S_i\right), \tag{7.69}$$

as discussed in Section 7.3. It is the above equation that gives us the basic idea of the relaxation method: a guessed solution that satisfies the required boundary condition is gradually modified to satisfy the difference equation within the given tolerance. So the key is to establish an updating scheme that can modify the guessed solution gradually toward the correct direction, namely, the direction with the functional minimized. In practice, one uses the following updating scheme

$$n_i^{(k+1)} = (1-p)n_i^{(k)} + pn_i, \tag{7.70}$$

where $n_i^{(k)}$ is the solution of the kth iteration at the ith lattice point; n_i is given from Eq. (7.69) with the terms on the right-hand side calculated under $n_i^{(k)}$. Here p is an adjustable parameter restricted in the region $p \in [0, 2]$. The reason for such a restriction on p is that the procedure has to ensure the optimization of the functional during the iterations, and this is equivalent to having the solution of Eq. (7.70) approach the true solution of the diffusion equation defined in Eq. (7.68). More discussion on the quantitative aspect of p can be found in Young and Gregory (1988, pp. 1026–39). The points at the boundaries can be updated under the constraints of the boundary condition. Later in this section we will show how to achieve this numerically.

Here let us reexamine the bench problem solved in the preceding section with the LU decomposition. Equation (7.64) is equivalent to Eq. (7.68) with $D(x) = 1$ and $S(x) = -f(x)/(YI)$. The following example program is an implementation of the relaxation scheme for the problem of a person sitting on a bench.

```
// A program to solve the problem of a person sitting
// on a bench with the relaxation scheme.

import java.lang.*;
public class Bench2 {
  final static int n = 100, m = 2;
  public static void main(String argv[]) {
  double u[] = new double[n+1];
  double d[] = new double[n+1];
  double s[] = new double[n+1];
```

```
    double l = 3, l2 = l/2, h = l/n, h2 = h*h;
    double x0 = 0.25, x2 = x0*x0, e0 = 1/Math.E;
    double x = 0, rho = 3, g = 9.8, f0 = 200;
    double y = 1e9*Math.pow(0.03,3)*0.2/3;
    double u0 = 0.032, p = 1.5, del =1e-3;
    int nmax = 100;

  // Evaluate the source in the equation
      for (int i=0; i<=n; ++i) {
        s[i] = rho*g;
        x = h*i-l2;
        if (Math.abs(x) < x0)
          s[i] += f0*(Math.exp(-x*x/x2)-e0);
        s[i] *= h2/y;
      }
      for (int i=1; i<n; ++i) {
        x = Math.PI*h*i/l;
        u[i] = u0*Math.sin(x);
        d[i] = 1;
      }
      d[0] = d[n] = 1;
      relax(u, d, s, p, del, nmax);

  // Output the result in every m time step
      x = 0;
      double mh = m*h;
      for (int i=0; i<n; i+=m) {
        System.out.println(x + " " + 100*u[i]);
        x += mh;
      }
    }

 // Method to complete one step of relaxation.

    public static void relax(double u[], double d[],
      double s[], double p, double del, int nmax) {
      int n = u.length-1;
      double q = 1-p, fi = 0;
      double du = 2*del;
      int k = 0;

      while ((du>del) && (k<nmax)) {
        du = 0;
        for (int i=1; i<n; ++i) {
          fi = u[i];
          u[i] = p*u[i]
                +q*((d[i+1]+d[i])*u[i+1]
                +(d[i]+d[i-1])*u[i-1]+2*s[i])/(4*d[i]);
          fi = u[i]-fi;
          du += fi*fi;
        }
        du = Math.sqrt(du/n);
        k++;
      }
      if (k==nmax) System.out.println("Convergence not" +
        " found after " + nmax + " iterations");
    }
 }
```

Note that in the above method we have used the updated u_{i-1} on the right-hand side of the relaxation scheme, which is usually more efficient but less stable. We have also used $D_{i-1/2} \simeq (D_i + D_{i-1})/2$, $D_{i+1/2} \simeq (D_i + D_{i+1})/2$, and $D_{i+1/2} + D_{i-1/2} \simeq 2D_i$ in Eq. (7.69) in case the diffusion coefficient is given at the lattice points only. The multiplication of $S(x)$ by h^2 is done outside the method to keep the scheme more efficient. In general, the solution with the LU decomposition is faster and stabler than the solution with the relaxation method for the bench problem. But the situation reverses in the case of a higher-dimensional system.

The points at the boundaries are not updated in the method. For the Dirichlet boundary condition, we can just keep them constant. For the Neumann boundary condition, we can use either a four-point or a six-point formula for the first-order derivative to update the solution at the boundary. For example, if we have

$$\left.\frac{dn(x,t)}{dx}\right|_{x=0} = 0, \tag{7.71}$$

we can update the first point, n_0, by setting

$$n_0 = \frac{1}{3}(4n_1 - n_2), \tag{7.72}$$

which is the result of the four-point formula of the first-order derivative at $x = 0$,

$$\left.\frac{dn(x)}{dx}\right|_{x=0} \simeq \frac{1}{6h}(-2n_{-1} - 3n_0 + 6n_1 - n_2) = 0, \tag{7.73}$$

where $n_{-1} = n_1$ is the result of the zero derivative. A partial derivative is dealt with in the same manner. Higher accuracy can be achieved if we use a formula with more points. We have to use even-numbered point formulas because we want to have n_0 in the expression.

