

Numerical Methods for Partial Differential Equations

Lt Col Kyle Novak
Air Force Institute of Technology
Math 676, Summer 2009

Revised: August 28, 2009

Preface

These notes are a distilled adaptation of the lecture notes for Math 712-713 taught by Professor Shi Jin at the University of Wisconsin-Madison during Fall 2003 and Spring 2004 semesters which provided a first year applied math PhD student with the essential tools of scientific computing. While time limitations prevent this one quarter course from being a complete survey of numerical methods for PDEs, the course is far from being a cursory overview. In this course, we shall examine three classifications of partial differential equations (PDEs): parabolic, hyperbolic and elliptic.

Parabolic equations include equations such as the heat or diffusion equation. Solutions are always smooth and grow smoother over time. An important related set of problems includes the reaction-diffusion equations which describe ice melting in a pool of water, the patterning of stripes on a zebra or spots on a leopard, and the aggregation of bacteria or tumor cells in response to chemical signals.

Hyperbolic equations are often used to describe low viscosity gas and particle dynamics. Nonlinear hyperbolic equations are often synonymous with shock waves and are used to model anything from supersonic flight to bomb blasts to tsunamis to traffic flow. Without a smoothing term, discontinuities may appear but do not disappear. To handle these equations mathematically, an alternative weak solution is needed. Adding viscosity to the equation regularizes the solution and brings us back to parabolic equations.

In some regards, elliptic equations may be viewed as the steady-state solutions to parabolic equations. They include the Laplace's equation and the Poisson equation. Unlike parabolic and hyperbolic equations which are typically time-dependent, elliptic equations are usually time-independent. They are used to model strain in materials, steady-state distribution of electric charge, and so forth.

Often, a problem does not neatly fit into any given category. And sometimes the problem behaves very differently over different length and time scales. For example, the Navier-Stokes equation which describes fluid flow is predominantly diffusive on one length/time scale but it is predominately hyperbolic on another. The melting of ice occurs only at thin interface region separating the liquid and solid. Deflagration (and detonation) has two timescales—a slow diffusion and a fast chemical reaction timescale. These so-called stiff problems require some consideration.

In this course, we examine three main numerical tools for solving PDEs: finite difference methods, finite element methods, and Fourier spectral methods.

Finite difference (and finite volume) methods are the oldest. They were employed in the 1950s and developed throughout the twentieth century (and before that). Their simple

formulation allows them to be applied to a large class of problems. However, the finite difference methods become cumbersome when the boundaries are complicated.

Finite element methods were developed in the 1960s as a means of solving problems that finite difference methods are not well adapted to such as handling complicated boundaries. Hence, finite element methods are attractive solutions to engineering problems. Finite element methods often require much more numerical and mathematical machinery than finite difference methods.

Fourier spectral methods were also developed in the 1960s with the development of the Fast Fourier Transform (FFT). The FFT allows efficient employment of Fourier spectral solutions. They are important in fluid modeling and analysis where boundary effects are not important and can be assumed periodic.

Finally, for each method we shall explore the three mathematical requirements for a numerical method: consistency, stability and convergence. Consistency says that the discrete numerical method is a correct approximation to the continuous problem. Stability says that the numerical solution does not blow up. Finally, convergence says you can always reduce error in the numerical and exact solutions by refining the mesh. In other words, convergence says you get the correct solution. The Lax Equivalence Theorem says that a consistent and stable method is always convergent.

References

1. Kendall Atkinson and Weimin Han. *Theoretical numerical analysis*, volume 39 of *Texts in Applied Mathematics*. Springer, New York, second edition, 2005. ISBN 978-0387-25887-4; 0-387-25887-6. A nice (but mathematically intensive) look at functional analysis as it pertains to numerical analysis. This book provides a rigorous development of finite element analysis.
2. David Kincaid and Ward Cheney. *Numerical analysis: Mathematics of scientific computing*. Brooks/Cole Publishing Co., Pacific Grove, CA, third edition, 2001. ISBN 0-534-38905-8.
3. J. D. Lambert. *Numerical methods for ordinary differential systems*. John Wiley & Sons Ltd., Chichester, 1991. ISBN 0-471-92990-5. The initial value problem.
4. Cleve B. Moler. *Numerical computing with MATLAB*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2004. ISBN 0-89871-560-1. An introductory book on MATLAB and computational mathematics. You can download it for free at <http://www.mathworks.com/moler/chapters.html>.
5. K. W. Morton and D. F. Mayers. *Numerical solution of partial differential equations*. Cambridge University Press, Cambridge, second edition, 2005. ISBN 978-0-521-60793-0; 0-521-60793-0. An introduction.
6. Alfio Quarteroni and Fausto Saleri. *Scientific computing with MATLAB and Octave*, volume 2 of *Texts in Computational Science and Engineering*. Springer-Verlag, Berlin, second edition, 2006. ISBN 978-3-540-32612-0; 3-540-32612-X. An easy-to-read undergraduate version of Quarteroni's and Saleri's other book. But it has no information on PDEs.
7. Alfio Quarteroni, Riccardo Sacco, and Fausto Saleri. *Numerical mathematics*, volume 37 of *Texts in Applied Mathematics*. Springer-Verlag, Berlin, second edition, 2007. ISBN 978-3-540-34658-6; 3-540-34658-9. This book covers an entire three-quarter graduate course in modern numerical analysis. The book is up-to-date and includes significant MATLAB code. The index is unfortunately rather sparse.
8. Pavel Šolín. *Partial differential equations and the finite element method*. Pure and Applied Mathematics (New York). Wiley-Interscience [John Wiley & Sons], Hoboken, NJ, 2006. ISBN 978-0-471-72070-6; 0-471-72070-4. A good textbook for finite element method—theoretical but not obtuse.

9. Gilbert Strang. 18.085 Mathematical Methods for Engineers I, fall 2005. MIT OpenCourseWare, a. An excellent series of lectures and notes available at <http://ocw.mit.edu/OcwWeb/Mathematics/18-085Fall-2005/CourseHome/index.htm>.
10. Gilbert Strang. 18.086 Mathematical Methods for Engineers II, spring 2006. MIT OpenCourseWare, b. An excellent series of lectures and notes available at <http://ocw.mit.edu/OcwWeb/Mathematics/18-086Spring-2006/CourseHome/index.htm>.
11. Lloyd N. Trefethen. *Spectral methods in MATLAB*, volume 10 of *Software, Environments, and Tools*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2000. ISBN 0-89871-465-6. Fourier and Chebyshev spectral methods.
12. Lloyd N. Trefethen. Finite difference and spectral methods for ordinary and partial differential equations. Unpublished text, 1996. An set of lectures notes available at <http://web.comlab.ox.ac.uk/oucl/work/nick.trefethen/pdetext.html>.

Ordinary Differential Equations

We begin a course in numerical methods for partial differential equations (PDEs) by examining numerical methods to solve ordinary differential equations (ODEs). The reason is that modern numerical techniques to solve time-dependent partial differential equations often employ the **numerical method of lines** in which the space is discretized and the semi-discrete problem is solved using standard ODE solvers. For example, consider the heat equation $u_t = \Delta u$ where $\Delta = \sum_i \partial^2 / \partial x_i^2$ is the Laplacian operator. The heat equation can be formally solved as a first-order linear ordinary differential equation to get the solution

$$u(x, t) = e^{t\Delta} u(x, 0)$$

where $e^{t\Delta}$ is a linear operator acting on the initial conditions $u(x, 0)$. A typical numerical solution involves discretizing the Laplacian operator Δ in space and employing an ODE solver in time. Understanding stability, consistency and convergence of the numerical schemes for ordinary differential equations will help us to understand stability, consistency and convergence of the numerical schemes for partial differential equations.

1.1 Brief theory of ODEs

Consider the initial value problem

$$\frac{dy}{dt} = f(t, y) \text{ with } y(0) = y_0. \quad (1.1)$$

An initial value problem is said to be **well-posed** if (1) the solution exists, (2) the solution is unique, and (3) the solution depends continuously on the initial conditions. Let's examine these conditions for well-posedness.

Well-posed = Existence + Uniqueness + Stability

A function $f(x)$ is **Lipschitz continuous** if there exists some constant $L < \infty$ such that $|f(x) - f(x^*)| \leq L|x - x^*|$ for all $x, x^* \in \mathbb{R}$. For example,

$$\begin{array}{ll} f(x) = |x| & \text{is Lipschitz on } \mathbb{R} \\ f(x) = x^2 & \text{is not Lipschitz on } \mathbb{R} \text{ (it is locally Lipschitz)} \\ f(x) = \sqrt{|x|} & \text{is not Lipschitz on } \mathbb{R} \end{array}$$

Theorem 1. *If $f(t, y)$ is continuous in the region $0 \leq t \leq T$ and it is Lipschitz continuous in y , then there exists a unique continuously differentiable function $y(t)$ such that $y'(t) = f(t, y(t))$ and $y(0) = y_0$.*

Example. Let's see what happens when $f(t, y)$ is not Lipschitz. If $f(t, y) = y^2$ with $y(0) = 1$ over the interval $t \in [0, 2]$. Then $y' = y^2$ has the solution

$$\frac{dy}{y^2} = dt \quad \Rightarrow \quad -\frac{1}{y} = t + C \quad \Rightarrow \quad y(t) = \frac{1}{1-t}.$$

The solution blows up at $t = 1$. The solution does not exist on the interval $[1, 2]$.

Now, consider $f(t, y) = \sqrt{y}$ with $y(0) = 0$ over the interval $t \in [0, 2]$. The ODE $y' = \sqrt{y}$ has the solution $y(t) = \frac{1}{4}t^2$. Another solution is $y(t) = 0$. In fact, there are infinitely many solutions such as

$$y(t) = \begin{cases} 0 & t < t_c \\ \frac{1}{4}(t - t_c)^2 & t \geq t_c \end{cases}$$

where $t_c \in [0, 2]$ is an arbitrary constant. So, when $f(t, y)$ is not Lipschitz continuous in y , the solution may fail to exist; or if it exists, it may fail to be unique. \circ

We say that the initial value problem (1.1) is **stable**, if a small perturbation in the initial data or the coefficients in $f(t, y)$ will not become unbounded. Let's clarify what we mean by this definition. The initial value problem (1.1) is stable with respect to the initial conditions if there exists $K > 0$ such that for ε sufficiently small, the perturbed problem

$$z'(t) = f(t, z) + \delta(t) \text{ with } z(0) = y_0 + \varepsilon_0$$

satisfies $|z(t) - y(t)| \leq K\varepsilon$ whenever $|\varepsilon_0|, |\delta(t)| < \varepsilon$ for all $0 \leq t \leq T$.

Under what conditions is an initial value problem stable? Consider the original problem

$$y'(t) = f(t, y) \quad \text{with} \quad y(0) = y_0$$

and the perturbed problem

$$z'(t) = f(t, z) + \delta(t) \quad \text{with} \quad z(0) = y_0 + \varepsilon_0$$

where $|\delta(t)| \ll 1$ and $|\varepsilon| \ll 1$ and $t \in [0, T]$. The error is $e(t) = z(t) - y(t)$. So,

$$\begin{aligned} e'(t) &= z'(t) - y'(t) = f(t, z) - f(t, y) - \delta(t) \\ e(0) &= z(0) - y(0) = \varepsilon_0 \end{aligned}$$

If $f(t, y)$ is Lipschitz continuous in y , then

$$\begin{aligned}
 |e'(t)| &\leq |f(t, z) - f(t, y) - \delta(t)| \\
 &\leq |f(t, z) - f(t, y)| + |\delta(t)| \\
 &\leq L|z(t) - y(t)| + |\delta(t)| \\
 &\leq L|e(t)| + |\delta(t)|
 \end{aligned}$$

Let $\varepsilon = \max(\varepsilon_0, \max_{0 \leq t \leq T} |\delta(t)|)$. Then

$$|e'(t)| \leq L|e(t)| + \varepsilon \quad \text{and} \quad |e(0)| \leq \varepsilon.$$

It follows that

$$|e(t)| \leq \frac{\varepsilon}{L}[(L+1)e^{Lt} - 1] \leq \frac{\varepsilon}{L}[(L+1)e^{LT} - 1] = \varepsilon K \quad \text{for } 0 \leq t \leq T$$

which says that the error is bounded for $0 \leq t \leq T$. So, if $f(t, y)$ is Lipschitz continuous in y , then the initial value problem $y' = f(t, y)$ is stable.

If $f(t, y)$ is continuous in t and Lipschitz in y then (1.1) is well-posed.

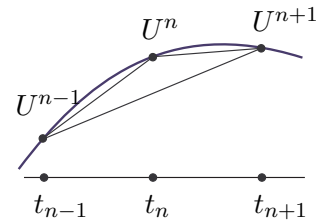
1.2 Numerical methods and stability

Let $u \equiv u(t)$. Suppose we are given with the initial value problem





$$u' = f(u) \quad \text{where } u(0) = u_0. \quad (1.2)$$

For simplicity we will discretize t using a uniform time step, i.e., we take $t_n = nk$ where k is the time step. We can approximate $\frac{d}{dt}$ by a numerical difference operator:

$$\begin{aligned}
 \text{Forward Difference} \quad \delta_+ U^n &= \frac{U^{n+1} - U^n}{k} \\
 \text{Backward Difference} \quad \delta_- U^n &= \frac{U^n - U^{n-1}}{k} \\
 \text{Centered Difference} \quad \delta_0 U^n &= \frac{U^{n+1} - U^{n-1}}{2k}
 \end{aligned}$$



where U^n is the numerical approximation to $u(t_n)$. Using these difference operators, we can generate several numerical methods to solve the initial value problem (1.2):

	Forward Euler	$\frac{U^{n+1} - U^n}{k} = f(U^n)$	$O(k)$	Explicit
	Backward Euler	$\frac{U^{n+1} - U^n}{k} = f(U^{n+1})$	$O(k)$	Implicit
	Leap Frog	$\frac{U^{n+1} - U^{n-1}}{2k} = f(U^n)$	$O(k^2)$	Explicit
	Trapezoidal	$\frac{U^{n+1} - U^n}{k} = \frac{f(U^{n+1}) + f(U^n)}{2}$	$O(k^2)$	Implicit

In each of the above schemes we can express U^{n+1} in terms of U^n or U^{n-1} either explicitly or implicitly. A numerical method of U^{n+1} is **implicit** if the right-hand side contains a term $f(U^{n+1})$. Otherwise, the numerical method is **explicit**. An explicit method, while easy to implement, is typically less stable than an implicit method.

A numerical method is **stable** if there exists a $K > 0$ such that a change in the starting values by a fixed amount produces a bounded change in the numerical solution for all $0 \leq k \leq K$ and $0 \leq nk \leq T$. Furthermore, a numerical method is **absolutely stable** if for a given step size k and a given differential equation, the change due to a perturbation of size δ in one of the time steps t_n is not larger than δ in any of the subsequent time steps. In other words, a method is stable if perturbations are bounded in finite time and absolutely stable if perturbations shrink over time.

Clearly, absolute stability is a desirable condition for a numerical method to possess. Let's determine under what conditions a numerical method is absolutely stable. Let $U^0 = u_0$ be the initial condition and V^0 be a perturbation of the initial condition. The error in the initial condition is defined as $e^0 = V^0 - U^0$ and the error at time $t_n = nk$ is $e^n = V^n - U^n$. Since

$$|e^n| = \left| \frac{e^n}{e^{n-1}} \right| \cdot \left| \frac{e^{n-1}}{e^{n-2}} \right| \cdots \left| \frac{e^1}{e^0} \right| \cdot |e^0|,$$

a sufficient condition for the error to be nonincreasing is for $|e^{n+1}/e^n| \leq 1$ for all n . Consider the linear ODE

$$u' = f(u) + g(t)$$

and take the perturbation $v = u + e$ for an error e . Then

$$v' = f(v) + g(t) \Rightarrow u' + e' = f(u + e) + g(t) = f(u) + f(e) + g(t) \Rightarrow e' = f(e)$$

and so the error has the same growth rate as the solution to the homogeneous problem. Therefore, to determine the region of absolute stability we will determine under what conditions $r = |U^{n+1}/U^n| \leq 1$.

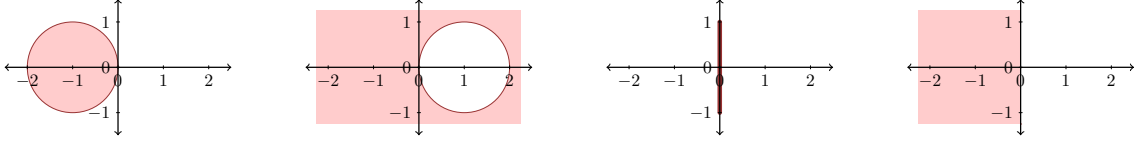


Figure 1.1: Regions of absolute stability (shaded) in the λk -plane for the forward Euler, backward Euler, leapfrog, and trapezoidal methods (left to right).

Regions of Absolute Stability

Consider the linear differential equation $u' = \lambda u$ for an arbitrary complex value λ . For what time step k is a numerical scheme absolutely stable? To answer this question, we'll determine for what values $z = \lambda k$ the growth factor $|r| = |U^{n+1}/U^n| \leq 1$.

Example. Determine the region of absolute stability for the forward Euler scheme. The forward Euler method for $u' = \lambda u$ is

$$\frac{U^{n+1} - U^n}{k} = \lambda U^n$$

from which we have that

$$U^{n+1} = (1 + \lambda k)U^n$$

and therefore

$$|U^{n+1}| \leq |1 + \lambda k| |U^n|.$$

Absolute stability requires $|1 + \lambda k| \leq 1$ —inside a unit circle centers at $\lambda k = -1$. See Figure 1.1. This means that we must limit the size of our step size k to ensure a stable method. \circ

Example. Determine the region of absolute stability for the backward Euler scheme. For the backward Euler method

$$\frac{U^{n+1} - U^n}{k} = \lambda U^{n+1}$$

from which we have that

$$(1 - \lambda k)U^{n+1} = U^n$$

and therefore

$$|U^{n+1}| \leq \frac{1}{|1 - \lambda k|} |U^n|$$

Absolute stability requires $|1 - \lambda k| \geq 1$. See Figure 1.1. This means that we can take any size stepsize as long as we are outside of the unit circle centered at $\lambda k = 1$. A numerical method is said to be **A-stable**, if the region of absolute stability includes the entire left half λk -plane. The backward Euler method is A-stable. \circ

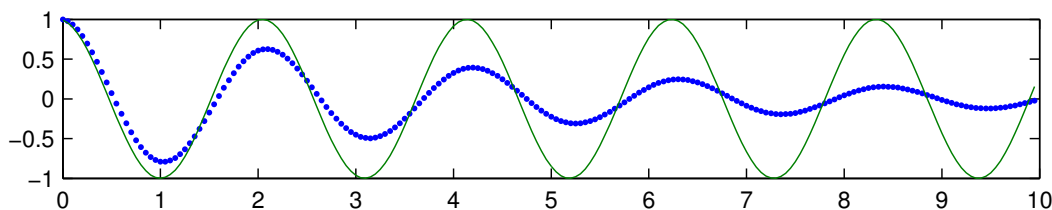


Figure 1.2: Solution to $u'' = -9u$ using the backward Euler method with $k = \frac{1}{20}$.

Note that absolute stability says that the error in a perturbed numerical solution decreases over time (numerical solution minus numerical solution). It does not say anything about accuracy of a numerical method (numerical solution minus analytic solution). For example, the numerical method $U^{n+1} = 0$ is stable but the scheme is always inconsistent. Let's examine at a nontrivial example. Consider the IVP $u'' = -9u$ with $u(0) = 1$ and $u'(0) = 0$. The eigenvalues of this equation are $\lambda = \pm 3i$. Using a backward Euler method,

$$U^{n+1} = \begin{bmatrix} 1 & k \\ -9k & 1 \end{bmatrix}^{-1} U^n.$$

with eigenvalues $\pm \frac{1}{3}k$. While the analytic solution is $u(t) = \cos 3t$, the numerical solution dissipates within a few oscillations. See Figure 1.2.

Example. Determine the region of absolute stability for the leapfrog scheme. For the leapfrog scheme

$$\frac{U^{n+1} - U^{n-1}}{k} = 2\lambda U^n.$$

By letting $r = U^{n+1}/U^n$, we have that

$$r^2 - 2\lambda k r - 1 = 0.$$

We want to find when $|r| \leq 1$ in the complex plane, so let's let $r = e^{i\theta}$. From this we have that

$$e^{2i\theta} - 2\lambda k e^{i\theta} - 1 = 0$$

and so

$$e^{i\theta} - e^{-i\theta} = 2\lambda k.$$

Therefore

$$2i \sin \theta = 2\lambda k.$$

The leapfrog scheme is absolute stability only on the imaginary axis $\lambda k \in [-1, 1]$. See Figure 1.1 on the previous page. ◦

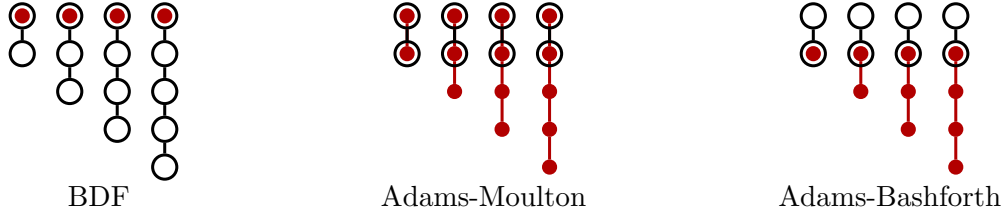


Figure 1.3: Stencils for multistep methods—BDF, Adams–Moulton and Adams–Bashforth. Note that the first order BDF is the same as the backward Euler, the second-order Adams–Moulton is the same as the trapezoidal and the first-order Adams–Bashforth is the same as the forward-Euler methods.

Example. Determine the region of absolute stability for the trapezoidal scheme. For the trapezoidal scheme, letting $r = U^{n+1}/U^n$.

$$\frac{U^{n+1} - U^{n-1}}{k} = \lambda(U^{n+1} + U^{n-1})$$

from which we have that

$$r^2 - 1 = \lambda k(r^2 + 1)$$

or equivalently

$$\lambda k = \frac{r^2 - 1}{r^2 + 1}.$$

As before, take $r = e^{i\theta}$, then

$$\lambda k = \frac{e^{2i\theta} - 1}{e^{2i\theta} + 1} = i \tan \theta$$

So $|r| = 1$ along the entire imaginary axis and the region of absolute stability includes the entire left half λk -plane. Therefore, the trapezoidal scheme is A-stable. See Figure 1.1 on page 5. ◦

A method is consistent if and only if the region of stability contains $\lambda k = 0$. We call such a condition **zero-stability**.

1.3 Multistep methods

Consider the backward Euler solution to $u'(t) = f(u)$:

$$\frac{U^n - U^{n-1}}{k} = f(U^n).$$

The Taylor series expansion of $u(t - k)$ about $t = nk$ is

$$u(t - k) = u(t) - ku'(t) + \frac{1}{2}k^2u''(t) + O(k^3)$$

from which we have that

$$\frac{U^n - U^{n-1}}{k} = \frac{u(t) - u(t-k)}{k} = u'(t) + O(k)$$

Therefore, the backward Euler method gives a first-order method.

The forward and backward Euler methods both approximate the solution using piecewise linear functions. We can get a more accurate approximation to the derivative by using higher order piecewise polynomials—piecewise quadratic, piecewise cubic, and so forth. For an n th order polynomial approximation, we require n terms of the Taylor series and hence n steps in time.

Backwards difference formulas (BDF)

We can get a better approximation by using more terms to cancel out more terms of the Taylor series. Let's derive a second order implicit method by using a piecewise quadratic approximation for u . Consider $u(t_n)$, $u(t_{n-1})$, and $u(t_{n-2})$:

$$\begin{aligned} u(t) &= u(t) \\ u(t-k) &= u(t) - ku'(t) + \frac{1}{2}k^2u''(t) + O(k^3) \\ u(t-2k) &= u(t) - 2ku'(t) + \frac{4}{2}k^2u''(t) + O(k^3) \end{aligned}$$

We want to cancel as many terms as possible to be left $O(k^3)$ by combining these in some fashion. Let a , b and c be unknowns, then

$$\begin{aligned} au(t) &= au(t) \\ bu(t-k) &= bu(t) - bku'(t) + \frac{1}{2}bk^2u''(t) + O(k^3) \\ cu(t-2k) &= cu(t) - 2cku'(t) + \frac{4}{2}ck^2u''(t) + O(k^3) \end{aligned}$$

We have three variables, so we are allowed three constraints. For *consistency*, we require the coefficient of $u(t)$ to be zero and the coefficient of $u'(t)$ to be one; for *accuracy*, we require the coefficient $u''(t)$ to be zero. Therefore

$$\begin{aligned} a + b + c &= 0 \\ 0 - b - 2c &= 1/k \\ 0 + \frac{1}{2}b + \frac{4}{2}c &= 0 \end{aligned}$$

We solve this linear system for (a, b, c) and we get

$$a = \frac{3}{2} \frac{1}{k}, \quad b = -2 \frac{1}{k}, \quad c = \frac{1}{2} \frac{1}{k}.$$

The method obtained is the **backward difference formula** of second order (BDF2)

$$\frac{3}{2}U^n - 2U^{n-1} + \frac{1}{2}U^{n-2} = kf(U^n).$$

Note that

$$\frac{1}{k} \left(\frac{3}{2}u(t_n) - 2u(t_{n-1}) + \frac{1}{2}u(t_{n-2}) \right) = u'(t_n) + O(k^2)$$

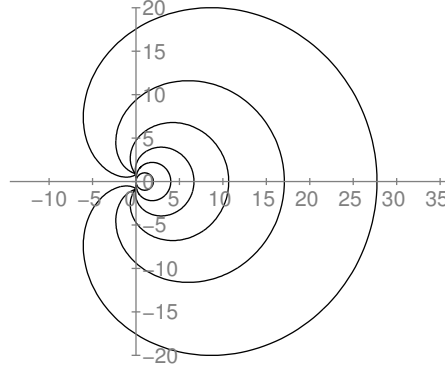


Figure 1.4: Inner contour of the regions of absolute stability for the BDF methods (BDF1–BDF5). Regions of absolute stability for BDF methods are *unbounded*. The regions of stability get progressively smaller with increased accuracy.

An r -step (and r th order) backward differentiation formula takes the form

$$a_0 U^n + a_1 U^{n-1} + \cdots + a_{n-r} U^{n-r} = k f(U^n)$$

where the coefficients a_0, \dots, a_{n-r} are chosen appropriately. The backward Euler is also the BDF1.

Example. Determine the region of absolute stability for BDF2: We take $f(u) = \lambda u$. Then

$$\frac{3}{2} U^n - 2 U^{n-1} + \frac{1}{2} U^{n-2} = k f(U^n)$$

Letting $r = U^n / U^{n-1}$, we have

$$\frac{3}{2} r^2 - 2r + \frac{1}{2} = k \lambda r^2$$

For which λk is $|r| \leq 1$? While we can easily compute this answer analytically using a quadratic formula, it becomes difficult or impossible for higher order methods. So, instead, we will simply plot the boundary of the region of absolute stability. Since we want to know when $|r| \leq 1$, i.e., when r is inside the unit disk in the complex plane, we will consider $r = e^{i\theta}$. Then the variable λk , can be expressed as the rational function

$$\lambda k = \frac{\frac{3}{2} r^2 - 2r + \frac{1}{2}}{r^2}$$

We can plot this function using MATLAB:

```
r = exp(2i*pi*(0:.01:1));
plot((1.5*r.^2 - 2*r + 0.5)./r.^2);
```

o

The regions of absolute stability for backward difference formulas are *unbounded* and always contain the negative real axis. This stability feature makes backward difference

formulas particularly nice for stiff problems. See Section 1.6. The regions of absolute stability for the BDF methods get progressively smaller with increased accuracy. The trade off between stability and accuracy is usual, but not necessary (higher order Runge-Kutta methods are typically more stable than their lower order counterparts).

General multistep methods

We are really interested in getting a more accurate approximation to the whole equation $u' = f(u)$ and not simply the left hand side u' . A general r -step multistep method may be written as

$$\sum_{j=0}^r a_j U^{n-j} = k \sum_{j=0}^r b_j f(U^{n-j})$$

where the coefficients a_j and b_j are often zero. When $b_0 = 0$, the method is explicit. When $b_0 \neq 0$, the method is implicit.

For the initial value equation $u' = f(u)$ clearly $u' - f(u) = 0$. The **local truncation error** of the finite difference approximation is defined as

$$\tau_{\text{local}} = \sum_{j=0}^r a_j u(t_{n-j}) - k \sum_{j=0}^r b_j f(u(t_{n-j}))$$

and it used to quantify the error for one time step. The **global truncation error** of the finite difference approximation is defined as

$$\tau_{\text{global}} = \frac{1}{k} \sum_{j=0}^r a_j u(t_{n-j}) - \sum_{j=0}^r b_j f(u(t_{n-j})).$$

and it is used to quantify the accumulation of error over several times steps. Note that

$$\tau_{\text{global}} = \frac{1}{k} \tau_{\text{local}}.$$

For example, the local truncation error of the leap frog scheme is $O(k^3)$ but the global truncation error is $O(k^2)$.

From Taylor series expansion, we have that

$$\begin{aligned} u(t_{n-j}) &= u(t_n) - jku'(t_n) + \frac{1}{2}(jk)^2 u''(t_n) + \cdots \\ f(u(t_{n-j})) &= \frac{d}{dt} u(t_{n-j}) = u'(t_n) - jku''(t_n) + \frac{1}{2}(jk)^2 u'''(t_n) + \cdots \end{aligned}$$

Then

$$\begin{aligned} \tau(t_n) &= \frac{1}{k} \sum_{j=0}^r a_j u(t_n) + \sum_{j=0}^r (ja_j - b_j) u'(t_n) + k \sum_{j=0}^r (\frac{1}{2}j^2 a_j - b_j j) u''(t_n) + \cdots \\ &\quad \cdots + \frac{1}{p!} k^{p-1} \sum_{j=0}^r (j^p a_j - pj^{p-1} b_j) u^{(p)}(t_n) + \cdots \end{aligned}$$

From this equation, for consistency ($\lim_{k \rightarrow 0} \tau(t_n) = 0$), we need

$$\sum_{j=0}^r a_j = 0 \quad \text{and} \quad \sum_{j=0}^r j a_j - b_j = 0 \quad (1.3a)$$

and to get an $O(k^p)$ method, we need to set the first $p + 1$ coefficients to 0:

$$\sum_{j=0}^r (j^p a_j - p j^{p-1} b_j) = 0. \quad (1.3b)$$

Suppose that we want to maximize the order of an r -step method. It may seem reasonable that we could build a $2r - 1$ -order method using r -steps using (1.3) because we have $2r$ unknowns—namely a_0, a_1, \dots, a_r and b_0, b_1, \dots, b_r . But such an approach does not assure stability of the scheme. In general, the order of a method is limited by the first Dahlquist barrier.

Theorem 2 (First Dahlquist Stability Barrier). *The order of accuracy p of a stable r -step linear multistep formula satisfies*

$$p \leq \begin{cases} r + 2 & \text{if } r \text{ is even,} \\ r + 1 & \text{if } r \text{ is odd,} \\ s & \text{if the formula is explicit.} \end{cases}$$

Multistep methods may require several starting values, i.e., to compute U^r for an r -step method, we first need U^0, \dots, U^{r-1} . For example, to implement the leapfrog method, first use forward Euler method to compute U^1 . Then use the leapfrog after that. The local truncation error from the Euler method is $O(k^2)$ and the global truncation error from the leapfrog method is $O(k^2)$. Therefore, the overall global error from the leapfrog method is $O(k^2)$.

Adams methods

The Adams methods have the form $a_0 = 1$ and $a_1 = -1$ and $a_j = 0$ for $2 \leq j \leq r$. That is,

$$U^n = U^{n-1} + k \sum_{j=0}^r b_j f(U^{n-j}).$$

The explicit **Adams-Bashforth method** takes $b_0 = 0$ and the implicit **Adams-Moulton method** takes $b_0 \neq 0$. Note that the forward Euler is equivalent to the first-order Adams-Bashforth method (AB1) and that the trapezoidal method is equivalent to the second order Adams-Moulton method (AM1).

1.4 Multistage: Runge-Kutta method

By directly integrating $u'(t) = f(u, t)$ we can change the differential equation into an integral equation

$$u(t_{n+1}) - u(t_n) = \int_{t_n}^{t_{n+1}} f(t, u(t)) dt.$$

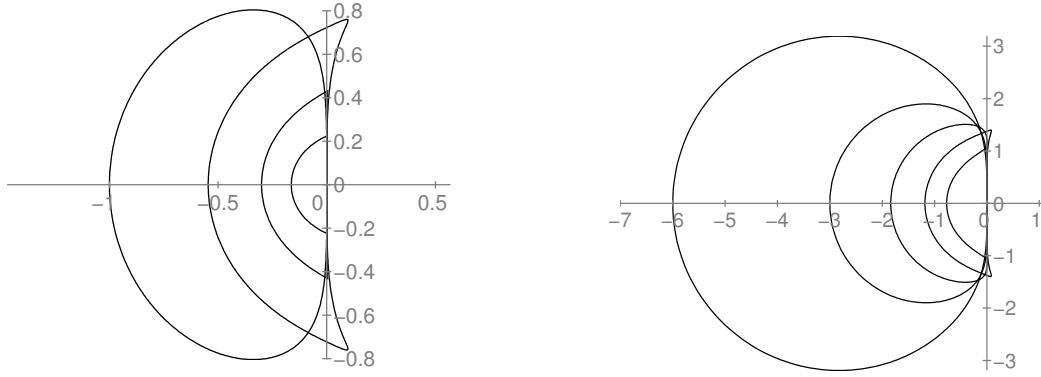


Figure 1.5: External contour of the regions of absolute stability for the Adams-Bashforth method (AB2–AB5) and Adams-Moulton method (AM2–AM6), left and right respectively. Note that the plots are at different scales. The regions of stability get progressively smaller with increased accuracy.

