CHAPTER 16

Partial Differential Equations

CONTENTS

INTUITION for ordinary differential equations largely stems from the time evolution of physical systems. Equations like Newton's second law, determining the motion of physical objects over time, dominate the literature on ODE problems; additional examples come from chemical concentrations reacting over time, populations of predators and prey interacting from season to season, and so on. In each case, the initial configuration—e.g., the positions and velocities of particles in a system at time zero—is known, and the task is to predict behavior as time progresses. Derivatives only appear in a single time variable.

In this chapter, we entertain the possibility of *coupling* relationships between different derivatives of a function. It is not difficult to find examples where this coupling is necessary. When simulating gases or fluids, quantities like "pressure gradients," which encode the derivatives of pressure in *space*, figure into how material moves over *time*. These gradients appear since gases and fluids naturally move from high-pressure regions to low-pressure regions. In image processing, coupling the horizontal and vertical partial derivatives of an image can be used to describe its edges, characterize its texture, and so on.

Equations coupling together derivatives of functions in more than one variable are known as *partial differential equations*. They are the subject of a rich, nuanced theory worthy of

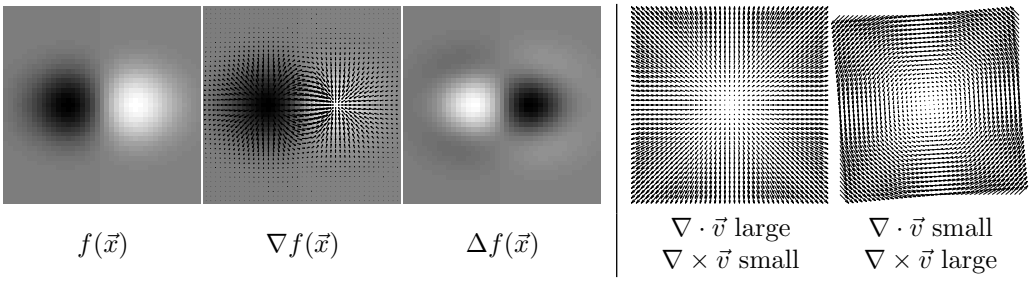


Figure 16.1 Vector calculus notation. On the left, we show a function $f(\vec{x})$ for $\vec{x} \in \mathbb{R}^2$ colored from black to white, its gradient ∇f, and its Laplacian $\nabla^2 f$; on the right are vector fields $\vec{v}(\vec{x})$ with different balances between divergence and curl.

larger-scale treatment, so we simply will summarize key ideas and provide sufficient material to approach problems commonly appearing in practice.

16.1 MOTIVATION

Partial differential equations (PDEs) provide one or more relationships between the partial derivatives of a function $f : \mathbb{R}^n \to \mathbb{R}^m$; the goal is to find an f satisfying the criteria. PDEs appear in nearly any branch of applied mathematics, and we list just a few below. Unlike in previous chapters, the algorithms in this chapter will be far from optimal with respect to accuracy or speed when applied to many of the examples. Our goals are to explore the vast space of problems that can be expressed as PDEs, to introduce the language needed to determine necessary numerical machinery, and to highlight key challenges and techniques for different classes of PDEs.

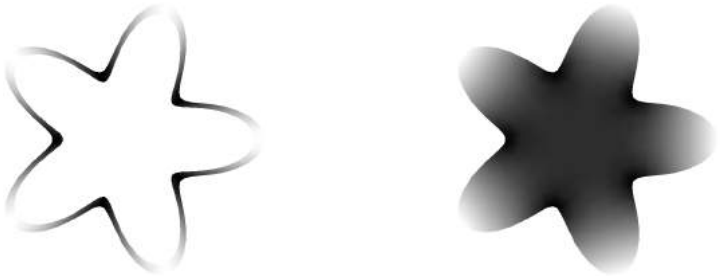
There are a few combinations of partial derivatives that appear often in the world of PDEs. If $f : \mathbb{R}^3 \to \mathbb{R}$ is a function and $\vec{v} : \mathbb{R}^3 \to \mathbb{R}^3$ is a vector field, then the following operators from vector calculus, illustrated in Figure 16.1, are worth remembering:

Name	**Notation**	**Definition**
Gradient	∇f	$\left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial x_3}\right)$
Divergence	$\nabla \cdot \vec{v}$	$\frac{\partial v_1}{\partial x_1} + \frac{\partial v_2}{\partial x_2} + \frac{\partial v_3}{\partial x_3}$
Curl	$\nabla \times \vec{v}$	$\left(\frac{\partial v_3}{\partial x_2} - \frac{\partial v_2}{\partial x_3}, \frac{\partial v_1}{\partial x_3} - \frac{\partial v_3}{\partial x_1}, \frac{\partial v_2}{\partial x_1} - \frac{\partial v_1}{\partial x_2}\right)$
Laplacian	$\nabla^2 f$	$\frac{\partial^2 f}{\partial x_1^2} + \frac{\partial^2 f}{\partial x_2^2} + \frac{\partial^2 f}{\partial x_3^2}$

For PDEs involving fluids, electrodynamics, and other physical quantities, by convention we think of the derivatives above as acting on the *spatial* variables (x, y, z) rather than the time variable t. For instance, the gradient of a function $f : (x, y, z; t) \to \mathbb{R}$ will be written $\nabla f \equiv (\partial f/\partial x, \partial f/\partial y, \partial f/\partial z)$; the partial derivative in time $\partial f/\partial t$ is treated separately.

Example 16.1 (Fluid simulation)**.** The flow of fluids and smoke is governed by the *Navier-Stokes equations*, a system of PDEs in many variables. Suppose a fluid is moving in a region $\Omega \subseteq \mathbb{R}^3$. We define the following quantities:

$t \in [0, \infty)$	Time
$\vec{v}(t) : \Omega \to \mathbb{R}^3$	Velocity
$p(t) : \Omega \to \mathbb{R}$	Pressure
$\vec{f}(t) : \Omega \to \mathbb{R}^3$	External forces (e.g., gravity)



Boundary conditions (on $\partial\Omega$) Laplace solution (on Ω)

Figure 16.2 Laplace's equation takes a function on the boundary $\partial\Omega$ of a domain $\Omega \subseteq \mathbb{R}^2$ (left) and interpolates it to the interior of Ω as smoothly as possible (right).

If the fluid has fixed viscosity μ and density ρ, then the (incompressible) Navier-Stokes equations state

$$\rho \cdot \left(\frac{\partial \vec{v}}{\partial t} + \vec{v} \cdot \nabla \vec{v} \right) = -\nabla p + \mu \nabla^2 \vec{v} + \vec{f} \qquad \text{with} \qquad \nabla \cdot \vec{v} = 0.$$

This system of equations determines the time dynamics of fluid motion and can be constructed by applying Newton's second law to tracking "particles" of fluid. Its statement involves derivatives in time $\partial/\partial t$ and derivatives in space ∇, making it a PDE.