Now let us turn to the two-dimensional case, and we will use the Poisson equation

$$\nabla^2\phi(\mathbf{r}) = -\rho(\mathbf{r})/\epsilon_0 = -s(\mathbf{r}), \tag{7.74}$$

as an illustrative example. Here $s(\mathbf{r})$ is introduced for convenience. Now if we consider the case with a rectangular boundary, we have

$$\frac{\phi_{i+1j} + \phi_{i-1j} - 2\phi_{ij}}{h_x^2} + \frac{\phi_{ij+1} + \phi_{ij-1} - 2\phi_{ij}}{h_y^2} = -s_{ij}, \tag{7.75}$$

where i and j are used for the x and y coordinates and h_x and h_y are the intervals along the x and y directions, respectively. We can rearrange the above equation so that the value of the solution at a specific point is given by the corresponding values at the neighboring points,

$$\phi_{ij} = \frac{1}{2(1+\alpha)}\left[\phi_{i+1j} + \phi_{i-1j} + \alpha(\phi_{ij+1} + \phi_{ij-1}) + h_x^2 s_{ij}\right], \tag{7.76}$$

with $\alpha = (h_x/h_y)^2$. So the solution is obtained if the values of ϕ_{ij} at all the lattice points satisfy the above equation and the boundary condition. The relaxation scheme is based on the fact that the solution of the equation is approached iteratively. We first guess a solution that satisfies the boundary condition. Then we can update or improve the guessed solution with

$$\phi_{ij}^{(k+1)} = (1 - p)\phi_{ij}^{(k)} + p\phi_{ij}, \tag{7.77}$$

where p is an adjustable parameter close to 1. Here $\phi_{ij}^{(k)}$ is the result of the kth iteration, and ϕ_{ij} is obtained from Eq. (7.76) with $\phi_{ij}^{(k)}$ used on the right-hand side.

The second part of the above equation can be viewed as the correction to the solution, because it is obtained through the differential equation. The choice of p determines the speed of convergence. If p is selected outside the allowed range by the specific geometry and discretization, the algorithm will be unstable. Usually, the optimized p is somewhere between 0 and 2. In practice, we can find the optimized p easily by just running the program for a few iterations, say, ten, with different values of p. The convergence can be analyzed easily with the result from iterations of each choice of p. Mathematically, one can place an upper limit on p but in practice, this is not really necessary, because it is much easier to test the choice of p numerically.

7.6 Groundwater dynamics

Groundwater dynamics is very rich because of the complexity of the underground structures. For example, a large piece of rock may modify the speed and direction of flow drastically. The dynamics of groundwater is of importance in the construction of any underground structure.

In this section, because we just want to illustrate the power of the method, we will confine ourselves to the relatively simple case of a two-dimensional aquifer with a rectangular geometry of dimensions $L_x \times L_y$. Readers interested in learning more about the subject can find detailed discussions on groundwater modeling in Wang and Anderson (1982), Konikow and Reilly (1999), or Charbeneau (2000).

Steady groundwater flow is described by the so-called Darcy's law

$$\mathbf{q} = -\boldsymbol{\sigma} \cdot \nabla\phi, \tag{7.78}$$

where $\mathbf{q}$ is the *specific discharge* vector (the flux density), which is a measure of the volume of the fluid passing through a unit cross-sectional area perpendicular to the velocity of the flow in a unit of time. The average velocity $\mathbf{v}$ of the flow at a given position and time can be related to the specific discharge at the same position and time by $\mathbf{v} = \mathbf{q}/\beta$, where β is the *porosity*, which is the percentage of the empty space (voids) on a cross section perpendicular to the flow. Here $\boldsymbol{\sigma}$ is the *hydraulic conductivity* tensor of rank 2 (or 3×3 matrix); it has nine elements given by the specified porous medium and carries a unit of velocity. For example, for an isotropic medium, $\boldsymbol{\sigma}$ reduces to a diagonal matrix with three identical elements σ, and Darcy's law is simplified to $\mathbf{q} = -\sigma\nabla\phi$. The scalar field ϕ is referred to as the *head*, which is a measure of relative energy of the water in a standpipe from the datum (zero point). The head at elevation z, measured from the datum, can be related to the hydraulic pressure P and speed of the flow at the same elevation as

$$\phi \simeq \frac{v^2}{2g} + \frac{P}{\rho g} + z \simeq \frac{P}{\rho g} + z, \tag{7.79}$$

where ρ is the density of the fluid, g is the magnitude of the gravitational acceleration, and $v \simeq 0$ is the speed of the flow. Note that Darcy's law here is analogous to Ohm's law for electric conduction or Fourier's law for thermal conduction. For an ideal fluid under the steady flow condition, $\rho g\phi$ is a constant, a consequence of Bernoulli's equation.

