4.3.5 Beyond Euler’s Method For Reaction-Diffusion PDE: Diffusion of Gas in a Tunnel, Gas in Porous Media, Second-Order in Time Methods, and Unconditional Stability

The forward Euler method is easy to program, it runs fast, and with acceleration (extrapolation applied to increase accuracy) it can be used to produce reasonably efficient and accurate results. On the other hand, it has at least two weaknesses: (1) the ratio of the time-step size and the square of the spatial discretization size must be small to avoid numerical instability (see Exercise 4.62) and (2) the method is first-order in time. Both of these deficiencies have been overcome with the development of discretization methods that are unconditionally stable and second-order in time. One such method that is often used is due to John Crank and Phyllis Nicolson [21]; its derivation, order estimates, numerical stability, and implementation are discussed in the following sections.

Second-order methods are introduced after development of two model problems: gas diffusion in air and gas motion in porous media. These models have independent interest, but their main function here is to demonstrate via realistic applications that numerical simulations using Euler’s method are not efficient due to the Courant-Lewy-Friedricks restriction on step-size. Of course, to appreciate this fact, the reader is encouraged to perform the suggested exercises. Further on, after second-order methods are introduced, the exercises can be repeated to compare and contrast the efficacy of first and second order numerical algorithms in practice.

Gas Diffusion in a Tunnel

A portion of a cylindrical tunnel with radius \( a = 4 \) m is \( \ell = 25 \) m long. A gas with molecular weight of 21.3 g/mol leaks near the entrance of the tunnel at the rate of 2 L/s. A sensor, installed in the tunnel 22 m from the entrance, records the concentration of this gas in mg/L once per minute. The gas is absorbed in a filter at the far end of the tunnel, at 90% of its concentration in the 1.5 m³/s of air passing through the filter. The problem is to simulate the sensor output as a function of time from the instant the gas is released until the sensor reads 2 mg/L. Assume the following: The diffusivity of the gas in air is \( 0.5 \times 10^{-4} \) m²/s, 1 mol of gas occupies a volume of 22.4 L, and \( 1 \) L = 0.001 m³.
Let $u$ denote the concentration of gas in mol/m$^3$ and assume the gas concentration is constant in each cross section of the tunnel so that $u$ is a function of the distance $x$ measured in meters from the entrance and time $t$ measured in seconds. In this case, the diffusion in the tunnel is modeled by the usual PDE

$$u_t + \kappa u_{xx} = 0$$

for $0 < x < \ell$, $\kappa = 0.6 \times 10^{-4}$ m$^2$/s, and initial data is $u(x, 0) = 0$ on the same spatial domain.

At the end of the tunnel, gas is filtered out. Thus, there is a net loss of gas corresponding to out-flux across the cross section at $x = \ell$. The out-flux is

$$\pi a^2 \kappa u_x,$$

a quantity measured in mol/s. This is due to the filter processing air at the rate $R = 1.5$ m$^3$/s with an efficiency $E = 0.9$ times the gas concentration of $u(\ell, t)$ mol/s. Thus, the boundary condition at the end of the tunnel is

$$\pi a^2 \kappa u_x(\ell, t) = -ERu(\ell, t).$$

The negative sign corresponds to gas leaving the tunnel. In fact, $u(\ell - \Delta x, t)$ must be larger than $u(\ell, t)$ because gas is removed from the air at the boundary. This means the approximate derivative

$$u_x(\ell, t) \approx \frac{u(\ell - \Delta x, t) - u(\ell, t)}{-\Delta x}$$

is negative as in the boundary condition.

At the entrance of the tunnel, there seems to be more than one choice for a viable boundary condition. In case the gas is injected into the tunnel, it is natural to consider its influx across the cross section at $x = 0$, which is the rate of injection

$$r = 2\frac{L}{s} \times \frac{1}{22.4} \frac{\text{mol}}{\text{L}} = \frac{1}{11.2} \frac{\text{mol}}{\text{s}}.$$

The corresponding boundary condition is

$$\pi a^2 \kappa u_x(0, t) = -r(t).$$

The minus sign indicates that the concentration of the gas is larger outside the tunnel.
An alternative boundary condition corresponding to pure diffusion across the boundary is obtained by setting the concentration at the entrance to be the concentration of gas in an adjoining chamber or part of the tunnel, perhaps with volume $V = 1.0 \text{ m} \times \pi a^2$. In this case, the concentration at $x = 0$ is set equal to the concentration $C$ (measured in mol/m$^3$) in the chamber that is given by

$$C(t) = \frac{t}{11.2\pi a^2}.$$  

The boundary condition is

$$u(0,t) = C(t) \quad (4.131)$$

(see Exercise 4.71).

Building a mathematical model for a real application is not a series of precise mathematical deductions. Assumptions are made as in the models of the two boundary conditions at the entrance of the contaminated tunnel. The best choice of boundary condition would be determined by experiments in controlled conditions for a scaled physical model of the tunnel in a laboratory or perhaps in some real tunnel. The model derived here has several shortcomings: The filtering process will cause the air in the tunnel to move. Convection is expected to transport gas (or heat) at a much faster rate than diffusion. The gas from the leak might increase the gas pressure at the entrance of the tunnel. This force would cause the mixture of gas and air to move away from the entrance. Perhaps you will produce a more accurate model after reading further in this book (see Exercise 5.149).

The contaminated tunnel model for the concentration $u$ of the contaminant gas (in moles per cubic meter) discussed in subsequent sections is

$$u_t + \kappa u_{xx} = 0,$$

$$u(x, 0) = 0,$$

$$\pi a^2 \kappa u_x(0, t) = -r(t),$$

$$\pi a^2 \kappa u_x(\ell, t) = -ERu(\ell, t),$$

$$\quad (4.132)$$

where

$$\kappa = 0.5 \times 10^{-4}, \quad a = 4.0, \quad \ell = 25, \quad E = 0.9, \quad R = 1.5,$$

$$r = 1/11.2.$$  The desired output $st$ (the sensor time trace) is the concentration at $x = 22$ given in mg/L with time measured in minutes, which is given by

$$st(\tau) = 21.3u(\xi, 60\tau), \quad (4.133)$$
where $\tau$ is measured in minutes and $\xi = 22$.

**Exercise 4.71.** (a) Approximate the sensor output for the model (4.132). In particular, determine the length of time in hours until the sensor output is $2.0 \text{ mg/L}$. (b) Approximate the sensor output with the boundary condition (4.131) and determine the length of time in hours until the sensor output is $2.0 \text{ mg/L}$. (c) Compare and contrast the simulated sensor output for the two models, and discuss the viability of the model and the suggested boundary conditions. (d) Suppose the gas leak persists for 24 hours after which it ceases. Determine the time in hours when the concentration at the sensor reaches its maximum value and specify this maximum. (e) Consider Exercise 4.104 after reading the following sections on the trapezoidal and Crank-Nicolson methods.

**Gas in a Porous Medium**

For the motion of a gas in a porous medium, diffusion due to the concentration gradient of the gas is generally so slow (due to the obstruction of the porous material) that it is ignored in the modeling process in favor of motion due to gas pressure.

To make a model, reconsider the conservation law (4.55) and the constitutive law (4.56) that lead to the dynamical equation

$$u_t + \text{div}(−K \text{grad} u + \mu V) = f,$$

where $u$ is the amount of the substance under consideration (the gas), $V$ is the velocity of the medium moving the substance, and $f$ is the function representing a model of the sources or sinks of the substance. For simplicity, assume that there are no sources or sinks, so that $f = 0$; and, ignore the diffusion by setting the diffusivity $K$ to zero. These assumptions reduce the conservation law to the equation

$$u_t + \text{div}(\mu V) = 0.$$

For gas flow in a porous medium, the usual state variable is the density $\rho$ measured in some average void in the porous material. In the derivation of the conservation law, $u$ is the density of some substance in the space that it occupies, which in this case would include the porous media. More precisely, the density of the substance in the conservation law was to be determined by taking the limit of the total amount of the substance in an open set divided
Conservation of Mass

by the volume of the open set as the size of the set shrinks to a point. This viewpoint in the case of a gas in a porous medium would produce a density that would be some fraction of the density $\rho$ of the fluid itself, a quantity given by $\epsilon \rho$, where $0 < \epsilon < 1$ is the ratio of the volume of the voids to the total volume of the material. The dimensionless number $\epsilon$ is called the porosity of the material. Using this dimensionless parameter, the conservation law for the density of a gas in a porous medium is

$$\epsilon \rho_t + \text{div}(\rho V) = 0,$$  \hspace{1cm} (4.134)

where $V$ is the velocity of the gas and the function $\mu$ in the conservation law equals $\rho$ because all the gas is moving with its own flow.

Reusing the symbol $\mu$ for the viscosity of the gas and $k$ for the permeability of the medium, Darcy’s (constitutive) law relates the velocity of the fluid to the pressure $p$ (that is, force per unit area) acting on the fluid:

$$\mu V = -k \text{grad} p.$$ \hspace{1cm} (4.135)

The minus sign is there because a positive pressure gradient drives the gas toward a region of lower pressure.

Using Darcy’s law and the conservation of mass, the governing equation takes the form

$$\epsilon \rho_t = \text{div}\left(\frac{k}{\mu} \rho \text{grad} p\right).$$

The model is closed by specifying the pressure or relating the pressure to the gas density via an equation of state, (a relation between state variables) that in this case relates pressure to density. In general, useful equations relating these two states are functional relations $F(\rho, p) = 0$ determined by the modeling process. For gases, the usual choice (which can be partially justified by using the theory of thermodynamics) is a power law

$$p = p_0 \left(\frac{\rho}{\rho_0}\right)^\gamma,$$ \hspace{1cm} (4.136)

where $p_0$ is a reference pressure, $\rho_0$ is some reference density, and the exponent $\gamma$ is in the range $\gamma \geq 1$. Note that the presence of the reference pressure and density in the formula ensure that $p$ has the units of pressure (force per area).

The (constitutive) equation of state combined with the governing equation is the dynamical equation for gas flow in a porous medium:

$$\epsilon \rho_t = \text{div}\left(\frac{k}{\mu} \rho \text{grad}\left(p_0 \left(\frac{\rho}{\rho_0}\right)^\gamma\right)\right) = \text{div}\left(\frac{p_0 k \gamma}{\mu} \rho \left(\frac{\rho}{\rho_0}\right)^\gamma \text{grad} \left(\frac{\rho}{\rho_0}\right)\right).$$ \hspace{1cm} (4.137)
By defining the dimensionless and positive state variable \( u = \frac{\rho}{\rho_0} \), the dimensionless time
\[
s = \frac{t}{\alpha}, \quad \alpha := \frac{\epsilon \mu}{\gamma \rho_0},
\]
and dimensionless length
\[
\xi = \frac{x}{\sqrt{k}}
\]
the dynamical equation (4.137) is recast into the dimensionless porous medium equation
\[
u_s = \nabla \cdot (u^\gamma \nabla u).
\] (4.138)
It is a nonlinear PDE whose dimensionless form is suitable for theoretical work.

As a specific test case, consider a rectangular block of sandstone (located underground) with length 2.0 m, width and height 1.0 m. Its porosity is 0.1 and its permeability is \( 10^{-8} \) cm\(^2\). A gas (perhaps methane) exists in a reservoir adjacent to one end of the block so that a face with area 1.0 m\(^2\) is exposed to the gas reservoir. The other end of the block is open to the atmosphere, and the lateral surface is bounded by impermeable rock. The gas has viscosity \( 12.119 \times 10^{-6} \) Pa s and pressure 6000 kPa. Assume the equation of state (4.136) has exponent \( \gamma = 2 \). The density of methane is approximately 16.04 grams per mole. The universal gas constant \( R \) is approximately 8.3 joules per mole per kelvin and the ideal gas law is \( PV = nRT \), where \( P \) is pressure, \( V \) is volume, \( n \) is number of moles, and \( T \) is absolute temperature (kelvins). The problem is to determine the transient to a steady state and the gas flux through the downstream end of the block at steady state (see Exercise 4.72).

Using the porous medium model, determine the flux of the gas (in kg/s) through the cross section of the block located 1.0 m from the reservoir after one minute, the flux through the end 2.0 m from the reservoir at steady state.