Example 16.2 (Maxwell's equations)**.** Maxwell's equations determine the interaction between electric fields $\vec{E}$ and magnetic fields $\vec{B}$ over time. As with the Navier-Stokes equations, we think of the gradient, divergence, and curl operators as taking partial derivatives in space (x, y, z) and not time t. In a vacuum, Maxwell's system (in "strong" form) can be written:

$$\begin{aligned} \text{Gauss's law for electric fields: } & \nabla \cdot \vec{E} = \frac{\rho}{\varepsilon_0} \\ \text{Gauss's law for magnetism: } & \nabla \cdot \vec{B} = 0 \\ \text{Faraday's law: } & \nabla \times \vec{E} = -\frac{\partial \vec{B}}{\partial t} \\ \text{Ampère's law: } & \nabla \times \vec{B} = \mu_0 \left(\vec{J} + \varepsilon_0 \frac{\partial \vec{E}}{\partial t} \right) \end{aligned}$$

Here, ε_0 and μ_0 are physical constants and $\vec{J}$ encodes the density of electrical current. Just like the Navier-Stokes equations, Maxwell's equations relate derivatives of physical quantities in time t to their derivatives in space (given by curl and divergence terms).

Example 16.3 (Laplace's equation)**.** Suppose Ω is a domain in $\mathbb{R}^2$ with boundary $\partial\Omega$ and that we are given a function $g : \partial\Omega \to \mathbb{R}$, illustrated in Figure 16.2. We may wish to interpolate g to the interior of Ω as smoothly as possible. When Ω is an irregular shape, however, our strategies for interpolation from Chapter 13 can break down.

Take $f(\vec{x}) : \Omega \to \mathbb{R}$ to be an interpolating function satisfying $f(\vec{x}) = g(\vec{x})$ for all $\vec{x} \in \partial\Omega$. One metric for evaluating the quality of f as a smooth interpolant is to define an energy functional:

$$E[f] = \int_\Omega \|\nabla f(\vec{x})\|_2^2 \, d\vec{x}.$$

Here, the notation $E[\cdot]$ does not stand for "expectation" as it might in probability theory, but rather is an "energy" functional; it is standard notation in variational analysis. $E[f]$ measures the "total derivative" of f measured by taking the norm of its gradient and integrating this quantity over all of Ω. Wildly fluctuating functions f will have high values of $E[f]$ since the slope ∇f will be large in many places; smooth functions f, on the other hand, will have small $E[f]$ since their slope will be small everywhere.

We could ask that f interpolates g while being as smooth as possible in the interior of Ω using the following optimization:

$$\begin{aligned} &\text{minimize}_f \; E[f] \\ &\text{subject to } f(\vec{x}) = g(\vec{x}) \, \forall x \in \partial\Omega. \end{aligned}$$

This setup *looks* like optimizations we have solved elsewhere, but now our unknown is a function f rather than a point in $\mathbb{R}^n$.

If f minimizes E subject to the boundary conditions, then $E[f + h] \geq E[f]$ for all functions $h(\vec{x})$ with $h(\vec{x}) = 0$ for all $\vec{x} \in \partial\Omega$. This statement is true even for small perturbations $E[f + \varepsilon h]$ as $\varepsilon \to 0$. Subtracting $E[f]$, dividing by ε, and taking the limit as $\varepsilon \to 0$, we must have $\frac{d}{d\varepsilon} E[f + \varepsilon h]|_{\varepsilon=0} = 0$; this expression is akin to setting directional derivatives of a function equal to zero to find its minima. We can simplify:

$$\begin{aligned} E[f + \varepsilon h] &= \int_\Omega \|\nabla f(\vec{x}) + \varepsilon \nabla h(\vec{x})\|_2^2 \, d\vec{x} \\ &= \int_\Omega (\|\nabla f(\vec{x})\|_2^2 + 2\varepsilon \nabla f(\vec{x}) \cdot \nabla h(\vec{x}) + \varepsilon^2 \|\nabla h(\vec{x})\|_2^2) \, d\vec{x}. \end{aligned}$$

Differentiating with respect to ε shows

$$\begin{aligned} \frac{d}{d\varepsilon} E[f + \varepsilon h] &= \int_\Omega (2\nabla f(\vec{x}) \cdot \nabla h(\vec{x}) + 2\varepsilon \|\nabla h(\vec{x})\|_2^2) \, d\vec{x} \\ \implies \frac{d}{d\varepsilon} E[f + \varepsilon h]|_{\varepsilon=0} &= 2 \int_\Omega [\nabla f(\vec{x}) \cdot \nabla h(\vec{x})] \, d\vec{x}. \end{aligned}$$

Applying integration by parts and recalling that h is zero on $\partial\Omega$,

$$\frac{d}{d\varepsilon} E[f + \varepsilon h]|_{\varepsilon=0} = -2 \int_\Omega h(\vec{x}) \nabla^2 f(\vec{x}) \, d\vec{x}.$$

This expression must equal zero for *all* perturbations h that are zero on $\partial\Omega$. Hence, $\nabla^2 f(\vec{x}) = 0$ for all $\vec{x} \in \Omega \backslash \partial\Omega$ (a formal proof is outside of the scope of our discussion).

We have shown that the boundary interpolation problem above amounts to solving the following PDE:

$$\begin{aligned} \nabla^2 f(\vec{x}) &= 0 \; \forall \vec{x} \in \Omega \backslash \partial\Omega \\ f(\vec{x}) &= g(\vec{x}) \; \forall \vec{x} \in \partial\Omega. \end{aligned}$$

This PDE is known as *Laplace's equation*.

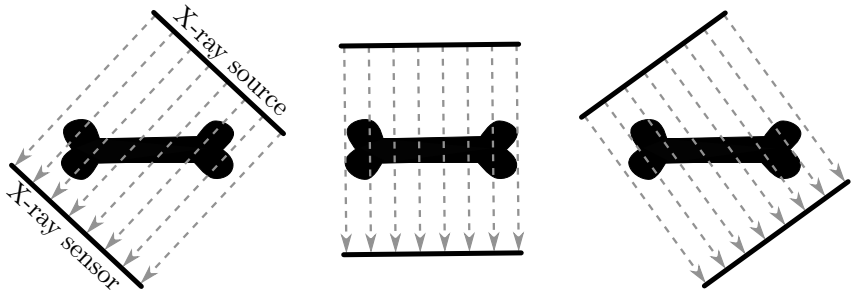


Figure 16.3 A CT scanner passes x-rays through an object; sensors on the other side collect the energy that made it through, giving the integrated density of the object along the x-ray path. Placing the source and sensor in different rotated poses allows for reconstruction of the pointwise density function.