If we combine the above equation with the continuity equation resulting from mass conservation (volume conservation if ρ is constant), we have

$$\mu\frac{\partial\phi(\mathbf{r},t)}{\partial t} - \nabla\cdot\boldsymbol{\sigma}\cdot\nabla\phi(\mathbf{r},t) = f(\mathbf{r},t), \tag{7.80}$$

where μ is the *specific storage*, a measure of the influence on the rate of head change, and f is the rate of *infiltration*. For an isotropic, steady flow, the above equation reduces to a generalized Poisson equation

$$\nabla^2\phi(\mathbf{r}) = -f(\mathbf{r})/\sigma, \tag{7.81}$$

which describes the groundwater dynamics reasonably well in most cases. When there is no infiltration, the above equation reduces to its simplest form

$$\nabla^2\phi(\mathbf{r}) = 0, \tag{7.82}$$

which is the Laplace equation for the head.

Now let us demonstrate how one can solve the groundwater dynamics problem by assuming that we are dealing with a rectangular geometry with nonuniform conductivity and nonzero infiltration. Boundary conditions play a significant role in determining the behavior of groundwater dynamics. We discretize the equation with the scheme discussed in Section 7.3, and the equation becomes

$$\begin{aligned}\phi_{ij} = {} & \frac{1}{4(1+\alpha)\sigma_{ij}}\{(\sigma_{i+1j}+\sigma_{ij})\phi_{i+1j} + (\sigma_{ij}+\sigma_{i-1j})\phi_{i-1j} \\ & + \alpha[(\sigma_{ij+1}+\sigma_{ij})\phi_{ij+1} + (\sigma_{ij}+\sigma_{ij-1})\phi_{ij-1}] + 2h_x^2 f_{ij}\},\end{aligned} \tag{7.83}$$

where $\alpha = (h_x/h_y)^2$. The above difference equation can form the basis for an iterative approach under the relaxation method with

$$\phi_{ij}^{(k+1)} = (1-p)\phi_{ij}^{(k)} + p\phi_{ij}, \tag{7.84}$$

where $\phi_{ij}^{(k)}$ is the value of the kth iteration and ϕ_{ij} is evaluated from Eq. (7.83) with the right-hand side evaluated under $\phi_{ij}^{(k)}$.

We illustrate this procedure by studying an actual example, with $L_x = 1000$ m, $L_y = 500$ m, $\sigma(x, y) = \sigma_0 + ay$ with $\sigma_0 = 1.0$ m/s and $a = -0.04$ s^{-1}, and $f(x, y) = 0$. The boundary condition is $\partial\phi/\partial x = 0$ at $x = 0$ and $x = 1000$ m, $\phi = \phi_0$ at $y = 0$, and $\phi = \phi_0 + b\cos(x/L_x)$ at $y = 500$ m, with $\phi_0 = 200$ m and $b = -20$ m. The four-point formula for the first-order derivative is used to ensure zero partial derivatives for two of the boundaries. The following program is an implementation of the relaxation method to this groundwater dynamics problem.

```
// An example of studying the 2-dimensional groundwater
// dynamics through the relaxation method.

import java.lang.*;
public class Groundwater {
  final static int nx = 100, ny = 50, ni = 5;
  public static void main(String argv[]) {
    double sigma0 = 1, a = -0.04, phi0 = 200, b = -20;
    double lx = 1000, hx = lx/nx, ly = 500, hy =ly/ny;
    double phi[][] = new double[nx+1][ny+1];
    double sigma[][] = new double[nx+1][ny+1];
    double f[][] = new double[nx+1][ny+1];
    double p = 0.5;

 // Set up boundary values and a trial solution
    for (int i=0; i<=nx; ++i) {
      double x = i*hx;
      for (int j=0; j<=ny; ++j) {
        double y = j*hy;
        sigma[i][j]  = sigma0+a*y;
        phi[i][j] = phi0+b*Math.cos(Math.PI*x/lx)*y/ly;
        f[i][j] = 0;
      }
    }
    for (int step=0; step<ni; ++step) {

   // Ensure boundary conditions by 4-point formula
      for (int j=0; j<ny; ++j) {
         phi[0][j] = (4*phi[1][j]-phi[2][j])/3;
         phi[nx][j] = (4*phi[nx-1][j]-phi[nx-2][j])/3;
      }
      relax2d(p, hx, hy, phi, sigma, f);
    }

 // Output the result
    for (int i=0; i<=nx; ++i) {
      double x = i*hx;
      for (int j=0; j<=ny; ++j) {
        double y = j*hy;
        System.out.println(x + " " + y + " "
          + phi[i][j]);
      }
    }
  }

// Method to perform a relaxation step in 2D.

  public static void relax2d(double p, double hx,
```