The challenge now is to numerically evaluate the integral. One way of doing this is to use the Runge-Kutta method. Unlike multistep methods, Runge-Kutta methods can get high order results without saving the solutions of previous time steps, making them good choices for adaptive step size. Because of this, Runge-Kutta methods are often a method of choice.

We can approximate the integral using quadrature

$$\int_{t_n}^{t_{n+1}} f(u(t), t) dt \approx k \sum_{i=1}^s b_i f(t_i^*, u(t_i^*))$$

with appropriate weights b_j and nodes $t_i^* \in [t_n, t_{n+1}]$. In fact, by choosing the s nodes and weights so that we have a Gaussian quadrature, we can get a method that has order $2s$. But to apply this method we must already know the values $u(t_i^*)$. So, the method is circular. Of course, we can always approximate the values by using quadrature.

Example. The simplest quadrature rule using a Riemann sum:

$$\int_a^b f(x) dx \approx (b - a)f(a)$$

Let's use this to compute U^{n+1} given U^n .

$$u(t_{n+1}) = u(t_n) + \int_{t_n}^{t_{n+1}} f(t, u(t)) dt \Rightarrow U^{n+1} = U^n + k f(t_n, U^n)$$

which is of course the forward Euler method. ◻

Example. The midpoint rule is another (more accurate) one-point quadrature rule

$$\int_a^b f(x) dx \approx (b - a)f\left(\frac{b + a}{2}\right).$$

This time we'll need to approximate $f(t_{n+1/2}, U^{n+1/2})$ —we can use the forward Euler method. Take

$$K_1 = f(t_n, U^n)$$

and

$$K_2 = f(t_n + \frac{1}{2}k, U^n + \frac{1}{2}kf(t_n, U^n)) = f(t_n + \frac{1}{2}k, U^n + \frac{1}{2}kK_1)$$

Then the midpoint rule says

$$u(t_{n+1}) = u(t_n) + \int_{t_n}^{t_{n+1}} f(t, u(t)) dt \Rightarrow U^{n+1} = U^n + kK_2$$

This gives us the second-order Heun method. ◦

Example. The trapezoidal rule is another (second-order) one-point quadrature rule

$$\int_a^b f(x) dx \approx (b-a) \left[\frac{1}{2}f(a) + \frac{1}{2}f(b) \right].$$

To compute U^{n+1} given U^n , we'll need to approximate $f(t_{n+1}, U^{n+1})$ using the forward Euler method. Take

$$K_1 = f(t_n, U^n)$$

and

$$K_2 = f(t_n + k, U^n + kf(t_n, U^n)) = f(t_n + k, U^n + kK_1)$$

Then the trapezoidal quadrature rule says

$$u(t_{n+1}) = u(t_n) + \int_{t_n}^{t_{n+1}} f(u(t), t) dt \Rightarrow U^{n+1} = U^n + k \left[\frac{1}{2}K_1 + \frac{1}{2}K_2 \right]$$

◦

Example. To get high order methods, we add quadrature points. The next step up is Simpson's method which says

$$\int_a^b f(x) dx \approx (b-a) \left[\frac{1}{6}f(a) + \frac{2}{3}f\left(\frac{b+a}{2}\right) + \frac{1}{6}f(b) \right].$$

This time we'll need to approximate both $f(t_{n+1/2}, U^{n+1/2})$ and $f(t_{n+1}, U^{n+1})$. ◦

The general s -stage Runge-Kutta method is given by

$$U^{n+1} = U^n + k \sum_{i=1}^s b_i K_i \text{ with } K_i = f\left(t_n + c_i k, U^n + k \sum_{j=1}^s a_{ij} K_j\right) \quad (1.4)$$

For consistency we need to take $c_i = \sum_{j=1}^s a_{ij}$ and $\sum_{i=1}^s b_i = 1$. The coefficients are often conveniently given as a **Butcher tableau**

$$\begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b}^T \end{array}$$

where $[\mathbf{A}]_{ij} = a_{ij}$. If \mathbf{A} is a strictly lower triangular matrix, then the Runge-Kutta method is explicit. Otherwise, the method is implicit. If \mathbf{A} is a lower triangular matrix, then the method is a diagonally implicit Runge-Kutta Method (DIRK). In general, computing the coefficients \mathbf{A} is not a trivial exercise.

The second order Heun method (RK2) is given by

$$\begin{array}{l} K_1 = f(t_n, U^n) \\ K_2 = f(t_n + \frac{1}{2}k, U^n + \frac{1}{2}kK_1) \\ U^{n+1} = U^n + kK_2 \end{array} \quad \begin{array}{c|c} 0 & \\ \frac{1}{2} & \frac{1}{2} \\ \hline & 0 \quad 1 \end{array}$$

The fourth-order Runge-Kutta (RK4) is given by

$$\begin{array}{l} K_1 = f(t_n, U^n) \\ K_2 = f(t_n + \frac{1}{2}k, U^n + \frac{1}{2}kK_1) \\ K_3 = f(t_n + \frac{1}{2}k, U^n + \frac{1}{2}kK_2) \\ K_4 = f(t_n + k, U^n + kK_3) \\ U^{n+1} = U^n + \frac{1}{6}k(K_1 + 2K_2 + 2K_3 + K_4). \end{array} \quad \begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

Example. Plot the regions of stability for the Runge-Kutta methods given by the following tableaus:

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

The region of absolute stability is bounded by the $|r| = 1$ contour where $r = U^{n+1}/U^n$ where U^{n+1} is the solution to $u' = \lambda u$ using a general Runge-Kutta method. For a Runge-Kutta method we can get an explicit expression for λk in terms of r . For $f(t, u) = \lambda u$, a general Runge-Kutta method is by (1.4)

$$U^{n+1} = U^n + k\mathbf{b}^T \mathbf{K}$$

with \mathbf{K} given implicitly by

$$\mathbf{K} = \lambda(U^n \mathbf{E} + k\mathbf{A}\mathbf{K}) \quad (1.5)$$

where \mathbf{K} is an $s \times 1$ vector, \mathbf{b} is a $1 \times s$ vector, \mathbf{A} is an $s \times s$ matrix, and \mathbf{E} is an $s \times 1$ vector of ones. We solve (1.5) for \mathbf{K} to get

$$\mathbf{K} = \lambda U^n (\mathbf{I} - \lambda k \mathbf{A})^{-1} \mathbf{E}$$

where \mathbf{I} is an $s \times s$ identity matrix. Then

$$r = 1 + \lambda k \mathbf{b}^T (\mathbf{I} - \lambda k \mathbf{A})^{-1} \mathbf{E}. \quad (1.6)$$

We can use MATLAB to compute the $|r| = 1$ contours of (1.6) for each of the tableaus. The regions of absolute stability are plotted in Figure 1.6 on the next page. \circ

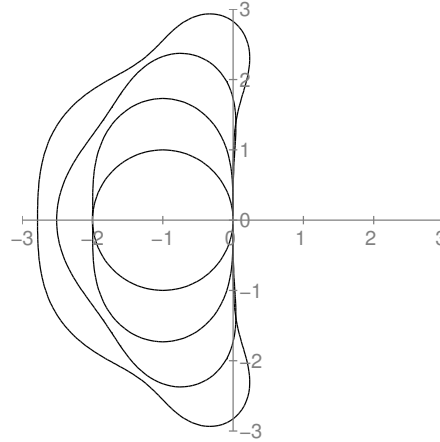


Figure 1.6: External contour of the regions of absolute stability for the explicit Runge-Kutta methods (RK1–RK4), left and right respectively. The regions of stability get progressively larger with increased accuracy.

Adaptive step size

One advantage Runge-Kutta methods have over multistep methods is that it is easy to vary the step size k , taking smaller steps we needed to minimize error and maintain stability and larger steps otherwise. The Dormand–Prince method combines a fourth-order and a fifth-order Runge–Kutta method using the same quadrature points.

0							
$\frac{1}{5}$	$\frac{1}{5}$						
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$					
$\frac{4}{5}$	$\frac{44}{45}$	$-\frac{56}{15}$	$\frac{32}{9}$				
$\frac{8}{9}$	$\frac{19372}{6561}$	$-\frac{25360}{2187}$	$\frac{64448}{6561}$	$-\frac{212}{729}$			
1	$\frac{9017}{3169}$	$-\frac{355}{33}$	$\frac{46732}{5247}$	$\frac{49}{176}$	$-\frac{5103}{18656}$		
1	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	
	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	0
	$\frac{5179}{57600}$	0	$\frac{7571}{16695}$	$\frac{393}{640}$	$-\frac{92097}{339200}$	$\frac{187}{2100}$	$\frac{1}{40}$

For a p th order Runge-Kutta method,

$$u(t_{n+1}) = u(t_n) + k \sum_{i=1}^n b_i K_i + O(k^p) \text{ with } K_i = f \left(t_n + c_i k, u(t_n) + k \sum_{j=1}^s a_{ij} K_j \right)$$

For the Runge-Kutta-Fehlberg, K_i are the same for the fourth-order and the fifth-order methods. So, by subtracting we have that

$$0 = 0 + k \sum_{i=1}^n (\hat{b}_i - b_i) K_i + O(k^4)$$

Hence the truncation error of the fourth order method is approximately $k \sum_{i=1}^n (\hat{b}_i - b_i) K_i$ giving us a means of measuring the error.

1.5 Nonlinear ODEs

So far, we've looked at linear ODEs $u' = \lambda u + c(t)$, which for the most part can be solved analytically. But what is the stability condition for a nonlinear ODE $u' = f(u)$? To determine this, we linearize the right hand side

$$f(u) \approx f(u_0) + (u - u_0)f'(u_0) = [f(u_0) - u_0f'(u_0)] + f'(u_0)u.$$

This says that we should consider $\lambda = \max_{u_0} f'(u_0)$ for our eigenvalue. For a system, we take the spectral radius (largest eigenvalue) of the Jacobian determinant.

Implicit methods are more difficult to implement but they are more stable. To solve a nonlinear equation $\Phi(x) = 0$, one often requires Newton's method

$$X_{j+1} = X_j - [\Phi'(X_j)]^{-1}\Phi(X_j) \quad (1.7)$$

where $\Phi'(X_j)$ is the Jacobian matrix of Φ . or a fixed-point iteration

$$X_{j+1} = X_j - \Phi(X_j) \quad (1.8)$$

Example. Solve the nonlinear ODE $u' = f(u)$ using the backward Euler method. The backward Euler method is

$$\frac{U^n - U^{n-1}}{k} = f(U^n).$$

So,

$$U^n = U^{n-1} + kf(U^n). \quad (1.9)$$

Since we can't solve for U^n explicitly, we use Newton's method. Let the $X_0 = U^{n-1}$. Then from (1.9)

$$\Phi(X_j) = X_j - X_0 - kf(X_j)$$

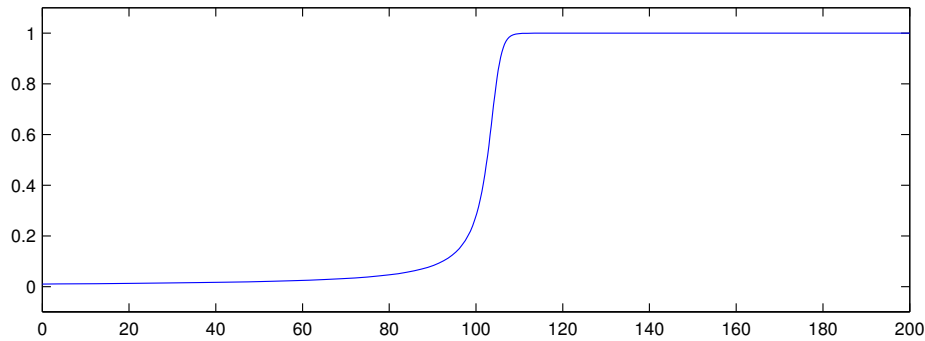
which we use in (1.7). ◦

Both Newton's method and fixed point iteration have global convergence issues. Therefore, it is often useful to use an explicit method to get a sufficiently close solution and then use an implicit method to get even closer for each time step. Such a method is called a predictor-corrector methods. Predictor corrector methods often employ a combination of Adams-Bashforth and Adams-Moulton because the Adams-Moulton has significantly larger stability regions. Unfortunately, predictor-corrector methods do not perform well on stiff nonlinear problems.

1.6 Stiff equations

The initial value problem $u' = f(u)$ is called **stiff** if the Jacobian matrix $f'(u)$ has a wide range of eigenvalues:

$$\frac{\max |\lambda|}{\min |\lambda|} \gg 1.$$

Figure 1.7: Solution to $y' = y^2 - y^3$ with $y(0) = 1/100$.

This means that there are at least two scales of different orders of magnitude.

Example. Consider the behavior of the solution of the stiff ODE

$$u'(t) = \lambda(\cos t - u) - \sin t \quad u(0) = u_0 \quad \text{with} \quad \lambda > 0 \quad \text{and} \quad |\lambda| \gg 1.$$

The solution is

$$u(t) = (u_0 - 1)e^{-\lambda t} + \cos t.$$

There are two scales—a slow time scale and a fast time scale—over which the solution evolves. Initially, the solution behaves like $u(t) = u_0 e^{-\lambda t}$ which quickly decays. After a long time, the solution behaves like $u(t) = \cos t$. \circ

Example. A simple model of flame propagation is

$$y' = y^2(1 - y) \quad \text{with} \quad y(0) = \delta$$

where $y(t)$ is the radius of a ball of flame (of a match for example). See Moler [2004]. The problem has a slowly evolving solution with $y(t) \approx 0$ until $t \approx 1/\delta$. At this time the solution quickly changes to $y(t) \approx 1$. After this rapid change, the solution again evolves very slowly to $y(\infty) = 1$. See Figure 1.6. \circ

Numerical methods for stiff problems

If we are solving a stiff problem, at least one eigenvalue λ is relatively large. Consider the problem $u' = \lambda u$. For an explicit method, we need $|\lambda k|$ smaller than a constant C for stability. So, $|k| \leq C/|\lambda|$ which says that k needs to be very small. Therefore, it is natural to use implicit methods, especially A-stable methods such as the backward Euler and trapezoidal (Crank-Nicolson) methods. A trapezoidal method is implicit and second-order. But the trapezoidal method has transient behavior.

Example. Determine the transient (long term) behavior of the trapezoidal method for the initial value problem $u' = -\lambda u$ with $u(0) = 1$ where $\lambda > 0$. The trapezoidal method is

$$\frac{U^n - U^{n-1}}{k} = -\lambda \frac{U^n + U^{n-1}}{2}$$

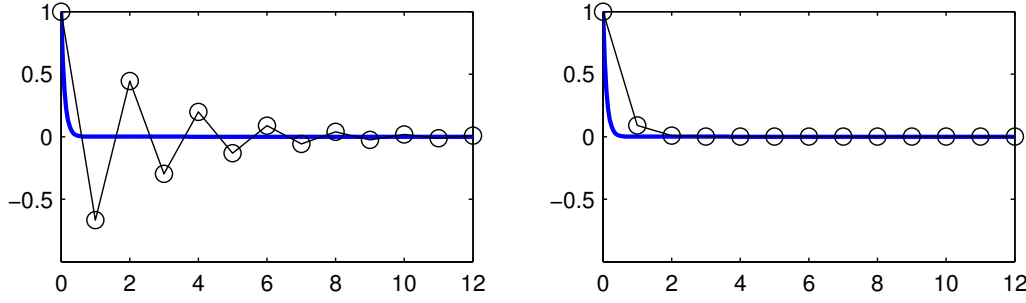


Figure 1.8: Numerical solutions to $u' = -10u$ using second-order trapezoidal method (left) and first-order L-stable backward Euler method (right) with stepsize $k = 1$. Since the trapezoidal method is not L-stable, it exhibits oscillations and slow decay.

Take $|\lambda k| \gg 1$. Then

$$U^n = \frac{1 - \frac{1}{2}\lambda k}{1 + \frac{1}{2}\lambda k} U^n = \left(\frac{1 - \frac{1}{2}\lambda k}{1 + \frac{1}{2}\lambda k} \right)^n U^0 \approx (-1)^n U^0.$$

So, the numerical solution will oscillate for a long time. See Figure 1.8. \circ

Example. Determine the transient (long term) behavior of the backward Euler method for the initial value problem $u' = -\lambda u$ with $u(0) = 1$ where $\lambda > 0$. The backward Euler method is

$$\frac{U^n - U^{n-1}}{k} = -\lambda U^n$$

Then

$$U^n = \frac{1}{1 + \lambda k} U^{n-1} = \left(\frac{1}{1 + \lambda k} \right)^n U^0 = \sigma(\lambda k)^n U^0 \rightarrow 0 \quad \text{as } |\lambda k| \rightarrow \infty$$

See Figure 1.8. \circ

For stiff problems, A-stability is not enough. A method $U^{n+1} = \sigma(\lambda k)U^n$ is **L-stable** if $\lim_{|\lambda k| \rightarrow \infty} |\sigma(\lambda k)| < 1$.

An L-stable scheme has more weights in the implicit terms than the explicit terms. All backward difference schemes are L-stable.

1.7 Splitting methods

Often PDEs have stiff linear term coupled with a nonlinear term:

Navier-Stokes equation	$\rho \frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} = -\nabla p + \mu \Delta \mathbf{v} + f$
reaction-diffusion equation	$\frac{\partial u}{\partial t} = \Delta u + u(1 - u^2)$
nonlinear Schrödinger equation	$i \frac{\partial \psi}{\partial t} = -\frac{1}{2} \Delta \psi + \psi ^2 \psi$

The stiff linear term is best treated implicitly, while the nonlinear term is best treated explicitly. Such an approach is called **splitting**.

Operator splitting

Let's examine the stiff nonlinear ODE

$$u' = \cos^2 u - 10u \quad \text{with} \quad u(0) = u_0. \quad (1.10)$$

We can write the problem as $u' = \mathcal{N}u + \mathcal{L}u$, where $\mathcal{N}u = \cos^2 u$ and $\mathcal{L}u = -10u$. In this problem, the solution u is being forced by both $\mathcal{N}u$ and $\mathcal{L}u$ *concurrently*. We can't solve this problem analytically, but we can approximate the solution for small times by considering $\mathcal{N}u$ and $\mathcal{L}u$ as acting *successively*. We can do this in two ways:

1. First solve $u' = \mathcal{N}u$. Then, use the solution as the initial conditions for $u' = \mathcal{L}u$.
2. First solve $u' = \mathcal{L}u$. Then, use the solution as the initial conditions for $u' = \mathcal{N}u$.

Let's evaluate each of these approximate solutions after a time k .

1. The solution to $u' = \cos^2 u$ with initial condition $u(0) = u_0$ is $u(t) = \tan^{-1}(t + u_0)$.
The solution to $u' = -10u$ with initial condition $\tan^{-1}(k + u_0)$ is

$$u(k) = e^{-10k} \tan^{-1}(k + \tan u_0).$$

2. The solution to $u' = -10u$ with initial condition u_0 is $u(t) = e^{-10t}u_0$. The solution to $u' = \cos^2 u$ with initial condition $e^{-10k}u_0$ is

$$u(k) = \tan^{-1}(k + \tan(e^{-10k}u_0)).$$

The natural question to ask is “how close are these approximations to the exact solution?”

Suppose we are given the problem $u' = Au + Bu$ where A and B are arbitrary operators. Consider a simple splitting where we alternate by solving $u' = Au$ and then solving $u' = Bu$ for the one time step. We can write (and implement) this procedure as

$$\begin{aligned} U^* &= S_1(k)U^n \\ U^{n+1} &= S_2(k)U^* \end{aligned}$$

or equivalently as

$$U^{n+1} = S_2(k)S_1(k)U^n$$

where $S_1(k)$ and $S_2(k)$ are solution operators.

What is the splitting error (of applying the operators A and B successively rather than concurrently)? Suppose that $S_1(k)$ and $S_2(k)$ are exact solution operators

$$S_1(k)U^n = e^{kA}U^n \quad \text{and} \quad S_2(k)U^n = e^{kB}U^n$$

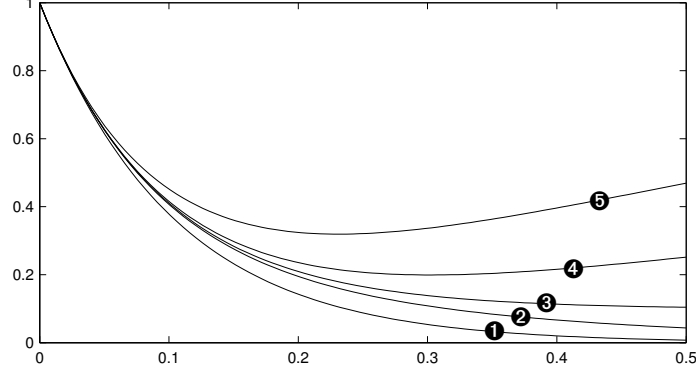


Figure 1.9: Solution to (1.10) using general splitting in ❶ and ❺, Strang splitting in ❷ and ❹, and exact solution in ❸. Each operator is inverted exactly without numerical error over a time step $k = 0.5$.

Then

$$\begin{aligned} \text{Concurrently:} \quad & U^{n+1} = e^{k(A+B)}U^n \\ \text{Successively:} \quad & \tilde{U}^{n+1} = e^{kB}e^{kA}U^n \end{aligned}$$

The splitting error is $|U^{n+1} - \tilde{U}^{n+1}|$. By Taylor series expansion, we have that

$$\begin{aligned} e^{kA} &= I + kA + \frac{1}{2}k^2A^2 + O(k^3) \\ e^{kB} &= I + kB + \frac{1}{2}k^2B^2 + O(k^3) \end{aligned}$$

and hence

$$\begin{aligned} e^{kB}e^{kA} &= I + k(A+B) + \frac{1}{2}k^2(A^2 + B^2 + 2BA) + O(k^3) \\ e^{k(A+B)} &= I + k(A+B) + \frac{1}{2}k^2(A^2 + B^2 + AB + BA) + O(k^3) \end{aligned}$$

Subtracting these two operators gives us

$$e^{k(A+B)} - e^{kB}e^{kA} = \frac{1}{2}k^2(AB + BA - 2BA) + O(k^3)$$

There is no splitting error if $AB = BA$. But, in general, A and B do not commute. So, the splitting error is $O(k^2)$ for each time step. After n time steps, the error is $O(k)$.

Strang splitting

We can reduce the splitting error in operator splitting by using **Strang Splitting**. For each time step, we solve $u' = Au$ for a half time step; then solve $u' = Bu$ for a full time step; and finally, solve $u' = Au$ for a half time step. Because

$$e^{kA/2}e^{kB}e^{kA/2} - e^{k(A+B)} = O(k^3),$$

Strang splitting is second order. Note that

$$\begin{aligned} U^{n+1} &= S_1(\tfrac{1}{2}k)S_2(k)S_1(\tfrac{1}{2}k)U^n \\ &= S_1(\tfrac{1}{2}k)S_2(k)S_1(\tfrac{1}{2}k)S_1(\tfrac{1}{2}k)S_2(k)S_1(\tfrac{1}{2}k)U^{n-1} \end{aligned}$$

But since $S_1(\tfrac{1}{2}k)S_1(\tfrac{1}{2}k) = S_1(k)$, this becomes

$$= S_1(\tfrac{1}{2}k)S_2(k)S_1(k)S_2(k)S_1(\tfrac{1}{2}k)U^{n-1}$$

Continuing like this we get

$$= S_1(\tfrac{1}{2}k)[S_2(k)S_1(k)]^{n-1}S_2(k)S_1(\tfrac{1}{2}k)U^0$$

So, to implement Strang splitting, we only need to solve $u' = Au$ for a half time step on the initial and final time steps. In between, we use simple splitting. In this manner, we get $O(k^2)$ global splitting error without much more computation. It is impossible to have a splitting method with splitting error $O(k^3)$ or smaller.

IMEX methods

Another way to approach stiff problems is to use an implicit-explicit (IMEX) method. This method keeps the same discretization for the time derivative but uses an implicit discretization for the stiff linear term and an explicit discretization for the nonlinear term.

Example. A typical second-order IMEX method combines the second-order Adams-Bashforth and the second-order Adams-Moulton methods. Recall that the second-order Adams-Moulton method is really just the trapezoidal (Crank-Nicolson) method. For both of these methods, the time derivatives are approximated to second-order at $t_{n+1/2}$. For $u' = \mathcal{L}u + \mathcal{N}u$, we have

$$\frac{U^{n+1} - U^n}{k} = \mathcal{L}U^{n+1/2} + \mathcal{N}U^{n+1/2} \approx \tfrac{1}{2}\mathcal{L}U^{n+1} + \tfrac{1}{2}\mathcal{L}U^n + \tfrac{3}{2}\mathcal{N}U^n - \tfrac{1}{2}\mathcal{N}U^{n-1} \quad (1.11)$$

o

Example. We can also build a second-order IMEX scheme using BDF2 for the implicit part:

$$u' = \mathcal{L}u \quad \Rightarrow \quad \frac{1}{k} \left(\tfrac{3}{2}U^{n+1} - 2U^n + \tfrac{1}{2}U^{n-1} \right) = \mathcal{L}U^{n+1}$$

Backward difference formulae expand the ODE about t_{n+1} . This means that we should find an approximation for U^{n+1} in terms of U^n and U^{n-1} .

$$\begin{aligned} u(t_n) &= u(t_{n+1}) - ku'(t_{n+1}) + \frac{k^2}{2}u''(t_{n+1}) + O(k^3) \\ u(t_{n-1}) &= u(t_{n+1}) - 2ku'(t_{n+1}) + \frac{4k^2}{2}u''(t_{n+1}) + O(k^3) \end{aligned}$$

Combining these equations we have that

$$2u(t_n) - u(t_{n-1}) = u(t_{n+1}) - k^2u''(t_{n+1}) + O(k^3)$$

So, the IMEX method is

$$\frac{\frac{3}{2}U^{n+1} - 2U^n + \frac{1}{2}U^{n-1}}{k} = \mathcal{L}U^{n+1} + \mathcal{N}[2U^n - U^{n-1}]$$

◦

Other ways of implementing IMEX methods include IMEX Adams Methods which combine Adams–Bashforth and Adams–Moulton methods. Another way to solve stiff problems is to use an implicit-explicit Runge-Kutta method. See A. Uri, S. Ruuth and R. Spiteri, Implicit-explicit Runge-Kutta methods for time-independent partial differential equations, *Applied Numerical Mathematics*, 25 (1997) 151–167.

Integrating factors

Consider the problem

$$u' = \mathcal{L}u + \mathcal{N}u \quad \text{with} \quad u(0) = u_0$$

where \mathcal{L} is a linear operator and \mathcal{N} is a nonlinear operator. Set $v = e^{-t\mathcal{L}}u$, then

$$v' = e^{-t\mathcal{L}}\mathcal{N}(e^{t\mathcal{L}}v) \quad \text{with} \quad v(0) = e^{-t\mathcal{L}}u_0.$$

There is no splitting error, but such a method of solution may only mollify the stiffness.

1.8 Theory: consistency, stability, convergence

Let's consider the general initial value problem $u_t = \mathcal{L}u$ where \mathcal{L} is a linear operator. In the next chapter, we will consider \mathcal{L} to be the Laplacian operator and we will get the heat equation $u_t = \Delta u$. In Chapter 3, we will consider $\mathcal{L} = c \cdot \nabla$ and we will get the advection equation $u_t = c \cdot \nabla u$.

Suppose that $u(t, \cdot)$ is the analytic solution to $u_t = \mathcal{L}u$. Let $t_n = nk$ be the uniform discretization of time and U^n denote the finite difference approximation of $u(t_n, \cdot)$. Define a finite difference operator H_k such that $U^{n+1} = H_k U^n$. We will use the subscript k in H_k to denote the dependence on the stepsize k . Of course, $U^{n+1} = H_k^{n+1} U^0$.

For an ODE, U^n and H_k are scalars. For a system of ODEs or a PDE, U^n is a vector and H_k is a matrix. For example, the forward Euler method of the heat equation $u_t = u_{xx}$ is

$$U_j^{n+1} - U_j^n = \frac{k}{h^2} (U_{j+1}^n - 2U_j^n + U_{j-1}^n).$$

Letting $\lambda = k/h^2$ and assuming Dirichlet boundary conditions

$$\begin{bmatrix} U_1^{n+1} \\ \vdots \\ U_{j-1}^{n+1} \end{bmatrix} = \underbrace{\begin{bmatrix} 1-2\lambda & \lambda & & \\ & \lambda & \ddots & \lambda \\ & & \lambda & 1-2\lambda \end{bmatrix}}_{H_k} \begin{bmatrix} U_1^n \\ \vdots \\ U_{j-1}^n \end{bmatrix}.$$

Define the error to be $E_k(t_n, x) = U^n - u(t_n, x)$. A method is **convergent** in some particular norm $\|\cdot\|$ if the error $\|E_k(\cdot, t)\| \rightarrow 0$ as $k \rightarrow 0$ for any fixed $t \geq 0$ and for any initial data u_0 . Define the **truncation error** to be

$$\tau_k(t_n, \cdot) = \frac{1}{k} [u(t + k, \cdot) - H_k u(t, \cdot)].$$

A method is **consistent** if the truncation error $\|\tau_k(\cdot, t)\| \rightarrow 0$ as $k \rightarrow 0$. A method is of **order** p if for all sufficiently smooth initial data with compact support, there exists a constant C_τ such that $\|\tau_k(\cdot, t)\| \leq C_\tau k^p$ for all $k < k_0$ and $0 \leq t < T$.

Convergence:	Comparing two solutions
Consistency:	Comparing two equation

Define an operator norm $\|\cdot\|$ as

$$\|A\| = \max_{x \in \mathbb{R}^n} \frac{\|Ax\|}{\|x\|} = \max_{\|x\|=1} \|Ax\|$$

A method is **stable** if for each time T , there exists a C_S and $k_0 > 0$ such that $\|H_k^n\| \leq C_S$ for all $nk \leq T$ and $k \leq k_0$.

Remark: If $\|H_k\| \leq 1$ then $\|H_k^n\| \leq \|H_k\|^n \leq 1$ which implies stability. We can even relax conditions to $\|H_k\| \leq 1 + \alpha k$ for some constant α , since

$$\|H_k^n\| \leq \|H_k\|^n \leq (1 + \alpha k)^n \leq e^{\alpha kn} \leq e^{\alpha T} \Rightarrow \text{stability}$$

Consistency + Stability \Rightarrow Convergence

Theorem 3 (Lax Equivalence Theorem). *A consistent method is convergent if and only if it is stable.*

Proof. (Consistency + Stability \Rightarrow Convergence)

For a finite difference operator H_k , the truncation error is

$$\tau_k(t_n, x) = \frac{1}{k} (u(t_{n+1}, \cdot) - H_k(u(t_n, \cdot))).$$

Equivalently (for t_{n-1}),

$$\tau_k(t_{n-1}, x) = \frac{1}{k} (u(t_n, \cdot) - H_k(u(t_{n-1}, \cdot))).$$

Therefore, we can write the exact solution as

$$u(t_n, \cdot) = H_k(u(t_{n-1}, \cdot)) + k\tau_k(t_{n-1}, \cdot)$$

Furthermore, the error at t_n is

$$\begin{aligned}
 E_k(t_n, \cdot) &= U^n - u(t_n, \cdot) \\
 &= H_k U^{n-1} - H_k u(t_{n-1}, \cdot) - k\tau_k(t_{n-1}) \\
 &= H_k(E(t_{n-1}, \cdot)) - k\tau_k(t_{n-1}, \cdot) \\
 &= H_k[H_k(E(t_{n-2}, \cdot)) - k\tau_k(t_{n-2}, \cdot)] - k\tau_k(t_{n-1}, \cdot) \\
 &= \vdots \\
 &= H_k^n E(0, \cdot) - k \sum_{j=1}^n H_k^{j-1}(t_{j-1}, \cdot) \tau_k(t_{j-1}, \cdot).
 \end{aligned}$$

By the triangle inequality

$$\|E_k(t_n, \cdot)\| \leq \|H_k^n\| \|E(0, \cdot)\| + k \sum_{j=1}^n \|H_k^{j-1}\| \|\tau_k(t_{j-1}, \cdot)\|$$

From the stability condition, we have that $\|H_k^j\| \leq C_S$ for all $1 \leq j \leq n$ and all $0 \leq nk \leq T$.

$$\|E_k(t_n, \cdot)\| \leq C_S \left(\|E(0, \cdot)\| + k \sum_{j=1}^n \|\tau_k(t_{j-1}, \cdot)\| \right)$$

From the consistency condition, we have that $\|\tau_k(t_{j-1}, \cdot)\| \rightarrow 0$ as $k \rightarrow 0$. If the method is p th order consistent, then $\|\tau_k(\cdot, t)\| \leq C_\tau k^p$

$$\begin{aligned}
 \|E_k(t_n, \cdot)\| &\leq C_S(\|E_k(0, \cdot)\| + C_\tau k^p kn) \\
 &\leq C_1(\|E_k(0, \cdot)\| + Tk^p), \quad p > 0
 \end{aligned}$$

If the initial error $\|E_k(0, \cdot)\| \leq C_2 k^p$ then $\|E_k(t_n, \cdot)\| \leq Ck^p$ for some C for all $0 \leq nk \leq T$. \square

Exercises

1.1. A θ -scheme is of the form

$$\frac{U^{n+1} - U^n}{k} = (1 - \theta)f(U^{n+1}) + \theta f(U^n)$$

Find the regions of absolute stability for the θ -scheme. *Hint:* Express the complex variable λk as $x + iy$ and find the locus of points for $r^2 \leq 1$.

1.2. Use the system of equation (1.3) to compute the 3-step Adams-Bashforth, Adams-Moulton and BDF methods. Plot the regions of absolute stability for these methods.

1.3. Show that for a linear problem, one step of the Crank-Nicolson method is the same as applying a forward-Euler method for one-half time-step followed by applying the backward Euler methods for one-half time-step.