The change of variables used to derive the porous medium equation is not suitable for numerical approximations of solutions for the gas-sandstone application (4.137). Why? Instead, let \( \ell \) be the characteristic length of 1.0 m, \( \rho_0 \) be a reference density, and use the scaling
\[
x = \ell \xi, \quad t = \frac{\ell^2 \mu}{k \rho_0} s, \quad \rho = \rho_0 U
\] (4.139)
to obtain the dimensionless porous medium model
\[
U_s = \frac{\gamma}{\epsilon} \nabla \cdot (U^\gamma \nabla U),
\] (4.140)
where it is essential to note that $\nabla$ now denotes differentiation with respect to the dimensionless variable $\xi$. In fact, $u$ is to be viewed as a function $U = U(\xi, s)$ The corresponding function in the original variables is

$$u(x, t) := U\left(\frac{x}{\ell}, \frac{kp_0}{(\ell^2\mu)}t\right).$$

Boundary conditions are problematic as there is no obvious choice. One approach is to relate the gas flux through a boundary to the pressure difference inside and outside the medium.

Under the assumption of a homogeneous and isotropic medium, so that the geometry of the physical domain is idealized to one spatial dimension, the gas flux through a cross section is then (in dimensioned variables) area times density times velocity

$$\ell^2\frac{p_0\rho_0k\gamma}{\mu}u^\gamma u_x$$

at the cross section. It has units of mass per time. This quantity is related to the (dimensionless) pressure difference

$$c\left(\frac{\rho}{\rho_0}\right)^\gamma - \frac{p_a}{p_0}$$

using the ambient pressure $p_a$ (of the gas) and a new constant $c$ that has units of mass per time. The corresponding boundary condition is

$$\ell^2\frac{p_0\rho_0k\gamma}{\mu}u^\gamma u_x = \pm c(u^\gamma - \frac{p_a}{p_0}); \quad (4.141)$$

or, in dimensionless variables,

$$U^\gamma U_\xi = \pm \alpha(U^\gamma - \frac{p_a}{p_0}), \quad (4.142)$$

where

$$\alpha := \frac{c\mu}{\ell p_0\rho_0k\gamma}.$$
By choosing the reference pressure of $p_0 = 6000$ kPa, computing an approximate reference density $\rho_0 = 1.7$ kg/m$^3$ via the ideal gas law, and (absent experimental data) taking $c = 0.01$ kg/s, the dimensionless constant $\alpha$ is computed to be approximately $\alpha = 0.00594$.

While all the ingredients are in place to make a numerical computation using Euler’s method for time-stepping and the second-order central difference approximation for second derivatives, there are several issues that need to be addressed in writing a code.

The porous medium PDE written in the dimensionless form (4.140) may not reveal the many options for discretization of the spatial derivative $\nabla \cdot (U^\gamma \nabla U)$. One possibility is to expand the derivative to

$$
\nabla \cdot (U^\gamma \nabla U) = \gamma U^{\gamma - 1} \nabla U \cdot \nabla U + U^\gamma \Delta U
$$

and discretize (in one space-dimension) as follows

$$
\gamma U_i^{\gamma - 1} \left( \frac{U_{i+1} - U_{i-1}}{2\Delta x} \right)^2 + U_i^\gamma \frac{U_{i+1} - 2U_i + U_{i-1}}{\Delta x^2}.
$$

(4.143)

The central difference approximation of the first-derivative seems natural, but it could be replaced by a forward or backward difference. An alternative method is obtained by compressing the spatial derivative to

$$
\nabla \cdot (U^\gamma \nabla U) = \frac{1}{\gamma + 1} \Delta (U^{\gamma + 1})
$$

and using the discretization

$$
\frac{1}{\gamma + 1} \frac{U_{i+1}^{\gamma + 1} - 2U_i^{\gamma + 1} + U_{i-1}^{\gamma + 1}}{\Delta x^2}.
$$

(4.144)

The boundary conditions (4.142) may also be discretized in several ways. A natural choice at the left boundary, using the interior nodes from $i = 1$ to $i = n - 1$ as the computational domain and $i = 0$ as the left boundary, is

$$
U_0^\gamma \frac{U_1 - U_0}{\Delta x} = \alpha (U_0^\gamma - \frac{p_a}{p_0}).
$$

(4.145)

At each time-step, $U_1$ is known; the required value of $U_0$ can be obtained by approximating the solution of the nonlinear equation. A similar approach can be used at the right boundary.
Exercise 4.72. (a) For the case of one spatial variable, use the dimensionless porous medium equation (4.138), the boundary conditions (4.142), zero density in the porous medium at the initial time, and the other data given in this section to approximate the gas flux at the middle of the porous block as a function of time until one minute after the initial time, which should be taken to be \( t = 0 \). Report the flux in \( \text{kg/s} \) and the time in seconds. (b) For the case of one spatial variable, use pencil and paper to determine the general solution of the PDE at steady state. Impose the boundary conditions and use a numerical computation to approximate the solution of the boundary value problem. Use your result to determine the gas flux at the downstream boundary when the flow is in steady state. (c) How long is the transient in hours measured from the instant the flow starts until the root mean square distance of the density profile in the porous block is within 1% of the steady state density profile. Hints: One good way to debug a numerical code is to use a stable steady state as initial data. Stepping forward in time should leave the steady state unchanged. For part (a) it might be wise to use a second-order accurate numerical method to approximate the velocity at the cross section. This can be achieve by using a centered difference across the section; that is, by approximating the first derivative of an appropriate function \( f \) via \( (f(x + \Delta x) - f(x - \Delta x))/(2\Delta x) \). The root mean square distance between two functions is the square root of the integral over the spatial domain of the square of the difference between the two functions divided by the length of the interval \( \sqrt{1/(b - a)} \int_a^b |f(x) - g(x)|^2 \, dx \). (d) Consider Exercise 4.105 after reading the following sections on the trapezoidal and Crank-Nicolson methods.

The Trapezoidal Method and Crank-Nicolson in One Spatial Dimension

Consider a solution \( t \mapsto u(t) \) of the ODE \( \dot{u} = f(u) \); it satisfies the equation

\[
 u'(t) = f(u(t)).
\]

By integrating both sides of this identity with respect to the independent variable, we obtain the integral equation

\[
 u(t + \Delta t) - u(t) = \int_t^{t+\Delta t} f(u(s)) \, ds.
\]

The left-hand rectangle rule approximation of the integral (that is, \( ((t + \Delta t) - t) \times f(u(t)) \)) yields the forward Euler method; the trapezoidal rule approximation

\[
 u(t + \Delta t) \approx u(t) + \frac{\Delta t}{2} (f(u(t + \Delta t)) + f(u(t)))
\]
yields the trapezoidal method also called the implicit improved Euler method

\[ U^{k+1} = U^k + \frac{\Delta t}{2} (f(U^{k+1}) + f(U^k)). \] (4.146)

As might be expected, the trapezoidal method is more accurate than Euler’s method; in fact, it is a second-order method. To prove this fact, we simply assume that the solution and its approximation agree at time \( t \) and estimate the norm of the difference between the true and approximate solutions at \( t + \Delta t \) as follows:

\[
\begin{align*}
|u(t + \Delta t) - U^1| &= |u(t + \Delta t) - u(t) - \frac{\Delta t}{2} \left( f(u(t + \Delta t)) + f(u(t)) \right)| \\
&= |u'(t)\Delta t + \frac{1}{2} u''(t)\Delta t^2 - \frac{\Delta t}{2} \left( 2f(u(t)) + f'(u(t))u'(t)\Delta t \right)| \\
&\quad + O(\Delta t^3) \\
&= |f(u(t))\Delta t + \frac{1}{2} f'(u(t))u'(t)\Delta t^2 \\
&\quad - \frac{\Delta t}{2} \left( 2f(u(t)) + f'(u(t))u'(t)\Delta t \right)| + O(\Delta t^3) \\
&= O(\Delta t^3).
\end{align*}
\]

Since the local truncation error is \( O(\Delta t^3) \), the method is second-order; that is

\[
\frac{u(t + \Delta t) - u(t)}{\Delta t} = -\frac{1}{2} \left( f(u(t + \Delta t)) + f(u(t)) \right) + O(\Delta t^2)
\]

(cf. Exercise 4.74).

As a test for numerical stability, numerical methods for approximating solutions of ordinary differential equations are tested against the prototype equation for exponential decay:

\[ \dot{x} = -\lambda x, \] (4.147)

where \( \lambda > 0 \). The solution \( x(t) = x(0)e^{-\lambda t} \) decreases exponentially fast to zero.

Euler’s method applied to the ODE (4.147) is the iteration scheme

\[ x^{j+1} = x^j + \Delta t(-\lambda x^j) = (1 - \lambda \Delta t)x^j = (1 - \lambda \Delta t)^j x^0. \]

The iterates \( x^j \) converge to zero if and only if \( |1 - \lambda \Delta t| < 1 \). The method is numerically stable provided the time-step is restricted to \( 0 < \Delta t < 2/\lambda \).
Conservation of Mass

A larger time-step will result in a sequence of iterates whose absolute values grow without bound.

This result is closely related to the Courant-Lewy-Friedricks condition (4.72) for numerical stability of Euler’s method (4.70) applied to approximate solutions of the heat equation. Recall the numerical method is

\[ U_{i}^{j+1} = U_{i}^{j} + \Delta t \frac{\kappa}{\Delta x^2} (U_{i-1}^{j} - 2U_{i}^{j} + U_{i+1}^{j}), \quad (4.148) \]

which has exactly the form of Euler’s method except that the function \( f \) in the ODE \( \dot{x} = f(x) \) (for example \( f(x) := -\lambda x \) in the exponential decay equation (4.147)) is replaced by an operator \( F(u) := \kappa u_{xx} \), which also happens to be discretized in the scheme (4.148).

Similarly, the trapezoidal method applied to the exponential decay model is the iteration scheme

\[ x^{j+1} = x^{j} - \lambda \frac{\Delta t}{2} (x^{j+1} - x^{j}). \]

By a simple rearrangement, the iterates are given by

\[ x^{j+1} = \frac{1 - \lambda \frac{\Delta t}{2}}{1 + \lambda \frac{\Delta t}{2}} x^{j}. \quad (4.149) \]

Thus, the trapezoidal method produces a scheme that is \textit{unconditionally stable}. Indeed, the coefficient of \( x^{j} \) has absolute value less than one for every positive \( \Delta t \). No restriction on the step-size is required to maintain numerical stability.

The Crank-Nicolson method, which will be discussed in detail in this section, is the trapezoidal scheme applied to partial differential equations. It is unconditionally stable for many PDEs; that is, the method is numerically stable with no restriction on the size of the positive time-step.

For the heat equation \( u_t = \kappa u_{xx} \) in one spatial dimension, the Crank-Nicolson method with \( U_{i}^{j} := u(i\Delta x, j\Delta t) \) is given by the difference scheme

\[ U_{i}^{j+1} = U_{i}^{j} + \frac{\kappa \Delta t}{2\Delta x^2} (U_{i-1}^{j} - 2U_{i}^{j} + U_{i+1}^{j}) + U_{i}^{j+1} + U_{i+1}^{j+1} - 2U_{i}^{j+1} + U_{i+1}^{j+1}), \quad (4.150) \]

where, for example, \( i = 1, 2, 3, \ldots n \). Let

\[ \alpha := \frac{\kappa \Delta t}{2\Delta x^2} \]
and define $W^j$ to be the $n$-dimensional vector with components $U^j_i$ for $i = 1, 2, 3, \ldots n$. Also, let $A$ be the tridiagonal matrix with all components on the main diagonal equal to $-2$, all elements on the first super and sub-diagonals equal to 1 and all other components equal to zero. Using these notations and an algebraic rearrangement, the difference scheme (4.150) is given in the vector form

$$(I - \alpha A)W^{j+1} = (I + \alpha A)W^j,$$

where $\alpha > 0$.

Using the Gerschgorin theorem (see Appendix A.5) and the symmetry of $A$, every eigenvalue of this matrix is a real number in the closed interval $[-4, 0]$. An easy calculation shows that $\mu$ is an eigenvalue of $I - \alpha A$ (that is, $(I - \alpha A)v = \mu v$ for some nonzero vector $v$) if and only if $\mu = 1 - \alpha \lambda$ for some eigenvalue $\lambda$ of $A$. Since every eigenvalue of $A$ is non positive, every eigenvalue of $I - \alpha A$ is positive. In particular, this matrix has no zero eigenvalue; it is invertible.