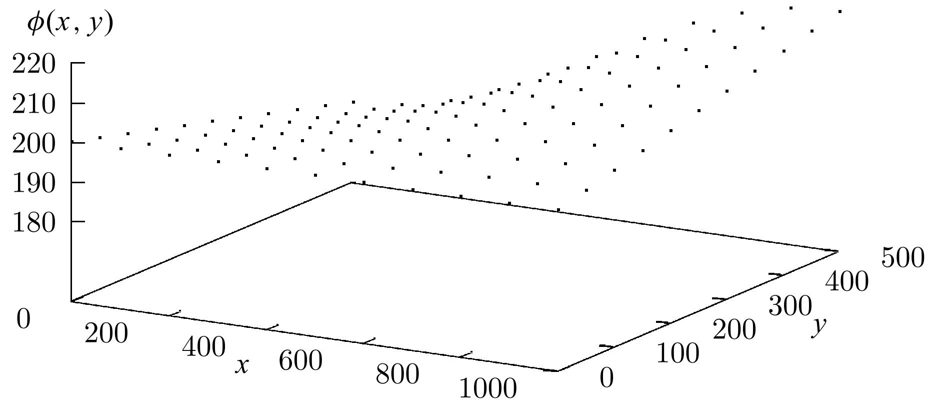

Fig. 7.2 The head of the groundwater obtained from the program given in the text. Here x, y, and $\phi(x, y)$ are all plotted in meters.

```
      double hy, double u[][], double d[][],
      double s[][]) {
      double h2 = hx*hx, a = h2/(hy*hy),
        b = 1/(4*(1+a)), ab = a*b, q = 1-p;
      for (int i=1; i<nx; ++i) {
        for (int j = 1; j <ny; ++j) {
          double xp = b*(d[i+1][j]/d[i][j]+1);
          double xm = b*(d[i-1][j]/d[i][j]+1);
          double yp = ab*(d[i][j+1]/d[i][j]+1);
          double ym = ab*(d[i][j-1]/d[i][j]+1);
          u[i][j] = q*u[i][j]+p*(xp*u[i+1][j]
                   +xm*u[i-1][j]+yp*u[i][j+1]
                   +ym*u[i][j-1]+h2*s[i][j]);
        }
      }
    }
  }
```

Note that we have used the four-point formula to update the boundary points at $x = 0$ and $x = 1000$ m to ensure zero partial derivatives there. We can use the six-point formula to improve the accuracy if needed. The output of the above program is shown in Fig. 7.2.

The transient state of groundwater dynamics involves the time variable, and is similar to the situations that we will discuss in the next two sections. Dynamics involving time can be solved with the combination of the discrete scheme discussed in the preceding two sections for the spatial variables and the scheme discussed in Chapter 4 for the initial-value problems. It is interesting that the time evolution involved in the problem is almost identical to the iterations of the relaxation scheme. A stabler scheme requires the use of the tridiagonal-matrix scheme discussed in Section 7.4.

7.7 Initial-value problems

A typical initial-value problem can be either the time-dependent diffusion equation or the time-dependent wave equation. Some initial-value problems are nonlinear equations, such as the equation for a stretched elastic string or the Navier–Stokes equation in fluid dynamics. We can, in most cases, apply the Fourier transform for the time variable of the equation to reduce it to a stationary equation, which can be solved by the relaxation method discussed earlier in this chapter. Then the time dependence can be obtained with an inverse Fourier transform after the solution of the corresponding stationary case is obtained.

For equations with higher-order time derivatives, we can also redefine the derivatives as new variables in order to convert the equations to ones with only first-order time derivatives, as we did in Chapter 4. For example, we can redefine the first-order time derivative in the wave equation, that is, the velocity

$$v(\mathbf{r}, t) = \frac{\partial u(\mathbf{r}, t)}{\partial t} \tag{7.85}$$

as a new variable. Then we have two coupled first-order equations,

$$\frac{\partial u(\mathbf{r}, t)}{\partial t} = v(\mathbf{r}, t), \tag{7.86}$$

$$\frac{1}{c^2}\frac{\partial v(\mathbf{r}, t)}{\partial t} = \nabla^2 u(\mathbf{r}, t) + R(\mathbf{r}, t), \tag{7.87}$$

which describe the same physics as the original second-order wave equation. Note that the above equation set now has a mathematical structure similar to that of a first-order equation such as the diffusion equation

$$\frac{\partial n(\mathbf{r}, t)}{\partial t} = \boldsymbol{\nabla} \cdot D(\mathbf{r})\boldsymbol{\nabla} n(\mathbf{r}, t) + S(\mathbf{r}, t). \tag{7.88}$$

This means we can develop numerical schemes for equations with first-order time derivatives only. In the case of higher-order time derivatives, we will always introduce new variables to reduce the higher-order equation to a first-order equation set. In the next chapter, however, we will introduce some numerical algorithms designed to solve second-order differential equations directly. As one can see, after discretization of the spatial variables, we have practically the same initial-value problem as that discussed in Chapter 4. However, there is one more complication. The specific scheme used to discretize the spatial variables as well as the time variable will certainly affect the stability and accuracy of the solution. Even though it is not the goal here to analyze all aspects of various algorithms, we will still make a comparison among the most popular algorithms with some actual examples in physics and discuss specifically the relevant aspects of the instability under the spatial and time intervals adopted.