- 1.4. Plot the region of stability for the Merson method, whose Butcher tableau is

0					
$\frac{1}{3}$	$\frac{1}{3}$				
$\frac{1}{3}$	$\frac{1}{6}$	$\frac{1}{6}$			
$\frac{1}{2}$	$\frac{1}{8}$	0	$\frac{3}{8}$		
1	$\frac{1}{2}$	0	$-\frac{3}{2}$	2	
	$\frac{1}{6}$	0	0	$\frac{2}{3}$	$\frac{1}{6}$

- 1.5. Consider the ODE $u'(t) = -\varepsilon^{-1}u(t) + f(t)$ where $\varepsilon \ll 1$. Suppose that you want to use a compute the solution $u(1)$ to within $\delta \ll 1$. In terms of ε and δ , what step size should you take if using a second order Adams-Moulton method? What about a third-order Adams-Moulton method?

- 1.6. Let A and B be two arbitrary operators. Show that

$$e^{kA/2}e^{kB}e^{kA/2} - e^{k(A+B)} = O(k^3)$$

from which it follows that the splitting error using Strang Splitting is $O(k^2)$.

- 1.7. Sometimes it is easier to implement

$$\mathcal{N}U^{n+1/2} \approx \mathcal{N}\left(\frac{3}{2}U^n - \frac{1}{2}U^{n-1}\right)$$

instead of

$$\mathcal{N}U^{n+1/2} \approx \frac{3}{2}\mathcal{N}U^n - \frac{1}{2}\mathcal{N}U^{n-1}$$

in the second-order Adams-Bashforth method. Show that these two approximation are the same up to $O(k^2)$.

- 1.8. Develop a third-order L-stable IMEX scheme. Study the stability of the method by writing the characteristic polynomial and plotting the regions of absolute stability in the complex plane for the implicit part and the explicit part of the IMEX scheme.

- 1.9. MATLAB has several built-in “black box” routines for solving ODEs numerically: `ode23`, `ode45`, `ode113`, `ode15s`, `ode23s`, `ode23t`, `ode23tb`, `ode15i`. For each function, give a short description of the method it uses and what the advantages/disadvantages of using that method. You may wish to use the `help` and `type` commands along with the MATLAB’s function reference and Moler [2004] as references.

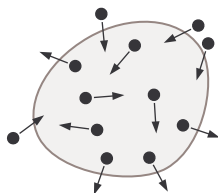
- 1.10. Find the numerical solution to the following initial value problem:

$$u'(t) = -100(u - \cos t) - \sin t \quad \text{with} \quad u(0) = 1$$

using the forward Euler, the trapezoidal and the fourth-order Runge-Kutta methods using an appropriate step-size. Plot the error between the numerical solution and the analytic

solution at time $t = 2$ as a function of the time step-size k using a log-log plot. Compute the slope of the plot to confirm that you are getting the correct rate of convergence for the methods.

Parabolic Equations



A group of molecules, bacteria, insects, people, etc. will often move in a random way. As a result these “particles” tend to spread out. On the large scale, this evolution can be modeled by the diffusion process. Let $u(t, \mathbf{x})$ be the concentration of particles in \mathbb{R}^n and consider an *arbitrarily shaped (but fixed)* domain $\Omega \subset \mathbb{R}^n$. Then the mass of particles at some time t in Ω is $\int_{\Omega} u(t, \mathbf{x}) dV$. Since the particles are moving, the mass of particles in Ω changes as some particles enter the domain and other particles leave it. The change in the number of particles in Ω is equal to the integral of the flux of particles $J(\mathbf{x})$ through the boundary:

$$\frac{d}{dt} \int_{\Omega} u(t, \mathbf{x}) dV = \int_{\partial\Omega} J(t, \mathbf{x}) \cdot \mathbf{n} dA$$

By invoking the divergence theorem, we have that

$$\int_{\Omega} \frac{\partial}{\partial t} u(t, \mathbf{x}) dV = \int_{\Omega} \nabla \cdot J(t, \mathbf{x}) dV$$

Since the domain Ω was arbitrarily chosen, it follows that

$$\frac{\partial}{\partial t} u(t, \mathbf{x}) = \nabla \cdot J(t, \mathbf{x})$$

Fickian diffusion says that the flux $J(t, \mathbf{x})$ (the number of particles crossing the interface per period of time) is proportional the gradient of the concentration $u(\mathbf{x})$. This makes sense if we consider a random walk process. In one-dimension, consider adjacent cells at $x - \frac{1}{2}h$ and $x + \frac{1}{2}h$ separated by an interface at x . If a particle moves to the right or the left with constant velocity $m(x)$, then the flux across the interface can be approximated by

$$J(t, x) \approx \frac{m(x)(u(x + h/2) - u(x - h/2))}{h}$$

In the limit, as $h \rightarrow 0$ we have that

$$J = m(x) \frac{\partial}{\partial x} u.$$

So, in general,

$$\frac{\partial u}{\partial t} = \nabla \cdot (m(\mathbf{x}) \nabla u).$$

Example. Consider a heat conducting bar. If u is temperature, then

$$u_t + J_x = 0$$

where $J(x)$ is the heat flux. Fourier's law states that $J(x) = -m(x)u_x$ where $m(x) > 0$ is the heat conductivity. Then

$$u_t = (m(x)u_x)_x.$$

This initial value boundary value problem is well-posed, and it can be solved analytically by separation of variables or using a Fourier transform. Suppose that the bar has a uniform heat conductivity $m(x) = a$ and the ends of the bar at $x = 0$ and $x = 1$ are in contact with heat sinks of constant temperature $u = 0$. Furthermore, suppose that the bar has an initial temperature distribution $u_0(x)$. Then the heat equation is given by the initial-boundary value problem

$$u_t = au_{xx}, \quad u(0, x) = u_0(x), \quad u(t, 0) = 0 \quad u(t, 1) = 0 \quad (2.1)$$

◦

Example. Important examples of diffusion occur frequently in biology. One such example includes chemotaxis. Animals and bacteria often communicate by releasing chemicals. Note only do the bacteria spread out due to normal diffusion mechanisms, the bacteria may move also in a direction up a concentration gradient, following the strongest attractant, which itself is subject to diffusion.

◦

Parabolic partial differential equations are linear partial differential equations that are first order in time and second order in space. That is, the highest order derivative in space is a second derivative. They include the heat or diffusion equations, reaction-diffusion problems, and viscous fluid dynamics. One property of parabolic systems is the **maximum principle** which says that the maximum value of a solution always decreases in time unless it lies on a boundary. Equivalently, the minimum value of a solution always increases unless it lies on a boundary. Another property is that the system is dissipative. That is, the L^2 -norm of the solution decreases in time. A third property is that the solution for $t > 0$ is smooth and grows smoother in time. For nonlinear diffusion, we'll treat these properties as rules of thumb.

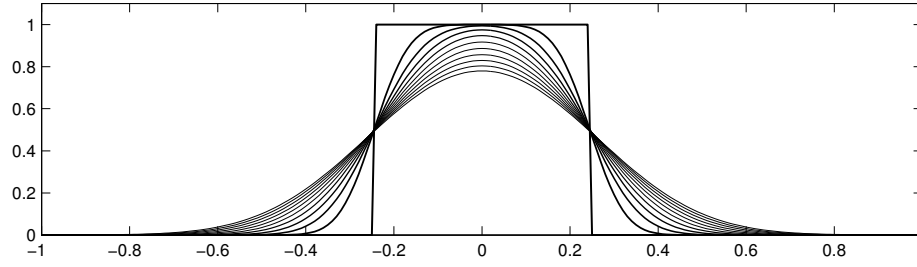


Figure 2.1: Solution to the heat equation $u_t = u_{xx}$ over an unbounded domain. Several snapshots in time are superimposed with the thicker lines occurring earlier.

2.1 Method of lines

A typical means of numerically solving a time-dependent PDE is using the **method of lines**. We discretize space and keep time continuous to derive a semidiscrete method. The advantage of this approach is that we can decouple time and space discretizations, thereby combining the best methods for each. Take the grid points in space $0 = x_0 < x_1 < \cdots < x_N = 1$. For simplicity, define the a uniform mesh $x_j = jh$ where h is the grid spacing. Let $U_j(t) \approx u(x_j, t)$. We get a second-order approximation of the second derivative, we use a centered-difference scheme

$$\frac{\partial^2}{\partial x^2} U_j \approx \frac{\partial}{\partial x} \left(\frac{U_{j+1/2} - U_{j-1/2}}{h} \right) \approx \frac{U_{j+1} - 2U_j + U_{j-1}}{h^2}$$

Hence, $u_t = au_{xx}$ becomes

$$\partial_t U_j = a \frac{U_{j+1} - 2U_j + U_{j-1}}{h^2} \quad 0 \leq j \leq N \quad (2.2)$$

We now have a system of $N + 1$ ordinary differential equations, and we can use numerical methods for ordinary differential equations to solve the system of equations. Note that the system is not closed. In addition to the interior mesh points ($0 \leq j \leq N$), we also have mesh points outside of the domain ($j = -1$ and $j = N + 1$). In the next section, we will use the boundary conditions to explicitly remove the ghost points and close the system.

This approach of employing the numerical method of lines to solve a partial differential equation is often called **time-marching**. We can combine any consistent ODE solver with any consistent discretization of the Laplacian. In the previous chapter, we discussed the forward Euler, the backward Euler, leap frog and the trapezoidal methods to solve ODEs. Let's look at applying these methods to solve PDEs.



Forward (Explicit) Euler Method

$O(k + h^2)$

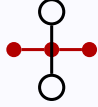
$$\frac{U_j^{n+1} - U_j^n}{k} = a \frac{U_{j+1}^n - 2U_j^n + U_{j-1}^n}{h^2} \quad (2.3)$$



Backward (Implicit) Euler Method

 $O(k + h^2)$

$$\frac{U_j^{n+1} - U_j^n}{k} = a \frac{U_{j+1}^{n+1} - 2U_j^{n+1} + U_{j-1}^{n+1}}{h^2} \quad (2.4)$$



Richard Method (Unstable!)

 $O(k^2 + h^2)$

$$\frac{U_j^{n+1} - U_j^{n-1}}{2k} = a \frac{U_{j+1}^n - 2U_j^n + U_{j-1}^n}{h^2} \quad (2.5)$$



Crank-Nicolson (Trapezoidal) Method

 $O(k^2 + h^2)$

$$\frac{U_j^{n+1} - U_j^n}{k} = \frac{1}{2}a \frac{U_{j+1}^{n+1} - 2U_j^{n+1} + U_{j-1}^{n+1}}{h^2} + \frac{1}{2}a \frac{U_{j+1}^n - 2U_j^n + U_{j-1}^n}{h^2} \quad (2.6)$$

The implementation of the forward Euler method is straight-forward—at each iteration apply (2.3) and then enforce the boundary conditions. Because the backward Euler method is implicit, we will need to do some extra work. Let $\lambda = ak/h^2$. In this case, the backward Euler method for (2.1) is

$$\begin{bmatrix} 1+2\lambda & -2\lambda & & & \\ -\lambda & 1+2\lambda & -\lambda & & \\ & \ddots & \ddots & \ddots & \\ & & -\lambda & 1+2\lambda & -\lambda \\ & & & -2\lambda & 1+2\lambda \end{bmatrix} \begin{bmatrix} U_0^{n+1} \\ U_1^{n+1} \\ \vdots \\ U_N^{n+1} \end{bmatrix} = \begin{bmatrix} U_0^n \\ U_1^n \\ \vdots \\ U_{N-1}^n \\ U_N^n \end{bmatrix} \quad (2.7)$$

At each time step we need to invert a tridiagonal matrix by using a tridiagonal solver. A tridiagonal solver efficiently uses Gaussian elimination by only operating on the diagonal and two off-diagonals. By doing so, a tridiagonal solver requires only $O(3N)$ operations to solve a linear system whereas general Gaussian elimination requires $O(\frac{2}{3}N^3)$ operations. For example, the MATLAB code in the following section takes about 25 times longer when using full matrices instead of sparse matrices.

► In MATLAB, the command `mldivide (\)` will automatically implement a tridiagonal solver on a sparse matrix. But the matrix must be sparse. The MATLAB command `help sparsfun` returns a list of MATLAB's functions for working with sparse matrices.

2.2 Boundary conditions

In the previous section, we used the method of lines to convert a PDE into a system of ODEs with $N + 3$ unknowns and $N + 1$ equations. We use the boundary conditions to eliminate the two unknowns U_{-1} and U_{N+1} and close the system. In addition to the interior mesh points at $x_0, \dots, x + N$, we also have **ghost points** outside the domain. We must use the

boundary constraints to explicitly remove the ghost points from the system. We should also try to maintain a tridiagonal system so that the method is efficient. Let's examine how we add various types of boundary conditions.

A **Dirichlet boundary condition** specifies the value of the solution on the boundary, such as in (2.1) where

$$u(0) = c_0, u(1) = c_1 \text{ for some constants } c_0 \text{ and } c_1$$

physically models a bar in contact with heat sinks (at temperatures of c_0 and c_1) at $u(0)$ and $u(1)$. This boundary condition provides a constraint to the PDE; the discretization of the boundary condition provides a constraint to the finite difference approximation. This constraint allows us to remove variables from and thereby “close” the system.

Consider the uniformly spaced meshpoints $\{x_0, x_2, \dots, x_N\}$ with corresponding solution $\{U_0, U_2, \dots, U_N\}$. We'll eliminate the left ghost point, the right ghost point can similarly be eliminated. We should also try to maintain a tridiagonal system so that the method is efficient. We can eliminate U_{-1} from the backward Euler equation (2.4) by using the second-order approximation $\frac{1}{2}(U_{-1} + U_1) \approx U_0 = u(0) = c_0$. Now, we can eliminate U_{-1}^{n+1} :

$$\begin{array}{rcl} -\lambda U_{-1}^{n+1} + (1 + 2\lambda)U_0^{n+1} - \lambda U_1^{n+1} & = & aU_0^n \\ U_{-1}^{n+1} & + & U_1^{n+1} = 2c_1 \\ \hline (1 + 2\lambda)U_0^{n+1} - 2\lambda U_1^{n+1} & = & aU_0^n + 2c_1\lambda \end{array}$$

A **Neumann boundary condition** specifies the value of the first derivative of the solution (or the gradient) on the boundary. For example, to model a bar that is insulated at both ends, we set the heat flux $q(x)$ to be zero on the boundary giving $u_x(0) = u_x(1) = 0$. Zero Neumann boundary conditions are also called reflecting boundary conditions.

Example. Let's implement reflecting boundary conditions into a Crank-Nicolson scheme. The Crank-Nicolson scheme is second-order in space, so we should also approximate the boundary conditions with the same order. Otherwise, we will lose accuracy. In addition to the interior equations given by (2.6), we must add boundary constraints. As before we should try to maintain a tridiagonal system so that the method is efficient. Consider the uniformly spaced meshpoints $\{x_0, x_2, \dots, x_N\}$ with corresponding solution $\{U_0, U_2, \dots, U_N\}$. Then the Crank-Nicolson method (2.6) is

$$-\lambda U_{j-1}^{n+1} + (2 + 2\lambda)U_j^{n+1} - \lambda U_{j+1}^{n+1} = \lambda U_{j-1}^n + (2 - 2\lambda)U_j^n + \lambda U_{j+1}^n \quad 0 \leq j \leq N \quad (2.8)$$

Of course, the system is not closed—it spills over into U_{-1} and U_{N+1} . We enforce the boundary conditions $u_x(x_0) = 0$ on the left and $u_x(x_N) = 0$ by using the second-order center-difference approximation

$$\frac{\partial}{\partial x}u(x_0) \approx \frac{U_2 - U_0}{2h} \quad \text{and} \quad \frac{\partial}{\partial x}u(x_N) \approx \frac{U_{N+1} - U_{N-1}}{2h}.$$

We have that

$$U_1^n - U_{-1}^n = U_1^{n+1} - U_{-1}^{n+1} = 0 \quad \text{and} \quad U_{N+1}^n - U_{N-1}^n = U_{N+1}^{n+1} - U_{N-1}^{n+1} = 0$$

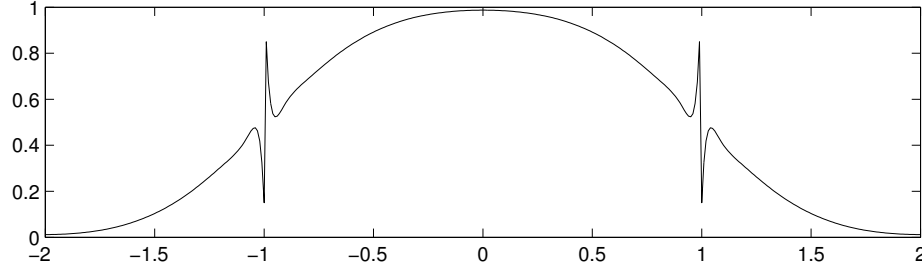


Figure 2.2: Numerical solution to the heat equation starting with a discontinuous initial condition using the Crank-Nicholson method after 8 time steps. The Crank-Nicholson method is not L-stable and hence the transient spikes.

Hence, the solution at the ghost points are given by

$$U_{-1}^n = U_1^n, U_{-1}^{n+1} = U_1^{n+1}, U_{N+1}^n = U_{N-1}^n, U_{N+1}^{n+1} = U_{N-1}^{n+1}$$

Now we can eliminate the ghost points from the system and close it:

$$\lambda U_{j-1}^{n+1} + (2 + 2\lambda)U_j^{n+1} + \lambda U_{j+1}^{n+1} = \lambda U_{j-1}^n + (2 - 2\lambda)U_j^n + \lambda U_{j+1}^n \quad \text{for } 1 < j < N \quad (2.9a)$$

$$(2 + 2\lambda)U_0^{n+1} + 2\lambda U_1^{n+1} = (2 - 2\lambda)U_0^n + 2\lambda U_1^n \quad (2.9b)$$

$$2\lambda U_{N-1}^{n+1} + (2 + 2\lambda)U_N^{n+1} = 2\lambda U_{N-1}^n + (2 - 2\lambda)U_N^n \quad (2.9c)$$

This system now consists of two tridiagonal matrices—one on the left and one on the right. The following MATLAB code implements the Crank-Nicholson method with reflecting boundary conditions.

```
function heat
dx = .01; dt = .01; L = 2;
x = (-L:dx:L)'; n = length(x);
lambda = dt/dx^2;
Q = spdiags(repmat([1 -2 1],[n 1]),-1:1,n,n)*lambda;
A = (2*speye(n) + Q);
B = (2*speye(n) - Q);
A(1,2)=2*lambda; A(n,n-1)=2*lambda; B(1,2)=-2*lambda; B(n,n-1)=-2*lambda;
u = (abs(x)<1);
for i = 1:1000
    u = B\ (A*u);
    plot(x,u); ylim([0 1]);drawnow;
end
```

Running this code, we observe transient spikes, which are clearly numerical artifacts. See Figure 2.2. What causes these artifacts? Recall that the Crank-Nicholson method is A-stable, but not L-stable. How can we get a better solution? Think about this when we discuss numerical stability in the next section and when we discuss dispersive versus dissipative schemes in the next chapter. \circ

A third type of boundary condition, the **Robin or mixed boundary condition** which is a combination of Dirichlet and Neumann boundary conditions

$$\begin{aligned}u(0) + b_0 u_x(0) &= c_0, \\u(1) + b_1 u_x(1) &= c_1\end{aligned}$$

for some constants b_0 , b_1 , c_0 and c_1 .

If the problem has **periodic boundary conditions** $u(0) = u(1)$, the system can no longer be made to be tridiagonal. Instead, we are left with a circulant matrix which can be solved more simply with a Fourier spectral method as we will see in Chapter 5.

Suppose that we are interested in solving a problem over the entire real line, but we are only interested in some bound region $x \in [0, 1]$. One method of implementation would be to use a nonuniform mesh. Another method would be to use **absorbing boundary conditions** at $x = 0$ and $x = 1$. One way to do this is to use an absorbing sponge layer. Sponge layers can be difficult to implement. See Trefethen [1996].

2.3 Stability: von Neumann analysis

In Chapter 1, we found region of absolute stability for a numerical method for a linear ODE $u' = \lambda u$ by determining the values λk for which a perturbation of the numerical solution will decrease over time. We now want to determine under what conditions a numerical method for a linear PDE is stable. We can use a Fourier transform to decouple a PDE into a system of ODEs and then use the same theory we developed for ODEs. Because numerical stability is a local behavior, changing the boundary conditions from Dirichlet to periodic will not affect the analysis.

Take a uniform discretization in space $x_j = jh$. The discrete Fourier transform is defined as

$$\hat{u}(t, \xi) = \frac{1}{\sqrt{2N+1}} \sum_{j=-N}^N u(t, x_j) e^{-i\xi jh}$$

and the discrete inverse Fourier transform is defined as

$$u_j(t, x_j) = \int \hat{u}(\xi, t) e^{i\xi jh} d\xi = \frac{1}{\sqrt{2N+1}} \sum_{\xi=-N}^N \hat{u}(\xi, t) e^{i\xi jh}$$

where the wave number ξ is an integer (because the domain is bounded).

Use the method of lines to spatially discretize the heat equation

$$u_t = a u_{xx} \quad \text{with} \quad 0 < x < 2\pi$$

to get

$$\frac{\partial}{\partial t} u(t, x_j) = a \frac{u(t, x_{j+1}) - 2u(t, x_j) + u(t, x_{j-1}))}{h^2}.$$

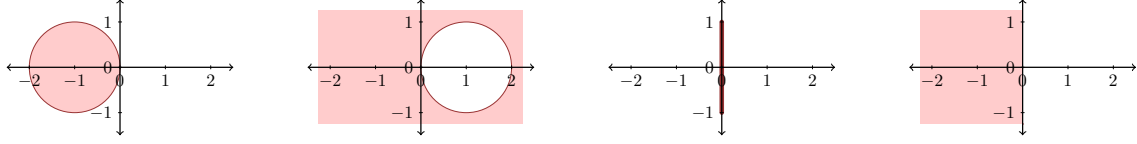


Figure 2.3: Regions of absolute stability (shaded) for the forward Euler, backward Euler, leapfrog, and trapezoidal methods (left to right).

Substituting the definition of the inverse Fourier transform:

$$\frac{1}{\sqrt{2N+1}} \sum_{\xi=-N}^N \partial_t \hat{u}(t, \xi) e^{i\xi jh} = \frac{1}{\sqrt{2N+1}} \sum_{\xi=-N}^N a \frac{e^{i\xi(j+1)h} - 2e^{i\xi jh} + e^{i\xi(j-1)h}}{h^2} \hat{u}(t, \xi).$$

Equivalently,

$$\frac{1}{\sqrt{2N+1}} \sum_{\xi=-N}^N \partial_t \hat{u}(t, \xi) e^{i\xi jh} = \frac{1}{\sqrt{2N+1}} \sum_{\xi=-N}^N a \frac{e^{i\xi h} - 2 + e^{-i\xi h}}{h^2} \hat{u}(t, \xi) e^{i\xi jh}.$$

This equality is true for each integer j , so

$$\partial_t \hat{u} = a \underbrace{\frac{e^{i\xi h} - 2 + e^{-i\xi h}}{h^2}}_{\mu} \hat{u}(\xi).$$

Note that μ is the eigenvalue of the ODE $\hat{u}' = \mu \hat{u}$. By simplifying the expression μ

$$\mu = a \frac{2 \cos \xi h - 2}{h^2} = \frac{2a}{h^2} (\cos \xi h - 1) = -4 \frac{a}{h^2} \sin^2 \frac{\xi h}{2}$$

we have that

$$\partial_t \hat{u} = -4 \frac{a}{h^2} \sin^2 \frac{\xi h}{2} \hat{u} \quad (2.10)$$

where $j = -N, \dots, N$. Note that (2.10) is now a linear ordinary differential equation $\hat{u}_t = \mu \hat{u}$. For consistency in notation, we now use μ to represent the eigenvalue instead of λ as we did in Chapter 1. Note that μ is a non-positive real number.

Recall the regions of stability for the forward Euler, backward Euler, leapfrog, and trapezoidal methods. See Figure 2.3. The method is stable as long as μk lies in the region of stability. The eigenvalue $\mu = -4 \frac{a}{h^2} \sin^2 \frac{\xi h}{2}$ can be as big as $\mu = -4 \frac{a}{h^2}$. Using this bound along with associated the region of stability we can determine the stability conditions for the various methods.

Example. Determine the conditions for stability if we implement (2.10) using the explicit Euler method. In Chapter 1, we found that the forward (explicit) Euler method is absolutely stable if $|1 + \mu k| \leq 1$. The forward Euler method for the heat equation (2.2) is stable if

$$\left| 1 - 4 \frac{ak}{h^2} \sin^2 \frac{\xi h^2}{2} \right| \leq 1$$

So,

$$-1 \leq 1 - 4 \frac{ak}{h^2} \sin^2 \frac{\xi h^2}{2} \leq 1$$

or equivalently

$$\frac{ak}{h^2} \sin^2 \frac{\xi h^2}{2} \leq \frac{1}{2} \quad (2.11)$$

for all ξ . Therefore, the stability condition for the forward Euler method requires that $k \leq h^2/2a$. Such a stability condition is called the **Courant–Friedrichs–Lewy condition** or **CFL condition**. \circ

Example. Determine the conditions for stability if we implement (2.10) using the implicit Euler method. In Chapter 1, we determined that for the backward (implicit) Euler method, there was no constraint on k . Therefore, the backward Euler method is **unconditionally stable**. The trade-off is we need to implement a tridiagonal solver at each time step. When h is small, this extra work is worth it. \circ

In the above analysis, we used that fact that $\{e^{i\xi jh}\}$ forms a linearly independent basis, and therefore, we only need to compare the Fourier coefficients. We can abbreviate the stability analysis a bit by setting $U_j^n = A e^{i\xi jh}$. A method is stable iff $|A| \leq 1$. We call this **von Neumann stability analysis**.

Example. Determine the stability condition of the Richard method (2.5).

$$\frac{U_j^{n+1} - U_j^{n-1}}{2k} = a \frac{U_{j+1}^n - 2U_j^n + U_{j-1}^n}{h^2}$$

We set $U_j^n = A^n e^{i\xi jh}$

$$\frac{A^{n+1} e^{i\xi jh} - A^{n-1} e^{i\xi jh}}{2k} = a A^n \frac{e^{i\xi h} - 2 + e^{-i\xi h}}{h^2} e^{i\xi jh}$$

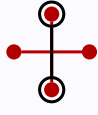
Then

$$\frac{A^2 - 1}{2k} = -a \frac{4}{h^2} \sin^2 \frac{\xi h}{2} A$$

from which we have that

$$A^2 + 8a \frac{k}{h^2} \sin^2 \frac{\xi h}{2} A - 1 = 0$$

From the constant term of the quadratic, we know that the product of the roots of this equation is -1 . Hence, both roots are real. (Complex roots would appear in conjugate pairs, the product of which is positive.) Since $\{+1, -1\}$ are not the roots, the absolute value of one root must be greater than one. Therefore, the Richard method is unconditionally unstable! This result should be clear from the region of absolute stability of the Leap frog method $(-i, i)$ and the fact that μ is a negative real number. A slight modification to the Richard method is the implicit Dufort-Frankel method. \circ



Dufort-Frankel Method

 $O(k^2 + h^2 + \frac{k^2}{h^2})$

$$\frac{U_j^{n+1} - U_j^{n-1}}{2k} = a \frac{U_{j+1}^n - (U_j^{n+1} + U_j^{n-1}) + U_{j-1}^n}{h^2} \quad (2.12)$$

Example. Determine the stability condition for the Dufort-Frankel method by von Neumann analysis. Setting $U_j^n = A^n e^{ij\xi h}$ in the Dufort-Frankel method (2.12) gives us

$$\begin{aligned} \frac{A^2 - 1}{2k} &= a \frac{Ae^{i\xi h} - (A^2 + 1) + Ae^{-i\xi h}}{h^2} \\ &= a \frac{2A \cos \xi h - (A^2 + 1)}{h^2}. \end{aligned}$$

Let $\lambda = ak/h^2$, then

$$(1 + 2\lambda)A^2 - 4\lambda \cos \xi h A - (1 - 2\lambda) = 0.$$

The roots to this quadratic are

$$\begin{aligned} A_{\pm} &= \frac{1}{2(1 + 2\lambda)} \left[4\lambda \cos \xi h \pm \sqrt{16\lambda^2 \cos^2 \xi h + 4(1 - 4\lambda^2)} \right] \\ &= \frac{1}{2(1 + 2\lambda)} \left[4\lambda \cos \xi h \pm \sqrt{4 - 16\lambda^2 \sin^2 \xi h} \right] \\ &= \frac{1}{1 + 2\lambda} \left[2\lambda \cos \xi h \pm \sqrt{1 - 4\lambda^2 \sin^2 \xi h} \right] \end{aligned}$$

We consider two cases.

Case 1: $(1 - 4\lambda^2 \sin^2 \xi h \geq 0)$

$$|A \pm| \leq \frac{2\lambda + 1}{1 + 2\lambda} = 1$$

Case 2: $(1 - 4\lambda^2 \sin^2 \xi h \leq 0)$

$$\begin{aligned} |A \pm| &= \frac{2\lambda \cos \xi h \pm i\sqrt{4\lambda^2 \sin^2 \xi h - 1}}{1 + 2\lambda} \\ |A \pm|^2 &= \frac{(2\lambda \cos \xi h)^2 + 4\lambda^2 \sin^2 \xi h - 1}{(1 + 2\lambda)^2} = \frac{4\lambda^2 - 1}{4\lambda^2 + 4\lambda + 1} \leq 1 \end{aligned}$$

So, the Dufort-Frankel scheme is unconditionally stable.

But we're not finished yet. Let's compute the truncation error of the Dufort-Frankel method by substituting for each term the Taylor expansion about (x_j, t_k) . First note that we can rewrite

$$\frac{U_j^{n+1} - U_j^{n-1}}{2k} = a \frac{U_{j+1}^n - (U_j^{n+1} + U_j^{n-1}) + U_{j-1}^n}{h^2}$$

as

$$\frac{U_j^{n+1} - U_j^{n-1}}{2k} = a \frac{U_{j+1}^n - 2U_j^n + U_{j-1}^n}{h^2} - a \frac{U_j^{n+1} - 2U_j^n + U_j^{n-1}}{h^2}$$

Then Taylor expansion simply gives us

$$u_t + \frac{k^2}{6}u_{ttt} + O(k^4) = a(u_{xx} + \frac{h^2}{12}u_{xxx} + O(h^4)) - a(\frac{k^2}{h^2}u_{tt} + O(k^4/h^2)).$$

The truncation error is $O(k^2 + h^2 + \frac{k^2}{h^2})$. If $k = h$, then the method is inconsistent, and we are actually finding the solution to the equation

$$u_t = au_{xx} + au_{tt}.$$

This says that although the Dufort-Frankel scheme is absolutely stable, it is only consistent when $k \ll h$. Furthermore, we don't get second order convergence unless $k = O(h^2)$. \circ

2.4 Higher dimensional methods

In two dimensions, the heat equation is

$$u_t = u_{xx} + u_{yy}.$$

We make the approximation

$$U_{jk}^n = u(x_j, y_k, t_n)$$

and defining the discrete operators in x and y

$$\delta_x^2 U = (U_{j+1,k} - 2U_{j,k} + U_{j-1,k})/h^2 \quad (2.13a)$$

$$\delta_y^2 U = (U_{j,k+1} - 2U_{j,k} + U_{j,k-1})/h^2 \quad (2.13b)$$

Then for $\Delta x = \Delta y = h$, the Crank-Nicolson method is

$$\frac{U_{jk}^{n+1} - U_{jk}^n}{k} = \frac{1}{2} (\delta_x^2 U^{n+1} + \delta_x^2 U^n + \delta_y^2 U^{n+1} + \delta_y^2 U^n) \quad (2.14)$$

The method is second order in time and space and unconditionally stable. For von Neumann analysis in two dimensions we substitute $U_{jk}^n = A^n e^{i(j\xi h + k\eta h)}$ and stability requires that $|A| \leq 1$. But we no longer have a tridiagonal system. In fact, if we have 100 grid points in both the x - and y -directions, then we would need to invert a $10^4 \times 10^4$ (sparse) matrix.

Instead, we will use operator splitting. Let's examine two ways of splitting the right-hand operator of (2.14) while still keeping the method implicit—the fractional step method

$$\frac{U_{jk}^{n+1} - U_{jk}^n}{k} = \frac{1}{2} \left(\overline{\delta_x^2 U^{n+1}} + \overline{\delta_x^2 U^n} + \underline{\delta_y^2 U^{n+1}} + \underline{\delta_y^2 U^n} \right)$$

and the alternate direction implicit (ADI)

$$\frac{U_{jk}^{n+1} - U_{jk}^n}{k} = \frac{1}{2} \left(\overline{\delta_x^2 U^{n+1}} + \underline{\delta_x^2 U^n} + \underline{\delta_y^2 U^{n+1}} + \overline{\delta_y^2 U^n} \right).$$

where the overline and the underline notation is used to group terms for the operator splitting. For example, one iteration of ADI is

$$\begin{aligned} \frac{U_{jk}^* - U_{jk}^n}{k} &= \frac{1}{2} (\delta_x^2 U^* + \delta_y^2 U^n) \\ \frac{U_{jk}^{n+1} - U_{jk}^n}{k} &= \frac{1}{2} (\delta_x^2 U^* + \delta_y^2 U^{n+1}). \end{aligned}$$

Fractional step method:

Consider the splitting method

$$u_t = Du \quad \text{where} \quad D = D_1 + D_2 + \cdots + D_p \quad \text{and} \quad D_j = \partial_{x_j x_j}$$

We solve $u_t = D_j u$ successively for $1 \leq i \leq p$. In this way, a multidimensional problem becomes a succession of one-dimensional problems. Discretize $D_j \mapsto \delta_{x_j}^2$ and use the Crank-Nicholson method at each stage

$$\frac{U^{n+j/p} - U^{n+(j-1)/p}}{k} = D_j \frac{U^{n+j/p} + U^{n+(j-1)/p}}{2} \quad j = 1, \dots, p$$

This method is unconditionally stable. What about the accuracy? At each fractional step,

$$(I - \frac{1}{2}kD_j)U^{n+j/p} = (I + \frac{1}{2}kD_j)U^{n+(j-1)/p}.$$

So,

$$(I - \frac{1}{2}kD_1)(I - \frac{1}{2}kD_2) \cdots (I - \frac{1}{2}kD_p)U^{n+1} = (I + \frac{1}{2}kD_1)(I + \frac{1}{2}kD_2) \cdots (I + \frac{1}{2}kD_p)U^n.$$

Formally,

$$(I - \frac{1}{2}kD)^{-1} = I + \frac{1}{2}kD + \frac{1}{4}k^2D^2 + \cdots$$

and hence

$$U^{n+1} - U^n = \underbrace{\frac{1}{2}k \sum_i D_i (U^{n+1} + U^n)}_{\text{Crank-Nicolson}} + \underbrace{\frac{1}{4}k^2 \sum_{i,j} D_i D_j (U^{n+1} + U^n)}_{O(k^2)} + O(k^3)$$

So, the fractional step method is second order.