Example 16.4 (X-ray computerized tomography)**.** Computerized tomography (CT) technology uses x-rays to see inside an object without cutting through it. The basic model is shown in Figure 16.3. Essentially, by passing x-rays through an object, the density of the object integrated along the x-ray path can be sensed by collecting the proportion that makes it through to the other side.

Suppose the density of an object is given by a function $\rho : \mathbb{R}^3 \to \mathbb{R}^+$. For any two points $\vec{x}, \vec{y} \in \mathbb{R}^3$, we can think of a CT scanner abstractly as a device that can sense the integral u of ρ along the line connecting $\vec{x}$ and $\vec{y}$:

$$u(\vec{x}, \vec{y}) \equiv \int_{-\infty}^{\infty} \rho(t\vec{x} + (1-t)\vec{y})\, dt.$$

The function $u : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}^+$ is known as the *Radon transform* of ρ.

Suppose we take a second derivative of u in an $\vec{x}$ and then a $\vec{y}$ coordinate:

$$\begin{aligned}
\frac{\partial}{\partial x_i} u(\vec{x}, \vec{y}) &= \int_{-\infty}^{\infty} \frac{\partial}{\partial x_i} \rho(t\vec{x} + (1-t)\vec{y})\, dt \text{ by definition of } u \\
&= \int_{-\infty}^{\infty} t\vec{e}_i \cdot \nabla\rho(t\vec{x} + (1-t)\vec{y})\, dt \\
\implies \frac{\partial^2}{\partial y_j \partial x_i} u(\vec{x}, \vec{y}) &= \int_{-\infty}^{\infty} \frac{\partial}{\partial y_j} t\vec{e}_i \cdot \nabla\rho(t\vec{x} + (1-t)\vec{y})\, dt \\
&= \int_{-\infty}^{\infty} t(1-t)\vec{e}_i^{\top} H_\rho(t\vec{x} + (1-t)\vec{y})\vec{e}_j\, dt \text{ for Hessian } H_\rho \text{ of } \rho.
\end{aligned}$$

An identical set of steps shows that the derivative $\frac{\partial^2 u}{\partial x_j \partial y_i}$ equals the same expression after applying symmetry of H_ρ. That is, u satisfies the following relationship:

$$\frac{\partial^2 u}{\partial y_j \partial x_i} = \frac{\partial^2 u}{\partial x_j \partial y_i}.$$

This equality, known as the Fritz John equation [68], gives information about u without involving the unknown density function ρ. In a computational context, it can be used to fill in data missing from incomplete x-ray scans or to smooth data from a potentially noisy x-ray sensor before reconstructing ρ.

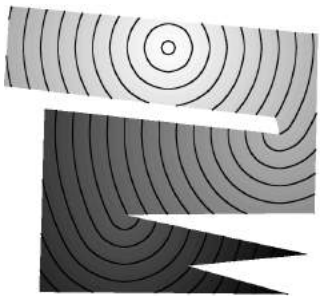


Figure 16.4 Shortest-path distances constrained to move within the interior of a non-convex shape have to wrap around corners; level sets of the distance function (shown as black lines) are no longer circles beyond these corner points.

Example 16.5 (Eikonal equation)**.** Suppose Ω is a closed region in $\mathbb{R}^n$. For a fixed point $\vec{x}_0 \in \Omega$, we might wish to find a function $d(\vec{x}) : \Omega \to \mathbb{R}^+$ measuring the length of the shortest path from $\vec{x}_0$ to $\vec{x}$ restricted to move only within Ω. When Ω is convex, we can write d in closed form as

$$d(\vec{x}) = \|\vec{x} - \vec{x}_0\|_2.$$

As illustrated in Figure 16.4, however, if Ω is non-convex or is a complicated domain like a surface, these distance functions become more challenging to compute. Solving for d, however, is a critical step for tasks like planning paths of robots by minimizing the distance they travel while avoiding obstacles marked on a map.

If Ω is non-convex, away from singularities, the function $d(\vec{x})$ still satisfies a derivative condition known as the *eikonal equation*:

$$\|\nabla d\|_2 = 1.$$

Intuitively, this PDE states that a distance function should have unit rate of change everywhere. As a sanity check, this relationship is certainly true for the absolute value function $|x - x_0|$ in one dimension, which measures the distance along the real line between x_0 and x. This equation is nonlinear in the derivative ∇d, making it a particularly challenging problem to solve for $d(\vec{x})$.

Specialized algorithms known as *fast marching methods* and *fast sweeping methods* estimate $d(\vec{x})$ over all of Ω by integrating the eikonal equation. Many algorithms for approximating solutions to the eikonal equation have structure similar to Dijkstra's algorithm for computing shortest paths along graphs; see Exercise 16.8 for one example.

Example 16.6 (Harmonic analysis)**.** Different objects respond differently to vibrations, and in large part these responses are functions of the *geometry* of the objects. For example, cellos and pianos can play the same note, but even an inexperienced listener can distinguish between the sounds they make.

From a mathematical standpoint, we can take $\Omega \subseteq \mathbb{R}^3$ to be a shape represented either as a surface or a volume. If we clamp the edges of the shape, then its *frequency spectrum* is given by eigenvalues coming from the following problem:

$$\begin{aligned} \nabla^2\phi &= \lambda\phi \\ \phi(\vec{x}) &= 0 \ \forall \vec{x} \in \partial\Omega, \end{aligned}$$

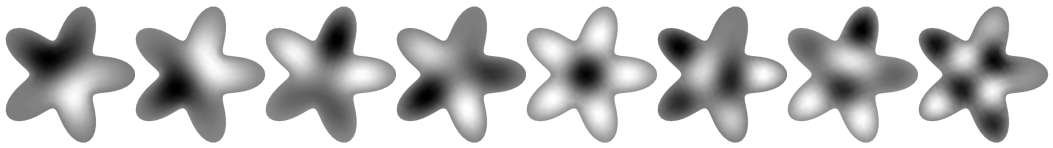


Figure 16.5 The first eight eigenfunctions ϕ_i of the Laplacian operator of the domain Ω from Figure 16.2, which satisfy $\nabla^2\phi_i = \lambda_i\phi_i$ in order of increasing frequency; we omit ϕ_1, which is the constant function with $\lambda = 0$.

where ∇^2 is the Laplacian of Ω and $\partial\Omega$ is the boundary of Ω. Figure 16.5 shows examples of these functions on a two-dimensional domain Ω.