In order to analyze the stability of the problem, let us first consider the one-dimensional diffusion equation

$$\frac{\partial n(x, t)}{\partial t} = D\frac{\partial^2 n(x, t)}{\partial^2 x} + S(x, t). \tag{7.89}$$

If we discretize the first-order time derivative by means of the two-point formula with an interval τ and the second-order spatial derivative by means of the three-point formula with an interval h, we obtain a difference equation

$$n_i(t+\tau) = n_i(t) + \gamma[n_{i+1}(t) + n_{i-1}(t) - 2n_i(t)] + \tau S_i(t), \tag{7.90}$$

which is the result of the Euler method, equivalent to that for the initial-value problems introduced in Chapter 4. Here $\gamma = D\tau/h^2$ is a measure of the relative sizes between the space and the time intervals. Note that we have used $n_i(t) = n(x_i, t)$ for notational convenience. So the problem is solved if we know the initial value $n(x, 0)$ and the source $S(x, t)$. However, this algorithm is unstable if γ is significantly larger than $1/2$. We can show this by examining the case with $x \in [0, 1]$ and $n(0, t) = n(L, t) = 0$. For detailed discussions, see Young and Gregory (1988, pp. 1078–84).

A better scheme is the Crank–Nicolson method, which modifies the Euler method by using the average of the second-order spatial derivative and the source at t and $t + \tau$ on the right-hand side of the equation, resulting in

$$n_i(t+\tau) = n_i(t) + \frac{1}{2}\{[H_i n_i(t) + \tau S_i(t)] + [H_i n_i(t+\tau) + \tau S_i(t+\tau)]\}, \tag{7.91}$$

where we have used

$$H_i n_i(t) = \gamma[n_{i+1}(t) + n_{i-1}(t) - 2n_i(t)] \tag{7.92}$$

to simplify the notation. The implicit iterative scheme in Eq. (7.91) can be rewritten into the form

$$(2 - H_i)n_i(t+\tau) = (2 + H_i)n_i(t) + \tau[S_i(t) + S_i(t+\tau)], \tag{7.93}$$

which has all the unknown terms at $t + \tau$ on the left. More importantly, Eq. (7.93) is a linear equation set with a tridiagonal coefficient matrix, which can easily be solved as discussed in Section 2.4 for the cubic-spline approximation, or as in Section 7.4 for the problem of a person sitting on a bench. We can also show that the algorithm is stable for any γ and converges as $h \to 0$, and that the error in the solution is on the order of h^2 (Young and Gregory, 1988).

However, the above tridiagonal matrix does not hold if the system is in a higher-dimensional space. There are two ways to deal with this problem in practice. We can discretize the equation in the same manner and then solve the resulting linear equation set with some other methods, such as the Gaussian elimination scheme or a general LU decomposition scheme for a full matrix. A more practical approach is to deal with each spatial coordinate separately. For example, if we are dealing with the two-dimensional diffusion equation, we have

$$H_{ij} n_{ij}(t) = (H_i + H_j) n_{ij}(t), \tag{7.94}$$

with

$$H_i n_{ij}(t) = \gamma_x[n_{i+1j}(t) + n_{i-1j}(t) - 2n_{ij}(t)], \tag{7.95}$$

$$H_j n_{ij}(t) = \gamma_y[n_{ij+1}(t) + n_{ij-1}(t) - 2n_{ij}(t)]. \tag{7.96}$$

Here $\gamma_x = D\tau/h_x^2$ and $\gamma_y = D\tau/h_y^2$. The decomposition of H_{ij} into H_i and H_j can be used to take one half of each time step along the x direction and the other half along the y direction with

$$(2 - H_j)n_{ij}\left(t + \frac{\tau}{2}\right) = (2 + H_j)n_{ij}(t) + \frac{\tau}{2}\left[S_{ij}(t) + S_{ij}\left(t + \frac{\tau}{2}\right)\right], \tag{7.97}$$

and

$$\begin{aligned}(2 - H_i)n_{ij}(t + \tau) = (2 + H_i)n_{ij}\left(t + \frac{\tau}{2}\right) \\ + \frac{\tau}{2}\left[S_{ij}\left(t + \frac{\tau}{2}\right) + S_{ij}(t + \tau)\right],\end{aligned} \tag{7.98}$$

which is a combined implicit scheme developed by Peaceman and Rachford (1955). As we can see, in each of the above two steps, we have a tridiagonal coefficient matrix. We still have to be careful in using the Peaceman–Rachford algorithm. Even though it has the same accuracy as the Crank–Nicolson algorithm, the convergence in the Peaceman–Rachford algorithm can sometimes be very slow in practice. We should compare the Peaceman–Rachford algorithm with the Crank–Nicolson method in test runs before deciding which one to use in a specific problem. We can easily monitor the convergence of the algorithm by changing h_x and h_y. For more discussions on and comparisons of various algorithms, see Young and Gregory (1988).

7.8 Temperature field of a nuclear waste rod

The temperature increase around nuclear waste rods is not only an interesting physics problem but also an important safety question that has to be addressed before building nuclear waste storage facilities. The typical plan for nuclear waste storage is an underground facility with the waste rods arranged in an array. In this section, we will study a very simple case, the temperature around a single rod.