Because $I - \alpha A$ is invertible, the Crank-Nicolson scheme for the heat equation reduces to the iteration process

$$W^{j+1} = (I - \alpha A)^{-1}(I + \alpha A)W^j$$

To prove that the Crank-Nicolson method for the heat equation is unconditionally numerically stable it suffices to show that every eigenvalue of the matrix $(I - \alpha A)^{-1}(I + \alpha A)$ is in the closed unit disk in the complex plane. Again, this fact follows by relating the eigenvalues of this matrix to the eigenvalues of $A$.

The number $\gamma$ is an eigenvalue of $(I - \alpha A)^{-1}(I + \alpha A)$ if there is a nonzero vector $v$ such that

$$(I + \alpha A)v = \gamma(I - \alpha A)v.$$ 

By an easy calculation, this eigenvalue must be given by

$$\gamma = \frac{1 + \alpha \lambda}{1 - \alpha \lambda},$$

where $\lambda$ is an eigenvalue of $A$. Because $\alpha > 0$ and $-4 \leq \lambda \leq 0$, the eigenvalue $\lambda$ is such that $-1 \leq \lambda \leq 1$ (compare equation (4.149)). In particular, every eigenvalue of the matrix $(I - \alpha A)^{-1}(I + \alpha A)$ lies in the closed unit disk; therefore, the Crank-Nicolson method for the heat equation is unconditionally numerically stable.
The numerical stability result does not take into account boundary conditions that alter the matrix $A$. Dirichlet and Neumann boundary conditions are considered in this context in Exercise 4.92.

Unconditional stability comes with a price: the trapezoidal method (4.146) is implicit. At each time-step or (equivalently) each iteration of the method, applied to approximate a solution of the ODE $\dot{x} = f(x)$, the equation

$$U^{k+1} - \frac{\Delta t}{2} f(U^{k+1}) = U^k + \frac{\Delta t}{2} f(U^k)$$

must be solved for the unknown $U^{k+1}$. This equation is nonlinear whenever the ODE is nonlinear.

The premier method for approximating the solutions of nonlinear equations is Newton’s method; it is one of the most important algorithms in analysis. Recall that if we wish to find a root $a$ of a function $g : \mathbb{R}^n \rightarrow \mathbb{R}^n$, we may suppose that if $x$ is close to $a$, then (by Taylor’s Theorem)

$$g(a) \approx g(x) + Dg(x)(a - x) \approx 0.$$

By solving for $a$ in the formula $g(x) + Dg(x)(a - x) = 0$, we expect to find an approximation of this vector. The solution is

$$a = x - Dg(x)^{-1}g(x),$$

where $a$ is an approximation of the root of $g$. This observation is turned into Newton’s method: Choose an approximation $x^0$ of the expected root $a$ and compute the sequence $\{x^k\}_{k=0}^\infty$ according to the iteration process

$$x^{k+1} = x^k - Dg(x^k)^{-1}g(x^k). \quad (4.153)$$

Under the three assumptions (1) the starting value $x^0$ is sufficiently close to $a$, (2) the function $g$ is continuously differentiable, and (3) $Dg(a)$ is invertible, the sequence of iterates defined by Newton’s method converges to the desired root $a$. Moreover, the rate of convergence is quadratic; that is, for $r = 2$ and some positive number $C$,

$$|x^{k+1} - a| \leq C|x^k - a|^r \quad (4.154)$$

(cf. Appendix A.14).
Figure 4.30: The iterates and asymptotic errors, using $|x^{k+1} - a|/|x^k - a|^r$ with $r = 2$ for Newton’s method and $r = 1$ for composition, to approximate the real root $x = 1$ of the polynomial $x^3 + x^2 - x - 1$.

To approximate a solution $x = a$ of the equation $g(x) = 0$ for $g : \mathbb{R} \to \mathbb{R}$ by Newton’s method (the scalar case), choose some initial approximation $x^0$ of $a$ and iterate using the formula

$$x^{k+1} = x^k - \frac{g(x^k)}{g'(x^k)}.$$  \hspace{1cm} (4.155)

In the vector case, it is best to avoid computing the inverse of the matrix $Dg(x^k)$. Instead, Newton’s method is implemented in an alternate form: Solve the linear system

$$Dg(x^k)y = -g(x^k)$$ \hspace{1cm} (4.156)

for $y$ and define $x^{k+1} = y + x^k$.

For an arbitrary convergent sequence $\{x^k\}^\infty_{k=1}$ with limit $a$, we say that the order of convergence is $r$ and the asymptotic error is $C$ whenever we have the inequality (4.154). In other words, the error at index $k + 1$ (which we will denote by $\text{err}(k+1)$) is approximately proportional (with a fixed constant $C$ of proportionality) to the $\text{err}(k)^r$; in particular, the error at index $k + 1$ is $O(\text{err}(C)^r)$. The quadratic convergence of Newton’s method makes it a pillar of numerical (and theoretical) analysis (cf. [41]).

As a simple example of the convergence properties of Newton’s method, consider the real root, $x = 1$, of the cubic polynomial $g(x) = x^3 + x^2 - x - 1$. The results of an implementation of Newton’s method with initial guess $x = 2$ are given in Figure 4.30. The asymptotic error computed with $r = 2$ as in display (4.154) seems to converge to $C = 1$. Thus, the experiment confirms that Newton’s method is quadratically convergent in this case (see Exercise 4.75 for a proof).
Figure 4.31: The iterates for Newton’s, Aitken’s and Steffensen’s methods applied to find the root ($x = 1$) of the polynomial $x^3 + x^2 - x - 1$.

For comparison, note that the root $x = 1$ can also be found as a fixed point of the function

$$G(x) = x - \frac{1}{8}(x^3 + x^2 - x - 1).$$

(4.157)

Starting from an initial guess $x = x_0$, the composition method (also called fixed point iteration) is to compute the compositional iterates of the function $G$; that is, to proceed inductively using the formula $x^{k+1} = G(x^k)$. In other words, we view $G$ as a dynamical system on the real line and seek the real root of the polynomial as a stable fixed point of this dynamical system. The factor $1/8$ is used to ensure that $G$ has a stable fixed point at $x = 1$ (see Exercise 4.82). The data in Figure 4.30 suggests that the iterates of $G$ are linearly convergent to the fixed point.

Newton’s method gives the correct root with an error of less than $10^{-2}$ after three iterates; the composition method requires five. To achieve an error less than $10^{-3}$, Newton’s method requires four iterates; the composition method requires nine. Let us also note that the computational cost of the two methods is comparable. It should be clear that Newton’s method is superior.

We have discussed acceleration by Richardson extrapolation of certain low-order methods with discretization errors. It is also possible to accelerate some linearly convergent sequences by a staple of numerical analysis called Aitken’s $\Delta^2$-method. The idea is simple. For a linearly convergent sequence $\{x_k\}_{k=1}^\infty$ with limit $x^\infty$, we can expect that (for large $k$)

$$|x^{k+1} - x^\infty| \approx C|x^k - x^\infty|,$$

for some constant $C$. In addition, let us assume that the errors all have the same signs. By taking one more iterate, we have two approximations

$$x^{k+1} - x^\infty \approx C(x^k - x^\infty), \quad x^{k+2} - x^\infty \approx C(x^{k+1} - x^\infty).$$
By eliminating \( C \), we find the relation
\[
(x^{k+1} - x^\infty)^2 \approx (x^{k+2} - x^\infty)(x^k - x^\infty)
\]
and solve for \( x^\infty \) to obtain
\[
x^\infty \approx \frac{x^k x^{k+2} - x^{k+1}}{x^{k+2} - 2x^{k+1} + x^k},
\]
or equivalently,
\[
x^\infty \approx x^k - \frac{(x^k - x^{k+1})^2}{x^{k+2} - 2x^{k+1} + x^k}.
\]
Of course, this last relation is turned into a numerical method by replacing the unknown \( x^\infty \) by the \( k \)th Aitken \( \Delta^2 \) approximation; that is,
\[
Aitken(k) := \frac{x^k x^{k+2} - x^{k+1}}{x^{k+2} - 2x^{k+1} + x^k}.
\]

Figure 4.31 lists the Aitken approximations obtained by applying formula (4.158) to the sequence of iterates of the function \( G \) defined in display (4.157). The convergence is faster than the linear convergence of the simple iterates of \( G \), but the new sequence does not converge quadratically (see Exercise 4.83). This deficiency can be remedied: Instead of applying Aitken’s method directly to the sequence of iterates, compute two iterations, apply formula (4.158) to \( x_0, x_1 \) and \( x_2 \) to obtain Aiken(0), use this value—which should be a better approximation of the fixed point than \( x_3 \)—as the new \( x_0 \), and repeat this process. This algorithm is called Steffensen’s method. The Steffensen sequence for our test example is listed in Figure 4.31. For this example, it performs as well as Newton’s method (see Exercise 4.83).

The assumption that the errors \( x^k - x^\infty \) in our sequence all have the same signs might not be satisfied; for example, the sequence given by \( x^0 = 1 \) and \( x^{k+1} = g(x^k) \) (where \( g(x) := -x/2 \)) converges to zero but alternates in sign. For a situation like this where the iterates alternate in sign, Steffensen’s method can be applied to every second iterate; or, in other words, it can be applied to the sequence generated by \( g^2(x) := g(g(x)) \).

Since we have Newton’s method, why are Aitken’s and Steffensen’s methods used? Answer: Aitken’s method applies to general linearly convergent sequences; hence, it can be applied in situations where no quadratically convergent method is known. Steffensen’s method is valuable in cases where the derivative of the function being iterated is difficult to obtain or expensive to
Conservation of Mass

evaluate (cf. Exercise 4.84). As might be expected, Richardson extrapolation, Aitken’s $\Delta^2$, and Steffensen’s are only a few of the important methods that have been developed to accelerate convergence of low-order methods. Of course, extrapolation methods, like all numerical methods, do not always work. The complete story is beyond the scope of this book (cf. [10]).

The Crank-Nicolson Method in Two Spatial Dimensions

The Crank-Nicolson method is simply the trapezoidal method adapted to partial differential equations, where we view our parabolic PDE as an abstract evolution equation \( \dot{u} = f(u) \) and \( f \) is a differential operator. In fact, the Crank-Nicolson method for our basic reaction-diffusion PDE (4.87) is the scheme

\[
U_{i,j}^{k+1} = U_{i,j}^k + \frac{\Delta t}{2} \left[ \frac{\lambda}{\Delta x^2} (U_{i+1,j}^{k+1} - 2U_{i,j}^{k+1} + U_{i-1,j}^{k+1}) + \frac{\lambda}{\Delta y^2} (U_{i,j+1}^{k+1} - 2U_{i,j-1}^{k+1} + U_{i,j-1}^{k+1}) \right] + f(U_{i,j}^{k+1}, V_{i,j}^{k+1})
\]

\[
V_{i,j}^{k+1} = V_{i,j}^k + \frac{\Delta t}{2} \left[ \frac{\lambda}{\Delta y^2} (V_{i+1,j}^{k+1} - 2V_{i,j}^{k+1} + V_{i-1,j}^{k+1}) + \frac{\lambda}{\Delta y^2} (V_{i,j+1}^{k+1} - 2V_{i,j-1}^{k+1} + V_{i,j-1}^{k+1}) \right] + g(U_{i,j}^{k+1}, V_{i,j}^{k+1})
\]

where the symbols \( f \) and \( g \) are now redefined to denote the reaction terms in the PDE.

To implement the Crank-Nicolson scheme directly, we must solve a nonlinear system of equations to compute each update. This can be done using Newton’s method or Steffensen’s method (see Exercise 4.81).

Another possibility is to modify the trapezoidal method into an explicit second-order method. Indeed, the idea is simple: Compute the implicit update using Euler’s method. This results in the explicit second-order scheme

\[
U^{k+1} = U^k + \frac{\Delta t}{2} (f(U^k + \Delta t f(U^k)) + f(U^k))
\]

which is often called the improved Euler method (see Exercise 4.85).