Alternating direction implicit method (ADI)

In the ADI method we take

$$U^{n+1/2} - U^n = \frac{1}{2} \frac{k}{h^2} \left(\delta_x^2 U^{n+1/2} + \delta_y^2 U^n \right), \quad (2.15a)$$

$$U^{n+1} - U^{n+1/2} = \frac{1}{2} \frac{k}{h^2} \left(\delta_x^2 U^{n+1/2} + \delta_y^2 U^{n+1} \right). \quad (2.15b)$$

To implement this we need two tridiagonal solvers for this two-stage method.

Let's look at stability. Taking the Fourier transform with $x \mapsto \xi$ and $y \mapsto \eta$

$$\begin{aligned} \hat{U}^{n+1/2} &= \hat{U}^n + \frac{1}{2} \lambda \left(-4 \sin^2 \frac{\xi h}{2} \right) \hat{U}^{n+1/2} + \frac{1}{2} \lambda \left(-4 \sin^2 \frac{\eta h}{2} \right) \hat{U}^n \\ \hat{U}^n &= \hat{U}^n + \frac{1}{2} \lambda \left(-4 \sin^2 \frac{\xi h}{2} \right) \hat{U}^{n+1/2} + \frac{1}{2} \lambda \left(-4 \sin^2 \frac{\eta h}{2} \right) \hat{U}^{n+1} \end{aligned}$$

from which

$$\hat{U}^{n+1} = \frac{1 - 2\lambda \sin^2 \frac{\eta h}{2}}{1 + 2\lambda \sin^2 \frac{\xi h}{2}} \hat{U}^{n+1/2}. \quad \text{and} \quad \hat{U}^{n+1/2} = \frac{1 - 2\lambda \sin^2 \frac{\eta h}{2}}{1 + 2\lambda \sin^2 \frac{\xi h}{2}} \hat{U}^n$$

So,

$$\hat{U}^{n+1} = \underbrace{\frac{\left(1 - 2\lambda \sin^2 \frac{\xi h}{2}\right) \left(1 - 2\lambda \sin^2 \frac{\eta h}{2}\right)}{\left(1 + 2\lambda \sin^2 \frac{\eta h}{2}\right) \left(1 + 2\lambda \sin^2 \frac{\xi h}{2}\right)}}_{\mu} \hat{U}^n.$$

For absolute stability $|\mu| \leq 1$ for all λ . Because the denominator is always larger than the numerator, $\mu \leq 1$ so the method is unconditionally stable.

Now let's determine the order of (2.15)

$$(2.15a) - (2.15b) : \quad 2U^{n+1/2} = U^{n+1} + \frac{1}{2}\lambda\delta_y^2(U^n - U^{n+1}) \quad (2.16a)$$

$$(2.15a) + (2.15b) : \quad U^{n+1} - U^n = \lambda\delta_x^2 U^{n+1/2} + \frac{1}{2}\lambda\delta_y^2(U^n + U^{n+1}) \quad (2.16b)$$

Substituting (2.16b) into (2.16a)

$$\begin{aligned} U^{n+1} - U^n &= \frac{1}{2}\lambda\delta_x^2 \left(U^n + U^{n+1} + \frac{1}{2}\lambda\delta_y^2(U^n - U^{n+1}) \right) + \frac{1}{2}\lambda\delta_y^2(U^n + U^{n+1}) \\ &= \underbrace{\lambda \left(\delta_x^2 \frac{U^n + U^{n+1}}{2} + \delta_y^2 \frac{U^n + U^{n+1}}{2} \right)}_{\text{Crank-Nicolson}} + \underbrace{\frac{\lambda^2}{4}\delta_x^2\delta_y^2(U^n - U^{n+1})}_{O(\lambda^2 h^4)} \end{aligned}$$

Since $\lambda = k/h^2$, $O(\lambda^2 h^4) = O(h^2)$. So the method is the same order as the Crank-Nicolson scheme $O(h^2 + k^2)$.

For a three dimensional problem

$$u_t = u_{xx} + u_{yy} + u_{zz}$$

Take $\Delta x = \Delta y = \Delta z = h$ and $\lambda = k/h^2$. Then the three dimensional ADI method

$$\begin{aligned} U^* - U^n &= \frac{1}{6}\lambda [\delta_x^2(U^* + U^n) + \delta_y^2(2U^n) + \delta_z^2(2U^n)] \\ U^{**} - U^n &= \frac{1}{6}\lambda [\delta_x^2(U^* + U^n) + \delta_y^2(U^{**} + U^n) + \delta_z^2(2U^n)] \\ U^{n+1} - U^{**} &= \frac{1}{6}\lambda [\delta_x^2(U^* + U^n) + \delta_y^2(U^{**} + U^n) + \delta_z^2(U^{n+1} + U^n)] \end{aligned}$$

The method is unconditionally stable and requires three tridiagonal solvers.

both the fractional step and ADI method are Crank-

2.5 Nonlinear diffusion equation

Consider the heat equation on a rod for which the heat conductivity $m(u) > 0$ changes as a function of the temperature. In this case,

$$\frac{\partial}{\partial t}u = \frac{\partial}{\partial x}(m(u)\frac{\partial}{\partial x}u) \quad (2.17a)$$

$$u(x, 0) = u_0(x) \quad (2.17b)$$

$$u(0, t) = u(1, t) = 0 \quad (2.17c)$$

In this case, we can solve the problem by using the method of lines

$$\begin{aligned} \frac{\partial}{\partial t}u &= \frac{\partial}{\partial x}(m(u)\frac{\partial}{\partial x}u) \\ &\approx \frac{m(U_{j+1/2})\frac{\partial}{\partial x}U_{j+1/2} - m(U_{j-1/2})\frac{\partial}{\partial x}U_{j-1/2}}{h} \\ &\approx m(U_{j+1/2})\frac{U_{j+1} - U_j}{h^2} - m(U_{j-1/2})\frac{U_j - U_{j-1}}{h^2} \\ &\approx m\left(\frac{U_j + U_{j+1}}{2}\right)\frac{U_{j+1} - U_j}{h^2} - m\left(\frac{U_{j-1} + U_j}{2}\right)\frac{U_j - U_{j-1}}{h^2} \end{aligned} \quad (2.18)$$

Because the diffusion equation is stiff, the ODE solver should be a stiff (implicit) solver. Otherwise, we will be forced to take several small timesteps to ensure stability. See Table 2.1. Because the problem is nonlinear, we will need to solve a system of nonlinear equations. Typically, this is done using Newton's method. **MATLAB** has several stiff ODE solvers (recall Problem 1.9). To implement Newton's method in the implicit ODE solvers, **MATLAB** numerically computes and inverts the Jacobian matrix. Luckily, our system of nonlinear equations is sparse, so the Jacobian matrix is also sparse—a tridiagonal matrix. We must explicitly tell **MATLAB** that the Jacobian is sparse by passing it the sparsity pattern for the Jacobian using the `odeset` command. In this case, the sparsity pattern is a tridiagonal matrix of ones.

Example. The following **MATLAB** function implements (2.18) with $m(u) = u^2$ using `ode15s`, which uses a variation of the adaptive-step BDF methods. The nonlinear diffusion equation $u_t = (u^p u_x)_x$ for some p is known as the porous medium equation. It is used to model the dispersion of grasshoppers and the flow of groundwater. The solution is plotted in Figure 2.4 on the next page.

```
function [t,x,U] = porousmedium
n = 400; L = 2; x = linspace(-L,L,n)'; dx = x(2)-x(1);
m = @(u) u.^2;
Du = @(t,u) [0;diff(m((u(1:n-1)+u(2:n))/2)).*diff(u))/dx^2;0];
u = double(abs(x)<1);
options = odeset('JPattern',spdiags(ones([n 3]),-1:1,n,n));
[t,U] = ode15s(Du,[0 2],u,options);
```

The solution U can be interpolated in time and animated using the following code:

```
U = interp1(t,U,linspace(0,2,200)');
for i=1:size(U,1), plot(x,U(i,:)); ylim([0 1]); drawnow; end
```

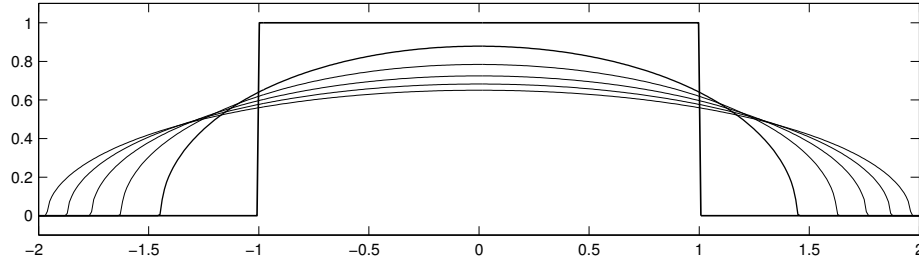


Figure 2.4: Solution to the PDE $u_t = (m(u)u_x)_x$ with $m(u) = u^2$. Several snapshots in time ($t = 0, 0.4, 0.6, 1.2, 1.6, 2.0$) are superimposed with the thicker lines occurring earlier.

solver	time steps	run time (sec)
ode15s	754	1.5
ode23t	1573	3.4
ode23s	1130	5.7
ode15s*	754	15.1
ode23t*	1573	26.9
ode23	11817	29.1
ode113	18784	69.6
ode23s*	1130	219.2
ode45	35777	220.8

Table 2.1: Run times of the MATLAB program `porousmedium` with 400 meshpoints in space. Note that the stiff solvers perform significantly faster (up to 150 times faster) than the nonstiff solvers. The duplicate stiff solvers marked by asterisks give the run times when the sparsity pattern was not supplied.

Note that $u^2 u_x = (\frac{1}{3}u^3)_x$. In this case, it may be simpler to implement $u_t = (\frac{1}{3}u^3)_{xx}$ instead, and we could simply replace the appropriate lines of the function `porousmedium` with the following:

```
m = @(u) u.^3/3;
Du = @(t,u) [0;diff(m(u),2)/dx^2;0];
```

◦

Stability

Linear von Neumann stability analysis cannot be applied to problems with *variable coefficients* and *nonlinear problems*. In these cases, the **energy method** is an important tool of checking stability. We define the “energy” of a system as the L^2 -norm of the variable u . The system is stable if the energy is non-increasing. Multiplying equation (2.17a) by u and integrating over the domain, we have that

$$\int_0^1 u \frac{\partial}{\partial t} u \, dx = \int_0^1 u \frac{\partial}{\partial x} (m(u) \frac{\partial}{\partial x} u) \, dx.$$

By using integration by parts on the right-hand side of the equation, we have

$$\frac{1}{2} \frac{\partial}{\partial t} \int_0^1 u^2 dx = - \int_0^1 m(u) \left(\frac{\partial}{\partial x} u \right)^2 dx$$

This expression says that $\frac{\partial}{\partial t} \|u(t, \cdot)\|_2 \leq 0$.

Example. We can apply the same idea to a numerical scheme by using the discrete l^2 -norm.

$$\begin{aligned} \sum_j U_j \frac{\partial}{\partial t} U_j &= \sum_j \frac{1}{h^2} [m_{j+1/2}(U_{j+1} - U_j)U_j - m_{j-1/2}(U_j - U_{j-1})U_j] \\ \partial_t \frac{1}{2} \sum_j U_j^2 &= \frac{1}{h^2} \sum_j m_{j+1/2}(U_{j+1} - U_j)U_j - m_{j-1/2}(U_j - U_{j-1})U_j \\ &= -\frac{1}{h^2} \sum_j m_{j-1/2}(U_j - U_{j-1})^2 \leq 0 \end{aligned}$$

So $\|U^n\|_2 \leq \|U^0\|_2$. ◦

Example. Discrete norms are equivalent. So, we can use other norms. For example, let's use the l^∞ -norm to confirm the stability condition of the forward Euler discretization of the linear heat equation (2.3)

$$\frac{U_j^{n+1} - U_j^n}{k} = a \frac{U_{j+1}^n - 2U_j^n + U_{j-1}^n}{h^2}$$

letting $\lambda = a \frac{k}{h^2}$, then

$$U_j^{n+1} = \lambda U_{j-1}^n + (1 - 2\lambda)U_j^n + \lambda U_{j+1}^n$$

If we take $0 \leq \lambda \leq \frac{1}{2}$, then

$$\begin{aligned} |U_j^{n+1}| &= \lambda |U_{j-1}^n| + (1 - 2\lambda)|U_j^n| + \lambda |U_{j+1}^n| \\ &\leq (\lambda + 1 - 2\lambda + \lambda) \|U^n\|_\infty = \|U^n\|_\infty \end{aligned}$$

for all j . So

$$\|U^{n+1}\|_\infty \leq \|U^n\|_\infty \leq \dots \leq \|U^0\|_\infty$$

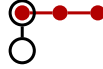
for all n , if we take $k \leq h^2/2a$. ◦

Exercises

2.1. Show that if the boundaries are insulating (the flux through the boundary is zero: $u_x(0) = u_x(1) = 0$), then the heat equation conserves “mass” ($\int_0^1 u dx$), i.e., no heat is lost or gained. Hint: Show $\frac{d}{dt} \int_0^1 u dx = 0$.

2.2. By Taylor series expansion, determine the truncation error of the Crank-Nicolson scheme for $u_t = u_{xx}$.

- 2.3. Suppose that we want to solve the heat equation $u_t = u_{xx}$ numerically. Instead of using $\{U_{j-1}, U_j, U_{j+1}\}$ to approximate the u_{xx} at x_j , one could instead use $\{U_j, U_{j+1}, U_{j+2}\}$ to approximate the u_{xx} at x_j . For example, the stencil for the backward Euler method is



This alternative approach is bad for a number of reasons. First, it provides only a first-order approximation, whereas the centered-difference approximation is second-order. More importantly, it affects stability. Discuss the stability conditions of using such a scheme with various time-stepping methods.

2.4. Solve the heat equation $u_t = u_{xx}$ over the domain $[0, 1]$ with initial conditions $u(0, x) = \sin \pi x$ and boundary conditions $u(0, t) = u(1, t) = 0$ using the forward Euler scheme. Use $N = 20$ grid points in space. Use the CFL condition to determine the stability requirement on the time step size k .

1. Examine the behavior of the numerical solution if k is slightly above or below the stability threshold.
2. Use the slope of the log-log plot of the error at $t = 1$ (with the analytic solution) to determine the order of convergence. To do this, you should use several values for the grid spacing h while keeping the time step $k \ll h$ constant. Then use several values for the time step k keeping the mesh $h \ll k$.

2.5. The Schrödinger equation

$$i\varepsilon \frac{\partial \psi}{\partial t} = -\frac{1}{2}\varepsilon^2 \frac{\partial^2 \psi}{\partial x^2} + V(x)\psi$$

models the quantum mechanical behavior of a particle in a potential field $V(x)$. The parameter ε is the scaled Plank constant, which gives the relative length and time scales, and $i = \sqrt{-1}$. The probability of finding a particle at a position x is given by $\rho(x, t) = |\psi(x, t)|^2$. (Note that the Schrödinger equation is not a parabolic PDE. The L^2 -norm of the solution is constant in time and has no maximum principal. The equation has attributes more closely related to hyperbolic PDEs.)

1. For $V(x) \equiv 0$, find the stability conditions (CFL conditions) using a second-order centered-space discretization along with forward Euler, leap-frog, Crank-Nicolson (trapezoidal), and Runge-Kutta (RK4) time-stepping.
2. Use Crank-Nicolson (trapezoidal) time-stepping to solve the initial value problem for the harmonic oscillator with $V(x) = \frac{1}{2}x^2$ and $\psi(x, 0) = e^{-(x-1)^2/2\varepsilon}$ and $\varepsilon = 0.3$ over a domain $[-3, 3]$. The analytic solution is given by

$$\rho(x, t) = e^{-(x - \cos t)^2/\varepsilon}.$$

Use the slope of the log-log plot of the error at $t = 2\pi$ (with the analytic solution) to determine the order of convergence. To do this, you should use several values for the

grid spacing h while keeping the time step $k \ll h$ constant. Then use several values for the time step k keeping the mesh $h \ll k$.

3. What happens when the mesh spacing h or the time-step k are about equal to or larger than ε ?

2.6. In polar coordinates, the heat equation $u_t = \Delta u$ is

$$\frac{\partial u}{\partial t} = \frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{\partial u}{\partial r} \right) + \frac{1}{r^2} \frac{\partial^2 u}{\partial \theta^2}.$$

For a radially symmetric problem, the heat equation simplifies to

$$\frac{\partial u}{\partial t} = \frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{\partial u}{\partial r} \right)$$

Suppose that we have with insulating boundary conditions at $|r| = 1$: $\frac{\partial u}{\partial r}|_{r=1} = 0$

1. Develop a Crank-Nicolson method for this problem and implement it for the initial conditions

$$u(r, 0) = \begin{cases} 1, & \text{if } |r| < \frac{1}{2} \\ 0, & \text{if } |r| > \frac{1}{2} \end{cases}.$$

Use symmetry to determine an appropriate boundary condition at $r = 0$.

2. Implement the method and plot the solution at several steps in time. Ensure the boundary conditions are correct.
3. Use the slope of the log-log plot of the error at $t = 1$ to determine the order of convergence. To do this, you should use several values for the grid spacing h while keeping the time step $k \ll h$ constant. Then use several values for the time step k keeping the mesh $h \ll k$. You can use the numerical solution with a very small k and h for an “exact” solution. To implement the l_2 -norm

$$\|U - V\|_2 = \left(\sum_{i=1}^N |U_i - V_i|^2 \Delta x \right)^{1/2}$$

in MATLAB use `error = norm(U-V)*sqrt(dx).`

Note about implementation of the boundary condition at $r = 0$. At $j = 0$, we have division by $r_0 = 0$. Furthermore, because of the boundary conditions the numerator on the right-hand side is zero. So, the equation is undefined at $j = 0$. We’ll need to fix this. The solution u must be an even function at the origin. So, all the odd derivatives (u_r , u_{rrr} , etc.) must vanish. When we take the Taylor series expansion of $u(r)$ at $r = 0$ we get

$$\begin{aligned} u(r) &= u(0) + ru_r(0) + \frac{1}{2}r^2u_{rr}(0) + \frac{1}{6}r^3u_{rrr}(0) + O(r^4) \\ &= u(0) + \frac{1}{2}r^2u_{rr}(0) + O(r^4) \quad (\text{odd derivatives are zero}) \end{aligned}$$

Furthermore,

$$\frac{1}{r}(ru_r)_r = 2u_{rr}(0) + O(r^2).$$

Letting $r \rightarrow 0$ we have that

$$\frac{1}{r}(ru_r)_r|_{r=0} = 2u_{rr}(0).$$

Also, from above (taking $r = h$)

$$u(h) = u(0) + \frac{1}{2}h^2 u_{rr}(0) + O(h^4)$$

So,

$$\frac{1}{r}(ru_r)_r|_{r=0} = 2u_{rr}(0) = 4 \frac{u(h) - u(0)}{h^2} + O(h^2)$$

Therefore, at $j = 0$ we will use the equation

$$\frac{\partial}{\partial t} U_j = 4 \frac{U_{j+1} - U_j}{h^2}.$$

2.7. Both the fractional step methods and ADI methods can be viewed as Crank-Nicolson methods with an additional second-order truncation term. Analyze both methods by comparing the size of the truncation terms plus the original truncation second-order term of the Crank-Nicolson method.

2.8. The Allen-Cahn equation

$$u_t = \Delta u + \varepsilon^{-1}u(1 - u^2)$$

is a reaction diffusion equation which describes a nonconservative phase transition. The solution $u(x, y, t)$ has stable equilibria at $u = \pm 1$ with a thin phase interphase given by $-1 < u < 1$. As a loose analogy, think of ice melting and freezing in pool of water with $u = 1$ representing the ice and $u = -1$ representing the water. Determine the numerical solution to the two-dimensional Allen-Cahn equation using a numerical method that is second order in time and space with stability condition independent of ε . Consider the domain $[-2, 2] \times [-2, 2]$ with reflecting boundary conditions. Take $\varepsilon = 1/20$ and take the initial conditions

$$u(0, x, y) = \begin{cases} 1, & \text{if } x^4 < x^2 - \frac{1}{2}y^2 \\ -1, & \text{otherwise} \end{cases}$$

Also, try random initial conditions: `u = randn(N)`. Observe the behavior of the solution over time. You can do this by plotting the solution $u(x, y, t)$ as a density map at several points in time

```
colormap(1-gray(200));
image((u+1)*100);
```

Hint: Use Strang splitting—the resulting ODE can be solved analytically. You can easily implement a two-dimensional fractional step method in MATLAB using `DX*U*DY`, where `DX` and `DY` are the operators in x and y and `U` is the dependent variable.

2.9. The Cahn-Hilliard equation

$$u_t = -\Delta(\Delta u + \varepsilon u(1 - u^2))$$

models the annealing of two metals in an alloy as it cools—think of the separation of oil and vinegar after a bottle of dressing has been vigorously shaken. Show that a **convexity splitting** scheme

$$\frac{U^{n+1} - U^n}{k} = -\Delta^2 U^{n+1} - 2\varepsilon \Delta U^{n+1} + \varepsilon \Delta U^n + \varepsilon \Delta (U^n)^3$$

is numerically stable.

Hyperbolic Equations

Nonlinear hyperbolic equations are synonymous with shock wave formation from the hydraulic lift of a tsunami as it approaches the shore to the sonic boom of a jet aircraft to the pressure wave of an atomic bomb. Hyperbolic equations also model magnetohydrodynamics of plasmas, acoustics, combustion, visco-elasticity of earthquakes, and traffic flow.

A principle difference between hyperbolic and parabolic equations is that while solutions to linear parabolic equations are always smooth (and become increasingly smoother over time). Solutions to hyperbolic equations often are not. In fact, the solutions of nonlinear hyperbolic equations with smooth initial conditions may become discontinuous in finite time resulting in shock waves. Such a discontinuity introduces difficulties when we try to approximate it using a high order polynomial interpolant.

Another difference is that the stability condition for explicit schemes for hyperbolic PDEs are typically $k = O(h)$ rather than the $k = O(h^2)$ that we had for parabolic PDEs. While there was a significant advantage to using a stiff (implicit) solver for parabolic equations, there is no significant advantage for hyperbolic PDEs. Therefore, we only need to consider explicit schemes.

The subject of numerical methods for hyperbolic equations has had considerable development in the last fifty years. Much of the interest arising out of a desire to efficiently simulate bomb blasts, supersonic flow, and semiconductors. This chapter is an introduction. See ? for a complete coverage.

3.1 Linear hyperbolic equations

Let $u(x, t)$ be some physical quantity such as density, pressure, velocity, etc. Recall from the previous chapter that if the flux of u is given by $f(u)$, then the time evolution of the quantity u is given by

$$\frac{\partial}{\partial t} u + \frac{\partial}{\partial x} f(u) = 0$$

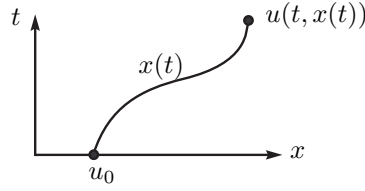


Figure 3.1: Method of characteristics

In Chapter 2, the flux was given by Fick's Law $f(x) = m(x) \frac{\partial}{\partial x} u$. Now, let's simply take $f(u) = cu$. Information propagating with velocity c is described by the linear advection equation

$$\frac{\partial}{\partial t} u + c \frac{\partial}{\partial x} u = 0 \quad (3.1a)$$

$$u(x, 0) = u_0(x) \quad (3.1b)$$

where $u_0(x) \in \mathbb{R}$. This problem can be solved by the **method of characteristics**. Note that by taking $dx/dt = c$, then

$$\frac{d}{dt} u(t, x(t)) = \frac{\partial}{\partial t} u + \frac{dx}{dt} \frac{\partial}{\partial x} u = \frac{\partial}{\partial t} u + c \frac{\partial}{\partial x} u = 0$$

which says that the solution $u(t, x(t))$ is constant along the characteristic curves $x(t)$. Since $dx/dt = c$, the characteristics curves are $x(t) = ct + x_0$ for some x_0 . The exact solution is then given by

$$u(x(t), t) = u(x(0), 0) = u(x_0, 0) = u_0(x_0) = u_0(x - ct)$$

The constant c is called the **characteristic speed**. It is not necessary that c be a constant—the method of characteristics also works for $c(u)$. We will return to the nonlinear case later in the chapter.

3.2 Numerical methods for linear hyperbolic equations

We first will develop numerical schemes for the linear problem $u_t + cu_x = 0$. The schemes developed using this simple linear model will serve as a prototypic methods which we will extend to solve nonlinear hyperbolic schemes later on.

Upwind

The forward Euler approximation in time and a first-order approximation in space is the simplest scheme for $u_t + cu_x = 0$ would be to take. Such a scheme is called an **upwind method**. Let's look at the stability conditions of this scheme.



Upwind Method ($c > 0$)

$O(k + h)$

$$\frac{U_j^{n+1} - U_j^n}{k} + c \frac{U_j^n - U_{j-1}^n}{h} = 0 \quad (3.2)$$

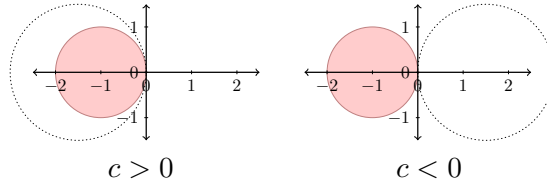


Figure 3.2: Stability for upwind method (3.2). Eigenvalues of $\frac{c}{h}(U_j^n - U_{j-1}^n)$ are overlaid on the region of absolute stability for forward Euler $\frac{1}{k}(U_j^{n+1} - U_j^n)$. For $c > 0$, the upwind method is made stable by scaling k so that $k < c/h$. The upwind method (3.2) is unconditionally unstable when $c < 0$.



Upwind Method ($c < 0$)

$O(k + h)$

$$\frac{U_j^{n+1} - U_j^n}{k} + c \frac{U_{j+1}^n - U_j^n}{h} = 0 \quad (3.3)$$

Consider the semidiscrete equation

$$\frac{\partial}{\partial t} u(t, x_j) = c \frac{u(t, x_j) - u(t, x_{j-1})}{h}$$

In the Fourier domain (replacing $u(t, x_j)$ with $\hat{u}(t, \xi)e^{i\xi jh}$, we have that the decoupled system

$$\frac{\partial}{\partial t} \hat{u}(t, \xi) = c \frac{e^{i\xi h} - 1}{h} \hat{u}(t, \xi)$$

First, assume that $c \geq 0$. The locus of $e^{i\xi h} - 1$ is a circle of unit radius centered at -1 . So $\frac{c}{h}(e^{i\xi h} - 1)$ is a circle centered at $-\frac{c}{h}$ with radius $\frac{c}{h}$. Recall that the region of stability for the forward Euler method is the unit circle centered at -1 . Therefore, the upwind method (3.2) is stable if $k < c/h$. Recall from the previous chapter that this is called the **CFL condition** for stability and $|c|k/h$ is called the **CFL number** for the upwind scheme.

Note that if $c < 0$, then $\frac{c}{h}(e^{i\xi h} - 1) = \frac{|c|}{h}(-e^{i\xi h} + 1)$ is a circle centered at $+\frac{c}{h}$ with radius $\frac{c}{h}$. In this case, the upwind method (3.2) is absolutely unstable. We can modify (3.2) when $c < 0$ by taking

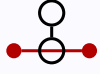
$$u_x(x_j, t) \approx \frac{U_{j+1}^n - U_j^n}{h}$$

giving us method (3.3). Similarly, the upwind scheme (3.3) is unconditionally unstable for $c > 0$. This should make intuitive sense—to predict the whether we need to look in the direction from which it is coming (upwind) not the direction in which it is going (downwind).

To implement the schemes, we must determine the sign of c . We can combine the two schemes (3.2) and (3.3) to get a scheme that does this automatically

$$\frac{U_j^{n+1} - U_j^n}{k} + \frac{c + |c|}{2} \frac{U_j^n - U_{j-1}^n}{h} + \frac{c - |c|}{2} \frac{U_{j+1}^n - U_j^n}{h} = 0.$$

How can we interpret the CFL condition physically? Information propagates at a finite speed along any characteristic of the advection equation (3.1) with $c > 0$. So, the solution at any specific point in space can only be influenced by the initial conditions in some bounded region of space. This region is called the **domain of dependence**. The analytic solution to the advection equation (3.1) is $u(t_n, x_j) = u_0(x_j - ct_n)$. Take uniform stepsize k in time and a uniform mesh h in space. The domain of dependence for the analytic solution is $[x_j, x_j + cnk]$. The domain of dependence for the numerical solution is $[x_j, x_j + nh]$. By keeping $ck/h < 1$, we ensure that the domain of dependence of the exact solution is contained in the domain of dependence of the numerical solution.



Centered-Difference Method (Unstable)

 $O(k + h^2)$

$$\frac{U_j^{n+1} - U_j^n}{k} + c \frac{U_{j+1}^n - U_{j-1}^n}{2h} = 0 \quad (3.4)$$

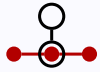
The upwind scheme gives a method that is $O(h + k)$. It seems reasonable that we could improve the order by using a centered-difference in space approximation (3.4). Such a scheme is $O(k + h^2)$. What about stability? By formally replacing $u(t, x_j)$ with $\hat{u}(t, \xi)e^{i\xi jh}$, we have that the decoupled system

$$\frac{\partial}{\partial t} \hat{u}(t, \xi) = c \frac{e^{i\xi h} - e^{-i\xi h}}{2h} \hat{u}(t, \xi) = -\frac{c}{h} i \sin \xi h \hat{u}(t, \xi)$$

in the Fourier domain. The eigenvalues are purely imaginary, but the region of stability for the forward Euler scheme is a unit circle in the left half plane, so the the method is unconditionally unstable.

Lax-Friedrichs

Let's modify the centered-difference scheme to make it more stable. One way to do this is to make is centered-difference in space and centered-difference in time (leapfrog). See Problem (3.3). Another way is to approximate the U_j^n term in the time derivative as $\frac{1}{2}(U_{j+1}^n + U_{j-1}^n)$. We call this method the Lax-Friedrich Method (3.5).



Lax-Friedrich Method

 $O(k + h^2/k)$

$$\frac{U_j^{n+1} - \frac{1}{2}(U_{j+1}^n + U_{j-1}^n)}{k} + c \frac{U_{j+1}^n - U_{j-1}^n}{2h} = 0 \quad (3.5)$$

The method is order $O(k + h^2/k)$ and conditionally stable. Von Neumann analysis says that

$$A - \frac{1}{2} (e^{i\xi h} + e^{-i\xi h}) = -c \frac{k}{h} i \sin \xi h$$

We solve for A

$$A = \cos \xi h - ic \frac{k}{h} \sin \xi h$$

from which we see that

$$|A|^2 = \cos^2 \xi h + \left(c \frac{k}{h}\right)^2 \sin^2 \xi h$$

The Lax-Friedrich scheme is stable if $|c|k/h \leq 1$. Therefore, the CFL number for the Lax-Friedrich scheme is $|c|k/h$.

Lax-Wendroff

Let's look at another way to improve the stability of the unstable centered-difference in space, forward-Euler scheme (3.4). Consider the Taylor series expansion about $u \equiv u(t_n, x_j)$:

$$\begin{aligned} U_j^n &= u \\ U_j^{n+1} &= u + ku_t + \frac{1}{2}k^2u_{tt} + O(k^3) \\ U_{j+1}^n &= u + hu_x + \frac{1}{2}h^2u_{xx} + \frac{1}{6}h^3u_{xxx} + O(h^4) \\ U_{j-1}^n &= u - hu_x + \frac{1}{2}h^2u_{xx} - \frac{1}{6}h^3u_{xxx} + O(h^4) \end{aligned}$$

Then the unstable centered-difference in space, forward-Euler in time scheme

$$\frac{U_j^{n+1} - U_j^n}{k} + c \frac{U_{j+1}^n - U_{j-1}^n}{2h} = 0$$

is consistent with

$$u_t + cu_x = -\frac{1}{2}ku_{tt} + O(k^2 + h^2) \quad (3.6)$$

Because $u_t + cu_x = 0$ (that is, $u_t = -cu_x$), we have that $u_{tt} = -cu_{tx} = c^2u_{xx}$. Therefore, (3.6) becomes

$$u_t + cu_x = -\frac{1}{2}c^2ku_{xx} + O(k^2 + h^3)$$

Note that $u_t = -\frac{1}{2}c^2ku_{xx}$ is a backward heat equation and therefore it is unstable. We can fix the scheme by counteracting $-\frac{1}{2}c^2ku_{xx}$ with $+\frac{1}{2}c^2ku_{xx}$. Not only will this improve stability, but it also improves the accuracy to $O(k^2 + h^2)$. That is, instead we can use

$$\frac{U_j^{n+1} - U_j^n}{k} + c \frac{U_{j+1}^n - U_{j-1}^n}{2h} = \frac{c^2}{2}k \frac{U_{j+1}^n - 2U_j^n + U_{j-1}^n}{h^2}.$$

The reason the original centered-difference in space, forward-Euler scheme failed was because the eigenvalues were along the imaginary axis and fell entirely outside of the region of stability of the forward-Euler scheme no matter how small k was. By adding a viscosity term u_{xx} to the problem, we push the eigenvalues into the left-half plane and by taking k sufficiently small, we can ensure stability.