Relating to the one-dimensional theory of waves, $\sin kx$ solves this problem when Ω is the interval $[0, 2\pi]$ and $k \in \mathbb{Z}$. To check, the Laplacian in one dimension is $\partial^2/\partial x^2$, and thus

$$\begin{aligned}\frac{\partial^2}{\partial x^2}\sin kx &= \frac{\partial}{\partial x}k\cos kx \\ &= -k^2\sin kx \\ \sin(k\cdot 0) &= 0 \\ \sin(k\cdot 2\pi) &= 0.\end{aligned}$$

That is, the eigenfunctions are $\sin kx$ with eigenvalues $-k^2$.

16.2 STATEMENT AND STRUCTURE OF PDES

Vocabulary used to describe PDEs is extensive, and each class of PDEs has substantially different properties from the others in terms of solvability, theoretical understanding of solutions, and discretization challenges. Our main focus eventually will be on developing algorithms for a few common tasks rather than introducing the general theory of continuous or discretized PDE, but it is worth acknowledging the rich expressive possibilities—and accompanying theoretical challenges—that come with using PDE language to describe numerical problems.

Following standard notation, in our subsequent development we will assume that our unknown is some function $u(\vec{x})$. For ease of notation, we will use subscript notation to denote partial derivatives:

$$u_x \equiv \frac{\partial u}{\partial x}, \qquad u_y \equiv \frac{\partial u}{\partial y}, \qquad u_{xy} \equiv \frac{\partial^2 u}{\partial x \partial y},$$

and so on.

16.2.1 Properties of PDEs

Just as ODEs couple the time derivatives of a function, PDEs typically are stated as relationships between two or more partial derivatives of u. By examining the algebraic form of a PDE, we can check if it has any of a number of properties, including the following:

- *Homogeneous* (e.g., $x^2u_{xx} + u_{xy} - u_y + u = 0$): The PDE can be written using linear combinations of u and its derivatives; the coefficients can be scalar values or func-

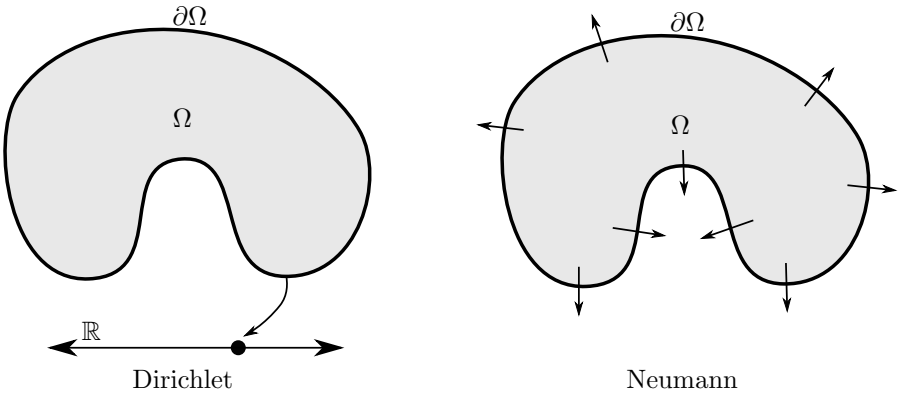


Figure 16.6 Dirichlet boundary conditions prescribe the values of the unknown function u on the boundary $\partial\Omega$ of the domain Ω, while Neumann conditions prescribe the derivative of u orthogonal to $\partial\Omega$.

tions of the independent variables. The equation can be nonlinear in the independent variables (x and y in our example).

- *Linear* (e.g., $u_{xx} - yu_{yy} + u = xy^2$): Similar to homogeneous PDE, but potentially with a nonzero (inhomogeneous) right-hand side built from scalars or the dependent variables. PDEs like the eikonal equation (or $u_{xx}^2 = u_{xy}$) are considered *nonlinear* because they are nonlinear in u.
- *Quasi-linear* (e.g., $u_{xy} + 2u_{xx} + u_y^2 + u_x^2 = y$): The statement is linear in the highest-order derivatives of u.
- *Constant-coefficient* (e.g., $u_{xx} + 3u_y = u_z$): The coefficients of u and its derivatives are not functions of the independent variables.

One potentially surprising observation about the properties above is that they are more concerned with the role of u than those of the independent variables like x, y, and z. For instance, the definition of a "linear" PDE allows u to have coefficients that are nonlinear functions of these variables. While this may make the PDE appear nonlinear, it is still linear in the unknowns, which is the distinguishing factor.

The *order* of a PDE is the order of its highest derivative. Most of the PDEs we consider in this chapter are second-order and already present considerable numerical challenges. Methods analogous to reduction of ODEs to first order (§15.2.1) can be carried out but do not provide as much benefit for solving PDEs.

16.2.2 Boundary Conditions

ODEs typically are considered *initial-value problems*, because given a configuration that is known at the initial time $t = 0$, they evolve the state forward indefinitely. With few exceptions, the user does not have to provide information about the state for $t > 0$.

PDE problems also can be *boundary-value problems* rather than or in addition to being initial value problems. Most PDEs require information about behavior at the boundary of the domain of all the variables. For instance, Laplace's equation, introduced in Example 16.3, requires fixed values on the boundary $\partial\Omega$ of Ω. Similarly, the heat equation used to

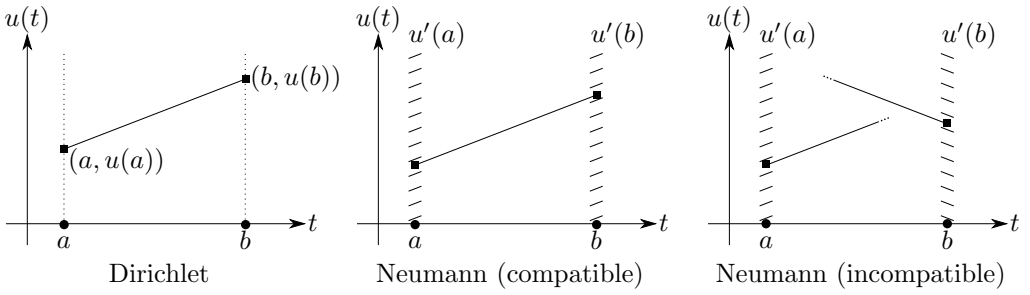


Figure 16.7 Boundary conditions for the PDE $u_{tt} = 0$ from Example 16.7.

simulate conductive material like metals admits a number of possible boundary conditions, corresponding to whether the material is attached to a heat source or dispersing heat energy into the surrounding space.