A third possibility is to modify the method to make it explicit (using the Euler approximation as in equation (4.160)) for the reaction terms and
leave the update equations implicit for the diffusion terms. To complete each
time step for this algorithm, a nonlinear solver is avoided as the modified
scheme requires only the solution of a system of linear equations. This
method is theoretically second-order with respect to both the space and time
discretization sizes; in other words, the method is \( O(\Delta t^2) \) and \( O(\Delta x^2 + \Delta y^2) \).

While Newton’s method is the premier choice for approximating the so-
lution of a set of nonlinear equations, there are superior methods for ap-
proximating the solution of large systems of linear equations. The subject of
solving systems of linear equations is itself an important branch of numerical
analysis (and pure mathematics); a glimpse of a few topics in this area is
provided here.

The simplest examples that lead to systems of linear equations are of
course PDEs with one spatial dimension. To apply new methods to the
reaction diffusion model (4.87)

\[
\begin{align*}
    u_t &= \lambda \Delta u + f(u, v), \\
    v_t &= \mu \Delta v + g(u, v),
\end{align*}
\]  

(4.161)

for a rectangular spatial domain with side lengths \( L_1 \) and \( L_2 \), some new
concepts are needed. This example illustrates some of the challenges that
arise in multi-dimensional problems. It is used here for this purpose.

For a spatial discretization, we may choose the spatial increments \( \Delta x = L_1/m \) and \( \Delta y = L_2/n \) and the nodes \((i - 2)\Delta x, (j - 2)\Delta y\) with \( i = 2, 3 \ldots, m + 2 \) and \( j = 2, 3 \ldots, n + 2 \). The corners of our spatial rectangle are
labeled by the indices \((2, 2)\) (lower left), \((2, n + 2)\) (upper left), \((m + 2, n + 2)\)
(upper right) and \((m + 2, 2)\) (lower right). Each of the \((m+1)\times(n+1)\)-nodes
in the closed rectangle will correspond to two linear equations, one for \(U_{ij}^{k+1}\)
and one for \(V_{ij}^{k+1}\). The resulting systems of linear equations can and will be
solved separately.

The equations for the approximate state variables at the nodes must
be ordered in some convenient manner; that is, we need a bijective function
defined on the set of nodes whose range is the set of integers \( \{1, 2, 3, \ldots, (m + 1)\times(n + 1)\} \). We will use the bijection that corresponds to the ordering of
the nodes on the spatial grid from top left to bottom right along rows from
left to right. The required function, called nodes, is given by

\[
\text{nodes}(i, j) = i - (m + 1)j + (n + 1) + m(n + 2)
\]  

(4.162)

(see Exercise 4.86).
Conservation of Mass

Using our bijection, we can write our system of linear equations (for the unknown updates of the state variables at the nodes) in matrix form

\[ AU^{k+1} = b, \]

where the \((m + 1)(n + 1) \times (m + 1)(n + 1)\)-matrix \(A\) and the \((m + 1)(n + 1)\)-vector \(b\) are defined using the Crank-Nicolson scheme. This procedure is accomplished in three steps:

1. Use the Neumann boundary condition to set the values of \(U\) and \(V\) at the fictitious nodes that lie outside the spatial domain, which are needed to compute the discretized spatial second derivatives; for example, \(U_{1,j}^k := U_{3,j}^k\) along the left-hand edge of the spatial domain.

2. Define the matrix \(A\) using the form of the equations at each node; for example, the generic nodes\((i, j)\)-row of the matrix \(A\) for the \(U\) variables is given by the assignments

\[
A(\text{nodes}(i, j), \text{nodes}(i, j)) := 1 + \lambda \left( \frac{\Delta t}{\Delta x^2} + \frac{\Delta t}{\Delta y^2} \right),
\]

\[
A(\text{nodes}(i, j), \text{nodes}(i + 1, j)) := -\lambda \frac{\Delta t}{\Delta x^2},
\]

\[
A(\text{nodes}(i, j), \text{nodes}(i - 1, j)) := -\lambda \frac{\Delta t}{\Delta x^2},
\]

\[
A(\text{nodes}(i, j), \text{nodes}(i, j - 1)) := -\lambda \frac{\Delta t}{\Delta y^2},
\]

\[
A(\text{nodes}(i, j), \text{nodes}(i, j + 1)) := -\lambda \frac{\Delta t}{\Delta y^2},
\]

and all other components are set to zero in this row. Unfortunately, the matrix contains many non generic rows. Perhaps, an example will help to illustrate the extra complications due to the Neumann boundary conditions.

For the case \(m = n = 2\) with \(\beta := \lambda \Delta t / (2\Delta x^2)\), the matrix \(A\) is the \(9 \times 9\)-matrix

\[
\begin{pmatrix}
1 + 4\beta & -2\beta & 0 & -2\beta & 0 & 0 & 0 & 0 & 0 \\
-\beta & 1 + 4\beta & -\beta & 0 & -2\beta & 0 & 0 & 0 & 0 \\
0 & -\beta & 1 + 4\beta & -\beta & 0 & -2\beta & 0 & 0 & 0 \\
-\beta & 0 & -\beta & 1 + 4\beta & -\beta & 0 & -\beta & 0 & 0 \\
0 & -\beta & 0 & -\beta & 1 + 4\beta & -\beta & 0 & -\beta & 0 \\
0 & 0 & -\beta & 0 & -\beta & 1 + 4\beta & -\beta & 0 & -\beta \\
0 & 0 & 0 & -2\beta & 0 & -\beta & 1 + 4\beta & -\beta & 0 \\
0 & 0 & 0 & 0 & -2\beta & 0 & -\beta & 1 + 4\beta & -\beta \\
0 & 0 & 0 & 0 & 0 & -2\beta & 0 & -\beta & 1 + 4\beta
\end{pmatrix},
\]

(4.165)
The factor two appears due to the boundary conditions. Note that the matrix structure is naturally separated into five strips from top to bottom: The top strip is the first row, strip two is rows 2 through \( m + 1 \), strip three (the generic strip) is rows \( m + 2 \) through \( (m + 1)(n + 1) - (m + 1) \), strip four is rows \( (m + 1)(n + 1) - m \) to \( (m + 1)(n + 1) - 1 \), and strip five is the last row. The upper nonzero diagonals are the first and the \( (m + 1) \) upper diagonals; the lower nonzero diagonals are the first and the \( (m + 1) \) lower diagonals.

The vector \( b \) is defined accordingly:

\[
b(\text{nodes}(i, j)) := (1 - \lambda \left( \frac{\Delta t}{\Delta x^2} + \frac{\Delta t}{\Delta y^2} \right))U_{i,j}^k + \lambda \frac{\Delta t}{2\Delta x^2}(U_{i+1,j}^k + U_{i-1,j}^k) + \lambda \frac{\Delta t}{2\Delta y^2}(U_{i,j+1}^k + U_{i,j-1}^k) + \frac{\Delta t}{2}(f(\text{FEU}_{i,j}^k, \text{FEV}_{i,j}^k) + f(U_{i,j}^k, V_{i,j}^k)),
\]

where \( (\text{FEU}_{i,j}^k, \text{FEV}_{i,j}^k) \) is the forward Euler approximation of the full reaction-diffusion PDE with boundary conditions imposed of the approximate states \((U_{i,j}^{k+1}, V_{i,j}^{k+1})\) computed from the known approximation \((U_{i,j}^k, V_{i,j}^k)\) and the same \( \Delta t, \Delta x \) and \( \Delta y \) used for the Crank-Nicolson scheme.

Our choice of ordering determines the structure of the matrix \( A \) and vector \( b \); but, in a computer code to implement the Crank-Nicolson method, we should avoid defining the matrix \( A \)—there is no need to reserve storage for its zero elements. Since the matrices we are contemplating are large (for instance, a 129 \( \times \) 129 grid requires solving a system of 16641 equations), it is natural to take advantage of the structure of our (sparse) matrices, which have only five diagonals with nonzero components. We do not want the computer to waste our time computing values that are known to be zero.

At this juncture, we could enter the world of numerical linear algebra. This is a vast subject (see [12, 82]), which has certainly been influenced by the necessity of dealing with the matrix systems that arise in solving partial differential equations. This book is mainly about differential equations in applied mathematics, so we will not develop the theory in detail here. On the other hand, we need a viable method to approximate the solutions of the (large) linear systems that arise in implementations of implicit schemes such as the Crank-Nicolson algorithm. Hence, we must at least have a working knowledge of some fundamental results of this subject.
There are two basic methods for the numerical solution of linear systems: Gaussian elimination (which should be a familiar method from linear algebra) and fixed point iteration.

A viable implementation of Gaussian elimination for our (large and sparse) matrices demands adaptations that take into account the matrix structures. In the special case of tridiagonal matrices (which arise for PDEs with one spatial dimension), there is an adaptation of Gaussian elimination that is simple to program, fast, and practical (see [12]). For banded matrices (such as the matrix in display (4.165)), there are similar but more complicated methods. Also, in applications to PDE (and many other places), numerical algorithms require solving \( AW = b \) repeatedly for the same system matrix \( A \) but with different column vectors \( b \). Efficient numerical schemes compute and store a factorization of \( A \), for example a factorization \( PA = LU \), where \( P \) is a permutation matrix, \( L \) is a lower triangular matrix, and \( U \) is upper triangular. The basic idea, for the case where \( P = I \), is to multiply \( A \) by a sequence of elementary matrices \( E_i \) that encode row reductions so that \( E_n E_{n-1} \cdots E_1 A = U \), where \( U \) is upper triangular. This is simply Gaussian elimination in case every pivot is nonzero. The \( P \) matrix is used to make sure every pivot is nonzero and more generally to permute rows so that the largest available pivot is used, a process called partial pivoting. The reason to use the large pivots is to avoid multiplying rows by large numbers—which might cause large numerical errors for some matrices that have very small nonzero eigenvalues. For the simple case where \( P = I \), the inverse \( L \) of the product of elementary matrices is lower triangular. Thus, \( A = LU \). The advantage of this decomposition is that it stores most of the work in solving the original system \( AW = b \). The factored matrix system \( LUW = b \) is solved in two steps: \( LY = b \) and \( UW = y \). Each of the two linear systems are solved immediately by simple recursive substitution. No elimination is necessary. In case \( PA = LU \) is used, simply multiple \( b \) by \( P \) and then use the \( LU \) decomposition. There are refinements of this method for special types of matrices, especially, symmetric matrices, banded matrices, and sparse matrices.

Iterative methods used to solve linear systems are also called methods of successive relaxation. In its simplest form, the basic method is a generalization of the idea used (recall display (4.157)) to approximate solutions of the nonlinear equation \( g(x) = 0 \) by iteration of the function \( G(x) = x - \omega g(x) \) for a choice of \( \omega \) that makes the desired solution a stable fixed point of the dynamical system defined by \( G \). Indeed, the solution of a linear system \( Az = b \)
might be solved by iterating the linear transformation $T$ given by
\begin{equation}
Tz = z - \omega(Az - b),
\end{equation}
where $\omega$ is a nonzero real number. There are at least three important questions related to this procedure: (1) What is a sufficient condition to guarantee that the process converges? (2) What is the best value to choose for the parameter $\omega$? (3) What is a good choice for $z$ to start the process?

Question (3) does not have a definite answer. Surely, the optimal choice of the initial point is the solution vector for the equation $Az = b$. Since this is the value we wish to find, we will have to settle for a less than optimal choice. One useful idea for our problem is to note that we plan to solve our system many times as we step along in time; hence, we can use current values of the concentrations at the nodes as the starting value to obtain the updated concentrations.

Questions (1) and (2) are answered by doing some numerical analysis. Recall that a fixed point of a map is asymptotically stable whenever all eigenvalues of the derivative of the map at the fixed point lie inside the open unit disk in the complex plane (see Exercise 4.82). Fortunately, the derivative of a linear transformation is itself and therefore does not depend on the point at which it is evaluated. The derivative of our proposed function (4.166), which defines our linear dynamical system, is
\begin{equation}
DT = I - \omega A,
\end{equation}
where $I$ is the identity matrix. The eigenvalues of $DT$ are in correspondence with the eigenvalues of $A$; in fact, $\mu$ is an eigenvalue of $DT$ if and only if $(1 - \mu)/\omega$ is an eigenvalue of $A$.