Lax-Wendroff Method

 $O(k^2 + h^2)$

$$\frac{U_j^{n+1} - U_j^n}{k} + c \frac{U_{j+1}^n - U_{j-1}^n}{2h} = \frac{c^2}{2} k \frac{U_{j+1}^n - 2U_j^n + U_{j-1}^n}{h^2} \quad (3.7)$$

Note that by rearranging the terms of the Lax-Friedrichs scheme

$$\frac{U_j^{n+1} - U_j^n}{k} + c \frac{U_{j+1}^n - U_{j-1}^n}{2h} = \frac{h^2}{2k} \frac{U_{j+1}^n - 2U_j^n + U_{j-1}^n}{h^2}$$

we get a form which looks very similar to the Lax-Wendroff scheme. Like the Lax-Wendroff scheme, the Lax-Friedrichs scheme achieves stability by adding a viscosity (diffusion) term onto the (unstable) centered-difference scheme. Whereas the Lax-Wendroff scheme adds just enough viscosity ($c^2 k/2$) to counteract the unstable second-order term, the Lax-Friedrichs scheme goes overboard adds a bit more viscosity ($h^2/2k$). We can generalize the Lax-Wendroff and Lax-Friedrichs methods by using an arbitrary diffusion coefficient Q :

$$\frac{\partial}{\partial t} U_j = -c \frac{U_{j+1} - U_{j-1}}{2h} + Q \frac{U_{j+1} - 2U_j + U_{j-1}}{h^2}.$$

From earlier in this chapter and the previous chapter, we know that the Fourier transform of this equation is

$$\frac{\partial}{\partial t} \hat{u}(t, \xi) = -4 \frac{Q}{h^2} \sin^2 \frac{\xi h}{2} \hat{u}(t, \xi) - i \frac{c}{h} \sin \xi h \hat{u}(t, \xi)$$

with eigenvalues

$$-4 \frac{Q}{h^2} \sin^2 \frac{\xi h}{2} - i \frac{c}{h} \sin \xi h.$$

The eigenvalues lie along an ellipse in the negative half plane bounded by $-4Q/h^2$ along the real axis and $\pm c/h$ along the imaginary axis. For a forward Euler discretization this ellipse must be scaled by k to lie entirely within the unit circle centered at -1 . For the Lax-Friedrichs method ($Q = h^2/2k$) and the k -scaled ellipse is

$$-2 \sin^2 \frac{\xi h}{2} - i \frac{ck}{h} \sin \xi h.$$

The ellipse always goes out to -2 along the real axis no matter the value of k . So, the CFL condition for the Lax-Friedrichs method is $k < |c|/h$ (from the imaginary direction). For the Lax-Wendroff method ($Q = c^2/2k$) and the k -scaled ellipse is

$$-2 \left(\frac{ck}{h} \right)^2 \sin^2 \frac{\xi h}{2} - i \frac{ck}{h} \sin \xi h.$$

When $k < |c|/h$ the k -scaled ellipse is completely inside the unit circle and when $k > |c|/h$ the k -scaled ellipse is completely outside the unit circle. The CFL numbers for the Lax-Wendroff method and the Lax-Friedrichs method are both $|c|k/h$.

The remainder of this section looks at consistency and stability of the Lax-Wendroff method one more time as an alternative approach to what is discussed above. Let's rederive the Lax-Wendroff scheme from scratch. The Lax-Wendroff scheme achieves second-order accuracy by adding extra terms cancel out the u_{tt} terms of the Taylor expansion of $u(t, x)$ in time. From the linear problem $u_t + cu_x = 0$ we have that $u_t = -cu_x$. Therefore, $u_{tt} = -cu_{tx} = c^2 u_{xx}$. Then the Taylor series expansion of $u(t_{n+1}, x_j)$ about $u \equiv u(t_n, x_j)$ is

$$\begin{aligned} u(t_{n+1}, x_j) &= u + ku_t + \frac{k^2}{2}u_{tt} + O(k^3) \\ &= u - ck u_x + c^2 \frac{k^2}{2}u_{xx} + O(k^3). \end{aligned}$$

By using a second order discretization of ∂_x and ∂_{xx} , we get the scheme

$$\frac{U_j^{n+1} - U_j^n}{k} + c \frac{U_{j+1}^n - U_{j-1}^n}{2h} = \frac{c^2}{2} k \frac{U_{j+1}^n - 2U_j^n + U_{j-1}^n}{h^2}.$$

We can check the stability condition using von Neumann analysis

$$\frac{A-1}{k} + i \frac{c}{h} \sin \xi h = -2c^2 \frac{k}{h^2} \sin^2 \frac{\xi h}{2}$$

from which we have that (again taking $\lambda = ck/h$)

$$A = 1 - 2\lambda^2 \sin^2 \frac{\xi h}{2} - i\lambda \sin \xi h$$

Therefore, $|A| \leq 1$ if and only if $|\lambda| \leq 1$. So, the CFL number for the Lax-Wendroff scheme is $|c|k/h$.

3.3 Numerical diffusion and numerical dispersion

By examining the truncation error for these numerical schemes, we will get a more complete understanding of the nature of the numerical solution. Let's start by looking at the upwind scheme. From Taylor series expansion we find that the upwind scheme is a numerical approximation of

$$u_t + cu_x = \frac{1}{2}ch \left(1 - \frac{ck}{h}\right) u_{xx} + O(h^2) \quad (3.8)$$

That is, the upwind scheme is a first order approximation to

$$u_t + cu_x = 0 \quad (3.9)$$

but it is a second order approximation to

$$u_t + cu_x = \frac{1}{2}ch \left(1 - \frac{ck}{h}\right) u_{xx} \quad (3.10)$$

The right-hand side of (3.10) contributes **numerical viscosity** (also called numerical diffusion or numerical dissipation) to the computed solution. You can think of it this way: while the upwind scheme does a decent job of approximating the original problem (3.9), it does a better job approximating the parabolic equation (3.10). Note what happens if we take smaller step sizes in time (keeping the mesh size in space fixed). As $ck/h \rightarrow 0$, we get $u_t + cu_x = \frac{1}{2}chu_{xx}$ and the solution is overly dissipative. On the other hand, note that if we take k equal to the CFL number ($k = h/c$) then not only does the u_{xx} term of (3.8) disappear, but the whole right side disappears. That is, the upwind scheme exactly solves $u_t + cu_x = 0$ when $k = h/c$.

The Lax-Wendroff scheme is a second-order approximation to (3.9), but it is a third-order approximation to

$$u_t + cu_x = -\frac{1}{6}ch^2 \left(1 - \left(\frac{ck}{h}\right)^2\right) u_{xxx} \quad (3.11)$$

The right-hand side of (3.11) contributes **numerical dispersion** to the computed solution.

There is an important distinction between dissipation and dispersion. If we take the Fourier component

$$u(t, x) = e^{i(\omega t - \xi x)}$$

as an **ansatz** to the modified upwind equation (3.10) $u_t + cu_x = \alpha u_{xx}$, we have the relationship

$$i\omega e^{i(\omega t - \xi x)} + c(-i\xi)e^{i(\omega t - \xi x)} = \alpha(-\xi^2)e^{i(\omega t - \xi x)}$$

from which we have that

$$i\omega + c(-i\xi) = \alpha(-\xi^2).$$

From this equation, we can determine the **dispersion relation** $\omega \equiv \omega(\xi)$

$$\omega = c\xi + i\alpha\xi^2.$$

Plugging this ω back into our ansatz gives us

$$\begin{aligned} u(t, x) &= e^{i(\omega t - \xi x)} \\ &= e^{ic\xi t} e^{-\alpha\xi^2 t} e^{-i\xi x} \\ &= \underbrace{e^{i\xi(ct-x)}}_{\text{advection}} \cdot \underbrace{e^{-\alpha\xi^2 t}}_{\text{dissipation}} \end{aligned}$$

Using the same ansatz in the modified Lax-Wendroff equation (3.11) $u_t + cu_x = \beta u_{xxx}$ gives

$$i\omega e^{i(\omega t - \xi x)} + c(-i\xi)e^{i(\omega t - \xi x)} = \beta(-i\xi^3)e^{i(\omega t - \xi x)}$$

which we can simplify to derive the dispersion relation

$$\omega = a\xi + \beta\xi^3$$

For the Lax-Wendroff scheme we have

$$u = e^{i(c\xi + \beta\xi^3)t} e^{-i\xi x}.$$

The **group velocity** is given by the first derivative of the dispersion relationship

$$v_{\text{group}} = \omega'(\xi) = c + 3\beta\xi^2.$$

If the group velocity is a function of k , we say that the wave is **dispersive**. The **phase velocity** is given by the ratio of the dispersion relation and ξ :

$$v_{\text{phase}} = \frac{\omega}{\xi} = c + \beta\xi^2.$$

For a dispersive wave, waves of different wave numbers ξ travel with different velocities resulting in oscillations in the numerical solution. On the other hand, numerical viscosity damps out high wave numbers. If you recall Fourier analysis, high wave numbers are associated with large derivatives, i.e., discontinuities.

A dispersive scheme oscillates around a discontinuity.
A dissipative scheme smooths out the discontinuity.

3.4 Linear hyperbolic systems

Let's turn our attention to the one-dimensional linear wave equation

$$u_{tt} - c^2 u_{xx} = 0 \tag{3.12}$$

We can rewrite this equation as the system of partial differential equations by setting $u_t = v$ and $u_x = w$. Then

$$\begin{aligned} v_t - c^2 w_x &= 0 \\ w_t - v_x &= 0 \end{aligned}$$

which is equivalent to

$$\begin{bmatrix} v \\ w \end{bmatrix}_t + \begin{bmatrix} 0 & c^2 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix}_x = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

In general, one has

$$\mathbf{u}_t + \mathbf{A}\mathbf{u}_x = \mathbf{0} \quad \mathbf{u} \in \mathbb{R}^n \quad \mathbf{A} \in \mathbb{R}^{n \times n}.$$

The equation $\mathbf{u}_t + \mathbf{A}\mathbf{u}_x = \mathbf{0}$ with $\mathbf{u} \in \mathbb{R}^n$ and $A \in \mathbb{R}^{n \times n}$ is called **hyperbolic** if \mathbf{A} has n real eigenvalues and a complete set of linearly independent eigenvectors. In this case, the matrix \mathbf{A} can be diagonalized in real space:

$$\mathbf{T}^{-1}\mathbf{A}\mathbf{T} = \mathbf{\Lambda} = \text{diag}(\lambda_1, \dots, \lambda_n)$$

where \mathbf{T} is a matrix of eigenvectors $\mathbf{v}_1, \dots, \mathbf{v}_n$. That is, $\mathbf{A}\mathbf{v}_i = \lambda_i\mathbf{v}_i$. The eigenvalues λ_i are the **characteristic speeds**. For the wave equation (3.12)

$$\mathbf{A} = \begin{bmatrix} 0 & c^2 \\ -1 & 0 \end{bmatrix}$$

and the characteristic speeds are $\lambda_{\pm} = \pm c$.

Because we can diagonalize the matrix \mathbf{A} . The system can be completely decoupled into a system of independent advection equations. Let's look at this idea using the method of characteristics. Consider

$$\mathbf{u}_t + \mathbf{A}\mathbf{u}_x = \mathbf{0}.$$

Multiply the equation by \mathbf{T}^{-1}

$$\mathbf{T}^{-1}\mathbf{u}_t + \mathbf{T}^{-1}\mathbf{A}\mathbf{u}_x = \mathbf{0}$$

which is the same as

$$\mathbf{T}^{-1}\mathbf{u}_t + \mathbf{T}^{-1}\mathbf{A}\mathbf{T}\mathbf{T}^{-1}\mathbf{u}_x = \mathbf{0}$$

which is simply

$$\mathbf{T}^{-1}\mathbf{u}_t + \mathbf{\Lambda}\mathbf{T}^{-1}\mathbf{u}_x = \mathbf{0}$$

By letting $\mathbf{v} = \mathbf{T}^{-1}\mathbf{u}$ we have the equivalent system

$$\mathbf{v}_t + \mathbf{\Lambda}\mathbf{v}_x = \mathbf{0}$$

in which each term uncouples

$$\partial_t v_i + \lambda_i \partial_x v_i = 0 \quad i = 1, \dots, n. \quad (3.13)$$

We solved the advection equation earlier. The solution is

$$v_i(t, x) = v_i(0, x - \lambda_i t)$$

The initial conditions are given by $\mathbf{v}_0 = \mathbf{T}^{-1}\mathbf{u}_i$. By setting $\mathbf{u} = \mathbf{T}\mathbf{v}$, the problem is solved.

Now, let's solve the problem numerically. We can discretize the uncoupled system (3.13) using, for example, the upwind scheme

$$\frac{(V_i)_j^{n+1} - (V_i)_j^n}{K} + \frac{\lambda_i + |\lambda_i|}{2} \frac{(V_i)_j^n - (V_i)_{j-1}^n}{h} + \frac{\lambda_i - |\lambda_i|}{2} \frac{(V_i)_{j+1}^n - (V_i)_j^n}{h} = 0 \quad (3.14)$$

If we let

$$\Lambda^{\pm} = \begin{bmatrix} \ddots & & \\ & \frac{1}{2}(\lambda_i \pm |\lambda_i|) & \\ & & \ddots \end{bmatrix}$$

then scheme (3.14) is the same as

$$\frac{\mathbf{V}_j^{n+1} - \mathbf{V}_j^n}{k} + \Lambda^+ \frac{\mathbf{V}_j^n - \mathbf{V}_{j-1}^n}{h} + \Lambda^- \frac{\mathbf{V}_{j+1}^n - \mathbf{V}_j^n}{h} = 0.$$

We can rewrite this equation as

$$\mathbf{T}^{-1} \frac{\mathbf{U}_j^{n+1} - \mathbf{U}_j^n}{k} + \Lambda^+ \mathbf{T}^{-1} \frac{\mathbf{U}_j^n - \mathbf{U}_{j-1}^n}{h} + \Lambda^- \mathbf{T}^{-1} \frac{\mathbf{U}_{j+1}^n - \mathbf{U}_j^n}{h} = 0$$

which is equivalent to

$$\frac{\mathbf{U}_j^{n+1} - \mathbf{U}_j^n}{k} + \mathbf{T}\mathbf{\Lambda}^+\mathbf{T}^{-1}\frac{\mathbf{U}_j^n - \mathbf{U}_{j-1}^n}{h} + \mathbf{T}\mathbf{\Lambda}^-\mathbf{T}^{-1}\frac{\mathbf{U}_{j+1}^n - \mathbf{U}_j^n}{h} = 0$$

which is equivalent to

$$\frac{\mathbf{U}_j^{n+1} - \mathbf{U}_j^n}{k} + \mathbf{A}^+\frac{\mathbf{U}_j^n - \mathbf{U}_{j-1}^n}{h} + \mathbf{A}^-\frac{\mathbf{U}_{j+1}^n - \mathbf{U}_j^n}{h} = 0$$

where $\mathbf{A}^\pm = \mathbf{T}\mathbf{\Lambda}^\pm\mathbf{T}^{-1}$ is called the **characteristic decomposition**.

3.5 Nonlinear hyperbolic equations

In this section, we will extend the ideas previously discussed for the prototypic linear hyperbolic equation (3.1) to the nonlinear hyperbolic equation. We start with

$$\frac{\partial}{\partial t}u + \frac{\partial}{\partial x}f(u) = 0 \quad (3.15)$$

where $u(t, x)$ is defined along the real line. This problem is also called a **conservation law** and the function $f(x)$ is called the **flux**. If we integrate (3.15) in x , we have that

$$\frac{d}{dt} \int_{\Omega} u \, dx + \int_{\Omega} \frac{\partial}{\partial x} f(u) \, dx = 0$$

If there is no flux through the boundary, $\frac{d}{dt} \int_{\Omega} u \, dx = 0$ and hence $\int_{\Omega} u \, dx$ is constant. So, u is a conserved quantity.

Recall the solution to (3.1) that we obtained using the method of characteristics

$$u(t, x) = u_0(x - ct).$$

If u_0 is initially smooth, $u(t, x)$ will always remain smooth. This is because all information is merely advected by a constant velocity. This is not necessarily true for solutions of nonlinear hyperbolic equations.

Example. Describe the time evolution of the inviscid Burgers' equation

$$u_t + \left(\frac{u^2}{2} \right)_x = 0 \quad (3.16a)$$

$$u(x, 0) = u_0(x) \quad (3.16b)$$

As long as $u(t, x)$ remains smooth in x , we can use the chain rule to rewrite (3.16a) as

$$u_t + uu_x = 0. \quad (3.17)$$

In this equation u now takes the place of the wave speed c that we had in the linear hyperbolic equation. Heuristically, it seems that now when u is larger, the solution $u(t, x)$ moves faster. The higher part of the wave will move faster than the lower part of the

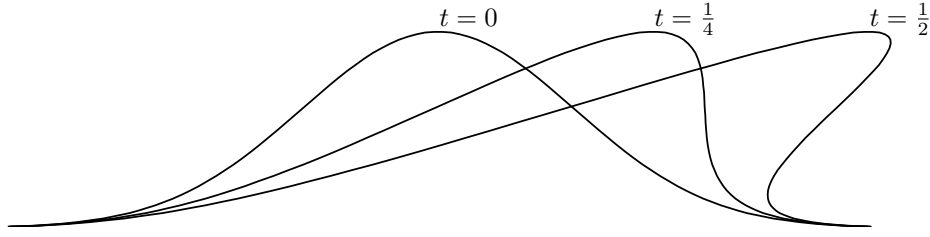


Figure 3.3: Solution to Burgers' equation

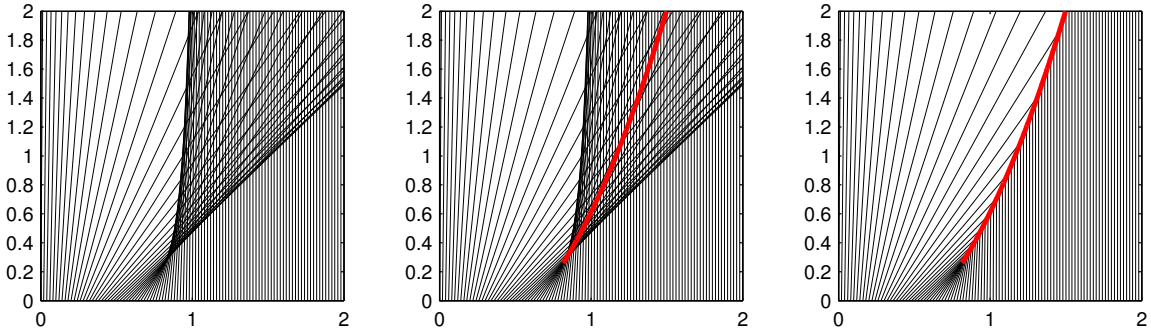


Figure 3.4: Characteristic curves of Burgers' equation (left) and weak solution (right).

wave and eventually catch up with it or possibly overtake it, say some critical time t_b . At this time the first derivative u_x is infinite and u develops a discontinuity or a **shock**. Can the faster part really overtake the slower moving part? No. We relied on the fact that $u(t, x)$ was smooth in order to apply the chain rule to take us from (3.16a) to (3.17). When $(f(u))_x = \infty$, this argument is clearly invalidated.

Just as we did for the linear hyperbolic equation (3.1), we can construct an analytic solution to the Burgers' equation (3.17) using the method of lines. We consider the ansatz that initial information is advected (and not changed) along characteristic curves $x(t)$, i.e.,

$$0 = \frac{d}{dt}u(t, x(t)) = u_t + \frac{dx}{dt}u_x$$

Comparing this equation with our problem $u_t + uu_x = 0$, we see that by defining $dx/dt = u$, u will be constant along the characteristics $x(t)$:

$$\frac{dx}{dt} = u(t, x(t)) = u(0, x(0)) = u_0(x_0)$$

Integrating this equation, we see that the characteristics are straight lines

$$x = u_0(x_0)t + x_0.$$

Each initial position x_0 has its own characteristic, but characteristics starting at different points may intersect. See Figure 3.4. Since the information merely travels along the

characteristics, u may have two (or more) values at a point of intersection. We can determine when this happens by finding the point when solution overturns, i.e., the derivative du/dx is infinite.

$$\frac{du}{dx} = \frac{du}{dx_0} \frac{dx_0}{dx} = u'_0(x_0) \frac{1}{\frac{dx}{dx_0}} = \frac{u'_0(x)}{u'_0(x)t + 1}. \quad (3.18)$$

Since we want to find out when du/dx blows up, we set the right-hand side of (3.18) to ∞ and solve for t . In this case, we have that

$$t = -\frac{1}{u'_0(x_0)}$$

The time that the first shock develops will be $t_b = -1/\min_{x_0} u'_0(x_0)$. Note that shocks develop only if there is at some point the gradient of the initial condition is negative, i.e., $u'_0(x) < 0$ for some x .

Weak solutions

As stated above, when the shock develops $u_t + uu_x = 0$ is no longer valid and the classical solution no longer exists. Instead, we need to find a more *physical* equation to use. Mathematically, we will extend the solution by a “generalized weak solution.” To construct the “weak solution” of

$$u_t + f(u)_x = 0 : \quad (3.19)$$

1. Multiply (3.19) by a sufficiently smooth **test function** φ that vanishes at the boundaries;
2. Integrate over space (and time); and
3. Use integration by parts to pass the derivatives over to φ . Since φ vanishes at the boundaries, there will be no boundary terms.

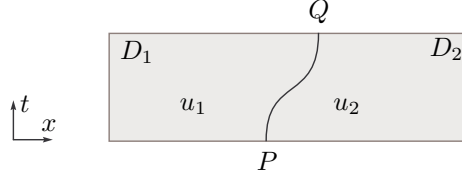
Suppose that we have a test function φ , which is differentiable and has **compact support**. Compact support means that the closure of the set over which φ is nonzero is bounded. Notably, φ is zero at the boundaries where we need it to be zero. We denote this by $\varphi = C_0^1(\Omega)$. The C^1 denotes that the function and its first derivative are continuous, and the subscript 0 denotes that the function has compact support over the region D . In this case, (3.19) implies

$$\int_0^\infty \int_\Omega (\varphi u_t + \varphi f(u)_x) dx dt = 0.$$

We integrate by parts to get

$$\int_0^\infty \int_{x_L}^{x_R} (\varphi_t u + \varphi_x f(u)) dx dt = 0. \quad (3.20)$$

If (3.20) is satisfied for all $\varphi \in C_0^1(\Omega)$, then we say that $u(t, x)$ is a **weak solution** of $u_t + f(u)_x = 0$. If $u(t, x)$ is smooth, the weak solution is the classical (strong) solution.

Figure 3.5: Region $D \in \{(x, t)\} \subset \mathbb{R}^2$.

Consider a domain $D \in \{(x, t)\} \subset \mathbb{R}^2$ and an arbitrary curve PQ dividing into D_1 and D_2 ($D = D_1 \cup D_2$). See Figure Figure 3.5. Suppose the $u(t, x)$ is a weak solution to $u_t + f(u)_x = 0$. Then (because φ vanishes on the boundary)

$$\begin{aligned}
 0 &= - \iint_D \varphi u_t + \varphi f(u)_x dx dt + \iint_D \varphi u_t + \varphi f(u)_x dx dt \\
 &= \iint_D \varphi_t u + \varphi_x f(u) dx dt + \iint_D \varphi u_t + \varphi f(u)_x dx dt \\
 &= \iint_D (\varphi u)_t + (\varphi f(u))_x dx dt \\
 &= \iint_{D_1} (\varphi u)_t + (\varphi f(u))_x dx dt + \iint_{D_2} (\varphi u)_t + (\varphi f(u))_x dx dt \\
 &= \int_{\partial D_1} -\varphi u dx + \varphi f(u) dt + \int_{\partial D_2} -\varphi u dx + \varphi f(u) dt \quad [\text{By Green's Theorem}] \\
 &= \int_P^Q -\varphi u_L dx + \varphi f(u_L) dt + \int_Q^P -\varphi u_R dx + \varphi f(u_R) dt \quad [\varphi(\partial D) = 0] \\
 &= \int_P^Q \varphi (u_R - u_L) dx - \varphi (f(u_R) - f(u_L)) dt
 \end{aligned}$$

So, for all test functions $\varphi \in C_0^1(D)$, $(u_R - u_L)dx - (f(u_R) - f(u_L))dt = 0$ along PQ . Therefore,

$$s = \frac{dx}{dt} = \frac{f(u_R) - f(u_L)}{u_R - u_L}.$$

Note that in the limit as $u_L \rightarrow u_R$, $s \rightarrow f'(u)$. This is called the **Rankine-Hugoniot jump condition** for shocks and s is called the shock speed. To simplify notation, one often uses brackets to indicate difference:

$$s = \frac{f(u_R) - f(u_L)}{u_R - u_L} \equiv \frac{[f]}{[s]}.$$

We can now continue to use the PDE with the weak solution.

A **Riemann problem** is a PDE for which the initial condition is given by a two constant states. Consider the solution to the inviscid Burgers' equation with the initial conditions

$$u_t + \left(\frac{u^2}{2}\right)_x = 0 \quad \text{with} \quad u(x, 0) = \begin{cases} u_L & x < 0 \\ u_R & x > 0 \end{cases} \quad (3.21)$$

Let's find the solution. If we make a change of variable by scaling t and x by a positive constant c

$$\tilde{t} = t/c, \quad \tilde{x} = x/c$$

then the problem becomes

$$\partial_{\tilde{t}} u + \partial_{\tilde{x}} f(u) = 0$$

The initial data remains unchanged, so

$$u(x, t) \equiv u\left(\frac{x}{t}\right).$$

Let $\xi = x/t$. We will find a **self-similar solution** $u(x, t) = u(\xi)$:

$$\begin{aligned} u_t + \left(\frac{u^2}{2}\right)_x &= 0 \\ \xi_t u' + \xi_x u u' &= 0 \\ \left(-\frac{x}{t^2}\right) u' + \left(\frac{1}{t}\right) u u' &= 0 \\ -\xi u' + u u' &= 0 \\ u'(u - \xi) &= 0 \end{aligned}$$

So either $u' = 0$ in which cases u is constant, or $u = \xi$. Cases:

1. $u_L > u_R$. Then

$$s = \frac{\frac{u_R^2}{2} - \frac{u_L^2}{2}}{u_R - u_L} = \frac{1}{2}(u_R + u_L)$$

and the exact solution is

$$u(x, t) = \begin{cases} u_L, & \text{if } x/t < s \\ u_R, & \text{if } x/t > s \end{cases}$$

2. $u_L < u_R$. Then

$$s = \frac{1}{2}(u_R + u_L)$$

One exact solution is

$$u(x, t) = \begin{cases} u_L, & \text{if } x/t < s \\ u_R, & \text{if } x/t > s \end{cases}$$

Another solution is

$$u(x, t) = \begin{cases} u_L, & \text{if } x/t < u_L \\ x/t, & \text{if } u_L < x/t < u_R \\ u_R, & \text{if } x/t > u_R \end{cases}$$

A third solution is

$$u(x, t) = \begin{cases} u_L, & \text{if } x/t < u_L \\ u_m, & \text{if } s < x/t < u_m \\ x/t, & \text{if } u_m < x/t < u_R \\ u_R, & \text{if } x/t > u_R \end{cases}$$



Figure 3.6: Possible solutions to the Riemann problem (3.21)

In fact, there are *infinitely* many weak solutions. What is the *physically* relevant weak solution? To find it we need an entropy condition. A solution satisfies the **Lax entropy condition** if $f'(u_L) > s > f'(u_R)$ where $f(u)$ is convex. That is,

$$\text{Lax entropy condition: } f'(u_L) > \frac{f(u_R) - f(u_L)}{u_R - u_L} > f'(u_R)$$

If we consider the problem $u_t + f(u)_x = 0$, then $f'(u)$ gives the characteristic speed and s gives the shock speed. The characteristics from both sides of the shock converge into the shock.

Let's reexamine the example. Recall that $f'(u) = u$. The two cases are

1. $u_L > u_R$.

$$f'(u_L) = u_L > \frac{1}{2}(u_R + u_L) > u_R = f'(u_R)$$

Therefore, the solution is an entropy shock.

2. $u_L < u_R$. The shock does not satisfy the entropy condition. So, the only possible weak solution is the (continuous) rarefaction wave.

Physically, there is a quantity called “entropy” which is a constant along smooth particle trajectories but jumps to a higher value across a discontinuity. The mathematical definition of entropy $\varphi(u)$ is the negative physical entropy. A convex function $\varphi(u)$ (that is, $\varphi''(u) \geq 0$) is called the **entropy** for $u_t + f(u)_x = 0$ if there exists a $\psi(u)$, called the **entropy flux** such that $\varphi'(u)f'(u) = \psi'(u)$. If u is smooth, then $\varphi'(u)u_t + \varphi'f'(u)u_x = 0$ from which we have that $\frac{\partial}{\partial t}\varphi(u) + \frac{\partial}{\partial x}\psi(u) = 0$. If u is not smooth, then the entropy condition $\frac{d}{dt}\phi = \frac{\partial}{\partial t}\varphi(u) + \frac{\partial}{\partial x}\psi(u) \leq 0$.

Example. Remark on conservation form The conservation form is important for weak solutions. For example, consider the following PDE in conservation form

$$u_t + \left(\frac{u^2}{2}\right)_x = 0 \tag{3.22}$$

The shock speed is given by

$$s = \frac{1}{2}(u_L + u_R).$$

Now multiplying (3.22) by u we have that

$$uu_t + u \left(\frac{u^2}{2}\right)_x = 0$$

from which it follows that (if the solution is continuous)

$$\left(\frac{u^2}{2}\right)_t + \left(\frac{u^3}{3}\right)_x = 0.$$

The shock speed is now

$$s = \frac{\frac{u_R^3}{3} - \frac{u_L^3}{3}}{\frac{u_R^3}{2} - \frac{u_L^3}{2}} = \frac{2}{3} \frac{u_R^2 + u_R u_L + u_L^2}{u_R + u_L}.$$

Example. We can model traffic using a nonlinear hyperbolic equation. Suppose that we have a single-lane road without any on or off ramps (no sources or sinks). Let $\rho(x, t)$ be the density of cars measured in cars per car length. So, $\rho = 0$ says the road is completely empty and $\rho = 1$ says that bumper-to-bumper traffic. Then

$$\frac{\partial}{\partial t} \rho + \frac{\partial}{\partial x} f(\rho) = 0$$

models the traffic flow where the flux $f(\rho)$ tells us the density of cars passing a given point in a given time interval (number of cars per second). To model the flux function $f(\rho) = U(\rho)\rho$ where $U(\rho)$ simply tells us the speed that each car is traveling. Let's make a couple reasonable assumptions to model $U(\rho)$:

1. Everyone travels as fast as they can while still obeying the posted speed limit.
2. Everyone keeps a safe distance behind the car ahead and adjusts their speed accordingly. If there are no other cars on the road, then drive as fast as possible and stop moving if traffic is bumper-to-bumper.

In this case, the simplest model for $U(\rho)$ is

$$U(\rho) = u_{\max}(1 - \rho)$$

Notice the that flux

$$f(\rho) = U(\rho)\rho = u_{\max}(\rho - \rho^2)$$

is zero when the roads are completely empty (there are no cars to move) and when the roads are completely full (no one is moving). The flux $f(\rho)$ is maximum at

$$f'(\rho) = u_{\max}(1 - 2\rho) = 0$$

when $\rho = \frac{1}{2}$ —the roads are half full (with the cars traveling at half-the posted speed limit). So, our traffic equation is now

$$\frac{\partial}{\partial t} \rho + \frac{\partial}{\partial x} (u_{\max}(1 - \rho)\rho) = 0$$

This equation is very similar in form to the Burgers equation.

Let's consider two simple examples: cars approaching an red stop light and cars leaving after the stop light turns green. Both of these examples are Riemann problems.

Red Light We imagine several cars traveling at $\frac{1}{2}u_{\max}$ approaching a queue of stopped cars. In this case we have

$$\rho(x, 0) = \begin{cases} \rho_L, & x < 0 \\ 1, & x \geq 0 \end{cases}$$

The Rankine-Hugoniot relationship gives

$$s = \frac{f(\rho_L) - f(1)}{\rho_L - 1} = \frac{u_{\max}\rho_L(1 - \rho_L) - 0}{\rho_L - 1} = -u_{\max}\rho_L$$

There is a shockwave moving backwards as the cars line up in queue.

Green Light We imagine several cars in queue with $\rho_L = 1$ and speed 0. This time the Lax entropy condition tells us that there is a rarefaction wave. The leading car moves forwards at a speed of u_{\max} and a rarefaction wave moves backwards at a speed $f'(\rho) = u_{\max}(1 - 2\rho)$ with $\rho = 1$. That is, speed $f'(1) = -u_{\max}$.

◦

3.6 Hyperbolic systems of conservations laws

Let's extend the discussion to a system of equations

$$\frac{\partial}{\partial t} \mathbf{u} + \frac{\partial}{\partial x} \mathbf{F}(\mathbf{u}) = 0 \quad \text{with } \mathbf{u} \in \mathbb{R}^n. \quad (3.23)$$

If the Jacobian $\mathbf{F}'(\mathbf{u})$ has n real eigenvalues and a complete set of linearly independent eigenvectors then the system (3.23) is called **hyperbolic**. In this case, there is an invertible map $\mathbf{T}(\mathbf{u})$ such that

$$\mathbf{T}^{-1} \mathbf{F}' \mathbf{T} = \mathbf{\Lambda} = \text{diag}(\lambda_1, \dots, \lambda_n).$$

The λ_i are called the characteristic speeds and $\mathbf{T} = (\xi_1, \dots, \xi_n)$ are the eigenvectors. But nonlinear systems cannot be diagonalized globally!