If the unknown of a PDE is a function $u : \Omega \to \mathbb{R}$ for some domain $\Omega \subseteq \mathbb{R}^n$, typical boundary conditions include the following:

- *Dirichlet conditions* directly specify the values of $u(\vec{x})$ for all $\vec{x} \in \partial\Omega$.
- *Neumann conditions* specify the derivative of $u(\vec{x})$ in the direction orthogonal to $\partial\Omega$.
- *Mixed* or *Robin conditions* specify a relationship between the value and normal derivatives of $u(\vec{x})$ on $\partial\Omega$.

The first two choices are illustrated in Figure 16.6.

Improperly encoding boundary conditions is a subtle oversight that creeps into countless discretizations of PDEs. There are many sources of confusion that explain this common issue. Different discretizations of the same boundary conditions can yield qualitatively different outputs from a PDE solver if they are expressed improperly. Indeed, some boundary conditions are not realizable even in theory, as illustrated in the example below.

Example 16.7 (Boundary conditions in one dimension)**.** Suppose we are solving the following PDE (more precisely an ODE, although the distinction here is not relevant) in one variable t over the interval $\Omega = [a, b]$:

$$u_{tt} = 0.$$

From one-variable calculus, that solutions must take the form $u(t) = \alpha t + \beta$.

Consider the effects of assorted choices of boundary conditions on $\partial\Omega = \{a, b\}$, illustrated in Figure 16.7:

- Dirichlet conditions specify the values $u(a)$ and $u(b)$ directly. There is a unique line that goes through any pair of points $(a, u(a))$ and $(b, u(b))$, so a solution to the PDE always exists and is unique in this case.
- Neumann conditions specify $u'(a)$ and $u'(b)$. From the general form of $u(t)$, $u'(t) = \alpha$, reflecting the fact that lines have constant slope. Neumann conditions specifying different values for $u'(a)$ and $u'(b)$ are *incompatible* with the PDE itself. Compatible Neumann conditions, on the other hand, specify $u'(a) = u'(b) = \alpha$ but are satisfied for any choice of β.

16.3 MODEL EQUATIONS

In §15.2.3, we studied properties of ODEs and their integrators by examining the model equation $y' = ay$. We can pursue a similar analytical technique for PDEs, although we will have to separate into multiple special cases to cover the qualitative phenomena of interest.

We will focus on the linear, constant-coefficient, homogeneous case. As mentioned in §16.2.1, the non-constant coefficient and inhomogeneous cases often have similar qualitative behavior, and nonlinear PDEs require special consideration beyond the scope of our discussion. We furthermore will study *second-order* systems, that is, systems containing at most the second derivative of u. While the model ODE $y' = ay$ is first-order, a reasonable model PDE needs at least two derivatives to show how derivatives in different directions interact.

Linear, constant-coefficient, homogeneous second-order PDEs have the following general form, for unknown function $u : \mathbb{R}^n \to \mathbb{R}$:

$$\sum_{ij} a_{ij} \frac{\partial u}{\partial x_i \partial x_j} + \sum_i b_i \frac{\partial u}{\partial x_i} + cu = 0.$$

To simplify notation, we can define a formal "gradient operator" as the vector of derivatives

$$\nabla \equiv \left(\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, \ldots, \frac{\partial}{\partial x_n} \right).$$

Expressions like ∇f, $\nabla \cdot \vec{v}$, and $\nabla \times \vec{v}$ agree with the definitions of gradients, divergence, and curl on $\mathbb{R}^3$ using this formal definition of ∇. In this notation, the model PDE takes a matrix-like form:

$$(\nabla^\top A \nabla + \nabla \cdot \vec{b} + c)u = 0.$$

The operator $\nabla^\top A \nabla + \nabla \cdot \vec{b} + c$ acting on u abstractly looks like a quadratic form in ∇ as a vector; since partial derivatives commute, we can assume A is symmetric.

The definiteness of A determines the *class* of the model PDE, just as the definiteness of a matrix determines the convexity of its associated quadratic form. Four cases bring about qualitatively different behavior for u:

- If A is *positive or negative definite*, the system is *elliptic*.
- If A is *positive or negative semidefinite*, the system is *parabolic*.
- If A has only one eigenvalue of different sign from the rest, the system is *hyperbolic*.
- If A satisfies none of these criteria, the system is *ultrahyperbolic*.

These criteria are listed approximately in order of the difficulty level of solving each type of equation. We consider the first three cases below and provide examples of corresponding behavior by specifying different matrices A; ultrahyperbolic equations do not appear as often in practice and require highly specialized solution techniques.

16.3.1 Elliptic PDEs

Positive definite linear systems can be solved using efficient algorithms like Cholesky decomposition and conjugate gradients that do not necessarily work for indefinite matrices. Similarly, elliptic PDEs, for which A is positive definite, have strong structure that makes them the most straightforward equations to characterize and solve, both theoretically and computationally.

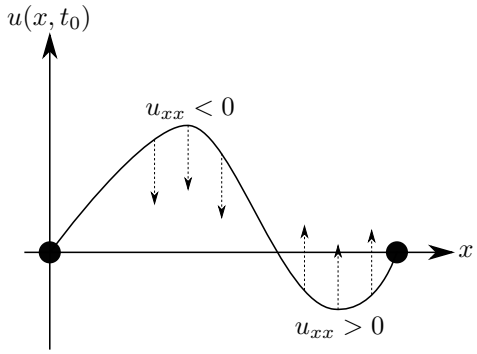


Figure 16.8 The heat equation in one variable $u_t = \alpha u_{xx}$ decreases u over time where it is curved down and increases u over time where u is curved up, as measured using the second derivative in space u_{xx}. Here, we show a solution of the heat equation $u(x, t)$ at a fixed time t_0; the arrows indicate how values of u will change as t advances.

The model elliptic PDE is the *Laplace equation*, given by $\nabla^2 u = 0$, as in Example 16.3. In two variables, the Laplace equation is written

$$u_{xx} + u_{yy} = 0.$$

Figure 16.2 illustrated a solution of the Laplace equation, which essentially interpolates information from the boundary of the domain of u to its interior.

Elliptic equations are well-understood theoretically and come with strong properties characterizing their behavior. Of particular importance is *elliptic regularity*, which states that solutions of elliptic PDEs automatically are differentiable to higher order than their building blocks. Physically, elliptic equations characterize stable equilbria like the rest pose of a stretched rubber sheet, which naturally resists kinks and other irregularities.