For the matrix $A$ given by the assignments in display (4.164) where no boundary condition is taken into account, all the elements on the main diagonal of $A$ are equal. In case $\Delta x = \Delta y$ and for computational convenience $\alpha := 2\beta$, these diagonal elements are all equal to $1 + 2\alpha$. The sum of the absolute values of the off-diagonal elements in each row of $A$ is at most $2\alpha$. Moreover, $A$ is a symmetric matrix; therefore, it has real eigenvalues. By Gerschgorin’s theorem (see Appendix A.5), the eigenvalues of $A$ lie in the interval $[1, 1 + 4\alpha]$, which consists of only positive real numbers.

Let $\sigma(A)$ denote the set of eigenvalues (the spectrum) of $A$ and note that $\xi \in \sigma(A)$ corresponds to the eigenvalue $\mu = 1 - \xi \omega$ in $\sigma(DT)$. The maximum
absolute value of the eigenvalues of $DT$ (which is its spectral radius $\rho(DT)$) is given by

$$\rho(DT) = \min_{\omega \in \mathbb{R}} \max_{\xi \in \sigma(A)} |1 - \xi \omega|.$$  \hfill (4.167)

Under the assumption that $\rho(DT) < 1$, the optimal value of $\omega$ is the value at which the minimum occurs. By Exercise 4.94,

$$\rho(DT) = \frac{2\alpha}{1 + 2\alpha}, \quad \omega = \frac{1}{1 + 2\alpha}.$$  

Thus, in this case, we have that $\rho(DT) < 1$ as desired, and the optimal choice is $\omega = 1/(1 + 2\alpha)$. The sequence of iterates of the transformation $T$ will be linearly convergent to the solution of $Az = b$ for every starting vector.

The implementation of our iterative method $z^{\ell+1} = Tz^\ell$ should take advantage of the structure of the matrix $A$. In particular, there is no reason to store this matrix. The update from $z^\ell$ to $z^{\ell+1} = Tz^\ell$ is given, for $\gamma = 1, 2, 3, \ldots (m+1)(n+1)$, by

$$z^{\ell+1}_\gamma = (1 - \omega(1 + \lambda \left(\frac{\Delta t}{\Delta x^2} + \frac{\Delta t}{\Delta y^2}\right)))z^\ell_\gamma$$

$$- \omega \lambda \frac{\Delta t}{2\Delta x^2} z^{\ell+1}_\gamma - \omega \lambda \frac{\Delta t}{2\Delta y^2} z^{\ell-1}_\gamma$$

$$- \omega \lambda \frac{\Delta t}{2\Delta y^2} z^{\ell}_\gamma + (m+1) - \omega \lambda \frac{\Delta t}{2\Delta x^2} z^{\ell}_{\gamma-(m+1)};$$  \hfill (4.168)

where $z^\ell_p$ is set to zero if the index

$$p \in \{\gamma + 1, \gamma - 1, \gamma + m + 1, \gamma - (m + 1)\}$$

is not in the range of $\gamma$. We will have to encode a stopping procedure for our iteration process; for example, the procedure can be stopped as soon as an iterative update does not change by more than some prespecified tolerance from its previous value. After gaining some experience by monitoring the performance of a code, we might fix the number of iterations to decrease its execution time. The final iterate $Z_\gamma$ is used to update the matrix $U^{k+1}$ in the code used to solve the PDE setting

$$U^{k+1}_{i,j} = Z_{\text{nodes}(i,j)}.$$  

The iteration process can be accelerated by an application of Steffensen’s method. For applications of this method to systems, there are at least two
natural questions: (1) Can Steffensen’s method be applied componentwise? 
(2) Can the signs of the errors be determined to check that the hypothesis 
for the convergence of Steffensen’s method is satisfied?

The idea underlying the answers to our questions can be illustrated by 
examining the iteration of linear transformation of the plane. Let us suppose 
that $T$ is a linear transformation of the plane whose spectrum is inside the 
open unit disk. In this case, the sequence of vectors $\{x^k\}_{k=0}^\infty$ defined by the 
itration process $x^{k+1} = T x^k$ (for an arbitrary choice of $x^0$) converges to 
zero. Under the assumption that $T$ is symmetric and positive definite, the 
eigenvalues of $T$ are positive real numbers $\lambda_1 \leq \lambda_2$ and there is a basis of $\mathbb{R}^2$ 
consisting of eigenvectors of $T$. In this basis, $T$ has the matrix representation 

\[
\begin{pmatrix}
\lambda_1 & 0 \\
0 & \lambda_2 
\end{pmatrix}
\]

The eigenspaces (that is, the subspaces spanned by the eigenvectors) in this 
basis correspond to the coordinate axes. Pick a vector in the plane, say 
the vector with components $x_0$ and $y_0$ and iterate the process. The $k$th 
iterate has components $\lambda_1^k x_0$ and $\lambda_2^k y_0$. Note that if $\lambda_1 < \lambda_2$, then the 
kth iterate becomes nearly parallel to the vertical coordinate axis. In other 
words, the dot product of the $k$th iterate and the usual basis vector $e_1$ is 
small compared with the dot product of the $k$th iterate and $e_2$. The first two 
ectors are nearly orthogonal. After a few iterations, the iterates align with 
the eigenspace corresponding to the largest eigenvalue. Thus, up to a small 
error, the iterates converge monotonically to zero along this one-dimensional 
ubspace. Also, all the components of the iterates converge linearly to zero 
and the errors with respect to each component have the same signs.

Exactly the same behavior occurs in general as long as the matrix that 
determines the iteration process is positive definite and symmetric. More 
generally, it suffices to have a positive real eigenvalue in the open unit disk 
that is larger than the absolute values of the real parts of all the other 
eigenvalues.

In practice, we should compute several iterates so that the expected align-
ment takes place, apply Steffensen’s method, compute several more iterations 
so that the expected alignment takes place, and continue in this manner until 
the iterations are no longer changing up to some preassigned tolerance or a 
maximum preset number of iterations is exceeded. It is possible to check 
internally that the alignment has occurred; for example, the computed dif-
Conservation of Mass

187

ferences \(x_i^{k+1} - x_i^k\) for the components can be tested to see if they are all of the same sign and approximately the same magnitude (see Exercise 4.98).

While the iterative method just described is viable (especially when it is accelerated via Aitken extrapolation), this method is seldom used because there are superior alternative methods due to Jacobi, Gauss, Seidel and others. All of these methods are based on a simple idea. To solve \(Az = b\) for \(z\), write the system matrix \(A\) as a sum, separate the product \(Az\) accordingly into a sum, move one summand to the right-hand side of the equation, and multiply both sides by the inverse of the remaining term to set up a fixed point iteration scheme. With an appropriate choice of the splitting, it is possible to obtain new iteration schemes that (usually) converge more rapidly than iterations of \(T\) defined in display (4.166).

Suppose \(A\) is an \(n \times n\)-matrix and \(\omega\) is a nonzero scalar. To solve for \(z\) in the linear system \(Az = b\), rewrite the equation as

\[
0 = -\omega(Az - b)
\]

and split \(A\) into the sum of three \(n \times n\)-matrices: \(D\), whose main diagonal is the main diagonal of \(A\) and all its other components are zero; \(L\), the lower triangular part of \(A\) (that is, \(L\) consists of the components of \(A\) below its main diagonal and all of its other components are zero); and \(U\), similarly, the upper triangular part of \(A\). Using this notation and with the intention of defining a transformation whose fixed point is a solution of the linear system, recast equation (4.169) into

\[
0 = Dz - Dz - \omega(Dz + Lz + Uz - b),
\]

which may be rearranged to obtain the matrix equation

\[
(D + \omega L)z = ((1 - \omega)D - \omega U)z + \omega b.
\]

If \(D + \omega L\) is invertible, the unknown vector \(z\) is a fixed point of the linear transformation

\[
\zeta \mapsto (D + \omega L)^{-1}(((1 - \omega)D - \omega U)\zeta + \omega b).
\]

Of course, the idea for this construction is to obtain approximations of the unknown vector \(z\) by iteration of this transformation starting from an arbitrary initial guess. This iteration process is sometimes called successive over
relaxation (SOR). It is called an over relaxation method if \( \omega > 1 \), an under relaxation method if \( \omega < 1 \), and the Gauss-Seidel method if \( \omega = 1 \).

The formula (4.170) requires the matrix
\[
S := D + \omega L
\]
to be invertible, which is the case if and only if every element on its main diagonal is not zero. In general, the inversion of matrices should be avoided in numerical computation because the number of operations required to invert a matrix increases rapidly with its size. In fact, the number of operations is on the order of \( n^3 \) for an \( n \times n \)-matrix. In contrast, the SOR method is practical because the required inversion of a lower triangular matrix is accomplished simply and efficiently by back substitution. By inspection of the matrix \( S \), it follows immediately that the components of \( z \) in the linear system
\[
Sz = v,
\]
where \( v \) is a given \( n \) vector, are
\[
z_1 = \frac{1}{S_{11}} v_1,
\]
\[
z_i = \frac{1}{S_{ii}} (v_i - \sum_{j=1}^{i-1} S_{ij} z_j).
\]
(4.171)

In this scheme, \( z_1 \) is used to solve for \( z_2 \), \( z_1 \) and \( z_2 \) are used to solve for \( z_3 \), and so on; that is, the components of the solution are used to solve for subsequent values as soon as they are obtained. This is the underlying reason why the SOR methods are expected to converge more rapidly than the naive iteration method (4.166).

To implement SOR, choose an initial approximation \( z^0 \) and compute successive approximations \( z^k \) by the scheme
\[
v^k := [(1 - \omega) D - \omega U] z^k + \omega b,
\]
\[
(D + \omega L) z^{k+1} = v^k.
\]
(4.172)
The components \( v^k_i, i = 1, 2, 3, \ldots n \), of \( v^k \) are
\[
v^k_i = (1 - \omega) a_{ii} z^k_i - \omega \sum_{j=i+1}^{n} a_{ij} z^k_j + \omega b_i,
\]
(4.173)
Figure 4.32: The order estimates in this figure are for an implementation of the Crank-Nicolson method using the data of Problem 4.65 and the Gauss-Seidel iteration with the stopping criterion: successive iterations that differ in Euclidean norm by less than $10^{-6}$. The floating point numbers are computed for the average of the concentrations $(u + v)/2$ using formula (4.125).

and the second step is completed using the back substitution formulas in display (4.171).

The SOR algorithm will converge if the eigenvalues of the matrix

$$(D + \omega L)^{-1}((1 - \omega)D - \omega U)$$

all lie inside the unit circle in the complex plane (see Exercise 4.90). The parameter $\omega$ may be adjusted to make the eigenvalues as close to zero as possible. Of course, an analysis of the structure of the decomposition of matrix $A = D + U + L$ or numerical computation of the required eigenvalues for a particular problem might lead to a theorem that would ensure convergence. The method does not converge for every $n \times n$-matrix $A$.

The SOR method for the five-diagonal matrix $A$ arising in the Crank-Nicolson scheme (4.163) can be programmed so that only the nonzero elements of the matrix are used (cf. equation (4.168)). For a positive-definite symmetric matrix, the spectrum of the matrix

$$(D + \omega L)^{-1}((1 - \omega)D - \omega U)$$

lies in the open unit disk in the complex plane whenever $0 < \omega < 2$; hence SOR converges whenever $A$ is a positive-definite symmetric matrix and $\omega$ is in this range (see [84]). In general, it is a difficult problem to determine the optimal value of $\omega$.