The relation between the thermal current density and the temperature gradient is given by Fourier's law

$$\mathbf{j} = -\boldsymbol{\sigma} \cdot \nabla T, \tag{7.99}$$

where $\boldsymbol{\sigma}$ is the thermal conductivity, which is typically a tensor of rank 2 or a 3×3 matrix. For an isotropic material, $\boldsymbol{\sigma}$ reduces to a diagonal matrix with identical diagonal elements σ. Note that this relation is a typical result from the linear-response theory, which provides all sorts of similar relations between the gradient of a field and the corresponding current density.

If we combine Fourier's law with the energy conservation, we have

$$c\rho\frac{\partial T}{\partial t} - \nabla \cdot \boldsymbol{\sigma} \cdot \nabla T = q(\mathbf{r}, t), \tag{7.100}$$

where c the specific heat and ρ the density of the material, and $q(\mathbf{r}, t)$ is the heat released into the system per unit volume per unit time at the position $\mathbf{r}$ and time t. Assuming that the system is isotropic, then we have

$$\frac{1}{\kappa}\frac{\partial T(\mathbf{r}, t)}{\partial t} - \nabla^2 T(\mathbf{r}, t) = S(\mathbf{r}, t), \tag{7.101}$$

where $\kappa = \sigma/c\rho$ is called the *diffusivity*, a system-dependent parameter, and $S = q/\sigma$ is the effective source.

For the specific geometry of a nuclear waste rod, we can assume that the system is effectively two-dimensional with the temperature field having a circular symmetry around the rod. The above equation then becomes

$$\frac{1}{\kappa}\frac{\partial T(r, t)}{\partial t} - \frac{1}{r}\frac{\partial}{\partial r} r \frac{\partial T(r, t)}{\partial r} = S(r, t), \tag{7.102}$$

with the boundary condition $T(\infty, t) = T_{\mathrm{E}}$, the environment temperature. For the purpose of the numerical evaluation, we assume that

$$S(r, t) = \begin{cases} \dfrac{T_0}{a^2} e^{-t/\tau_0} & \text{for } r \leq a, \\ 0 & \text{elsewhere,} \end{cases} \tag{7.103}$$

where T_0, a, and τ_0 are system-dependent parameters. If we take advantage of the cylindrical symmetry of the problem, the discretized equation along the radial direction is equivalent to a one-dimensional diffusion equation with a spatially dependent diffusion coefficient $D \propto r$, that is,

$$(2 - H_i)T(t + \tau) = (2 + H_i)T(t) + \tau\kappa[S_i(t) + S_i(t + \tau)], \tag{7.104}$$

with

$$H_i T(t) = \frac{\tau\kappa}{r_i h^2}[r_{i+1/2} T_{i+1}(t) + r_{i-1/2} T_{i-1}(t) - 2r_i T_i(t)]. \tag{7.105}$$

The numerical problem is now more complicated than the constant diffusion coefficient case, but the matrix involved is still tridiagonal. We can solve the problem iteratively with the simple method described in Section 2.4 by performing an LU decomposition first and then a forward substitution followed by a backward substitution.

Special care must be given at $r = 0$ and the cut-off radius r_{c}. Consider that $i = 0$ at $r = 0$ and $i = n + 1$ at $r = r_{\mathrm{c}}$. Because the energy cannot flow into the $r = 0$ region, we must have

$$\left.\frac{\partial T}{\partial r}\right|_{r=0} = 0. \tag{7.106}$$

Then we can use Eq. (7.72) to express the temperature at $r = 0$ in terms of the temperatures at the next two points. One way to fix the temperature at the cut-off radius is by extrapolation. We will leave this as an exercise for the reader.

Now let us see an actual numerical example. Assuming that the system (the rod and its environment) is close to concrete with $c = 789$ J/(kg K),

$\sigma = 1.00$ W/(m K), and $\rho = 2.00 \times 10^3$ kg/m^3, we have $\kappa = \sigma/c\rho = 6.34 \times 10^{-7}$ m^2/s $= 2.00 \times 10^7$ cm^2/100 yr. We will take $a = 25$ cm, $T_0 = 1.00$ K, and $\tau_0 = 100$ years.