Example. Let's examine the one-dimensional shallow water equations and show that they form a hyperbolic system. The shallow water equations are given by

$$h_t + (hu)_x = 0 \quad (3.24a)$$

$$(hu)_t + (hu^2 + \frac{1}{2}gh^2)_x = 0 \quad (3.24b)$$

where g is the gravitational acceleration, h is the height of the water, and u is the velocity of the water. Let $m = hu$ denote the momentum. Then (3.24) becomes

$$\begin{aligned} h_t + m_x &= 0 \\ m_t + \frac{2m}{h}m_x - \frac{m^2}{h^2}h_x + gh h_x &= 0 \end{aligned}$$

which can be written as

$$\begin{bmatrix} h \\ m \end{bmatrix}_t + \underbrace{\begin{bmatrix} 0 & 1 \\ gh - u^2 & 2u \end{bmatrix}}_{\mathbf{F}'} \begin{bmatrix} h \\ m \end{bmatrix}_x = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

The eigenvalues of the Jacobian matrix \mathbf{F}' are given by

$$\begin{vmatrix} -\lambda & 1 \\ gh - u^2 & 2u - \lambda \end{vmatrix} = \lambda^2 - 2u\lambda + u^2 - gh.$$

So $\lambda_{\pm} = \frac{1}{2} \left(2u \pm \sqrt{4u^2 - 4u^2 + 4gh} \right) = u \pm \sqrt{gh}$. If the height of the water h is positive, then there are two real eigenvalues and the system is hyperbolic.

The two eigenvalues give us the characteristic speeds for the shallow water equation. Notably the speed of water waves are functions of the depth of the water h (and the velocity of the current u). Tsunamis, for example, occur when a disturbance creates a wave in the ocean with wavelength of 100 km or more and a displacement of a meter or less. The wave travels quickly in the ocean where the depth may be several kilometers (speed $= \pm\sqrt{gh}$). Once it reaches the continental shelf, where the depth decreases dramatically, the wave slows considerably and the height of the wave grows. A shock forms as the faster characteristics intersect with the slower characteristics.

Refraction of water waves is also caused by the \sqrt{gh} speed dependence. Ocean waves turn to hit the beach perpendicularly even when the wind is not blowing straight into shore. Waves are produced by a Kelvin-Helmholtz instability resulting from the wind blowing over water. In the deeper water, waves are driven in the direction of the wind. When they reach shallow water, the wave moves at different speeds. The shallower side moves slower than the deeper side, curving the wave into shore. \circ

Riemann Invariants

As we saw at the beginning of the chapter information is advected along the characteristics. The quantity that is constant along a characteristic curve is called the **Riemann invariant**. For the advection equation $u_t + cu_x = 0$, the characteristics are given by $dx/dt = c$ and the Riemann invariant is u . For a general scalar conservation law $u_t + f(u)_x = 0$, we have that $u_t + f'(u)u_x = 0$ when u is differentiable. The characteristics are given by $dx/dt = f'(u)$ and it follows that

$$\frac{d}{dt}u(x(t), t) = \frac{\partial u}{\partial t} + \frac{dx}{dt} \frac{\partial u}{\partial x} = \frac{\partial u}{\partial t} + f'(u) \frac{\partial u}{\partial x} = 0.$$

So, $u(x, t)$ is the Riemann invariant. For systems of conservation laws, it's a little more complicated. Let's once again consider the system

$$\frac{\partial}{\partial t} \mathbf{u} + \frac{\partial}{\partial t} F(\mathbf{u}) = 0 \quad \text{with } \mathbf{u} \in \mathbb{R}^n.$$

If \mathbf{u} is differentiable,

$$\frac{\partial}{\partial t} \mathbf{u} + \mathbf{F}'(\mathbf{u}) \frac{\partial}{\partial x} \mathbf{u} = 0 \tag{3.26}$$

where $\mathbf{F}'(\mathbf{u})$ is the Jacobian matrix of \mathbf{F} with real eigenvalues $\{\lambda_1, \dots, \lambda_n\}$. Let ∇w_i —with $w \equiv w(\mathbf{u})$ —be the **left eigenvector** of $\mathbf{F}'(\mathbf{u})$ corresponding to the eigenvalues λ_i . Then left multiplying (3.26) by ∇w_i

$$\nabla w_i \frac{\partial}{\partial t} \mathbf{u} + \nabla w_i \mathbf{F}'(\mathbf{u}) \frac{\partial}{\partial x} \mathbf{u} = 0.$$

Because ∇w_i is the left eigenvector of $\mathbf{F}'(\mathbf{u})$, it follows that

$$\nabla w_i \frac{\partial}{\partial t} \mathbf{u} + \lambda_i \nabla w_i \frac{\partial}{\partial x} \mathbf{u} = 0.$$

And by the chain rule, we have

$$\frac{\partial}{\partial t} w_i + \lambda_i \frac{\partial}{\partial x} w_i = 0 \quad (3.27)$$

which is simply the scalar advection equation. So, w_i is a Riemann invariant, because it is constant along the characteristic curve given by $dx/dt = \lambda_i$. Notice that (3.27) is simply the diagonalization of the original system.

Example. Let's compute the Riemann invariants for the shallow water equations. Recall that

$$\mathbf{F}' = \begin{bmatrix} 0 & 1 \\ gh - u^2 & 2u \end{bmatrix}$$

with eigenvalues $\lambda_{\pm} = u \pm \sqrt{gh}$. We compute the left eigenvectors to be $(u \mp \sqrt{gh}, 1)$. Recall that we defined $\mathbf{u} = (h, m)$ with $m = hu$. Putting the eigenvectors in terms of h and m , we have

$$\left(\frac{m}{h} \mp \sqrt{gh}, 1 \right).$$

Finally, we need this vector to be an exact differential, so let's rescale this vector by $1/h$ —it will still be an eigenvector. We now have

$$\nabla_{\mathbf{u}} w_{\pm} = \left(\frac{m}{h^2} \mp \sqrt{\frac{g}{h}}, \frac{1}{h} \right).$$

So,

$$w_{\pm}(h, m) = \frac{m}{h} \mp 2\sqrt{gh} = u \mp 2\sqrt{gh}.$$

The Riemann invariants $u \mp 2\sqrt{gh}$ are constant along the characteristics $dx/dt = u \pm \sqrt{gh}$.
◦

Open boundary conditions

The advection equation $u_t + cu_x = 0$ has one space derivative and therefore it requires one influx boundary condition for uniqueness. Recall that the heat equation, which has two derivatives in space, requires two boundary conditions. Because the hyperbolic system of conservation laws (3.23) has n equations, we will need n boundary conditions.

Often we want to solve the shallow water equations or the Euler equations of gas dynamics using open boundary conditions. For example, in the ocean we may want to prescribe inflow boundary conditions to model incoming water waves. Or we may want to model airflow over an airfoil by injecting a flow upwind of the airfoil. Downwind of the airfoil, we want the gas to escape the domain without reflecting back into our simulation. Or maybe we may want to model the bomb blast in the semi-infinite domain above the ground. Without an open boundary, information will likely be reflected back into our domain of interest polluting our solution. It is typically a bad idea to force the boundary to be equal to the external data, because the external data does not perfectly match the information leaving the domain.

One simple solution is to use a larger domain and stop the simulation before the reflected information pollutes our problem. We use a thinner border by making it into a sponge layer with added viscosity to absorb the energy. Other approaches include creating an artificial boundary with tries to match the reflected waves with waves of opposite amplitude. Such non-reflecting boundary conditions were developed by Enquist and Majda in 1977. In this section, we'll look at a characteristic based approach to implement open boundary conditions.

Recall that the Riemann invariant is constant along a characteristics—as long as characteristics do not intersect. For the shallow water equation we had a right moving characteristic with slope $u + \sqrt{gh}$ and left characteristic with slope $u - \sqrt{gh}$. At the boundary, we will prescribe the Riemann invariant along the characteristic flowing into the domain. Suppose that we want an influx boundary condition on the right and an open boundary on the left. In this case,

Take the boundaries at $x = x_L$ and $x = x_R$ and suppose that the external information is given by $h(x_L, t) = h_L$ and $u(x_L, t) = u_L$ on the left and $h(x_R, t) = h_R$ and $u(x_R, t) = u_R$ on the right. We will match the Riemann invariants on the left by taking $w_-(h, u)$ to match the information outside entering our domain and $w_+(h, u)$ to match the solution inside our domain. Conversely, we will match the Riemann invariants on the right by taking $w_+(h, u)$ to match the solution inside our domain and $w_-(h, u)$ to match the information outside our domain. The Riemann invariants are

$$w_+ = u - 2\sqrt{gh} \quad \text{and} \quad w_- = u + 2\sqrt{gh},$$

so we have

$$\begin{aligned} \text{Left side: } & \begin{cases} w_- = \text{extrapolation of } u + 2\sqrt{gh} \\ w_+ = u_L - 2\sqrt{gh_L} \end{cases} \\ \text{Right side: } & \begin{cases} w_- = u_R + 2\sqrt{gh_R} \\ w_+ = \text{extrapolation of } u - 2\sqrt{gh} \end{cases} \end{aligned}$$

Solving this system of equations for u and h gives us the boundary conditions for our problem. Then solving for u and h gives us

$$\begin{aligned} h &= (w_- - w_+)/16g \\ u &= (w_- + w_+)/2. \end{aligned}$$

For example, consider the left boundary at x_0 . To determine w_- with second order accuracy we extrapolate

$$w_- = (2U_1 - U_2) + 2\sqrt{g(2H_1 - H_2)}.$$

and we set w_+ according to the external information u_L and h_L

$$w_+ = u_L + 2\sqrt{gh_L}.$$

If $|u| > \sqrt{gh}$, then both characteristics travel in the same direction and we match both Riemann invariants on the influx boundary and extrapolate both Riemann invariants on the outflux side.

The Riemann problem

First-order Godonov schemes approximate the solution by a series of Riemann problems—one for every mesh point in space. These Riemann problems are solved analytically and the resulting problems are numerically remeshed. The solution to the Riemann problem is good as long as characteristics don't cross—which happens to determine CFL condition.

The Riemann problem for a system of hyperbolic equations

$$\mathbf{u}_t + F(\mathbf{u})_x = 0 \quad \text{with} \quad \mathbf{u}(x, 0) = \begin{cases} \mathbf{u}_L, & x < 0 \\ \mathbf{u}_R, & x > 0 \end{cases}$$

can be solved analytically using a self-similar solution by taking $\mathbf{u}(x, t) = \mathbf{u}(x/t) = \mathbf{u}(\xi)$. Then

$$\left(-\frac{x}{t^2}\right) \mathbf{u}' + \frac{1}{t} F'(\mathbf{u}) \mathbf{u}' = 0$$

from which we have that

$$(F'(\mathbf{u}) - \xi I) \mathbf{u}' = 0$$

So either $\mathbf{u}' = 0$ in which case \mathbf{u} is constant in ξ or $\det(F'(\mathbf{u}) - \xi I) = 0$ in which case ξ is an eigenvalue of $F'(\mathbf{u})$. The solution is piecewise constant connected by shocks, rarefactions ($\det(F'(\mathbf{u}) - \xi I) = 0$) or contact discontinuity corresponding to eigenvalue ξ . A contact discontinuity is a linear discontinuity that travels with speed ξ .

3.7 Numerical methods for nonlinear hyperbolic conservation laws

Consider the Riemann problem

$$u_t + \left(\frac{u^2}{2}\right)_x = 0 \quad \text{with} \quad u(x, 0) = \begin{cases} 1 & x < 0 \\ 0 & x > 0 \end{cases}.$$

The shock speed is $s = [f(u)]/[u] = \frac{1}{2}$, so the exact solution is given by

$$u(x, 0) = \begin{cases} 1 & x < t/2 \\ 0 & x > t/2 \end{cases}.$$

The upwind scheme for

$$u_t + uu_x = 0$$

is

$$\frac{U_j^{n+1} - U_j^n}{k} + U_j^n \frac{U_j^n - U_{j-1}^n}{h} = 0 \quad (3.28)$$

with $x_j = jh$. For $j \leq 0$, $U_j^1 = U_j^0$ and for $j > 0$, $U_j^1 = U_j^0$. Therefore, $U_j^1 = U_j^0$ for all j , and hence $U_j^n = U_j^0$ for all n . This says that the numerical shock speed is identically zero. Although (3.28) is a reasonable scheme, it gives the *wrong* solution. The error is that $u_t + uu_x = 0$ is not in conservation form.

Theorem 4 (Lax-Wendroff Theorem). *If a consistent, conservative discretization of*

$$u_t + F(u)_x = 0 \quad (3.29)$$

converges, then it converges to a weak solution of (3.29).

A scheme is **conservative**, if it can be written as

$$\partial_t U_j + \frac{F_{j+1/2} - F_{j-1/2}}{h} = 0$$

where the numerical flux $F_{j+1/2} = F(u_{j-p}, \dots, u_{j+p})$. Then

$$\partial_t \sum_j U_j + \frac{1}{h} \sum_j (F_{j+1/2} - F_{j-1/2}) = 0$$

then

$$\partial_t \sum_j U_j = 0$$

if there is no flux through the boundaries. So, $\sum_j U_j$ is constant in time.

Example. The Lax-Friedrichs scheme is conservative

$$\frac{U_j^{n+1} - \frac{1}{2}(U_{j+1}^n + U_{j-1}^n)}{k} + \frac{f(U_{j+1}^n) - f(U_{j-1}^n)}{2h} = 0$$

can be rewritten as

$$\frac{U_j^{n+1} - U_j^n}{k} + \frac{U_j^n - \frac{1}{2}(U_{j+1}^n + U_{j-1}^n)}{k} + \frac{f(U_{j+1}^n) - f(U_{j-1}^n)}{2h} = 0.$$

So, we have

$$\frac{U_j^{n+1} - U_j^n}{k} + \frac{F_{j+1/2}^n - F_{j-1/2}^n}{h} = 0$$

where the numerical flux

$$F_{j+1/2}^n = \frac{f(U_{j+1}^n) + f(U_j^n)}{2} - \frac{h}{2k}(U_{j+1}^n - U_j^n).$$

Note that the Lax-Friedrichs scheme can also be written as

$$\underbrace{\frac{U_j^{n+1} - U_j^n}{k} + \frac{f(U_{j+1}^n) - f(U_{j-1}^n)}{2h}}_{\text{center difference}} = \underbrace{\frac{h^2}{2k}}_{O(h)} \underbrace{\frac{U_{j+1}^n - 2U_j^n + U_{j-1}^n}{h^2}}_{\text{numerical viscosity}}$$

We can improve the scheme a little bit by changing the numerical flux. The Local Lax-Friedrichs method defines the numerical flux by

$$F_{j+1/2} = \frac{f(U_{j+1}) + f(U_j)}{2} - \frac{\sqrt{a_j}}{2}(U_{j+1} - U_j)$$

where

$$\sqrt{a_j} = \sup_{U_j < u < U_{j+1}} |f'(u)|$$

and $\sqrt{a_j} \leq h/k$ by the CFL condition.

Finite volume methods

Rather than solving the problem using Taylor series expansion in the form of a finite difference scheme we can use an integral formulation in called a **finite volume method**. In such a method we consider breaking the space domain into a cells centered at $x_{j+1/2} = (x_{j+1} + x_j)/2$ with boundaries at x_j and x_{j+1} . We then compute the flux through the cell boundaries to determine the change inside the cell. Finite volume methods are expecially useful in two and three dimensions when the mesh can be irregularly shaped.

The problem $u_t + f(u)_x = 0$ where $u \equiv u(t, x)$ is equivalent to

$$\frac{1}{kh} \int_{t_n}^{t_{n+1}} \int_{x_{j-1/2}}^{x_{j+1/2}} (u_t + f(u)_x) dx dt = 0$$

which in turn can be integrated to be

$$\frac{1}{kh} \int_{x_{j-1/2}}^{x_{j+1/2}} u \Big|_{t_n}^{t_{n+1}} dx + \frac{1}{kh} \int_{t_n}^{t_{n+1}} f(u) \Big|_{x_{j-1/2}}^{x_{j+1/2}} dt = 0$$

or simply

$$\begin{aligned} & \frac{1}{kh} \int_{x_{j-1/2}}^{x_{j+1/2}} u(t_{n+1}, x) dx - \frac{1}{kh} \int_{x_{j-1/2}}^{x_{j+1/2}} u(t_n, x) dx \\ & + \frac{1}{kh} \int_{t_n}^{t_{n+1}} f(u(t, x_{j+1/2})) dt - \frac{1}{kh} \int_{t_n}^{t_{n+1}} f(u(t, x_{j-1/2})) dt = 0 \end{aligned} \quad (3.30)$$

Physically this says that the change in mass of a cell from time t_n to time t_{n+1} equals the flux through the boundaries at $x_{j+1/2}$ and $x_{j-1/2}$. Let's define

$$U_j^n = \frac{1}{h} \int_{x_{j-1/2}}^{x_{j+1/2}} u^n(t_n, x) dx$$

and the flux at x_j over the time interval $[t_n, t_{n+1}]$ to be

$$F_{j+1/2}^{n+1/2} = \frac{1}{k} \int_{t_n}^{t_{n+1}} f(u(t, x_{j+1/2})) dt$$

Think of u as density and U as mass, and think of f as flux density and F as flux. Then we have

$$\frac{U_j^{n+1} - U_j^n}{k} + \frac{F_{j+1/2}^{n+1/2} - F_{j-1/2}^{n+1/2}}{h} = 0 \quad (3.31)$$

Now we just need appropriate numerical approximations for these integrals.

For a *first order* approximation, we simply take *piecewise constant* approximations $u(t, x)$ in the cells. Then

$$U_j^n = \frac{1}{2} (U_{j-1/2}^n + U_{j+1/2}^n)$$

by the midpoint rule and

$$F_{j+1/2}^{n+1/2} = f(U_{j+1/2}^n).$$

This gives us a staggered version of the Lax-Friedrichs scheme

$$\frac{U_{j+1/2}^{n+1} - \frac{1}{2}(U_j^n + U_{j+1}^n)}{k} + \frac{f(U_{j+1}^n) - f(U_j^n)}{h} = 0$$

where the CFL condition is given by $|\lambda_p|k/h \leq 1/2$. Recall that approximating $U_{j+1/2}^n$ with $\frac{1}{2}(U_j^n + U_{j+1}^n)$ introduces numerical viscosity into the solution.

To extend the method to *second order* [Nessyahu-Tadmor, Sanders-Weiss '90], we need to use *piecewise linear* approximations. Define

$$P_j(x) = U_j^n + \partial_x U_j^n \cdot (x - x_j), \quad x_j < x < x_{j+1}.$$

Then

$$\begin{aligned} U_{j+1/2}^{n+1} &= \frac{1}{h} \int_{x_j}^{x_{j+1}} u(t_n, x) dx \approx \frac{1}{h} \int_{x_j}^{x_{j+1}} P_j(x) dx \\ &= \frac{1}{h} \int_{x_j}^{x_{j+1/2}} P_j(x) dx + \frac{1}{h} \int_{x_{j+1/2}}^{x_{j+1}} P_{j+1}(x) dx \\ &= \frac{1}{2}(U_j^n + U_{j+1}^n) + \frac{1}{8}h(\partial_x U_j^n - \partial_x U_{j+1}^n) \end{aligned}$$

So from (3.31), we have

$$U_{j+1/2}^{n+1} = \frac{1}{2}(U_j^n + U_{j+1}^n) + \frac{1}{8}h(\partial_x U_j^n - \partial_x U_{j+1}^n) - \frac{k}{h} (F_{j+1}^{n+1/2} - F_j^{n+1/2})$$

We still need a second order approximation for the flux $F_j^{n+1/2}$. Using the midpoint rule:

$$F_j^{n+1/2} = \frac{1}{k} \int_{t_n}^{t_{n+1}} f(u(t, x_j)) dt \approx f(u(t_{n+1/2}, x_j))$$

with

$$u(t_{n+1/2}, x_j) \approx u(t_n, x_j) + \frac{k}{2} \partial_t u(t_n, x_j) = u(t_n, x_j) - \frac{k}{2} \partial_x f(u(t_n, x_j)).$$

where we use $u_t + f(u)_x = 0$ for the equality above. This give us the staggered, two-step predictor-corrector method:

$$\begin{aligned} U_j^{n+1/2} &= U_j^n - \frac{k}{2} \partial_x f(U_j^n) \\ U_{j+1/2}^{n+1} &= \frac{1}{2}(U_j^n + U_{j+1}^n) + \frac{1}{8}h(\partial_x U_j^n - \partial_x U_{j+1}^n) - \frac{k}{h} \left[f(U_{j+1}^{n+1/2}) - f(U_j^{n+1/2}) \right] \end{aligned}$$

We need to use slope limiters to approximate the slopes $\partial_x U_j$ and $\partial_x f(U_j)$, otherwise we introduce oscillations. We define the slope limiter σ_j to be

$$\sigma_j = \frac{U_{j+1} - U_j}{h} \phi(\theta_j) \quad \text{where} \quad \theta_j = \frac{U_j - U_{j-1}}{U_{j+1} - U_j}$$

where common limiting functions are the minmod and the van Leer:

$$\begin{array}{ll} \text{minmod} & \phi(\theta) = \max(0, \min(1, \theta)) \\ \text{van Leer} & \phi(\theta) = \frac{|\theta| + \theta}{1 + |\theta|} \end{array}$$

The van Leer slope limiter gives a sharper shock than the minmod slope limiter. A slope limiter works by reducing a second-order method (which is typically dispersive) to a first-order method (which is typically dissipative) wherever there appears to be an oscillation. While the slope limiter removes oscillations and stabilizes the solution, it reduces the error to first order approximation near an oscillation. Hence, near discontinuities we can get at best first-order and more typically half-order convergence. To get higher orders, one may use essentially nonoscillatory (ENO) interpolation schemes of Harten and Osher.

Notice that the slope limiter σ_j is a symmetric function of U_{j-1} , U_j and U_{j+1} . That is,

$$\frac{U_{j+1} - U_j}{h} \phi\left(\frac{U_j - U_{j-1}}{U_{j+1} - U_j}\right) = \frac{U_j - U_{j-1}}{h} \phi\left(\frac{U_{j+1} - U_j}{U_j - U_{j-1}}\right).$$

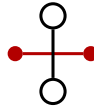
Also, note that the slope limiter as written above is undefined whenever $U_{j+1} = U_j$. But this happens precisely when we want the slope to be zero. We can fix the problem by adding a conditional to the denominator as in the following the MATLAB code

```
dU = [0;diff(U)];
s = dU.*phi( [dU(2:end);0]./(dU + (dU==0)) );
```

Notice that `dU` is padded with a zero on the left and then on the right, so that `s` has the same number of elements as `U`.

Exercises

- 3.1. Show that the order of convergence for the Lax-Friedrichs scheme is $O(k + h^2/k)$ and compute the dispersion relation. Taking the CFL condition and numerical viscosity into consideration, how should we best choose k and h to minimize error.
- 3.2. Solve the advection equation $u_t + u_x = 0$, with initial conditions $u(0, x) = 1$ if $|x - \frac{1}{2}| < \frac{1}{4}$ and $u(0, x) = 0$ otherwise, using the upwind scheme and the Lax-Wendroff scheme. Use the domain $x \in [0, 1]$ with periodic boundary conditions. Compare the solutions at $t = 1$.
- 3.3. Consider a finite difference scheme for the linear hyperbolic equation $u_t + cu_x = 0$ that uses the following stencil (leap-frog in time and center-difference in space):



Determine the order of the scheme and the stability conditions for a consistent scheme. Discuss whether the scheme is dispersive or dissipative. Comment on how dispersion or dissipation affects the stability, i.e., is this a good scheme? Show that you can derive this scheme starting with (3.4) and counteracting the unstable term.

3.4. Consider a finite difference scheme for $u_t + u_x = 0$ that uses a fourth-order centered-difference approximation for u_x and the fourth-order Runge-Kutta method in time. Determine the order of the scheme and discuss the stability. Determine whether the scheme is dispersive or dissipative.

- 3.5. The compressible Euler equations of gas dynamics are

$\rho_t + (\rho u)_x = 0$	Conservation of mass
$(\rho u)_t + (\rho u^2 + p)_x = 0$	Conservation of momentum
$E_t + ((E + p)u)_x = 0$	Conservation of energy

where ρ is the density, u is the velocity, E is the total energy and p is the pressure. This system is not a closed system—there are more unknowns than equations. To close the system we add the equation of state for an ideal polytropic gas:

$$E = \frac{1}{2}\rho u^2 + \frac{p}{\gamma - 1}$$

where $\gamma > 1$ is a constant. (For air, $\gamma \approx 1.4$) Sound speed is given by $c = \sqrt{\gamma p / \rho}$. Show that this system forms a hyperbolic system and find the eigenvalues of the Jacobian. Hint: Express the problem as a nonlinear hyperbolic system using the variables ρ , u and p .

- 3.6. Solve Burgers' equation $u_t + \frac{1}{2}(u^2)_x = 0$ over the domain $[0, 4]$, with initial conditions $u(0, x) = 1$ if $1 < x < 2$ and $u(0, x) = 0$ otherwise, both analytically (using the method of characteristics or some other means) and numerically using the Local Lax-Friedrichs method.

Use the analytic solution to confirm that the numerical solution is correct by plotting together at several time steps.

✎ 3.7. The “dam break” problem. Consider the shallow water equations

$$h_t + (hu)_x = 0 \tag{3.33a}$$

$$(hu)_t + (hu^2 + \tfrac{1}{2}gh^2)_x = 0 \tag{3.33b}$$

where g is the gravitational acceleration, h is the height of the water, and u is the velocity of the water. Set $g = 1$. Solve the problem with initial conditions $h = 1$ if $x < 0$, $h = 0.2$ if $x > 0$ and $u = 0$ over the domain $[-1, 1]$ using the first and second order central schemes with constant boundary conditions. (For the second order, you will need to use a slope limiter.) Use 100 and 200 points for space discretization and the time step according to the CFL condition. Plot the numerical result at $t = 0.25$ against the “exact” solution (using a very fine mesh in space and step-size in time).

Finite Element Method

In this chapter, we examine the Finite Element Method (FEM) and applications of it to elliptic equations. Typical elliptic equations are Laplace's equation or the Poisson's equation, which model the steady-state distribution of electrical charge or temperature either with a source term (Poisson's equation) or without (Laplace's equation), and the steady-state Schrödinger equation. Finite element methods, pioneered in the 1960s, are widely used for engineering problems because the flexible geometry allows one to use irregular elements, the variational formulation makes it easy to deal with boundary conditions, and it is mathematically convenient for error estimates.

4.1 Variational (or Weak) Forms

A one-dimensional example

Consider a heat conducting bar with an unknown temperature distribution $u(x)$, a known heat source $f(x)$ and a heat conductivity $\kappa > 0$ with constant temperature $u(0) = u(1) = 0$ at the ends of the bar. Fourier's law states that the heat flux $q = -\kappa u'(x)$ and the conservation of energy says that $q' = f$. Take $\kappa = 1$. The steady-state solution can be found by formulating the problem in three ways.

Ⓓ **Boundary value problem:** Find u such that $-u'' = f$ with $u(0) = u(1) = 0$.

Let $V = \{v \in C[0,1] \mid v \text{ piecewise continuous, } v(0) = v(1) = 0\}$. Define the **total potential energy**

$$F(v) = \frac{1}{2} \langle v', v' \rangle - \langle f, v \rangle$$

where the inner product

$$\langle u, v \rangle = \int_0^1 u(x)v(x) dx.$$

In the next section, we'll introduce the notation $a(\cdot, \cdot)$ and $b(\cdot)$ in the place of these inner products.

Ⓜ **Minimization problem:** Find $u \in V$ such that $F(u) \leq F(v)$ for all $v \in V$. In other words, find the solution that minimizes the energy.

Ⓥ **Variational problem:** Find $u \in V$ such that $\langle u', v' \rangle = \langle f, v \rangle$ for all $v \in V$

The variational problem is also called the **weak formulation**.

Theorem 5. Under suitable conditions the boundary value problem ⓓ, the minimization problem Ⓜ and the variational problem Ⓥ are all equivalent.

Proof. First, we show ⓓ \Leftrightarrow Ⓥ.

Assume ⓓ. So $-u'' = f$. Then for all $v \in V$,

$$\begin{aligned} -\int_0^1 u''v dx &= \int_0^1 f v dx \\ u'v'|_0^1 - \int_0^1 u'v' dx &= \int_0^1 f v dx \\ \langle u', v' \rangle &= \langle f, v \rangle \end{aligned}$$

Assume Ⓥ. Suppose that u'' is continuous. Then

$$\begin{aligned} \int_0^1 u'v' dx &= \int_0^1 f v dx \quad \text{for all } v \in V \\ u'v'|_0^1 - \int_0^1 u''v dx &= \int_0^1 f v dx \\ \int_0^1 (u'' + f)v dx &= 0 \end{aligned}$$

which is true for all $v \in V$. Therefore, $-u'' = f$.

Now, we show Ⓥ \Leftrightarrow Ⓜ.

Assume Ⓜ. Define the perturbation

$$g(\varepsilon) = F(u + \varepsilon w)$$

for an arbitrary function w with $\varepsilon > 0$. The variational derivative is defined as

$$\left. \frac{dg}{d\varepsilon} \right|_{\varepsilon=0}.$$

The energy $F(u)$ is at a minimum u when the variation derivative equals zero.

$$\begin{aligned} g(\varepsilon) &= \frac{1}{2} \langle u' + \varepsilon w', u' + \varepsilon w' \rangle - \langle f, u + \varepsilon w \rangle \\ &= \frac{1}{2} \langle u', u' \rangle + \varepsilon \langle u', w' \rangle + \frac{1}{2} \varepsilon^2 \langle w', w' \rangle - \langle f, u \rangle - \varepsilon \langle f, w \rangle \\ 0 &= \left. \frac{dg}{d\varepsilon} \right|_{\varepsilon=0} = \langle u', w' \rangle + \varepsilon \langle w', w' \rangle - \langle f, w \rangle \Big|_{\varepsilon=0} \end{aligned}$$

So $\langle u', w' \rangle = \langle f, w \rangle$ for all $w \in V$.

Now, assume **(V)**. Then

$$\begin{aligned} F(v) &= F(u + v - u) \\ &= F(u + w) \quad \text{where } w = v - u \in V \\ &= \frac{1}{2} \langle u' + w', u' + w' \rangle - \langle f, u + w \rangle \\ &= \frac{1}{2} \langle u', u' \rangle + \langle u', w' \rangle + \frac{1}{2} \langle w', w' \rangle - \langle f, u \rangle - \langle f, w \rangle \\ &= F(u) + \frac{1}{2} \langle w', w' \rangle \quad \text{because } \langle u', w' \rangle = \langle f, w \rangle \\ &\geq F(u) \end{aligned}$$

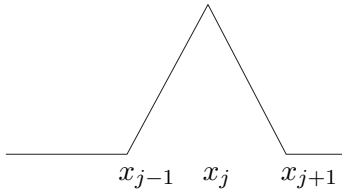
□

★ The basic idea of FEM is to find a solution to the variational problem **(V)** or the minimization problem **(M)** in a finite dimensional subspace V_h of V . The finite element solution u_h is a projection of u .

(M) Raleigh-Ritz: Find $u_h \in V_h$ such that $F(u_h) \leq F(v_h)$ for all $v_h \in V_h$.

(V) Galerkin: Find $u_h \in V_h$ such that $\langle u_h', v_h' \rangle = \langle f, v_h \rangle$ for all $v_h \in V_h$.

Let's apply the Galerkin formulation to the original boundary value problem. Consider V_h to be the subspace of continuous, piecewise linear functions with nodes at $\{x_i\}$. We may also consider piecewise quadratic or piecewise polynomial basis functions as a basis elements. We will restrict ourselves to a uniform partition so that the derivations are cleaner. The finite element solution for a nonuniform partition can similarly be derived. A basis element $\varphi_j(x) \in V_h$ is

$$\varphi_j(x) = \begin{cases} 1 + t_j, & t_j \in (-1, 0] \\ 1 - t_j, & t_j \in (0, 1] \\ 0, & \text{otherwise.} \end{cases}$$


with $t_j = (x - x_j)/h$ and h is the uniform grid spacing between the nodes j and $j - 1$. Note that $\{\varphi_j(x)\}$ form a **partition of unity**, i.e., $\sum_j \varphi_j(x) = 1$ for all x . Then $v_h \in V_h$ is defined by

$$v_h = \sum_{j=0}^{m+1} v_j \varphi_j(x)$$

with $v_j = v(x_j)$ and $v \in V$. Because $v(x) = 0$ on the boundaries we have that

$$v_h = \sum_{j=1}^m v_j \varphi_j(x).$$

The finite-element solution is given by

$$u_h = \sum_{i=1}^m \xi_i \varphi_i(x)$$

where $\xi_j = u(x_j)$ are unknowns. The Galerkin problem $\langle u'_h, v'_h \rangle = \langle f, v_h \rangle$ is then

$$\left\langle \sum_{i=1}^m \xi_i \varphi'_i(x), \sum_{j=1}^n v_j \varphi'_j(x) \right\rangle = \left\langle f, \sum_{j=1}^n v_j \varphi_j(x) \right\rangle$$

which by bilinearity is the same as

$$\sum_{j=1}^m v_j \left\langle \sum_{i=1}^n \xi_i \varphi'_i(x), \varphi'_j(x) \right\rangle = \sum_{j=1}^n v_j \langle f, \varphi_j(x) \rangle.$$

In order that this equality holds for all v_j it must follow that

$$\left\langle \sum_{i=1}^m \xi_i \varphi'_i(x), \varphi'_j(x) \right\rangle = \langle f, \varphi_j(x) \rangle \quad \text{for } j = 1, \dots, n.$$

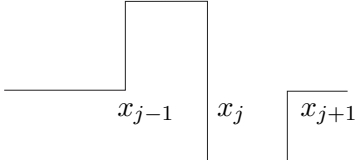
Again by bilinearity

$$\sum_{i=1}^m \langle \varphi'_i(x), \varphi'_j(x) \rangle \xi_i = \langle f, \varphi_j(x) \rangle \quad \text{for } j = 1, \dots, n.$$

Let $a_{ij} = \langle \varphi'_i, \varphi'_j \rangle$, $b = (f_1, \dots, f_m)^T$ where $f_j = \langle f, \varphi_j \rangle$, and $\xi = (\xi_1, \dots, \xi_m)^T$. Then we have the linear system of equation

$$\mathbf{A} \xi = \mathbf{b}.$$

\mathbf{A} is called the **stiffness matrix** and \mathbf{b} is called the **load vector**. Note that \mathbf{A} is symmetric positive definite and tridiagonal.

$$a_{ij} = \langle \varphi'_j, \varphi'_i \rangle = \begin{cases} \int_0^1 (\varphi'_i)^2 dx = \frac{2}{h}, & i = j \\ \int_0^1 \varphi'_i \varphi'_j dx = -\frac{1}{h}, & i = j \pm 1 \\ 0, & \text{otherwise.} \end{cases}$$


and

$$\eta^T A \eta = \sum_{i,j=1}^m \eta_i A_{ij} \eta_j = \sum_{i,j=1}^m \langle \eta_i \varphi'_i, \eta_j \varphi'_j \rangle = \left\langle \sum_{i=1}^m \eta_i \varphi'_i, \sum_{j=1}^m \eta_j \varphi'_j \right\rangle \geq 0$$

and equals 0 if and only if $\sum \eta_i \varphi'_i = 0$ then $\eta_i = 0$ for all i so $\eta' = 0$.