Computational results for Problem 4.65 using the Crank-Nicolson algorithm, where the solutions of the corresponding matrix systems are approximated using Gauss-Seidel iteration (SOR with $\omega = 1$ and with the stopping

<table>
<thead>
<tr>
<th>$h$</th>
<th>$32 \times 32$</th>
<th>$64 \times 64$</th>
<th>$128 \times 128$</th>
<th>$256 \times 256$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta t = 1/4$</td>
<td>1.93461</td>
<td>1.93451</td>
<td>1.93497</td>
<td>1.92266</td>
</tr>
<tr>
<td>$\Delta t = 1/8$</td>
<td>1.96882</td>
<td>1.96848</td>
<td>1.96688</td>
<td>2.00855</td>
</tr>
<tr>
<td>$\Delta t = 1/16$</td>
<td>1.98589</td>
<td>1.98510</td>
<td>1.98407</td>
<td>1.83655</td>
</tr>
<tr>
<td>$\Delta t = 1/32$</td>
<td>1.99551</td>
<td>1.99388</td>
<td>1.99383</td>
<td>2.07241</td>
</tr>
<tr>
<td>$\Delta t = 1/64$</td>
<td>2.0028</td>
<td>1.99951</td>
<td>1.99816</td>
<td>2.01279</td>
</tr>
</tbody>
</table>

$\Delta t = 1/128$ | 2.01155       | 2.00488        | 2.00157          | 2.00958          |
Figure 4.33: The error estimates in this figure are for an implementation of the Crank-Nicolson method using the data of Problem 4.65 and Gauss-Seidel iteration with the stopping criterion: successive iterations that differ in Euclidean norm by less than $10^{-6}$. The floating point numbers are computed by averaging the absolute error estimates (4.127) corresponding to the two concentrations.

\begin{table}
\centering
\begin{tabular}{|c|c|c|c|c|}
\hline
$h$ & $32 \times 32$ & $64 \times 64$ & $128 \times 128$ & $256 \times 256$ \\
\hline
$\Delta t = 1/4$ & 0.00022356 & 0.00022363 & 0.00022355 & 0.00022420 \\
$\Delta t = 1/8$ & 0.00005730 & 0.00005731 & 0.00005734 & 0.00005776 \\
$\Delta t = 1/16$ & 0.00001450 & 0.00001451 & 0.00001455 & 0.00001497 \\
$\Delta t = 1/32$ & $3.65 \times 10^{-6}$ & $3.65 \times 10^{-6}$ & $3.64 \times 10^{-6}$ & $4.14 \times 10^{-6}$ \\
$\Delta t = 1/64$ & $9.1 \times 10^{-7}$ & $9.1 \times 10^{-7}$ & $9.2 \times 10^{-7}$ & $9.8 \times 10^{-7}$ \\
$\Delta t = 1/128$ & $2.3 \times 10^{-7}$ & $2.3 \times 10^{-7}$ & $2.3 \times 10^{-7}$ & $2.4 \times 10^{-7}$ \\
\hline
\end{tabular}
\end{table}

Figure 4.34: The order estimates in this figure are for an implementation of the Crank-Nicolson method using the data of Problem 4.64 and Gauss-Seidel iteration with the stopping criterion: successive iterations that differ in Euclidean norm by less than $10^{-6}$. The floating point numbers are computed for the average of the concentrations $(u + v)/2$ using formula (4.125).

\begin{table}
\centering
\begin{tabular}{|c|c|c|c|c|}
\hline
$h$ & $32 \times 32$ & $64 \times 64$ & $128 \times 128$ & $256 \times 256$ \\
\hline
$\Delta t = 1/4$ & 1.20052 & 2.97511 & 1.97269 & 2.06121 \\
$\Delta t = 1/8$ & 1.42807 & 4.57200 & 2.17178 & -0.457879 \\
$\Delta t = 1/16$ & 1.65060 & 0.41385 & 1.94150 & 0.03870 \\
$\Delta t = 1/32$ & 0.99641 & 1.79209 & 1.96255 & 3.38090 \\
$\Delta t = 1/64$ & 4.77933 & 1.94427 & 1.98211 & 3.95762 \\
$\Delta t = 1/128$ & -4.55519 & 1.97100 & 1.99247 & 1.69587 \\
\hline
\end{tabular}
\end{table}

criterion: successive iterations that differ in Euclidean norm by less than $10^{-6}$), are reported in Figures 4.32 and 4.33. The order estimates in Figure 4.32 suggest that this implementation of the algorithm is performing as expected at order-two in time. The computed value of the average over both concentrations is consistently (over the viable grid and step sizes) 0.453021. The averaged $u$ concentration is consistently 0.808316 and the averaged $v$ concentration is 0.0977254. These values agree with the values obtained using the forward Euler method reported on page 161. We can have a high level of confidence that these values are within 1% of the exact corresponding values.

Computational results for Problem 4.64, using the Crank-Nicolson algorithm and Gauss-Seidel iteration, are reported in Figures 4.34 and 4.35. The
Figure 4.35: The error estimates in this figure are for an implementation of the Crank-Nicolson method using the data of Problem 4.64 and Gauss-Seidel iteration with the stopping criterion: successive iterations that differ in Euclidean norm by less than $10^{-6}$. The floating point numbers are computed by averaging the absolute error estimates (4.127) corresponding to the two concentrations.

<table>
<thead>
<tr>
<th>$h$</th>
<th>$32 \times 32$</th>
<th>$64 \times 64$</th>
<th>$128 \times 128$</th>
<th>$256 \times 256$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta t = 1/4$</td>
<td>$3.42 \times 10^{-9}$</td>
<td>$1.91 \times 10^{-3}$</td>
<td>$2.42 \times 10^{-3}$</td>
<td>$2.08 \times 10^{-3}$</td>
</tr>
<tr>
<td>$\Delta t = 1/8$</td>
<td>$1.70 \times 10^{-9}$</td>
<td>$4.80 \times 10^{-4}$</td>
<td>$5.63 \times 10^{-4}$</td>
<td>$1.00 \times 10^{-3}$</td>
</tr>
<tr>
<td>$\Delta t = 1/16$</td>
<td>$8.49 \times 10^{-10}$</td>
<td>$1.20 \times 10^{-4}$</td>
<td>$1.43 \times 10^{-4}$</td>
<td>$8.21 \times 10^{-4}$</td>
</tr>
<tr>
<td>$\Delta t = 1/32$</td>
<td>$4.24 \times 10^{-10}$</td>
<td>$3.01 \times 10^{-5}$</td>
<td>$3.60 \times 10^{-5}$</td>
<td>$1.06 \times 10^{-4}$</td>
</tr>
<tr>
<td>$\Delta t = 1/64$</td>
<td>$2.11 \times 10^{-10}$</td>
<td>$7.54 \times 10^{-6}$</td>
<td>$9.05 \times 10^{-6}$</td>
<td>$3.00 \times 10^{-5}$</td>
</tr>
<tr>
<td>$\Delta t = 1/128$</td>
<td>$1.07 \times 10^{-10}$</td>
<td>$1.89 \times 10^{-6}$</td>
<td>$2.97 \times 10^{-6}$</td>
<td>$1.40 \times 10^{-5}$</td>
</tr>
</tbody>
</table>

Figure 4.36: The average concentrations in this figure are for an implementation of the Crank-Nicolson method using the data of Problem 4.64 and Gauss-Seidel iteration with the stopping criterion: successive iterations that differ in Euclidean norm by less than $10^{-6}$. The floating point numbers are the computed averages of the concentrations $(u + v)/2$.

<table>
<thead>
<tr>
<th>$h$</th>
<th>$32 \times 32$</th>
<th>$64 \times 64$</th>
<th>$128 \times 128$</th>
<th>$256 \times 256$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta t = 1/4$</td>
<td>0.485018</td>
<td>0.463835</td>
<td>0.352109</td>
<td>0.335842</td>
</tr>
<tr>
<td>$\Delta t = 1/8$</td>
<td>0.485018</td>
<td>0.463835</td>
<td>0.352000</td>
<td>0.335848</td>
</tr>
<tr>
<td>$\Delta t = 1/16$</td>
<td>0.485018</td>
<td>0.463835</td>
<td>0.352098</td>
<td>0.335842</td>
</tr>
<tr>
<td>$\Delta t = 1/32$</td>
<td>0.485018</td>
<td>0.463835</td>
<td>0.352097</td>
<td>0.335842</td>
</tr>
<tr>
<td>$\Delta t = 1/64$</td>
<td>0.485018</td>
<td>0.463835</td>
<td>0.352097</td>
<td>0.335842</td>
</tr>
<tr>
<td>$\Delta t = 1/128$</td>
<td>0.485018</td>
<td>0.463835</td>
<td>0.352097</td>
<td>0.335842</td>
</tr>
</tbody>
</table>
order estimates in Figure 4.34 suggest that this implementation of the algorithm is performing at the expected order-two in time. The computed values of the averages over both concentrations converge, and these values are in good agreement with the values obtained by the forward Euler method. Perhaps the most trustworthy results are for the 128 × 128-grid with Δt = 1/4. The corresponding computed value for the averaged u concentration is 0.531848 and the averaged v concentration is 0.139837. These values also agree with the values obtained using the forward Euler method reported on page 161. The confidence level for these results extends to two or three decimal places at best.

How can we obtain more accurate results for Problem 4.64?

As already mentioned, we could use the trapezoidal together with Newton’s method to approximate the reaction terms.

We could use a direct method (some adapted form of Gaussian elimination) to solve the large linear systems that arise in the Crank-Nicolson method. This would avoid errors due to iteration. The difficulty is to incorporate the matrix structure into the method. Also, we could incorporate a more accurate approximation for the boundary conditions; for example, we could use an approximation based on the formula

\[
F'(a) = \frac{1}{12\Delta x}(F(a-2\Delta x)-8F(a-\Delta x)+8F(a+\Delta x)-F(a+2\Delta x))+O(\Delta x^2).
\]

(4.174)

Another idea is to use a more accurate spatial discretization. Instead of centered differences for the second derivatives, we could use higher-order approximations, for instance, the approximation

\[
F''(a) = -\frac{1}{12\Delta x^2}(F(a-2\Delta x) - 16F(a - \Delta x) + 30F(x) - 16F(a + \Delta x) + F(a + 2\Delta x)) + O(\Delta x^4),
\]

(4.175)

which is an alternate form of equation (4.121) (see Exercise 4.95).

Of course, there are many other methods to try. Can you devise a method that produces more accurate results for Problem 4.64?

Exercise 4.73. Suppose that \( f : \mathbb{R}^n \rightarrow \mathbb{R}^n \) is a smooth function. Prove that the initial value problem \( \dot{u} = f(u), \ u(0) = u_0 \) is equivalent to the integral equation

\[
u(t) = u_0 + \int_0^t f(u(s)) \, ds.
\]
Exercise 4.74. Suppose that the local truncation error with discretization step size $h$ for some method is $O(h^{n+1})$. Assume that the global error for the computation is the number of steps times the error per step. If the computation is carried out over an interval of length $L$ show that the global error is proportional to $Lh^n$; that is, the global error is $O(h^n)$.

Exercise 4.75. Prove that Newton’s method is quadratically convergent. Hint: Write the Newton iteration scheme as a function so that $x^{k+1} = F(x^k)$ and use the Taylor series approximation for $F$ at the fixed point of $F$.

Exercise 4.76. (a) Derive the vector form of Newton’s method and use it to write a numerical code formulated to approximate the rest points of the system of ODEs given by

$$
\dot{U}_{ij} = \epsilon(U_{i+1,j} - 2U_{ij} + U_{i-1,j} + U_{i,j+1} - 2U_{ij} + U_{i,j-1}) - U_{ij}V_{ij}^2 + F(1 - U_{ij}),
$$

$$
\dot{V}_{ij} = \mu(V_{i+1,j} - 2V_{ij} + V_{i-1,j} + V_{i,j+1} - 2V_{ij} + V_{i,j-1}) + U_{ij}V_{ij}^2 - (F + \kappa)V_{ij},
$$

for $i = 1, 2, 3, \ldots m$ and $j = 1, 2, 3, \ldots n$. Check that your implementation of Newton’s method is quadratically convergent. (b) Find all the rest points in case $\lambda = 2.0 \times 10^{-5}$, $\mu = 10^{-5}$, $\kappa = 0.05$, $F = 0.02725$, $m = 32$, and $n = 32$. (c) Repeat part (b) for $m = n = 64$, 128, and, 256. Check for convergence of the rest points to values independent of the choice of the grid size.

Exercise 4.77. Recall Exercise 1.17 concerning the ODE

$$
\dot{x} = 1, \quad \dot{y} = axy,
$$

where $a$ is a parameter. The solution of the initial value problem with data $x(0) = y(0) = -1$ has the exact value $(x(2), y(2)) = (1, -1)$ at $t = 2$ independent of $a$. Apply forward Euler, backward Euler, improved Euler, and the trapezoidal method to this ODE, and determine the largest value of $a$ for which each of your numerical codes returns the correct answer.