16.3.2 Parabolic PDEs

Positive *semi*definite linear systems are only marginally more difficult to deal with than positive definite ones, at least if their null spaces are known and relatively small. Positive semidefinite matrices have null spaces that prevent them from being invertible, but orthogonally to the null space they behave identically to definite matrices. In PDE, these systems correspond to *parabolic* equations, for which A is positive semidefinite.

The heat equation is the model parabolic PDE. Suppose $u_0(x, y)$ is a fixed distribution of temperature in some region $\Omega \subseteq \mathbb{R}^2$ at time $t = 0$. Then, the heat equation determines how heat diffuses over time $t > 0$ as a function $u(t; x, y)$:

$$u_t = \alpha(u_{xx} + u_{yy}),$$

where $\alpha > 0$. If $\nabla = (\partial/\partial x, \partial/\partial y)$, the heat equation can be written $u_t = \alpha \nabla^2 u$. There is no second derivative in time t, making the equation parabolic rather than elliptic.

Figure 16.8 provides a phenomenological interpretation of the heat equation in one variable $u_t = \alpha u_{xx}$. The second derivative $\nabla^2 u$ measures the convexity of u. The heat equation increases u with time when its value is "cupped" upward, and decreases u otherwise. This

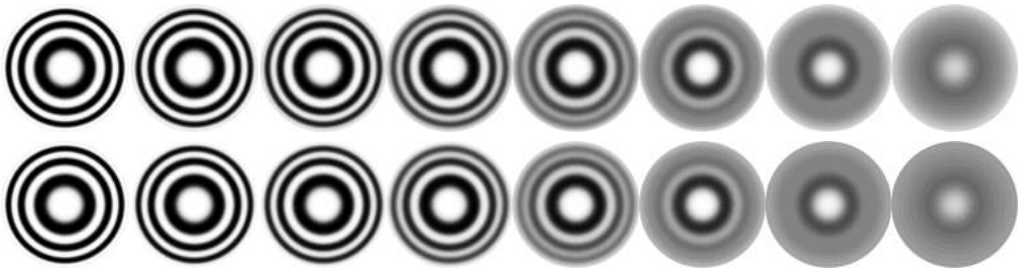


Figure 16.9 Solution to the heat equation $u_t = u_{xx} + u_{yy}$ on the unit circle with Dirichlet (top) and Neumann (bottom) boundary conditions. Solutions are colored from -1 (black) to 1 (white).

negative feedback is stable and leads to equilibrium as $t \to \infty$. Example solutions to the heat equation with different boundary conditions are shown in Figure 16.9.

The corresponding second-order term matrix A for the heat equation is:

$$A = \begin{array}{c} \\ t \\ x \\ y \end{array} \begin{array}{c} \begin{array}{ccc} t & x & y \end{array} \\ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{array}.$$

The heat equation is parabolic since this matrix has eigenvalues 0, 1, and 1.

There are two boundary conditions needed for the heat equation, both of which have physical interpretations:

- The distribution of heat $u(0; x, y) \equiv u_0(x, y)$ at time $t = 0$ at all points $(x, y) \in \Omega$.
- Behavior of u when $t > 0$ at boundary points $(x, y) \in \partial\Omega$. Dirichlet conditions fix $u(t; x, y)$ for all $t \geq 0$ and $(x, y) \in \partial\Omega$, e.g., if Ω is a piece of foil sitting next to a heat source like an oven whose temperature is controlled externally. Neumann conditions specify the derivative of f in the direction normal to the boundary $\partial\Omega$; they correspond to fixing the *flux* of heat out of Ω caused by different types of insulation.

16.3.3 Hyperbolic PDEs

The final model equation is the wave equation, corresponding to the indefinite matrix case:

$$u_{tt} = c^2(u_{xx} + u_{yy}).$$

The wave equation is hyperbolic because the second derivative in time t has opposite sign from the two spatial derivatives when all terms involving u are isolated on the same side. This equation determines the motion of waves across an elastic medium like a rubber sheet. It can be derived by applying Newton's second law to points on a piece of elastic, where x and y are positions on the sheet and $u(t; x, y)$ is the height of the piece of elastic at time t.

Figure 16.10 illustrates a solution of the wave equation with Dirichlet boundary conditions; these boundary conditions correspond to the vibrations of a drum whose outer boundary is fixed. As illustrated in the example, wave behavior contrasts considerably with

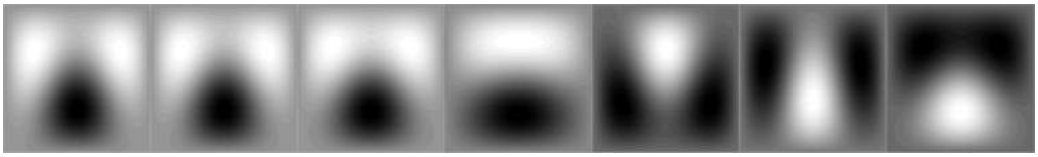


Figure 16.10 The wave equation on a square with Dirichlet boundary conditions; time is sampled evenly and progresses left to right. Color is proportional to the height of the wave, from -1 (black) to 1 (white).

heat diffusion in that as $t \to \infty$ the energy of the system does not disperse; waves can bounce back and forth across a domain indefinitely. For this reason, implicit integration strategies may not be appropriate for integrating hyperbolic PDEs because they tend to damp out motion.

Boundary conditions for the wave equation are similar to those of the heat equation, but now we must specify both $u(0; x, y)$ and $u_t(0; x, y)$ at time zero:

- The conditions at $t = 0$ specify the position and velocity of the wave at the start time.
- Boundary conditions on $\partial\Omega$ determine what happens at the ends of the material. Dirichlet conditions correspond to fixing the sides of the wave, e.g., plucking a cello string that is held flat at its two ends on the instrument. Neumann conditions correspond to leaving the ends of the wave untouched, like the end of a whip.

16.4 REPRESENTING DERIVATIVE OPERATORS

A key intuition that underlies many numerical techniques for PDEs is the following:

Derivatives act on functions in the same way that sparse matrices act on vectors.

Our choice of notation reflects this parallel: The derivative $d/dx[f(x)]$ looks like the product of an operator d/dx and a function f.

Formally, differentiation is a *linear operator* like matrix multiplication, since for all smooth functions $f, g : \mathbb{R} \to \mathbb{R}$ and scalars $a, b \in \mathbb{R}$,

$$\frac{d}{dx}(af(x) + bg(x)) = a\frac{d}{dx}f(x) + b\frac{d}{dx}g(x).$$

The derivatives act on functions, which can be thought of as points in an infinite-dimensional vector space. Many arguments from Chapter 1 and elsewhere regarding the linear algebra of matrices extend to this case, providing conditions for invertibility, symmetry, and so on of these abstract operators.