If the gridspacing h_j is independent of j , then we get the finite difference method

$$-\frac{u_{j+1} - 2u_j + u_{j-1}}{h^2} = f_j$$

with a stiffness matrix

$$A = \frac{1}{h^2} \begin{bmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix}.$$

4.2 A two-dimensional example

Write the two dimensional Poisson equation in variational form

$$\begin{cases} -\Delta u = f, \\ u|_{\partial\Omega} = 0 \end{cases}$$

Let $V = \{v \in C(\Omega) \mid v_x, v_y \text{ are piecewise continuous, } v|_{\partial\Omega} = 0\}$. Then for all $v \in V$,

$$-\iint_{\Omega} v \Delta u \, dx \, dy = \iint_{\Omega} f v \, dx \, dy. \quad (4.1)$$

Green's formula says that

$$\iint_{\Omega} v \Delta u \, dx \, dy = \int_{\partial\Omega} v \frac{\partial u}{\partial n} \, ds - \iint_{\Omega} \nabla v \cdot \nabla u \, dx \, dy$$

where $\frac{\partial u}{\partial n}$ is a directional derivative ($\hat{\mathbf{n}}$ is the unit outer normal). It's an easy exercise to prove this formula. Note that

$$\nabla \cdot (v \nabla u) = \nabla v \cdot \nabla u + v \Delta u$$

which is easily seen using component notation (using Einstein summation convention)

$$\partial_i (v \partial_i u) = (\partial_i v) (\partial_i u) + v \partial_{ii} u$$

Integrating both sides over Ω gives us

$$\int_{\Omega} \nabla \cdot (v \nabla u) \, dV = \int_{\Omega} \nabla v \cdot \nabla u + v \Delta u \, dV$$

From which it follows that

$$\begin{aligned} \int_{\Omega} v \Delta u \, dV &= \int_{\Omega} \nabla \cdot (v \nabla u) \, dV - \int_{\Omega} \nabla v \cdot \nabla u \, dV \\ &= \int_{\partial\Omega} (v \nabla u) \cdot \hat{\mathbf{n}} \, dA - \int_{\Omega} \nabla v \cdot \nabla u \, dV \\ &= \int_{\partial\Omega} v \frac{\partial u}{\partial n} \, dA - \int_{\Omega} \nabla v \cdot \nabla u \, dV. \end{aligned}$$

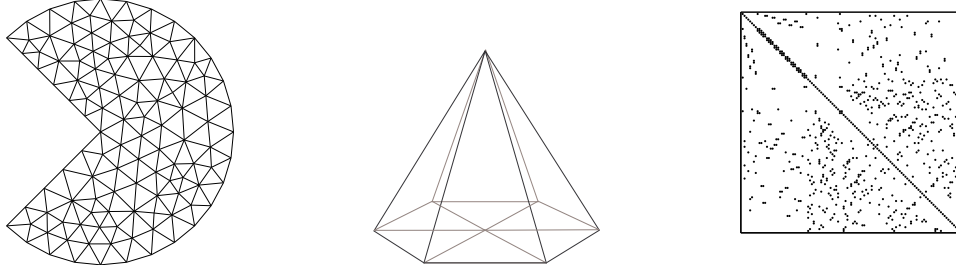


Figure 4.1: Triangulation of a domain Ω (left) using a simple pyramidal basis functions (middle) and the resultant sparse stiffness matrix (right).

Therefore, (4.1) is equivalent to

$$\iint_{\Omega} \nabla u \cdot \nabla v \, dx \, dy = \iint_{\Omega} f v \, dx \, dy.$$

Define the bilinear functional $a(\cdot, \cdot)$ as

$$a(u, v) = \iint_{\Omega} \nabla u \cdot \nabla v \, dx \, dy$$

and the linear functional $b(\cdot)$

$$b(v) = \iint_{\Omega} v f \, dx \, dy.$$

Then the variation formulation is

$$\textcircled{\mathbf{V}} \text{ Find } u \in V \text{ such that } a(u, v) = b(v) \text{ for all } v \in V.$$

Let's now look at the implementation using FEM. Consider some triangulation T_h . Let $\Omega = \bigcup_{K_j \in T_h} K_j$ where K_j are triangular elements and n_j are nodes on the vertices. Assume that there are no nodes at the edges. The finite element space is

$$V_h = \{v \in C(\Omega) \mid v_k \text{ piecewise linear, } v|_{\partial\Omega} = 0\}$$

The basis of V_h

$$\varphi_i(n_j) = \delta_{ij}$$

where $\varphi_i|_k$ is linear. For all $v_h \in V_h$, $v_h = \sum_{i=1}^m v(n_i) \varphi_i(x)$.

$$\textcircled{\mathbf{V}} \text{ Find } u_h \in V_h \text{ such that } a(u_h, v) = b(v) \text{ for all } v \in V_h.$$

This holds if and only if $a(u_h, \varphi_j) = b(\varphi_j)$ for $j = 1, \dots, m$. Let

$$u_h = \sum_{i=1}^m \xi_i \varphi_i(x)$$

where $\xi_i = u_h(N_i)$, then

$$a\left(\sum_{i=1}^m \xi_i \varphi_i(x), \varphi_j\right) = b(\varphi_j) \quad 1 \leq j \leq m$$

from which it follows by linearity that

$$\sum_{i=1}^m \xi_i a(\varphi_i, \varphi_j) = b(\varphi_j) \quad 1 \leq j \leq m$$

which we can write as

$$\mathbf{A}\xi = \mathbf{b}$$

where

$$a_{ij} = a(\varphi_i, \varphi_j) = \int_{\Omega} \nabla \varphi_i \cdot \nabla \varphi_j \, dx \, dy$$

and

$$b_j = b(\varphi_j) = \int_{\Omega} \varphi_j(x) f(x) \, dx \, dy.$$

Note that the stiffness matrix A is sparse, because $a_{ij} = 0$ if N_i and N_j are not the vertices of the same triangle. Element stiffness matrix for element K

$$\mathbf{A}^{(K)} = \begin{bmatrix} a^K \langle \varphi_i, \varphi_i \rangle & a^K \langle \varphi_i, \varphi_j \rangle & a^K \langle \varphi_i, \varphi_k \rangle \\ a^K \langle \varphi_j, \varphi_i \rangle & a^K \langle \varphi_j, \varphi_j \rangle & a^K \langle \varphi_j, \varphi_k \rangle \\ a^K \langle \varphi_k, \varphi_i \rangle & a^K \langle \varphi_k, \varphi_j \rangle & a^K \langle \varphi_k, \varphi_k \rangle \end{bmatrix}$$

Assembly of the global stiffness matrix $\mathbf{A}^{(K)} \rightarrow \mathbf{A}$

$$\mathbf{A} = \begin{bmatrix} & & & \\ & a_{ii}^K & a_{ij}^K & a_{ik}^K \\ & a_{ji}^K & a_{jj}^K & a_{jk}^K \\ & a_{ki}^K & a_{kj}^K & a_{kk}^K \\ & & & \end{bmatrix}$$

To implement a finite element method:

1. Input f , g and Ω .
2. Construct T_h . You want triangles as uniform as possible.
3. Construct element stiffness matrices $\{a_{ij}^K\}$ and element load vector $\{b_j^K\}$.
4. Assemble the global stiffness matrix \mathbf{A} and load vector \mathbf{b} .
5. Solve $\mathbf{A}\xi = \mathbf{b}$.
6. Present the results.

4.3 Neumann boundary conditions

Let's write the following Neumann problem as a FEM problem.

$$\textcircled{\mathbf{D}} \quad \begin{cases} -\Delta u + u = f, & x \in \Omega \\ \frac{\partial u}{\partial n} \Big|_{\partial\Omega} = g \end{cases}$$

Take $v \in H^1(\Omega)$. This time v is non-zero on the boundary. Then

$$\int_{\Omega} -v\Delta u + vu \, dx = \int_{\Omega} fv \, dx$$

Using Green's identity, we have that

$$\int_{\Omega} \nabla v \cdot \nabla u \, dx - \int_{\partial\Omega} v \frac{\partial u}{\partial n} \, ds + \int_{\Omega} vu \, dx = \int_{\Omega} fv \, dx$$

On the boundary we have that $\frac{\partial u}{\partial n} = g$, so

$$\int_{\Omega} \nabla v \cdot \nabla u \, dx + \int_{\Omega} vu \, dx = \int_{\Omega} fv \, dx + \int_{\partial\Omega} gv \, ds$$

Let

$$a(u, v) = \int_{\Omega} \nabla u \cdot \nabla v + uv \, dx$$

and let

$$b(v) = \langle f, v \rangle + \langle\langle g, v \rangle\rangle \equiv \int_{\Omega} fv \, dx + \int_{\partial\Omega} gv \, ds.$$

Then we have the variational form of the Neumann problem is

$$\textcircled{\mathbf{V}} \quad \text{Find } u \in H^1(\Omega) \text{ such that } a(u, v) = b(v) \text{ for all } v \in H^1(\Omega).$$

Recasting the problem as a FEM approximation

$$\textcircled{\mathbf{V}} \quad \text{Find } u_h \in V_h \text{ such that } a(u_h, v) = b(v) \text{ for all } v \in V_h \text{ where}$$

$$u_h = \sum_{i=0}^{m+1} \xi_i \varphi_i(x)$$

Then

$$\sum_{i=0}^{m+1} \xi_i a(\varphi_i, \varphi_j) = b(\varphi_j), \quad j = 0, \dots, m+1$$

So,

$$\mathbf{A}\xi = \mathbf{b}$$

where

$$a_{ij} = a(\varphi_i, \varphi_j) = \sum_{K \in T_h} \int_K \nabla \varphi_i \cdot \nabla \varphi_j + \varphi_i \varphi_j \, dx = \sum_{K \in T_h} a_{ij}^K$$

and

$$b_j = \sum_{K \in T_h} \int_{\partial\Omega \cap K} g \varphi_j \, ds + \int_K f \varphi_j \, dx = \sum_{K \in T_h} b_j^K$$

For a one-dimensional problem with $x \in [0, 1]$, a_{ij} and b_j are simply

$$a_{ij} = \sum_{i=0}^{m+1} \varphi_i' \varphi_j' + \varphi_i \varphi_j \, dx$$

and

$$b_j = \int_{\Omega} f(x) \varphi_j(x) \, dx + g(x) \varphi_j(x) \Big|_0^1$$

with $g(x) \varphi_j(x)|_0^1 = g(0)$ at $j = 0$, $g(x) \varphi_j(x)|_0^1 = g(1)$ at $j = m + 1$, and $g(x) \varphi_j(x)|_0^1 = 0$ otherwise. Note that this time our stiffness matrix \mathbf{A} and load vector \mathbf{b} includes the boundary elements.

4.4 Abstract formulation

The set V is a **vector space** if $\alpha v + w \in V$ for all real numbers α and $u, v \in V$. The mapping $L : V \rightarrow \mathbb{R}$ is a **linear functional** if

$$L(\alpha u + \beta v) = \alpha L(u) + \beta L(v)$$

for all real numbers α and β and $u, v \in V$. $V \times V \rightarrow \mathbb{R}$ is a **bilinear functional** if

$$\begin{aligned} a(\alpha u + \beta w, v) &= \alpha a(u, v) + \beta a(w, v) \\ a(u, \alpha v + \beta w) &= \alpha a(u, v) + \beta a(u, w) \end{aligned}$$

for all real numbers α and β and $u, v, w \in V$. If $a(u, v) = a(v, u)$, we say that a is symmetric. If $a(u, u) \geq 0$, we say that a is an **inner product**. Define $\|u\|_a = \sqrt{a(u, u)}$ to be the energy norm.

A vector space is **complete** if every Cauchy sequence converges under this norm. A **Banach space** is a complete vector space. A **Hilbert space** V is a Banach space with an inner product. The space of L^2 -functions (square integrable functions) is $L^2(\Omega) = \{v \mid \int_{\Omega} |v|^2 \, dx < \infty\}$. In this case, we define the L^2 -inner product as

$$\langle v, w \rangle = \int_{\Omega} v w \, dx$$

and the L^2 -norm as

$$\|v\|_{L^2} = \sqrt{\langle v, v \rangle}.$$

The Cauchy-Schwarz inequality says

$$|\langle v, w \rangle| \leq \|v\|_2 \|w\|_2$$

A **Sobolev space** is

$$H^1(\Omega) = \{v \in L^2(\Omega) \mid \nabla v \in L^2(\Omega)\}$$

We define the Sobolev inner product as as

$$\langle v, w \rangle_{H^1} = \int_{\Omega} vw + \nabla v \cdot \nabla w \, dx$$

and the Sobolev norm as

$$\|v\|_{H^1} = \sqrt{\int_{\Omega} v^2 + |\nabla v|^2 \, dx}$$

Finally, define

$$H_0^1(\Omega) = \{v \in H^1(\Omega) \mid v|_{\partial\Omega} = 0\}$$

be the set of all Sobolev functions which vanish on the the boundary of Ω . The Sobolev norms are extension the L^2 -norms and give a measure of the smoothness (or regularity) of a function.

Theorem 6 (Lax–Milgram Lemma). *Suppose that V is a Hilbert space, $\langle \cdot, \cdot \rangle_V$ is an inner product on V . Let $\|\cdot\|_V$ be a norm, $a(\cdot, \cdot)$ is a bilinear form which maps $V \times V \rightarrow \mathbb{R}$ and $b(\cdot)$ is a linear form which maps $V \rightarrow \mathbb{R}$. Assumptions on a and b :*

1. a is symmetric $a(u, v) = a(v, u)$
2. a is continuous $\exists \gamma > 0$ such that $|a(u, v)| \leq \gamma \|u\|_V \|v\|_V$ for all $u, v \in V$
3. a is coercive $\exists \alpha > 0$ such that $|a(u, u)| \geq \alpha \|u\|_V^2$ for all $u \in V$
4. b is continuous $\exists \Lambda > 0$ such that $|b(u)| \leq \Lambda \|u\|_V$ for all $u \in V$

Note that coercive is often called V -ellipticity and continuous is also called bounded. Under these assumptions,

(V) Find $u \in V$ such that $a(u, v) = b(v)$ for all $v \in V$.

are **well-posed**. That means that the solution exists, is unique, and moreover is stable.

Proof. Suppose that there are two solutions u_1 and u_2 . So, $a(u_1, v) = b(v)$ and $a(u_2, v) = b(v)$ for all $v \in V$. Then $|a(u_1 - u_2, v)| = 0$. By V -ellipticity

$$\alpha \|u_1 - u_2\|_V \leq a(u_1 - u_2, u_1 - u_2) = 0$$

So $u_1 = u_2$. Also,

$$\alpha \|u\|_V^2 \leq a(u, u) = b(u) \leq \Lambda \|u\|_V$$

So, $\|u\|_V \leq \Lambda/\alpha$. So, the solution is stable. □

Example. Show that the FEM formulation of the Poisson equation is well-posed (unique and stable):

$$-u'' = f, \quad u(0) = u(1) = 0$$

Define the finite element space as $V = H_0^1(\Omega)$.

$$\int_0^1 u'v' dx = \int_0^1 fv dx$$

Let

$$a(u, v) = \int_0^1 u'v' dx \quad \text{and} \quad b(v) = \int_0^1 fv dx.$$

(V) Find $u \in H_0^1(\Omega)$ such that $a(u, v) = b(v)$ for all $v \in H_0^1(\Omega)$.

Let

$$\|u\|_{H_0^1} = \sqrt{\int_{\Omega} u^2 + |u'|^2 dx}.$$

We need to check that the assumptions hold. Then the results follow by the theorem.

1. Checks
2. By the Cauchy-Schwarz inequality

$$a(u, v) = \left| \int_0^1 u'v' dx \right| \leq \sqrt{\int_0^1 |u'|^2 dx} \sqrt{\int_0^1 |v'|^2 dx} \leq \|u\|_{H^1} \|v\|_{H^1}$$

3. To show this we will first prove Poincaré's Inequality which says that for $u \in H_0^1$ and Ω bounded, then

$$\int u^2 dx \leq \beta \int (u')^2 dx \quad \text{for some } \beta.$$

Without loss of generality, take $u(0) = 0$ and $x \in [0, 1]$. Then by the Cauchy-Schwarz inequality

$$u(x) = u(x) - u(0) = \int_0^x u'(x) dx \leq \sqrt{\int_0^x 1 dx} \sqrt{\int_0^x (u')^2 dx} \leq \sqrt{\int_0^1 (u')^2 dx}$$

Squaring and integrating both sides give us

$$\int_0^1 u^2(x) dx \leq \int_0^1 \int_0^1 (u')^2 dx dx = \int_0^1 (u')^2 dx.$$

Back to our problem. $a(u, u) = \int_0^1 (u')^2 dx \geq \alpha \int_0^1 u^2 + (u')^2 dx$. Now, the assumption holds.

4. $|b(v)| = \left| \int fv dx \right| \leq \|f\|_{L^2} \|v\|_{L^2} \leq \|f\|_{L^2} \|v\|_{H^1}$

Theorem 7 (Céa's Lemma). *If a is bounded and coercive, then*

$$\|u - u_h\|_V \leq (\gamma/\alpha) \min_{w_h \in V_h} \|u - w_h\|_V.$$

Proof. For $w_h \in V_h \subset V$,

$$a(u_h, w_h) = \langle f, w_h \rangle \quad \forall w_h \in V_h \quad \text{and} \quad a(u, w_h) = \langle f, w_h \rangle \quad \forall w_h \in V$$

Therefore, $a(u - u_h, w_h) = 0$ for all $w_h \in V_h$. This says, that u_h is a projection of u in the inner-product $a(\cdot, \cdot)$. Let $e = u - u_h$. Then,

$$\begin{aligned} \alpha \|e\|_V^2 &\leq a(e, e) && \text{(because } a \text{ is coercive)} \\ &= a(e, u - w_h) + a(e, w_h - u_h) && \forall w_h \in V_h \\ &= a(e, u - w_h) && \forall w_h \in V_h \quad \text{(because } w_h - u_h \in V_h) \\ &\leq \gamma \|e\|_V \|u - w_h\|_V && \text{(because } a \text{ is bounded)} \\ \|e\|_V &\leq (\gamma/\alpha) \|u - w_h\|_V && \forall w_h \in V_h \end{aligned}$$

From which it follows that

$$\|u - u_h\|_V \leq (\gamma/\alpha) \min_{w_h \in V_h} \|u - w_h\|_V. \quad \square$$

Céa's Lemma says that the finite element solution u_h is the best solution up to the constant γ/α . One can use Céa's Lemma to estimate the Galerkin approximation error:

$$\begin{aligned} \|u - u_h\|_{H_0^1} &\leq (\gamma/\alpha) \min_{w_h \in V_h} \|u - w_h\|_{H_0^1} \\ &\leq (\gamma/\alpha) \|u - \Pi_h^1 u\|_{H_0^1} \\ &\leq (\gamma/\alpha) Ch \|u''\|_L^2 \\ &\leq (\gamma/\alpha) Ch \|u\|_{H_0^2} \end{aligned}$$

where $\Pi_h^1 u$ a piecewise linear polynomial interpolant of the solution $u \in H^2$. For the proof of the third inequality, see Theorem 8.3 in Quateroni, Sacco and Saleri. For k -order polynomial basis functions, then the H_0^1 -error is $O(h^k)$ if $u \in H^{k+1}$.

4.5 FEM for time-dependent problems

As an example of an application of Finite Element Method to time-dependent partial differential equations, consider the one-dimensional heat equation with a source term $f(x)$

$$\begin{cases} u_t - u_{xx} = f(x) \\ u(0, t) = u(1, t) = 0 \\ u(x, 0) = u_0(x) \end{cases} \quad (4.2)$$

Let $V = H_0^1([0, 1])$. Then for all $v \in V$,

$$\int_0^1 v u_t dx - \int_0^1 v u_{xx} dx = \int_0^1 v f dx$$

from which we have that

$$\int_0^1 v u_t dx - \int_0^1 v_x u_x dx = \int_0^1 v f dx$$

The Galerkin formulation is

$$(\mathbf{V}): \text{ Find } u_h \in V_h \text{ such that } \langle u_t, v \rangle + \langle u_x, v_x \rangle = \langle f, v \rangle \text{ for all } v \in V_h.$$

Let $\{\varphi_0, \dots, \varphi_n\}$ be a basis of V_h . Let

$$u^h(x, t) = \sum_{i=0}^n \xi_i(t) \varphi_i(x)$$

and take $v = \varphi_j$. Then we have the system

$$\sum_{i=0}^n \xi_i(t) \int_0^1 \varphi_i \varphi_j dx + \sum_{i=0}^n \xi_i(t) \int_0^1 \varphi'_i \varphi'_j dx = \int_0^1 f \varphi_j dx \quad \text{for } 0 \leq j \leq n$$

or more concisely, we have the system of ODEs

$$\mathbf{A} \xi'(t) + \mathbf{B} \xi(t) = \mathbf{c} \tag{4.3}$$

where $\mathbf{A} = (a_{ij})$, $\mathbf{B} = (b_{ij})$ with

$$\begin{aligned} a_{ij} &= \int_0^1 \varphi_i \varphi_j dx \\ b_{ij} &= \int_0^1 \varphi'_i \varphi'_j dx \\ c &= \int_0^1 f \varphi_j dx \end{aligned}$$

We can rewrite (4.3) as

$$\xi'(t) + \mathbf{A}^{-1} \mathbf{B} \xi(t) = \mathbf{A}^{-1} \mathbf{c}$$

which we can solve using an ODE method such as the backward Euler

$$\frac{\xi^{n+1} - \xi^n}{k} + \mathbf{A}^{-1} \mathbf{B} \xi^{n+1} = \mathbf{A}^{-1} \mathbf{c}$$

or the Crank-Nicolson

$$\frac{\xi^{n+1} - \xi^n}{k} + \frac{1}{2} \mathbf{A}^{-1} \mathbf{B} \xi^{n+1} + \frac{1}{2} \mathbf{A}^{-1} \mathbf{B} \xi^n = \mathbf{A}^{-1} \mathbf{c}.$$

Numerical stability can be shown by showing that the energy decreases over time. Consider the heat equation (4.2) with zero source term $u_t = u_{xx}$. Multiplying both sides by u gives us $u u_t = u u_{xx}$ from which it follows that

$$\frac{1}{2} \frac{d}{dt} \int_0^1 u^2 dx = - \int_0^1 (u_x)^2 dx \leq 0$$

So, $\|u(\cdot, t)\|_{L^2} \leq \|u(\cdot, 0)\|_{L^2}$.

Example. Show that the backward-Euler scheme is unconditionally stable. The FEM backward-Euler scheme is

$$\left\langle \frac{u_h^{n+1} - u_h^n}{k}, v \right\rangle - a(u_h^{n+1}, v) = 0 \quad \forall v \in V_h$$

where the energy norm $a(u, v) = \int_0^1 uv \, dx$. Take $v = u_h^{n+1}$

$$\langle u_h^{n+1}, u_h^{n+1} \rangle - \langle u_h^n, u_h^{n+1} \rangle = -ka(u_h^{n+1}, u_h^{n+1}) \leq 0$$

So,

$$\|u_h^{n+1}\|_{L^2}^2 \leq \langle u_h^n, u_h^{n+1} \rangle \leq \|u_h^n\|_{L^2} \|u_h^{n+1}\|_{L^2} \quad \text{by Cauchy-Schwarz}$$

Therefore, $\|u_h^{n+1}\|_{L^2} \leq \|u_h^n\|_{L^2}$ for all n and all step-sizes k . Hence, the backward-Euler scheme is unconditionally stable.

Exercises

- ✎ 4.1. Solve numerically the differential equation by the finite element method using piecewise linear elements:

$$-u_{xx} - u + 8x^2 = 0 \quad \text{for } 0 < x < 1$$

for the following sets of boundary conditions:

1. Dirichlet $u(0) = u(1) = 0$
2. Neumann $u_x(0) = 0, \quad u_x(1) = 1$

- ✎ 4.2. Consider the boundary value problem

$$\begin{aligned} \frac{d^4 u}{dx^4} &= f, & 0 < x < 1 \\ u(0) &= u'(0) = u(1) = u'(1) = 0. \end{aligned} \tag{4.4}$$

for the deflection of a uniform beam under the load $f(x)$.

1. Show that the problem (4.4) can be given the following variational formulation: Find $u \in V$ such that

$$a(u, v) = b(f) \quad \forall v \in V$$

where $V = \{v : v \text{ and } v' \text{ are continuous on } [0,1], v'' \text{ is piecewise continuous and } v(0) = v'(0) = v(1) = v'(1) = 0\}$.

2. Show that the variational formulation of the problem (4.4) is well-posed. Note: you will need to take $u \in H_0^2([0,1])$.

3. Construct a finite-dimensional subspace V_h of $H_0^2(0, 1)$. Formulate a finite element method based on the space V_h . Find the corresponding linear system of equations in the case of a uniform partition. Determine the solution for $f(x) = 384$. Compare it with the exact solution. Note: You will need to choose your solution in $H_0^2([0, 1])$. While you can get by with a quadratic spline, it may be easier to use cubic Hermite splines

$$u(x) = \sum_{i=0}^{m+1} \xi_i \phi_i(x) + \eta_i \psi_i(x)$$

where the cubic Hermite elements are given by

$$\phi_j(t) = \begin{cases} -2t^3 - 3t^2 + 1, & t \in [-1, 0] \\ +2t^3 - 3t^2 + 1, & t \in [0, 1] \\ 0, & \text{otherwise} \end{cases} \quad \text{and} \quad \psi_j(t) = \begin{cases} +t^3 + 2t^2 + t, & t \in [-1, 0], \\ +t^3 - 2t^2 + t, & t \in [0, 1], \\ 0, & \text{otherwise} \end{cases}$$

with $t = (x - x_j)/h$.

- 4.3. Show that the Crank-Nicolson FEM scheme for the heat equation $u_t = u_{xx}$ is unconditionally stable.

Fourier Spectral Methods

To increase the spatial accuracy (order of convergence) for a finite difference method, we use more grid points in order to achieve a better approximation of the derivatives. One might surmise that we could achieve the best accuracy by using all the gridpoints available. This is precisely what spectral methods do. In this chapter, we shall examine an important class of spectral methods—the Fourier Spectral Method—which uses trigonometric interpolation to approximate a solution. Rather than achieving m th order accuracy using a m -point finite difference approximation, Fourier spectral methods achieve **spectral accuracy**— $O(h^m)$ for all $m \geq 1$ —if the solution is smooth. If only the first p derivatives of the function exist, we have at most $O(h^{p+1})$.

Since we use trigonometric interpolation, we require that the solution be periodic. If we are not interested in modeling boundary interaction, this restriction is often acceptable. For non-periodic problems, absorbing “sponge” layers and reflecting boundaries can sometimes be added. One may also use other spectral method such as the Chebyshev spectral method (which uses Chebyshev polynomials) for non-periodic solutions.

5.1 Discrete Fourier Transform

Trigonometric Interpolation

Let the function $u(x)$ be periodic with period 2π , i.e., $u(x + 2\pi) = u(x)$. By rescaling x we can modify the discussion for any period $0 < s < \infty$. Consider a uniform grid $x_j = j \frac{2\pi}{2n+1}$ with $0 \leq j \leq 2n$. We can approximate u by a trigonometric polynomial with nodes at x_j

$$P_n(x) = \sum_{k=-n}^n c_k e^{+ikx} \quad \text{where} \quad u(x_j) = \sum_{k=-n}^n c_k e^{-ikx_j}.$$

Theorem 8. For all integers k

$$\sum_{j=0}^{2n} e^{ikx_j} = \begin{cases} 2n+1, & \text{if } k = 0, \\ 0, & \text{otherwise.} \end{cases}$$

Proof.

$$\sum_{j=0}^{2n} e^{ikx_j} = \sum_{j=0}^{2n} e^{ijx_k} = \sum_{j=0}^{2n} z_k^j = \begin{cases} \frac{z_k^{2n+1} - 1}{z_k - 1}, & \text{if } z_k \neq 1 \\ 2n+1, & \text{if } z_k = 1. \end{cases}$$

where $z_k = e^{ik2\pi/(2n+1)}$. Note that

$$z_k^{2n+1} = \left(e^{ik2\pi/(2n+1)} \right)^{2n+1} = e^{ik2\pi} = 1. \quad \square$$

Then

$$\sum_{j=0}^{2n} u(x_j) e^{-ikx_j} = \sum_{j=0}^{2n} \sum_{l=-n}^n c_k e^{i(l-k)x_j} = \sum_{l=-n}^n c_k \sum_{j=0}^{2n} e^{i(l-k)x_j} = c_k(2n+1)$$

$$\text{Discrete Fourier Transform} \quad c_k = \frac{1}{2n+1} \sum_{j=0}^{2n} u(x_j) e^{-ikx_j} \quad -n \leq k \leq n.$$

$$\text{Discrete Inverse Fourier Transform} \quad u(x_j) = \sum_{k=-n}^n c_k e^{ikx_j} \quad 0 \leq j \leq 2n.$$

If $f(x)$ is periodic with some period L , i.e., $u(x) = u(x+L)$, then

$$\begin{aligned} \int_{-\infty}^{\infty} e^{-ikx} f(x) dx &= \int_{-\infty}^{\infty} e^{-ikx} f(x+L) dx \\ &= \int_{-\infty}^{\infty} e^{-ik(x-L)} f(x) dx \\ &= e^{ikL} \int_{-\infty}^{\infty} e^{-ikx} f(x) dx \end{aligned}$$

It follows that $\hat{u}(k) = 0$ unless $k = 2\pi n/L$ for some integer n , and the inverse Fourier transform is simply

$$u(x) = \frac{1}{2\pi} \sum_{n=-\infty}^{\infty} e^{i2\pi nx/L} \hat{u}(2\pi n/L).$$

This gives us the Fourier series

$$\begin{aligned} c_k &= \frac{1}{L} \int_0^L f(x) e^{-ikx/L} dx \\ \text{Fourier Series} \quad f(x) &= \sum_{j=-\infty}^{\infty} c_k e^{ikx/L} \quad 0 \leq j \leq n \end{aligned}$$

Note: The discrete Fourier transform is the trapezoidal rule of integration applied to the Fourier transform:

$$\begin{array}{ll} \text{Fourier Transform} & \hat{u}(\xi) = \frac{1}{2\pi} \int_{-\infty}^{\infty} u(x) e^{-i\xi x} dx \\ \text{Inverse Fourier Transform} & u(x) = \int_{-\infty}^{\infty} \hat{u}(\xi) e^{i\xi x} d\xi \end{array}$$

Spectral differentiation

Let's compute the derivative for discrete Fourier transform $\{c_k\}$ of $\{u(x_j)\}$. Take the trigonometric polynomials approximation of $u(x)$

$$u_n(x) = \sum_{k=-n}^n c_k e^{ikx}$$

then

$$\frac{d}{dx} u_n(x) = \sum_{k=-n}^n c_k \frac{d}{dx} e^{ikx} = \sum_{k=-n}^n (ikc_k) e^{ikx}$$

So, $\text{DFT}[\frac{d}{dx} u_n(x)] = ik \text{DFT}[u_n(x)]$. The analogous statement for the Fourier Transform holds by integration by parts.

The discrete Fourier transform of $\frac{d}{dx} u$ is $ik\hat{u}$.

Example. Consider the heat equation $u_t = u_{xx}$ with initial conditions $u(0, x) = u_0(x)$. The Fourier Transform is

$$\hat{u}_t = -k^2 \hat{u}$$

which has the solution

$$\hat{u} = e^{-k^2 t} \hat{u}(0, k).$$

In physical space the solution is

$$u(t, x) = \text{IFT} \left[e^{-k^2 t} \text{FT} u_0(x) \right].$$

◦

Smoothness and spectral accuracy

Smooth functions have rapidly decaying Fourier transforms. If the Fourier transform of a function decays very rapidly, then the error introduced by discretization is small. These errors are due to aliasing of high wave numbers to lower wave numbers.

Theorem 9. Let $u \in L^2(\mathbb{R})$ have the Fourier transform $\hat{u}(k)$.