Exercise 4.78. An alternative to Newton’s method for vector functions is “Newton’s method one variable at a time (NOVAT):” Suppose we wish to apply Newton’s method to find a zero of the function $f : \mathbb{R}^n \to \mathbb{R}^n$ and this function is given in coordinates by

$$
y_1 = f_1(x_1, x_2, \ldots, x_n), \quad y_2 = f_2(x_1, x_2, \ldots, x_n), \quad \ldots, \quad y_n = f_n(x_1, x_2, \ldots, x_n).
$$
The alternative iteration scheme is

\[ x_1^{k+1} = x_1^k - \frac{f_1(x_1^k, x_2^k, \ldots, x_n^k)}{\frac{\partial f_1}{\partial x_1}(x_1^k, x_2^k, \ldots, x_n^k)} \]

\[ x_2^{k+1} = x_2^k - \frac{f_2(x_1^{k+1}, x_2^k, \ldots, x_n^k)}{\frac{\partial f_2}{\partial x_2}(x_1^{k+1}, x_2^k, \ldots, x_n^k)} \]

\[ \vdots \]

\[ x_n^{k+1} = x_n^k - \frac{f_n(x_1^{k+1}, x_2^{k+1}, \ldots, x_{n-1}^{k+1}, x_n^k)}{\frac{\partial f_n}{\partial x_n}(x_1^{k+1}, x_2^{k+1}, \ldots, x_{n-1}^{k+1}, x_n^k)} \]

(a) Write a code to implement NOVAT and test its convergence rate on several examples where the root is known in advance. Does it converge quadratically? Report your results. (b) Code an ODE solver using the trapezoidal method together NOVAT to solve the implicit equation for the updated state variable. Apply your code to several ODE systems and report your result. (c) Show that this version of Newton’s method applied to a linear system \( Ax = b \) produces Gauss-Seidel iteration. (d) Construct an example where Newton’s method converges but NOVAT does not. (e) Is there an example where NOVAT converges but Newton’s method does not? (f) Determine the criterion for convergence of NOVAT.

Exercise 4.79. Test the code for Exercise 4.78 on the ODE of Exercise 1.17. Compare the results obtained with the improved Euler method.

Exercise 4.80. (a) Working definition: An ODE with solutions that converge (or diverge) exponentially fast to (or from) other solutions is called stiff. There is no universally accepted definition of stiffness. An alternative definition might be the following: An ODE is stiff if some numerical methods require small step sizes to make accurate approximations of some of its solutions. The simplest example of a stiff ODE is \( \dot{x} = \lambda x \), where \( \lambda \neq 0 \). If \( \lambda \) is negative, then all solutions approach the zero solution exponentially fast. Write the formulas and prove this fact. We may gain some insight on the performance of ODE solvers on stiff equations by applying them to this equation. For the analysis, it suffices to consider the solution of the ODE with initial condition \( x(0) = 1 \). Show that Euler’s method applied to the test ODE gives the iteration scheme \( x^{k+1} = (1 + \lambda \Delta t)x^k \). In case \( \lambda < 0 \), we know the solution decays. Show that the Euler approximation will decay if and only if \( |1 + \lambda \Delta t| < 1 \) and that this puts a restriction on the step size \( \Delta t \): the step size must be smaller than \( 2/|\lambda| \). Thus, if \( \lambda \) is large, the step size must be small to have a chance of making an accurate approximation. Show that the same restriction applies in case \( \lambda > 0 \). An alternative view is the restriction \( |\lambda \Delta t| < 2 \). The set of points \( z \) in the complex plane (which must be considered for some more
Conservation of Mass

complicated examples) with \( |z| < 2 \) is called the region of absolute stability. (b) Determine the region of absolute stability for the improved Euler method. (c) Show that the region of absolute stability for the trapezoidal method is the entire complex plane; that is, there is no restriction on the step size. This result tells us that the trapezoidal method is useful to approximate solutions of stiff ODEs. It should perform well with larger step sizes than will be required for the one step methods we have discussed. The trapezoidal method is a viable method for many stiff ODEs if the accuracy requirements are not too restrictive. Higher order methods have been devised for stiff equations. To learn more, consult any book on the numerical analysis of ODE solvers. (d) Is the ODE in Exercise 1.17 stiff? Discuss your answer. (e) Consider the ODE

\[
\dot{x} = \lambda x - y - \lambda (x^2 + y^2)x, \quad \dot{y} = x + \lambda y - \lambda (x^2 + y^2)y
\]

with initial data \((x(0), y(0)) = (\xi, 0)\). Derive the exact solution

\[
x(t) = \frac{\xi e^{\lambda t}}{\sqrt{1 + \xi^2(e^{2\lambda t} - 1)}} \cos t, \quad y(t) = \frac{\xi e^{\lambda t}}{\sqrt{1 + \xi^2(e^{2\lambda t} - 1)}} \sin t.
\]

Hint: Change to polar coordinates and recall Bernoulli’s differential equation. (f) Show that the ODE of part (e) is stiff. (g) Test codes for improved Euler and trapezoidal integration with \(\lambda\) small (perhaps \(\lambda = 1\)) and big (perhaps \(\lambda = 10\)) for \(\rho = 4\) and integration until \(t = 6\). Do both codes return the correct answer? What is the maximum allowable step size for your codes to return an answer with relative error less than 1%?

**Exercise 4.81.** Write a code for solving the Gray-Scott model using the Crank-Nicolson method and the trapezoidal method for the reaction terms. Use Newton’s method to solve the nonlinear equations for the updates. Apply the code to Problems 4.64 and 4.65.

**Exercise 4.82.** Consider the function \(G : \mathbb{R}^n \to \mathbb{R}^n\) as a dynamical system; that is, the initial state \(x = x_0\) in \(\mathbb{R}^n\) evolves according to the rule \(x^{k+1} = G(x^k)\). A fixed point of \(G\) is a state \(x_0\) such that \(G(x_0) = x_0\). A fixed point is called stable if for each \(\epsilon > 0\) there is a \(\delta > 0\) such that all (forward) iterates of every initial state in the (open) ball of radius \(\delta\) centered at the fixed point are in the ball of radius \(\epsilon\). A fixed point is called asymptotically stable if it is stable and in addition if the sequence of iterates of every state starting in some ball of radius \(\delta\) centered at the fixed point converges to the fixed point. (a) Prove that a fixed point \(x_0\) is asymptotically stable if \(n = 1\) and derivative \(|G'(x_0)| < 1\). (b) The function \(G(x) = x - \alpha(x^3 + x^2 - x - 1)\) has a fixed point at \(x = 1\) for every \(\alpha \in \mathbb{R}\). For which \(\alpha\) is this fixed point asymptotically stable. (c) Prove that a fixed point \(x_0\)
Conservation of Mass

is asymptotically stable if all the eigenvalues of the derivative $DG(x_0): \mathbb{R}^n \to \mathbb{R}^n$ are in the open unit disk in the complex plane.

**Exercise 4.83.** (a) Show that Aitken’s $\Delta^2$ method applied to the sequence of iterates of the function $G$ defined in display (4.157) with starting value $x_0 = 2$ is not quadratically convergent. (b) Show that Steffensen’s method is quadratically convergent. (c) It is possible to prove that Aitken’s $\Delta^2$ method applied to a sequence of iterates of a function is superlinearly convergent; that is,

$$
\lim_{k \to \infty} \frac{|\text{Aitken}(k+1) - x^\infty|}{|\text{Aitken}(k) - x^\infty|} = 0.
$$

Give some (convincing) numerical evidence for this fact.

**Exercise 4.84.** Consider the following boundary value problem: Find a solution $t \mapsto \phi(t)$ of the differential equation $\ddot{x} + x - x^3 = 0$ that satisfies the conditions $\phi(0) = 0$, $\phi(2) = 0$, and $\dot{\phi}(t) > 0$ for $0 < t < 2$. (a) Approximate the solution (is there only one?) by the shooting method; that is, let $t \mapsto x(t, \xi)$ denote the solution of the ODE such that $x(0, \xi) = 0$ and $\dot{x}(0, \xi) = \xi$. Use a numerical method to find a root of the equation $x(2, \xi) = 0$. Suggestion: Draw the phase portrait of the ODE and use it to determine a reasonable starting value $\xi_0$. Perform some numerical experiments to determine a value for $\lambda$ so that the iterates of $G(\xi) = \xi - \lambda x(2, \xi)$ converge. Accelerate the convergence using Steffensen’s method. (b) Shoot with Newton’s method. Hint: Find the required derivative by solving (numerically) a variational equation. (c) The suggested methods in parts (a) and (b) may not be the best. Don’t shoot! Approximate the solution of the BVP by viewing the ODE as a PDE with Dirichlet boundary conditions. (d) Discuss the accuracy and efficiency of various methods for solving the boundary value problem. Perhaps you can find a new method that is better than any known method.

**Exercise 4.85.** Show that the improved Euler method (4.160) is second order.

**Exercise 4.86.** (a) Prove that the nodes function defined in display (4.162) is bijective. (b) Show how to determine the formula for the nodes function by supposing the function is affine and solving a system of linear equations.

**Exercise 4.87.** Consider the $(x, y)$ data

$$(0, 0.09), (1/4, 0.12), (1/2, 0.16), (3/4, 0.20), (1, 0.26), (5/4, 0.32), (3/2, 0.39),$$

$$(7/4, 0.46), (2, 0.54), (9/4, 0.61), (5/2, 0.68), (11/4, 0.75), (3, 0.80), (13/4, 0.85),$$

$$(7/2, 0.88), (15/4, 0.91), (4, 0.93), (17/4, 0.95), (9/2, 0.96), (19/4, 0.97), (5, 0.98).$$
Conservation of Mass

The theory underlying the experiment that produced the data implies that the data should be on the graph of a function of the form

\[ f(x, a, b) = \frac{e^{a+bx}}{10 + e^{a+bx}}, \]

where \( a \) and \( b \) are parameters. Use nonlinear regression to find the best constants \( a \) and \( b \). To do this, denote the data points with \((x_i, y_i)\), for \( i = 1, \ldots, 21 \), and define a new function by

\[ \Gamma(a, b) = \sum_{i=1}^{21} (y_i - f(x_i, a, b))^2. \]

The desired \((a, b)\) is the minimum of \( \Gamma \). To find this point, use calculus. Take the derivative and set it equal to zero. Use Newton’s method to solve the resulting nonlinear equations.

**Exercise 4.88.** Consider the \((x, y)\) data

\[
(250, 0.0967), (500, 0.159), (750, 0.201), (1000, 0.233), (1250, 0.257), (1500, 0.276), \\
(1750, 0.291), (2000, 0.305), (2250, 0.315), (2500, 0.325), (2750, 0.333), (3000, 0.340), \\
(3250, 0.346), (3500, 0.351), (3750, 0.356), (4000, 0.360).
\]

The theory (Michaelis-Menten kinetics) underlying the experiment that produced the data implies that the data should be on the graph of a function of the form

\[ f(x, a, b) = \frac{ax}{b + x}, \]

where \( a \) and \( b \) are parameters. Use nonlinear regression, as in Problem 4.87, to find the best constants \( a \) and \( b \).

**Exercise 4.89.** Find a nonsingular matrix \( A \) and a nonzero vector \( b \) such that there is no choice of \( \omega \) such that iteration of \( T \), defined to be \( Tz = z - \omega(Az - b) \), converges to the solution of the linear system \( Az = b \).

**Exercise 4.90.** (a) The iteration schemes mentioned in this chapter for approximating solutions of \( Ax = b \) all have the general form \( x^{k+1} = Mx^k + a \), where \( M \) is an \((n \times n)\)-matrix and \( a \) is a given \( n \)-dimensional vector. Show that if the sequence \( x^k \) produced by the method converges, then the limit of the sequence satisfies the equation \( x = Mx + a \); in other words, the limit is a fixed point of the map \( x \mapsto Mx + a \).

(b) Show: If \( M \) is a diagonal matrix and every eigenvalue of \( M \) has absolute value
less than unity, then the iteration scheme converges for every starting point \( x^0 \).