Nearly all techniques for solving linear PDEs make this analogy concrete. For example, recall the model equation $(\nabla^\top A\nabla + \nabla \cdot \vec{b} + c)u = 0$ subject to Dirichlet boundary conditions $u|_{\partial\Omega} = u_0$ for some fixed function u_0. We can define an operator $R_{\partial\Omega} : C^\infty(\Omega) \to C^\infty(\partial\Omega)$, that is, an operator taking functions on Ω and returning functions on its boundary $\partial\Omega$,

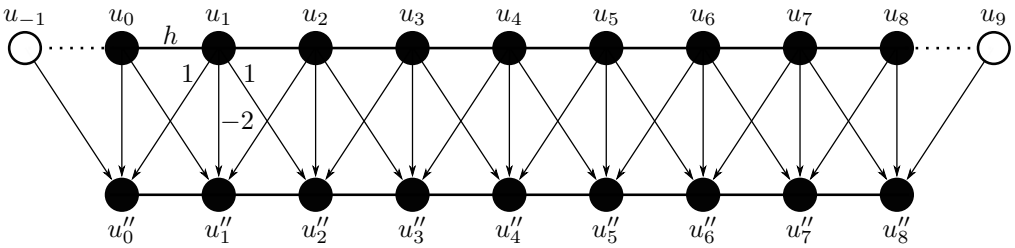


Figure 16.11 The one-dimensional finite difference Laplacian operator L takes samples u_i of a function $u(x)$ and returns an approximation of u'' at the same grid points by combining neighboring values using weights (1)—(−2)—(1); here $u(x)$ is approximated using nine samples $u_0, \ldots, u_8$. Boundary conditions are needed to deal with the unrepresented quantities at the white endpoints.

by restriction: $[R_{\partial\Omega}u](\vec{x}) \equiv u(\vec{x})$ for all $\vec{x} \in \partial\Omega$. Then, the model PDE and its boundary conditions can be combined in matrix-like notation:

$$\begin{pmatrix} (\nabla^\top A \nabla + \nabla \cdot \vec{b} + c) \\ R_{\partial\Omega} \end{pmatrix} u = \begin{pmatrix} 0 \\ u_0 \end{pmatrix}.$$

In this sense, we wish to solve $Mu = w$ where M is a linear operator. If we discretize M as a matrix, then recovering the solution u of the original equation is as easy as writing

$$\text{“}u = M^{-1}w.\text{”}$$

Many discretizations exist for M and u, often derived from the discretizations of derivatives introduced in §14.3. While each has subtle advantages, disadvantages, and conditions for effectiveness or convergence, in this section we provide constructions and high-level themes from a few popular techniques. Realistically, a legitimate and often-applied technique for finding the best discretization for a given application is to try a few and check empirically which is the most effective.

16.4.1 Finite Differences

Consider a function $u(x)$ on $[0, 1]$. Using the methods from Chapter 14, we can approximate the second derivative $u''(x)$ as

$$u''(x) = \frac{u(x+h) - 2u(x) + u(x-h)}{h^2} + O(h^2).$$

In the course of solving a PDE in u, assume $u(x)$ is discretized using $n+1$ evenly spaced samples $u_0, u_1, \ldots, u_n$, as in Figure 16.11, and take h to be the spacing between samples, satisfying $h = 1/n$. Applying the formula above provides an approximation of u'' at each grid point:

$$u_k'' \approx \frac{u_{k+1} - 2u_k + u_{k-1}}{h^2}.$$

That is, the second derivative of a function on a grid of points can be estimated using the (1)—(−2)—(1) stencil illustrated in Figure 16.12.

Boundary conditions are needed to compute u_0'' and u_n'' since we have not included u_{-1} or u_{n+1} in our discretization. Keeping in mind that $u_0 = u(0)$ and $u_n = u(1)$, we can incorporate them as follows:

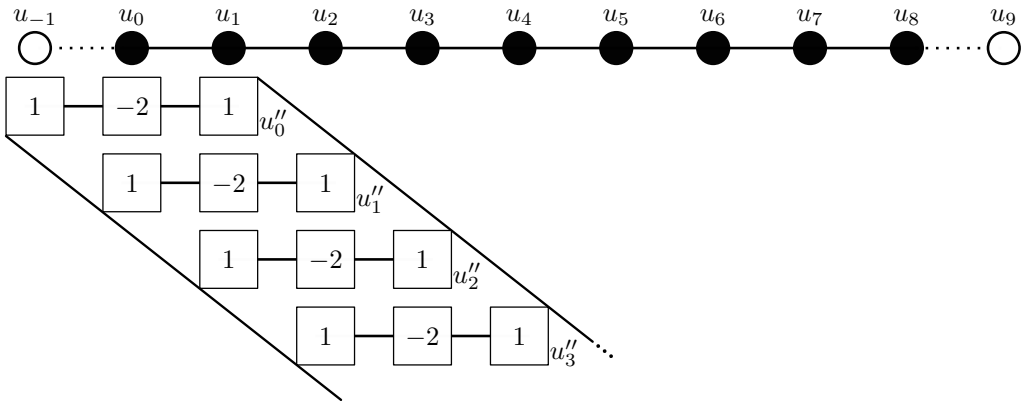


Figure 16.12 The one-dimensional finite difference Laplacian can be thought of as dragging a (1)—(−2)—(1) stencil across the domain.

- *Dirichlet:* $u_{-1} \equiv u_{n+1} = 0$, that is, fix the value of u beyond the endpoints to be zero.
- *Neumann:* $u_{-1} = u_0$ and $u_{n+1} = u_n$, encoding the condition $u'(0) = u'(1) = 0$.
- *Periodic:* $u_{-1} = u_n$ and $u_{n+1} = u_0$, making the identification $u(0) = u(1)$.

Suppose we stack the samples u_k into a vector $\vec{u} \in \mathbb{R}^{n+1}$ and the samples u_k'' into a second vector $\vec{w} \in \mathbb{R}^{n+1}$. The construction above shows that $h^2\vec{w} = L\vec{u}$, where L is one of the choices below:

Dirichlet

$$\begin{pmatrix} -2 & 1 & & & & \\ 1 & -2 & 1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & 1 & -2 & 1 & \\ & & & 1 & -2 & \end{pmatrix}$$

Neumann

$$\begin{pmatrix} -1 & 1 & & & & \\ 1 & -2 & 1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & 1 & -2 & 1 & \\ & & & 1 & -1 & \end{pmatrix}$$

Periodic

$$\begin{pmatrix} -2 & 1 & & & & 1 \\ 1 & -2 & 1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & 1 & -2 & 1 & \\ 1 & & & 1 & -2 & \end{pmatrix}$$

The matrix L can be thought of as a discretized version of the operator $\frac{d^2}{dx^2}$ acting on $\vec{u} \in \mathbb{R}^{n+1}$ rather than functions $u : [0,1] \to \mathbb{R}$.