1. If u has $p-1$ continuous derivatives in $L^2(\mathbb{R})$ for some $p \geq 0$ and a p th derivative of bounded variation, then $\hat{u}(k) = O(|k|^{-(p+1)})$ as $k \rightarrow \infty$. A function u has a derivative of bounded variation if $\int_{\Omega} u\varphi' dx < \infty$ for all $\varphi \in C^\infty(\Omega)$. We call $\int_{\Omega} u\varphi' dx$ the weak derivative of u .
2. If u has infinitely many continuous derivatives in $L^2(\mathbb{R})$, then $\hat{u}(k) = O(|k|^{-m})$ for all $m \geq 0$.
3. If u_h is the polynomial approximation $|u(x) - u_h(x)| < O(h^{p+1})$ where $h < 1$ is the grid spacing.
4. If w_j is the m th derivative of $u(x)$ at x_j , then $|u^{(m)}(x_j) - w_j| < O(h^{p-m})$.

Heuristically,

$$\hat{u}(\xi) = \int_{-\infty}^{\infty} e^{-i\xi x} u(x) dx = \frac{1}{i\xi} \int_{-\infty}^{\infty} e^{-i\xi x} u'(x) dx = \frac{1}{(i\xi)^p} \int_{-\infty}^{\infty} e^{-i\xi x} u^{(p)}(x) dx.$$

For example, the rectangle function (a piecewise constant B-spline)

$$f(x) = \begin{cases} 1, & x \in [-\frac{1}{2}, \frac{1}{2}] \\ 0, & \text{otherwise} \end{cases} \quad \text{has the Fourier transform} \quad \hat{f}(\xi) = \text{sinc } \xi = \frac{\sin \xi}{\xi}$$

The rectangle function is not differentiable at $x = \pm\frac{1}{2}$. In fact, it is not even continuous at $x = \pm 1$. Its Fourier transform decays as $O(|\xi|^{-1})$ as $|\xi| \rightarrow \infty$. The hat function is formed by taking the convolution of the rectangle function with itself producing a linear B-spline. The hat function

$$g(x) = \begin{cases} 1+x, & x \in [-1, 0] \\ 1-x, & x \in [0, 1] \\ 0, & \text{otherwise} \end{cases} \quad \text{has the Fourier transform} \quad \hat{g}(\xi) = (\text{sinc } \xi)^2 = \frac{\sin^2 \xi}{\xi^2}$$

The hat function is not differentiable at $x = -1, 0, 1$ but it is weakly differentiable everywhere. Its Fourier transform decays as $O(|\xi|^{-2})$ as $|\xi| \rightarrow \infty$. As functions become smoother the Fourier transform decays faster. The same is true for a discrete Fourier transform.

Numerically, we can't take all frequencies and must be satisfied working with a trigonometric polynomial as our approximation:

$$f(x) = \sum_{k=-\infty}^{\infty} c_k e^{-ikx} = \sum_{k=-n}^n c_k e^{-ikx} + \tau_n = P_n(x) + \tau_n$$

where the truncation $|\tau_n| = O(n^{-(p+1)})$ where p denotes the number of times that $f(x)$ is differentiable. For discontinuous functions $|\tau_n| = O(n^{-1})$ and we get slow convergence. The trigonometry polynomial approximation results in a Gibb's phenomenon.

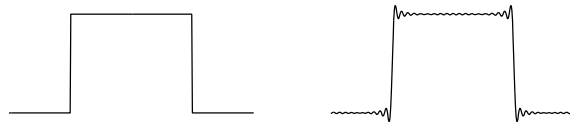


Figure 5.1: Gibbs's Phenomenon. The band-limited discrete Fourier transform of a discontinuous function results in oscillations.

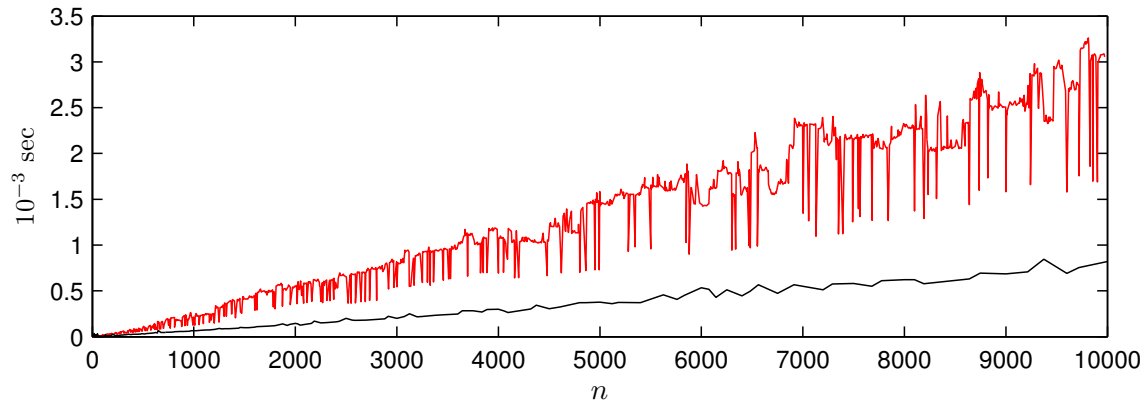


Figure 5.2: Computation time for to compute the DFT of random input vectors of size n using MATLAB's `fft`. The black curve shows composite number n of primes 2,3, and 5. For example, $720 = 2^4 \cdot 3^2 \cdot 5$. The red curve shows prime number n .

The FFT in MATLAB

Fourier spectral methods become important after 1965 with the Cooley-Tukey Fast Fourier Transform (FFT) algorithm (although Gauss had technically invented it around 1805). The DFT matrix is a full, scaled unitary matrix and direct multiplication (and inversion) by it requires $O(n^2)$ operations. The Cooley-Tukey algorithm reduces complexity to $O(n \log n)$ when n is a large composite number such as a power of 2. Computation can be fast even when n is not a composite number by exploiting properties of circulant and Toeplitz matrices. See Figure 5.2.

► The MATLAB function `fft` implements the Fastest Fourier Transform in the West (FFTW) library. The inverse FFT can be computed using `ifft` and multidimensional FFTs can be computed using `fftn` and `ifftn`.

The FFTW library has several FFT algorithms and chooses the fastest based on heuristics and trial. The first time that the library is called, it may try several different FFT algorithms for the same problem and thereafter use the fastest. The MATLAB command `fftw` can be used to tune the FFTW parameters. The FFTW library decomposes the problem using the composite Cooley-Tukey algorithm recursively until the problem can be solved using fixed-size codelets (which use split-radix, Cooley-Tukey, and a prime factor algorithm). If n is a prime number the FFTW library decomposes the problem using Rader

decomposition and then uses the Cooley-Tukey algorithm to compute the three $(n-1)$ -point DFTs.

5.2 Nonlinear Stiff PDES

Time Splitting

If the PDE combines a stiff linear term with a nonlinear term, we can consider time-splitting. This allows us to integrate the linear term exactly and avoid the stability issues. Consider the PDE

$$u_t = Lu + N(u) \quad \text{with} \quad u(0, x) = u_0(x)$$

where L be a stiff linear operator and N be a nonlinear operator. Take the simple splitting

1. $v_t = Lv$ where $v(0, x) = u_0(x)$
2. $w_t = Nw$ where $w(0, x) = v(t, x)$

then $u(t, x) \approx w(t, x)$ with $O(t^2)$ splitting error.

We can take the Fourier Transform of the linear equation $\hat{v}_t = \hat{L}\hat{v}$ which has the solution

$$\hat{v} = e^{t\hat{L}}\hat{v}(0, \xi) = e^{t\hat{L}}\hat{u}_0(\xi).$$

So, the solution $u(t, x) \approx w(t, x)$ where $w(t, x)$ solves

$$w_t = N(w) \quad \text{where} \quad w(t, 0) = v(t, x) \quad \text{with} \quad v(t, x) = \text{IFT} \left[e^{t\hat{L}} \text{FT} u(0, x) \right]$$

Recall from Chapter 1 that we may increase order of the method by using Strang splitting implementation.

Example. Consider the Allen-Cahn equation

$$u_t = u_{xx} + \varepsilon^{-1}u(1 - u^2)$$

with periodic boundary conditions. We can write the problem as

$$u_t = Lu + N(u)$$

where $Lu = (d^2/dx^2)u$ and $N(u) = \varepsilon^{-1}u(1 - u^2)$. The Fourier transform of d^2/dx^2 is $-\xi^2$ and the ODE

$$u_t = \varepsilon^{-1}u(1 - u^2)$$

has the analytical solution

$$u(\Delta t, x) = \frac{u_0}{\sqrt{u_0^2 - (u_0^2 - 1)e^{-2\varepsilon\Delta t}}}$$

where the initial conditions u_0 . Note that if the ODE did not have an analytic solution, we could have solved it numerically.

For each timestep t_n we have

1. $v(x) = \text{IFT} \left[e^{-\xi^2 \Delta t} \text{FT} [u(t_n, x)] \right]$
2. $u(t_{n+1}, x) = u_0 \left[u_0^2 - (u_0^2 - 1)e^{-2\varepsilon \Delta t} \right]^{-1/2}$ where $u_0 = v(x)$.

Each time step has a $O((\Delta t)^2)$ splitting error, so we have $O(\Delta t)$ error after n iterations. Remember that by using Strang splitting we can achieve $O((\Delta t)^2)$ error. \circ

Example. A electron in an external potential $V(x)$ is modeled quantum-mechanically by the Schrödinger equation

$$i\varepsilon u_t = -\frac{1}{2}\varepsilon^2 u_{xx} + V(x)u.$$

with scaled Planck constant ε . The Schrödinger equation can be rewritten as

$$u_t = \frac{1}{2}i\varepsilon u_{xx} - i\varepsilon^{-1}V(x)u.$$

We can separate the spatial operator into the kinetic and potential terms

$$\frac{1}{2}i\varepsilon u_{xx} \quad \text{and} \quad -i\varepsilon^{-1}V(x)u.$$

Both of these terms are linear in u and we can solve the problem using Strang splitting

$$u(x, t_{n+1}) = e^{-iV(x)\Delta t/2\varepsilon} \text{IFT} e^{-i\varepsilon k^2 \Delta t/2} \text{FT} e^{-iV(x)\Delta t/2\varepsilon} u(x, t_n)$$

giving a method that is spectral-order in space and second-order in time. \circ

Integrating Factors

We are unable to get better than second-order accuracy by using time-splitting. We can achieve arbitrary-order accuracy if we don't split the operators. Of course, we must use an implicit A-stable or L-stable method to avoid stability issues caused by the stiffness. And this can be difficult if we have nonlinear terms. We can use integrating factors to mollify the stiffness.

Let L be a linear operator and N be a nonlinear operator. Solve

$$u_t = Lu + N(u) \tag{5.1}$$

with initial conditions $u(0, x) = u_0(x)$ pseudo-spectrally using integrating factors.

We can rewrite (5.1) as

$$u_t - Lu = N(u).$$

Taking the Fourier transform, we get

$$\hat{u}_t - \hat{L}\hat{u} = \text{FT}[N(u)].$$

Multiple both sides of the equation by $\exp(-t\hat{L})$ and simplify to get

$$\left(e^{-t\hat{L}}\hat{u} \right)_t = e^{-t\hat{L}}\text{FT}[N(u)].$$

Let $\hat{w} = e^{-t\hat{L}}\hat{u}$. Then the solution

$$u(t, x) = \text{IFT} \left[e^{t\hat{L}} w(t, \xi) \right]$$

where

$$\frac{\partial}{\partial t} \hat{w}(t, \xi) = e^{-t\hat{L}} \text{FT} \left[N \left(\text{IFT} \left[e^{t\hat{L}} \hat{w}(t, \xi) \right] \right) \right] \quad (5.2)$$

with initial conditions

$$\hat{w} = e^{-t\hat{L}} \text{FT}[u_0(x)].$$

The benefit of such integrating factors is that there is no splitting error. But, integrating factors only mollify the impact of stiffness. For really stiff problems, a better approach is to use time-splitting. We can break the time interval into several increments $\Delta t = t_{n+1} - t_n$ and implement integrating factors over each interval.

In this case we have for each interval starting at t_n and ending at t_{n+1} the following:

$$\begin{aligned} \text{Set} \quad & \hat{w}(t_n, \xi) = e^{(t_n - t_n)\hat{L}} \text{FT} u(t_n, x) = \text{FT} u(t_n, x) \\ \text{Solve} \quad & \frac{\partial}{\partial t} \hat{w}(t, \xi) = e^{-(t - t_n)\hat{L}} \text{FT} \left[N \left(\text{IFT} \left[e^{(t - t_n)\hat{L}} \hat{w}(t, \xi) \right] \right) \right] \\ \text{Set} \quad & u(t_{n+1}, x) = \text{IFT} \left(e^{(t_{n+1} - t_n)\hat{L}} \hat{w}(t_{n+1}, \xi) \right) = \text{IFT} \left(e^{\Delta t \hat{L}} \hat{w}(t_{n+1}, \xi) \right) \end{aligned}$$

It's not necessary to switch back completely to the u variable. Instead we can solve the problem in the \hat{w} variable when we want to observe the solution.

1. Take $\hat{w} = \text{FT}[u(0, x)]$ as the initial conditions.
2. At each time step t_n to t_{n+1} , implement (5.2) with a time step Δt for initial conditions \hat{w} using an ODE solver such as a fourth-order Runge-Kutta to get a high-order method.
3. At the end of each time interval, take $\hat{w} = e^{\Delta t \hat{L}} \hat{w}$ as the initial condition for the next time step.
4. Take $u(t, x) = \text{IFT} \hat{w}$.

5.3 Incompressible Navier-Stokes equation

In this final section, we examine the two-dimensional incompressible Navier-Stokes equation. One-dimensional Fourier spectral methods can be easily modified to handle two- and three-dimensional problems by replacing the vector ξ with the tensor $\xi_x + \xi_y$ or $\xi_x + \xi_y + \xi_z$.

Consider the two-dimensional incompressible Navier-stokes equation

$$\mathbf{u}_t + \nabla \cdot (\mathbf{u} \otimes \mathbf{u}) = -\nabla p + \varepsilon \Delta \mathbf{u} \quad (5.3)$$

$$\nabla \cdot \mathbf{u} = 0 \quad (5.4)$$

which models the fluid flow. The vector $\mathbf{u} = (u, v)$ is the velocity field, $p(x, y)$ is the pressure and ε is the inverse of the Reynold's number. The symbol \otimes denotes a tensor

product— $\nabla \cdot (\mathbf{u} \otimes \mathbf{u})$ is $\sum_j \partial_j u_i u_j$ in index notation. Explicitly, the Navier-Stokes equation says

$$\begin{aligned} u_t + (u^2)_x + (uv)_y &= -p_x + \varepsilon(u_{xx} + u_{yy}) \\ v_t + (uv)_y + (v^2)_x &= -p_y + \varepsilon(v_{xx} + v_{yy}) \\ u_x + v_y &= 0 \end{aligned}$$

To solve the problem, we will find \mathbf{u} using the conservation of momentum equation (5.3) and use the conservation of mass (5.4) as a constraint by forcing \mathbf{u} to be divergence-free. Because (5.3) has stiff linear terms and nonlinear terms, we will use a second-order IMEX Adams-Bashforth/Crank-Nicholson method

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} = \underbrace{-\frac{1}{2}(\nabla p^{n+1} + \nabla p^n)}_{\text{Crank-Nicolson}} + \underbrace{\frac{3}{2}\mathbf{H}^n - \frac{1}{2}\mathbf{H}^{n-1}}_{\text{Adams-Bashforth}} + \underbrace{\frac{1}{2}\varepsilon(\Delta \mathbf{u}^{n+1} + \Delta \mathbf{u}^n)}_{\text{Crank-Nicolson}} \quad (5.5)$$

where $\mathbf{H}^n = \nabla \cdot (\mathbf{u}^n \otimes \mathbf{u}^n)$. There are a couple problems with this set-up. First, we don't explicitly know p^{n+1} . Second, we haven't enforced the constraint (5.3). So, let's modify the approach by introducing an intermediate solution \mathbf{u}^* . We'll be able to use the intermediate solution so that we don't explicitly need to compute the pressure p . By combining

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} = -\frac{1}{2}\nabla p^n + \frac{3}{2}\mathbf{H}^n - \frac{1}{2}\mathbf{H}^{n-1} + \frac{1}{2}\varepsilon(\Delta \mathbf{u}^* + \Delta \mathbf{u}^n) \quad (5.6a)$$

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^*}{\Delta t} = -\frac{1}{2}\nabla p^{n+1} + \frac{1}{2}\varepsilon(\Delta \mathbf{u}^{n+1} - \Delta \mathbf{u}^*) \quad (5.6b)$$

we have (5.5). First, solve (5.6a):

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} = -\frac{1}{2}\nabla p^n + \frac{3}{2}\mathbf{H}^n - \frac{1}{2}\mathbf{H}^{n-1} + \frac{1}{2}\varepsilon(\Delta \mathbf{u}^* + \Delta \mathbf{u}^n)$$

Formally, we have

$$(\frac{1}{\Delta t} - \frac{1}{2}\varepsilon\Delta)\mathbf{u}^* = -\frac{1}{2}\nabla p^n + \frac{3}{2}\mathbf{H}^n - \frac{1}{2}\mathbf{H}^{n-1} + (\frac{1}{\Delta t} + \frac{1}{2}\varepsilon\Delta)\mathbf{u}^n$$

from which we have that

$$\mathbf{u}^* = (\frac{1}{\Delta t} - \frac{1}{2}\varepsilon\Delta)^{-1} \left(-\frac{1}{2}\nabla p^n + \frac{3}{2}\mathbf{H}^n - \frac{1}{2}\mathbf{H}^{n-1} + (\frac{1}{\Delta t} + \frac{1}{2}\varepsilon\Delta)\mathbf{u}^n \right) \quad (5.7)$$

We need to determine ∇p^{n+1} in such a way as to implicitly enforce the Crank-Nicolson scheme (5.5) by using (5.6b) to get ∇p^{n+1}

$$-\frac{1}{2}\nabla p^{n+1} = (\frac{1}{\Delta t} - \frac{1}{2}\varepsilon\Delta)(\mathbf{u}^{n+1} - \mathbf{u}^*)$$

Equivalently,

$$-\frac{1}{2}\nabla p^n = (\frac{1}{\Delta t} - \frac{1}{2}\varepsilon\Delta)(\mathbf{u}^n - \mathbf{u}^{*-1}) \quad (5.8)$$

where \mathbf{u}^{*-1} is the intermediate solution at the previous timestep.

Substituting $-\frac{1}{2}\nabla p^n$ back into the first equation we have that

$$\begin{aligned}\mathbf{u}^* &= (\frac{1}{\Delta t} - \frac{1}{2}\varepsilon\Delta)^{-1} \left((\frac{1}{\Delta t} - \frac{1}{2}\varepsilon\Delta)(\mathbf{u}^n - \mathbf{u}^{*-1}) + \frac{3}{2}\mathbf{H}^n - \frac{1}{2}\mathbf{H}^{n-1} + (\frac{1}{\Delta t} + \frac{1}{2}\varepsilon\Delta)\mathbf{u}^n \right) \\ &= \mathbf{u}^n - \mathbf{u}^{*-1} + (\frac{1}{\Delta t} - \frac{1}{2}\varepsilon\Delta)^{-1} \left(\frac{3}{2}\mathbf{H}^n - \frac{1}{2}\mathbf{H}^{n-1} + (\frac{1}{\Delta t} + \frac{1}{2}\varepsilon\Delta)\mathbf{u}^n \right)\end{aligned}$$

Finally, enforce (5.4) by projecting the solution \mathbf{u}^* onto a divergence-free vector field. Note that if we define

$$\mathbf{u}^{n+1} = \mathbf{u}^* - \nabla\Delta^{-1}\nabla \cdot \mathbf{u}^*$$

Then

$$\nabla \cdot \mathbf{u}^{n+1} = \nabla \cdot \mathbf{u}^* - \nabla \cdot \nabla\Delta^{-1}\nabla \cdot \mathbf{u}^* = \nabla \cdot \mathbf{u}^* - \Delta\Delta^{-1}\nabla \cdot \mathbf{u}^* = 0$$

(You should recognize the projection operators $\mathbf{P}_\mathbf{A} = \mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$ and $\mathbf{I} - \mathbf{P}_\mathbf{A}$ in these equations.)

We now have

$$\begin{aligned}\mathbf{u}^* &= \mathbf{u}^n - \mathbf{u}^{*-1} + (\frac{1}{\Delta t} - \frac{1}{2}\varepsilon\Delta)^{-1} \left(\frac{3}{2}\mathbf{H}^n - \frac{1}{2}\mathbf{H}^{n-1} + (\frac{1}{\Delta t} + \frac{1}{2}\varepsilon\Delta)\mathbf{u}^n \right) \\ \mathbf{u}^{n+1} &= \mathbf{u}^* - \nabla\Delta^{-1}\nabla \cdot \mathbf{u}^*\end{aligned}$$

To initialize the multistep method, we'll take

$$\mathbf{u}^{-1} = \mathbf{u}^{*-1} = \mathbf{u}^0.$$

In the Fourier domain, we replace

$$\nabla \rightarrow i(\xi_x, \xi_y) \quad \text{and} \quad \Delta = \nabla \cdot \nabla \rightarrow -\xi_x^2 - \xi_y^2 = -|\xi|^2$$

and from (5.9) we get

$$\hat{\mathbf{u}}^* = \hat{\mathbf{u}}^n - \hat{\mathbf{u}}^{*-1} + (\frac{1}{\Delta t} + \frac{1}{2}\varepsilon|\xi|^2)^{-1} \left(\frac{3}{2}\hat{\mathbf{H}}^n - \frac{1}{2}\hat{\mathbf{H}}^{n-1} + (\frac{1}{\Delta t} - \frac{1}{2}\varepsilon|\xi|^2)\hat{\mathbf{u}}^n \right) \quad (5.10a)$$

$$\hat{\mathbf{u}}^{n+1} = \hat{\mathbf{u}}^* - \frac{(\xi \cdot \hat{\mathbf{u}}^*)}{|\xi|^2} \xi \quad (5.10b)$$

We must take some care when we divide by $|\xi|^2$ in (5.10b) because one of the components is zero. In this case we modify the term by replacing the zeros by ones. This “correction” is fixed when we subsequently multiply by ξ .

We haven't discussed implementation of the nonlinear operator $\hat{\mathbf{H}}$ yet. Note that for $\mathbf{u} = (u, v)$ we have that the x -component of \mathbf{H} is

$$H(u, v) = (uv)_y + (u^2)_x$$

with a similar form for a y -component. The Fourier transform of a product of two functions uv is the convolution $\hat{u} * \hat{v}$ —a very inefficient operator to implement numerically. Therefore, we'll evaluate \hat{H} in real space. Hence, we take

$$\hat{H}(\hat{u}, \hat{v}) = i\xi_y \text{FT}(\text{IFT}\hat{u}|\text{FT}\hat{v}) + i\xi_x \text{FT}((\text{IFT}\hat{u})^2)$$

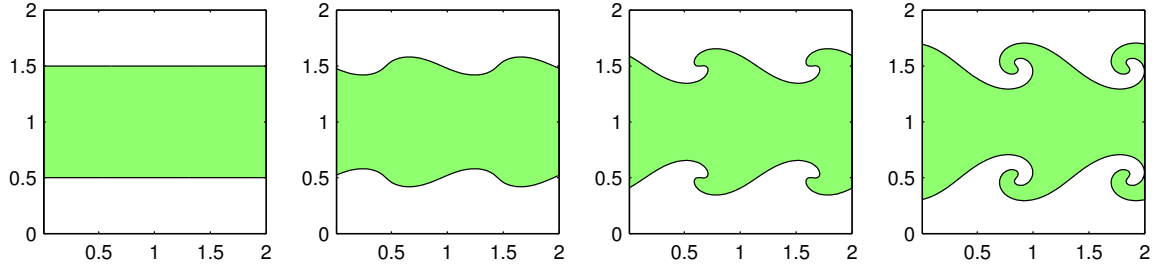


Figure 5.3: Kelvin-Helmoltz instability in solution to the Navier-Stokes equation at $t = 0, 0.4, 0.8$ and 1.2 .

To visualize the solution, we can use a tracer. Imagine dropping ink (or smoke) into a fluid and then following the motion of the ink. Numerically, we can simulate a tracer by solving the advection equation

$$Q_t + \mathbf{u} \cdot \nabla Q = 0 \quad (5.11)$$

using for example the Lax-Wendroff method where $\mathbf{u} = (u, v)$ is the solution to the Navier-Stokes equation at each time step.

Consider the stratified flow with shear

$$(u, v) = \left(\frac{1}{2}(1 + \tanh(10 - 5|1 - y|))(1 + \frac{1}{2} \sin 2\pi x), 0 \right)$$

The fluid is moving in the x -direction with two speeds separated by a narrow interface. See the first frame of Figure 5.3. Think of wind blowing over water. Such a situation leads to produces a Kelvin-Helmholtz instability, producing surface waves in water and leading to turbulence in other laminar fluids. See Figure 5.3.

In MATLAB we have the following

```
H = @(u,v,ikx,iky) ikx.*fft2(ifft2(u).^2) + iky.*fft2(ifft2(u).*ifft2(v));
flux = @(Q,c) c.*diff([Q(end,:);Q]) + 0.5*c.*(1-c).*diff([Q(end,:);Q;Q(1,:)],2);
L = 2; n = 128; e = 0.001; dt = .002;
k = [0:(n/2-1) 0 (-n/2+1):-1]*(2*pi/L);
[ikx,iky] = meshgrid(1i*k);
k2 = ikx.^2+iky.^2; k2i = k2; k2i(find(k2==0))=1;
x = (1:n)*(L/n); dx = x(2)-x(1); lambda = dt/dx;
[x,y] = meshgrid(x);
Q = 0.5*(1+tanh(10*(1-abs(L/2 -y)/(L/4))));
v = zeros(size(x)); u = Q.*(1+0.5*sin(L*pi*x));
u = fft2(u); v = fft2(v); us = u; vs = v;
Hx = H(u,v,ikx,iky); Hy = H(v,u,iky,ikx);
for i = 1:600
    Q = Q - flux(Q,lambda*real(ifft2(v))) - flux(Q',lambda*real(ifft2(u)))';
    image(U*100);colormap(jet(100));drawnow;
    Hxo = Hx; Hyo = Hy; Hx = H(u,v,ikx,iky); Hy = H(v,u,iky,ikx);
    us = u - us + (1.5*Hx - 0.5*Hxo + (1/dt + (e/2)*k2).*u)./(1/dt - (e/2)*k2);
    vs = v - vs + (1.5*Hy - 0.5*Hyo + (1/dt + (e/2)*k2).*v)./(1/dt - (e/2)*k2);
    phi = (ikx.*us + iky.*vs)./k2i;
    u = us - ikx.*phi;
```

```

v = vs - iky.*phi;
end

```

The variable `e` is used for the scaling parameter ε . `k` = ξ is a temporary variable used to construct `ikx` = $i\xi_x$, `iky` = $i\xi_y$, and `k2` = $-|\xi|^2 = -\xi_x^2 + \xi_y^2$. The variable `Q` gives the density of the tracer, which evolves using the advection equation (5.11) and solved with the Lax-Wendorff method. The flux is implemented with using the anonymous function `flux`. The variable `Hx`, computed using the function `H`, is the x -component of \mathbf{H}^n and `Hxo` x -component of \mathbf{H}^{n-1} . Similarly, `Hy` and `Hyo` are the y -components of \mathbf{H}^n and \mathbf{H}^{n-1} . The variables `u` and `v` are the x - and y -components of the velocity $\mathbf{u}^n = (u^n, v^n)$. Similarly, `uo`, `vo`, `us` and `vs` are u^{n-1} , v^{n-1} , u^* and v^* , respectively. The variable `phi` is a temporary variable used for $\Delta^{-1} \nabla \cdot \mathbf{u}^*$.

Exercises

5.1. Suppose that you use a fourth order Runge-Kutta scheme to solve the Burgers equation using a Fourier spectral method

$$u_t + \left(\frac{u^2}{2} \right)_x = 0.$$

What order can you expect? What else can you say about the numerical solution?

🔪 5.2. The Kortweg-deVries (KdV) equation

$$u_t + 3(u^2)_x + u_{xxx} = 0 \quad (5.12)$$

is a nonlinear equation used to model wave motion in a dispersive medium—for example, shallow water and fiber optics. The purpose of this exercise is to explore the behavior the KdV equation by first developing a high-order numerical scheme to solve the equation and then using the scheme to simulate soliton interaction.

Dispersion means that different frequencies travel at different speeds causing a solution to break up and spread out over time. For certain class of initial conditions, the nonlinearity of the KdV equation counteracts the dispersion and the solution maintains its shape. We call such a solution a solitary wave or a soliton. It can be easily shown that

$$u(x, t) = \frac{1}{2}c \operatorname{sech}^2 \left(\frac{\sqrt{c}}{2}(x - ct) \right)$$

is a soliton solution to the KdV equation. This solution is a traveling wave solution which simply moves with speed c and does not change shape.

Solve the KdV equation (5.2) on the interval $[-3, 3]$ for $t \in [0, 0.01]$ using the fourth-order Runge-Kutta method with integrating factors. Use perhaps $n = 256$ grid points in space over a domain $x \in [-20, 20]$. Take $\Delta t = 100/n^2$ to ensure stability. Let

$$\phi(x; x_0, c) = \frac{1}{2}c \operatorname{sech}^2 \left(\frac{\sqrt{c}}{2}(x - x_0) \right).$$

Observe the solutions to with initial conditions given by

1. $u(x, 0) = \phi(x; -4, 4)$
2. $u(x, 0) = \phi(x; -9, 9)$
3. $u(x, 0) = \phi(x; -4, 4) + \phi(x; -9, 9)$.

These initial conditions produce solitons with speeds 4 and 9 centered at $x = -4$ and $x = -9$, respectively. In the two soliton case, the two solitons should combine nonlinearly as the faster one overtakes the slower one. Comment on the behavior. You may also want to observe a three-soliton collision by adding another soliton (say $\phi(x; -12, 12)$). You may need to make the domain larger so that unstable oscillations at the domain boundaries do not pollute the results.

To implement the phase vector k in MATLAB use

```
k = [0:(n/2-1) 0 (-n/2+1):-1]*(2*pi/L);
```

where n is the number of grid points and L is the length of the domain.

You may want to use the following Runge-Kutta function:

```
function u = rk4(u,f,dt)
K1 = f(0,u);
K2 = f(0.5*dt,u+0.5*dt*K1);
K3 = f(0.5*dt,u+0.5*dt*K2);
K4 = f(dt,u+dt*K3);
u = u + dt*(K1+2*K2+2*K3+K4)/6;
```

5.3. The two-dimensional Swift-Hohenberg equation

$$u_t = \varepsilon u - u^3 - (\Delta + 1)^2 u$$

with $u \equiv u(x, y, t)$ models Rayleigh-Bernard convection, which results when a thin pan of water is heated from below. Compute the numerical solution for $\varepsilon = 0.8$ over a square with sides of length 200. Assume periodic boundary conditions with 256 gridpoints in x and y . Choose $u(x, y)$ randomly from the uniform distribution $[-1, 1]$. Output the solution at time $t = 50$ and several intermediate times.

Note: The Fourier transform of the Laplacian operator can be implemented as

```
k = [0:(n/2-1) 0 (-n/2+1):-1]*(2*pi/L);
[kx,ky] = meshgrid(k);
D2 = (1i*kx).^2 + (1i*ky).^2;
```

where n is the number of grid points and L is the length of the domain. A two dimensional FFT can be implemented in MATLAB with `fft2`

Note: the analytic solution to $y' = \varepsilon y - y^3$ is

$$y = \frac{y_0}{\sqrt{(1 - \varepsilon^{-1} y_0^2) e^{-2\varepsilon t} + \varepsilon^{-1} y_0^2}}.$$

Index

- A-stable, 5
- absolutely stable, 4
- absorbing boundary conditions, 33
- Adams-Bashforth method, 11
- Adams-Moulton method, 11
- ansatz, 54

- backward difference formula, 8
- Banach space, 83
- bilinear functional, 83
- Boundary value problem, 75
- Butcher tableau, 13

- CFL condition, 35, 49
- CFL number, 49
- characteristic decomposition, 57
- characteristic speed, 48
- characteristic speeds, 55
- compact support, 59
- complete, 83
- conservation law, 57
- conservative, 69
- consistent, 23
- convergent, 23
- convexity splitting, 46
- Courant–Friedrichs–Lewy condition, 35

- Dirichlet boundary condition, 31
- dispersion relation, 54
- dispersive, 55
- domain of dependence, 50

- energy method, 41
- entropy, 62
- entropy flux, 62
- explicit, 4

- finite volume method, 70
- flux, 57

- Galerkin, 77
- ghost points, 30
- global truncation error, 10
- group velocity, 55

- Hilbert space, 83
- hyperbolic, 55, 64

- implicit, 4
- inner product, 83

- L-stable, 18
- Lax entropy condition, 62
- left eigenvector, 66
- linear functional, 83
- Lipschitz continuous, 2
- load vector, 78
- local truncation error, 10

- maximum principle, 28
- method of characteristics, 48
- method of lines, 29
- Minimization problem, 76

- Neumann boundary condition, 31
- numerical dispersion, 54
- numerical method of lines, 1
- numerical viscosity, 54

- partition of unity, 77
- periodic boundary conditions, 33
- phase velocity, 55

- Raleigh-Ritz, 77
- Rankine-Hugoniot jump condition, 60

Riemann invariant, 65
Riemann problem, 60
Robin or mixed boundary condition, 33

self-similar solution, 61
shock, 58
Sobolev space, 84
spectral accuracy, 91
splitting, 19
stable, 2, 4, 23
stiff, 16
stiffness matrix, 78
Strang Splitting, 20

test function, 59
time-marching, 29
total potential energy, 75

unconditionally stable, 35
upwind method, 48

Variational problem, 76
vector space, 83
von Neumann stability analysis, 35

weak solution, 59
well-posed, 1, 84

zero-stability, 7