To simplify the problem here, we will assume that the temperature at the cut-off radius $r_c = 100$ cm is fixed at 300 K under ventilation. This restriction can be removed by extrapolation. The initial condition is given by $\Delta T(r, 0) = 0$ K or $T(r, 0) = T_E$, and the boundary condition $\Delta T(r_c, t) = 0$. The following program puts everything together for the temperature field change around a nuclear waste rod.

```
// A program to study the time-dependent temperature
// field around a nuclear waste rod in a 2D model.

import java.lang.*;
public class Nuclear {
  final static int nx = 100, n = 5, nt = 1000,
    mt = 1000;
  public static void main(String argv[]) {
    double d[] = new double[nx];
    double e[] = new double[nx];
    double c[] = new double[nx];
    double b[] = new double[nx];
    double p[] = new double[nx];
    double s[][] = new double[nt+1][nx];
    double T[][] = new double[nt+1][nx+1];
    double dt = 1.0/mt, tc = 1, T0 = 1, kappa = 2e7;
    double ra = 25, rb = 100, h = rb/nx, h2 = h*h;
    double s0 = dt*kappa*T0/(ra*ra), g = dt*kappa/h2;

 // Assign the elements in the matrix 2-H_i
    for (int i=0; i<nx; ++i) {
      d[i] = 2*(1+g);
      e[i] = -(1+0.5/(i+1))*g;
      c[i] = -(1-0.5/(i+2))*g;
    }

 // Modify the first equation from T"=0 at r=0
    d[0] -= 2*g/3;
    e[0] += g/6;

 // Assign the source of the radiation heat
    int na = (int) (ra/h);
    for (int i=0; i<=nt; ++i) {
      double t = -dt*i/tc;
      for (int j=0; j<na-1; ++j) {
        s[i][j] = s0*Math.exp(t);
      }
    }

 // Find the temperature field recursively
    for (int i=1; i<=nt; ++i) {

   // Assign the elements in the matrix 2+H_0
      double d0 = 2*(1-g);
      double e0 = (1+0.5)*g;
```

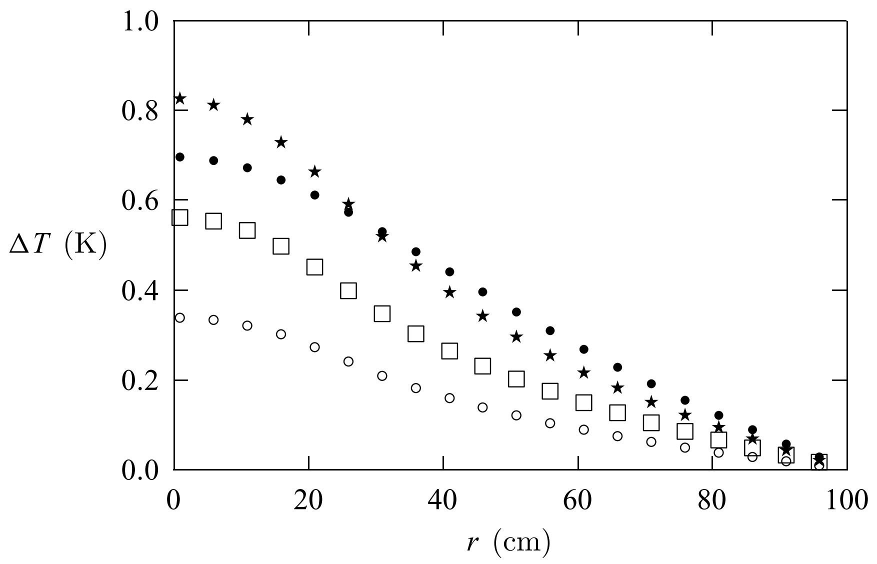

Fig. 7.3 Temperature change around a nuclear waste rod calculated with the program given in the text. Solid dots are for $t = 1$ year, stars are for $t = 10$ years, squares are for $t = 50$ years, and circles are for $t = 100$ years.

```
      double c0 = (1-0.5)*g;

   // Evaluate b[0] under the condition T"=0 at r=0
      b[0] = d0*T[i-1][0]+e0*T[i-1][1]
            +c0*(4*T[i-1][0]-T[i-1][1])/3
            +s[i-1][0]+s[i][0];

   // Find the elements in the array b[i]
      for (int j=1; j<nx; ++j) {

     // Assign the elements in the matrix 2+H_0
        d0 = 2*(1-g);
        e0 = (1+0.5/(j+1))*g;
        c0 = (1-0.5/(j+1))*g;

     // Obtain the elements from the last recursion
        b[j] = d0*T[i-1][j]+e0*T[i-1][j+1]
              +c0*T[i-1][j-1]+s[i-1][j]+s[i][j];
      }

   // Obtain the solution of the temperature field
      p = tridiagonalLinearEq(d, e, c, b);
      for (int j=0; j<nx; ++j) T[i][j] = p[j];
    }

 // Output the result at every n spatial data points
    for (int j=0; j<nx; j+=n) {
      double r = h*(j+1);
      System.out.println(r + " " + T[nt][j]);
    }
  }

// Method to solve the tridiagonal linear equation set.

  public static double[] tridiagonalLinearEq(double d[],
    double e[], double c[], double b[]) {...}
}
```

The parameters used in the above programs are not from actual storage units. However, if we want to study the real situation, we only need to modify the parameters for the actual system and environment. The numerical scheme and program are quite applicable to realistic situations. The output of the program is shown in Fig. 7.3. Note that the maximum change of the temperature (the peak) increases with time initially and then decreases afterward. The largest change of the temperature over time and the rate of the temperature change are critical in designing a safe and practical nuclear waste storage.