(c) Show: If \( M \) is a diagonalizable matrix and every eigenvalue of \( M \) has absolute value less than unity, then the iteration scheme converges for every starting point \( x^0 \).

(d) Show: If \( M \) is in Jordan canonical form and every eigenvalue of \( M \) has absolute value less than unity, then the iteration scheme converges for every starting point \( x^0 \).

(e) Show: If \( M \) every eigenvalue of \( M \) has absolute value less than unity, then the iteration scheme converges for every starting point \( x^0 \).

(f) Show: In all cases, the fixed point is unique; that is, the same fixed point is obtained independent of the starting point.

Hint: Define the error \( e^k := x - x^k \), where \( x \) is the fixed point. Show that the desired convergence follows if the sequence produced by \( e^{k+1} = Me^k \), independent of the starting point \( e^0 \), converges to zero. Also, the same is true, if the sequence of matrices \( M^k \) converges to zero.

**Exercise 4.91.** Is Newton’s method, which is quadratically convergent, a useful choice for solving linear systems of equations?

**Exercise 4.92.** (a) Consider the Crank-Nicolson scheme applied to the heat equation \( u_t = \kappa u_{xx} \) on a finite interval with zero Dirichlet boundary conditions at the end of the interval. Prove that this numerical method is unconditionally numerically stable. (b) Repeat part (a) for zero Neumann boundary conditions.

**Exercise 4.93.** (a) Derive the backward Euler method for \( \dot{x} = f(x) \):

\[
x^{k+1} = x^k + \Delta t f(x^{k+1}).
\]

(b) Show this method is first order. (c) Write a numerical code to implement backward Euler that incorporates Newton’s method to solve for the state variable update at each time step. (d) Repeat the computations for Problems 4.64 and 4.65 using the backward Euler method and compare results obtained with the forward Euler method. (e) Apply the backward Euler scheme to obtain a finite difference method for approximating solutions of the heat equation \( u_t = \kappa u_{xx} \). Is the method unconditionally numerically stable?

**Exercise 4.94.** Show that

\[
\rho(DT) = \frac{2\alpha}{1+2\alpha}, \quad \omega = \frac{1}{1+2\alpha}
\]

in the context of equation (4.167). Hint: The maximum occurs at an end point of the interval \([1 - \alpha, 1 + 3\alpha]\). Draw graphs of the functions \( \omega \mapsto |1 - (1 - \alpha)\omega| \) and \( \omega \mapsto |1 - (1 + 3\alpha)\omega| \).
Exercise 4.95.  (a) Derive formula (4.174) directly from Taylor series approximations of $F$. (b) Derive formula (4.175) directly from Taylor series approximations of $F$.

Exercise 4.96. Write a code to approximate solutions of the Gray-Scott model using the Crank-Nicolson method with periodic boundary conditions, and compare the results with those obtained using the forward Euler method.

Exercise 4.97.  [Method of Lines] The method of lines is a useful numerical method for some PDEs. For the case of reaction-diffusion equations, the idea is very simple: Discretize in space but not in time and treat the resulting equations (one for each spatial node) as a (perhaps large) system of ODEs (with time as the independent variable). The forward Euler method is exactly an example of the method of lines where Euler’s method is used to solve the system of ODEs. More sophisticated methods can be used to solve the ODEs. (a) Solve the BVP (4.114) by the method of lines using the improved Euler (explicit) method to solve the resulting systems of ODEs. Note: This is not the Crank-Nicolson method; the method of lines is explicit. (b) Solve Problem 4.65 by the method of lines using the improved Euler (explicit) method to solve the system of ODEs. The next part of this problem requires standard numerical methods for approximating the solutions of ODEs that are not explained in this book, but are found in standard textbooks (see, for example, [12]). (c) Solve the BVP (4.114) by the method of lines using 4th-order Runge-Kutta. (d) Solve the BVP (4.114) by the method of lines using Runge-Kutta-Fehlberg, a method that includes adaptive step size control. (e) Solve the BVP (4.114) by the method of lines using a 4th-order (Adams) predictor-corrector, a multi-step method that uses several previously computed steps. (f) Solve the BVP (4.114) by the method of lines using your favorite ODE integration method.

Exercise 4.98. Implement Steffensen’s method (incorporating the ideas discussed on Page 186) to solve five-diagonal $(m+1)^2 \times (m+1)^2$-matrix systems $Ax = b$ whose nonzero diagonals are the main diagonal of $A$, the lower $(m+1)$st diagonal, the lower first diagonal, the upper first diagonal, and the upper $(m+1)$st diagonal.

(a) Use your code to solve the system $Ax = b$, where the components of the vector $b$ all equal to one, all components on the main diagonal of $A$ are equal to five and all elements on the lower $(m+1)$st diagonal, the lower first diagonal, the upper first diagonal, and the upper $(m+1)$st diagonal are equal to one. (b) Incorporate Steffensen’s method into a Crank-Nicolson code and use it to solve the BVP (4.114).
**Exercise 4.99.** Jacobi’s method for approximating solutions of \( Ax = b \) is obtained by splitting the system matrix as \( A = D + (L + U) \), where \( D \) is the main diagonal and \( L+U \) is the off diagonal part of the matrix (which for lack of a better notation, is written here as the sum of the strict lower \( L \) and upper \( U \) triangular parts of \( A \)). The iterative method is simply obtained by moving the off diagonal part to the right hand side:

\[
Dx^{k+1} = b - (L + U)x^k.
\]

(a) Show that this method is given in components by

\[
x_i^{k+1} = \frac{1}{a_{ii}} \left( b - \sum_{j \neq i} a_{ij} x_j^k \right).
\]

(b) Write the Gauss-Seidel method in components.
(c) Discuss the statement: “The Jacobi method programmed badly is Gauss-Seidel.” Hint: Think about mathematical = and computer code =, which often means replacement. (d) Set up and run tests to compare the number of iterations required to achieve some prespecified accuracy for Jacobi and Gauss-Seidel iteration. Report on your results.

**Exercise 4.100.** [Research Project] The method of lines (Exercise 4.97) yields a window into pattern formation for the Gray-Scott model: The system of ODEs can be viewed as a perturbation problem where the unperturbed system is the reaction at each node and the discretized diffusion is the couplings between the nodes. The persistence and bifurcation from structures in the reaction equations might be used to account for the pattern formation. (a) Show that spatial pattern formation requires distinct diffusion coefficients. Hint: Use the system of PDEs and subtract the two equations. (b) Show that the coupled systems given by the method of lines have periodic solutions for sufficiently small diffusion coefficients.

**Exercise 4.101.** Recall the two-dimensional Oregonator reaction model (4.40). Add diffusion and explore spatial pattern formation for this model using the methods developed in this chapter.

**Exercise 4.102.** Suppose that the current dimensionless concentration of a substance \( u \) in space is \( u(w, t) \) at the dimensionless spatial position \( w = (x, y, z) \) and dimensionless time \( t \). Suppose the measured current concentration is approximately

\[
\frac{z(1 - e^{-(1-x^2-y^2)})}{(1 + z^2)^2}
\]
for \( w \) in the half cylinder \( \{(x, y, z) : x^2 + y^2 \leq 1, z \geq 0\} \) and zero everywhere else. The process that led to the current state is believed to be modeled by the three-dimensional diffusion equation \( u_t = \frac{1}{2} \Delta u \). Determine the dimensionless concentration at the point \( w = (1, 0, 2) \) at time \( t = -1 \). Hint: Solutions of the diffusion equation do not make sense for negative time. To treat the problem, consider the process beginning at time \( t = -1 \) and moving forward to time zero. Alternatively, reverse the direction of time in the dynamical equation; that is, work with the PDE \( u_t = -\frac{1}{2} \Delta u \). While it is possible to use the numerical methods developed in this chapter to obtain an approximation of the desired value, alternative methods are more efficient. The key result states that the solution of \( u_t = k \Delta u \) with bounded continuous initial data \( u(w, 0) = f(w) \) is

\[
    u(w, t) = \int_{\mathbb{R}^3} K(w - \omega) f(\omega) d\omega,
\]

where \( K \), called the heat kernel (or diffusion kernel), is given in \( n \)-dimensional space by

\[
    K(p) = \frac{1}{(4\pi kt)^{n/2}} e^{-|p|^2/(4kt)}.
\]

The desired concentration value may now be determined by approximating an integral over three-dimensional space. A numerical method that is widely used due to its simplicity is Monte Carlo integration. Suppose we desire the value of the integral \( \int_a^b g(x) \, dx \). We may use a random number generator to generate a finite sequence \( \{x_j\}_{j=1}^N \) of (uniformly distributed) random numbers in the interval \([a, b]\). An approximate value of the integral is given by

\[
    \int_a^b g(x) \, dx \approx \frac{b - a}{N} \sum_{j=1}^N g(x_j).
\]

Try this method on a few test cases to convince yourself that it gives reasonable approximations and discuss why the method should work. The integral of the heat kernel (for all \( k \) and \( t \) such that \( kt > 0 \)) over all of space is exactly one. Check this by Monte Carlo integration. (A better way to apply Monte Carlo integration in this case is to notice the relation between the heat kernel and the normal distribution function from probability theory.) Use Monte Carlo integration to solve the original problem. Check the result by using an alternative method to approximate the solution of the heat equation.

**Exercise 4.103.** [Modeling Project] According to folklore, starting a fire in a fireplace warms the room in which the fireplace resides and cools the outlying rooms. The mechanism for this process is the movement of air required by the
Conservation of Mass

fire. Air from the outlying rooms used by the fire is replaced by cold outside air. Develop a mathematical model to describe this physical situation and determine to what extent this bit of folklore is true.

Exercise 4.104. (a) Revisit Exercise 4.71 using a second-order in time method. Approximate the sensor output for the model (4.132). In particular, determine the length of time in hours until the sensor output is 2.0 mg / L. (c) Compare and contrast simulated sensor output and efficiency of your code relative to Euler’s method for time-stepping. In particular, discuss the time-step sizes that may be used in these simulations.

Exercise 4.105. Revisit Exercise 4.72 using a second-order in time method. (a) How long is the transient in hours measured from the instant the flow starts until the root mean square distance of the density profile in the porous block is within 1% of the steady state density profile. Compare and contrast simulated sensor output and efficiency of your code relative to Euler’s method for time-stepping. In particular, discuss the time-step sizes that may be used in these simulations.

4.4 Excitable Media: Transport of Electrical Signals on Neurons

The most important discovery of the 20th Century in biophysics is the understanding of how nerves transmit information. The basic discovery relates the transport of ions of sodium and potassium (also sodium and calcium) across the outer membrane of a nerve cell to electrical signals that may propagate along the membrane after an appropriate stimulation. Alan Hodgkin and Andrew Huxley (working in the early 1950s) described the biological basis of the ion transport, created a mathematical model, and explained experimental data on electrical signals excited in squid giant axons; they were awarded the Nobel Prize in Physiology or Medicine in 1963.

The original Hodgkin-Huxley model is a system of four ODEs. It was not meant as a predictive model as it does not include the details of the ion transport. The utility of the model lies in its aid to understanding the qualitative behavior or signals on neurons. Simplifications of the basic model, modifications for other excitable media (for example muscle cells) and spatial dependence have been extensively investigated. One of the most influential simplifications of the Hodgkin-Huxley model was introduced by Richard Fitzhugh, who also pioneered its mathematical and numerical analysis. An electric circuit analog for a similar model was constructed by Jin-Ichi Nagumo. Their
two-state model—which is still widely used in the study of the qualitative properties neural networks—describes the qualitative electrical behavior of stimulated nerve cells. We will investigate this model.

Excitable media in biology are far from being completely understood. Much contemporary work is focused on ion transport. Living membranes contain a variety of ion channels (across the membrane) that are selective to specific ions. The transport mechanisms and the switches that open and close ion channels are fundamental to the function of many biological processes. Also, networks of nerve cells and other excitable media are ubiquitous in biology. The study of such networks may lead to an understanding of how the brain works. Mathematics is playing an increasingly important role in this area of interdisciplinary research.

4.4.1 The Fitzhugh-Nagumo Model

The Fitzhugh-Nagumo model treats the nerve membrane as the electric circuit depicted in Figure 4.37. The differential equations for the important states, the voltage $V_D$ across the diode and the current through the inductor $I_L$ are obtained using standard circuit theory.