Exercises

7.1 Consider the Poisson equation

$$\nabla^2\phi(x,y) = -\rho(x,y)/\epsilon_0$$

from electrostatics on a rectangular geometry with $x \in [0, L_x]$ and $y \in [0, L_y]$. Write a program that solves this equation using the relaxation method. Test your program with: (a) $\rho(x,y) = 0$, $\phi(0,y) = \phi(L_x,y) = \phi(x,0) = 0, \phi(x,L_y) = 1$ V, $L_x = 1$ m, and $L_y = 1.5$ m; (b) $\rho(x,y)/\epsilon_0 = 1$ V/m^2, $\phi(0,y) = \phi(L_x,y) = \phi(x,0) = \phi(x,L_y) = 0$, and $L_x = L_y = 1$ m.

7.2 Develop a numerical scheme that solves the Poisson equation

$$\nabla^2\phi(r,\theta) = -\rho(r,\theta)/\epsilon_0$$

in polar coordinates. Assume that the geometry of the boundary is a circular ring with the potential at the inner radius, $\phi(a,\theta)$, and outer radius, $\phi(b,\theta)$, given. Test the scheme with some special choice of the boundary values.

7.3 If the charge distribution in the Poisson equation is spherically symmetric, derive the difference equation for the potential along the radius. Test the algorithm with $\rho(r) = \rho_0 e^{-r/r_0}$ in a program.

7.4 Derive the relaxation scheme for a three-dimensional system with rectangular boundaries. Analyze the choice of p in a program for the Poisson equation with constant potentials at the boundaries.

7.5 Modify the program for the groundwater dynamics problem given in Section 7.6 to study the general case of the transient state, that is, the case where the time derivative of the head is nonzero. Apply the program to study the stationary case given there as well as the evolution of the solution with time if the infiltration $f(\mathbf{r},t) = f_0 e^{-t/\tau}$ for various f_0 and τ.

7.6 Write a program that solves the wave equation of a finite string with both ends fixed. Assume that the initial displacement and velocity are given. Test the program with some specific choice of the initial condition.

7.7 Solve the time-dependent Schrödinger equation using the Crank–Nicolson method. Consider the one-dimensional case and test it by applying it to the problem of a square well with a Gaussian initial state coming in from the left.

7.8 Obtain the algorithm for solving the three-dimensional wave equation with 1/3 of the time step applied to each coordinate direction. Test the algorithm with the equation under the homogeneous Dirichlet boundary condition. Take the initial condition as $u(\mathbf{r};0) = 0$ and $v(\mathbf{r};0) = \sin(\pi x/L_x)\sin(\pi y/L_y)\sin(\pi z/L_z)$, where L_x, L_y, and L_z are the lengths of the box along the three directions.

7.9 Consider an elastic rope that is fixed at both ends at $x = 0$ and $x = L = 8.00$ m, subject to a tension $T = 5.00 \times 10^3$ N. Two friends are sitting on the rope, each with a mass distribution (mass per unit length) from a truncated Gaussian

$$\rho_i(x) = \frac{m_i}{2a_i} e^{-(x-x_i)^2/a_i^2},$$

for $|x - x_i| \leq a_i$. Assume that $m_1 = 40.0$ kg, $m_2 = 55.0$ kg, $x_1 = 3.50$ m, $x_2 = 4.10$ m, $a_1 = 0.300$ m, and $a_2 = 0.270$ m, and the rope has a density of $\rho_0 = 4.00$ kg/m. (a) Show that the motion of the rope is described by

$$\frac{\partial^2 u(x,t)}{\partial t^2} = \frac{T}{\rho(x)} \frac{\partial^2 u(x,t)}{\partial x^2} - g,$$

where $u(x,t)$ is the displacement of the rope at the position x and time t, $\rho(x) = \rho_0 + \rho_1(x) + \rho_2(x)$, and $g = 9.8\,\text{m/s}^2$ is the magnitude of the gravitational constant. (b) Find the displacement of the rope when it is in equilibrium. Where is the maximum displacement of the rope? (c) Find the first five angular frequencies if the system is in vibration and plot out the corresponding eigenstates.

7.10 Solve the nuclear waste rod problem discussed in Section 7.8 with the extrapolation of the temperature made at the cut-off radius. Compare various extrapolation schemes, linear extrapolation, quadratic extrapolation, the Lagrange extrapolation with multipoints, and the extrapolation based on the cubic spline. Which scheme works the best? Are there any significant differences between the result here and that found in Section 7.8 with a fixed temperature at the cut-off radius and why?

7.11 Simulate the process of burning a hole in the middle of a large silver sheet with a propane torch numerically. Assuming that the torch head can generate a power of about 5000 W within a circle that has a 4 mm radius, estimate how long it will take to create a hole in a sheet that has a thickness of 2 mm.

7.12 The macroscopic state of Bose–Einstein condensate is described by the Gross–Pitaevskii equation

$$i\hbar \frac{\partial \phi(\mathbf{r}, t)}{\partial t} = \left[-\frac{\hbar^2}{2m}\nabla^2 + V_{\text{ext}}(\mathbf{r}) + g|\phi(\mathbf{r}, t)|^2 \right] \phi(\mathbf{r}, t),$$

where $\phi(\mathbf{r}, t)$ is the macroscopic wavefunction, m and g are two system-related parameters, and $V_{\text{ext}}(\mathbf{r})$ is the external potential. Develop a program that solves this equation numerically. Consider various potentials, such as a parabolic or a square well.