In two dimensions, we can use a similar approximation of the Laplacian $\nabla^2 u$ of $u : [0,1] \times [0,1] \to \mathbb{R}$. Now, we sample using a grid of values shown in Figure 16.13. In this case, $\nabla^2 u = u_{xx} + u_{yy}$, so we sum up x and y second derivatives constructed in the one-dimensional example above. If we number our samples as $u_{k,\ell} \equiv u(kh, \ell h)$, then the formula for the Laplacian of u becomes

$$(\nabla^2 u)_{k,\ell} \approx \frac{u_{(k-1),\ell} + u_{k,(\ell-1)} + u_{(k+1),\ell} + u_{k,(\ell+1)} - 4u_{k,\ell}}{h^2}.$$

This approximation implies a (1)—(−4)—(1) stencil over a 3×3 box. If we once again combine our samples of u and ∇u into $\vec{u}$ and $\vec{w}$, respectively, then $h^2\vec{w} = L_2\vec{y}$ where L_2 comes from the stencil we derived. This two-dimensional grid Laplacian L_2 appears in many image processing applications, where (k, ℓ) is used to index pixels on an image.

Regardless of dimension, given a discretization of the domain and a Laplacian matrix L, we can approximate solutions of elliptic PDEs using linear systems of equations. Consider the Poisson equation $\nabla^2 u = w$. After discretization, given a sampling $\vec{w}$ of $w(\vec{x})$, we can obtain an approximation $\vec{u}$ of the solution by solving $L\vec{u} = h^2\vec{w}$.

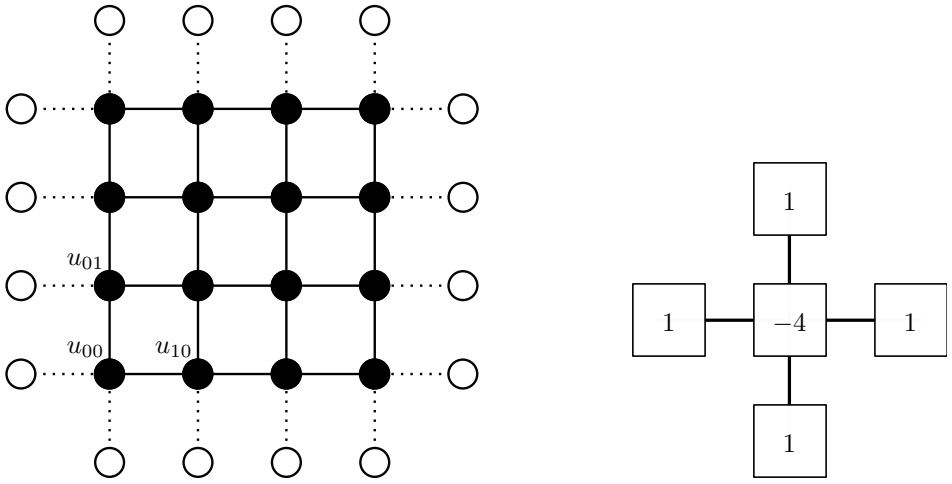


Figure 16.13 For functions $u(x, y)$ discretized on a two-dimensional grid (left), the Laplacian L_2 has a (1)—(−4)—(1) stencil.

This approach can be extended to inhomogeneous boundary conditions. For example, if we wish to solve $\nabla^2 u = w$ on a two-dimensional grid subject to Dirichlet conditions prescribed by a function u_0, we can solve the following linear system of equations for $\vec{u}$:

$$u_{k,\ell} = u_0(kh, lh) \text{ when } k \in \{0, n\} \text{ or } \ell \in \{0, n\}$$
$$u_{(k-1),\ell} + u_{k,(\ell-1)} + u_{(k+1),\ell} + u_{k,(\ell+1)} - 4u_{k,\ell} = 0 \text{ otherwise.}$$

This system of equations uses the 3×3 Laplacian stencil for vertices in the interior of $[0, 1]^2$ while explicitly fixing the values of u on the boundary.

These discretizations exemplify the *finite differences* method of discretizing PDEs, usually applied when the domain can be approximated using a grid. The finite difference method essentially treats the divided difference approximations from Chapter 14 as linear operators on grids of function values and then solves the resulting discrete system of equations.

Quoting results from Chapter 14 directly, however, comprises a serious breach of notation. When we write that an approximation of $u'(x)$ or $u''(x)$ holds to $O(h^k)$, we implicitly assume that $u(x)$ is sufficiently differentiable. Hence, what we need to show is that the result of solving systems like $L\vec{u} = h^2\vec{w}$ produces a $\vec{u}$ that actually approximates samples from a smooth function $u(x)$ rather than oscillating crazily. The following example shows that this issue is practical rather than theoretical, and that reasonable but non-convergent discretizations can fail catastrophically.

Example 16.8 (Lack of convergence)**.** Suppose we again sample a function $u(x)$ of one variable and wish to solve an equation that involves a first-order u' term. Interestingly, this task can be more challenging than solving second-order equations.

First, if we define u'_k as the forward difference $\frac{1}{h}(u_{k+1} - u_k)$, then we will be in the unnaturally asymmetric position of needing a boundary condition at u_n but not at u_0 as shown in Figure 16.14. Backward differences suffer from the reverse problem.

We might attempt to solve this problem and simultaneously gain an order of accuracy by using the symmetric difference $u'_k \approx \frac{1}{2h}(u_{k+1} - u_{k-1})$, but this discretization suffers

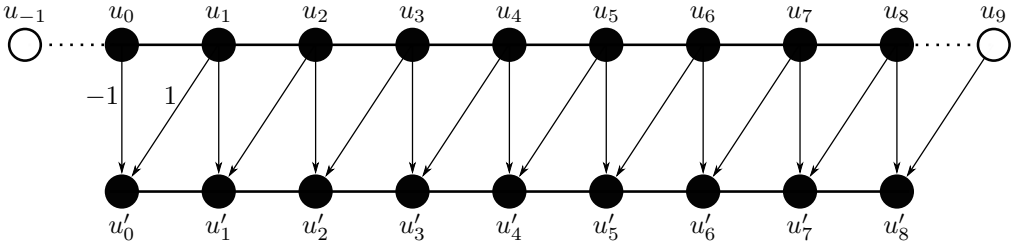


Figure 16.14 Forward differencing to approximate $u'(x)$ asymmetrically requires boundary conditions on the right but not the left.

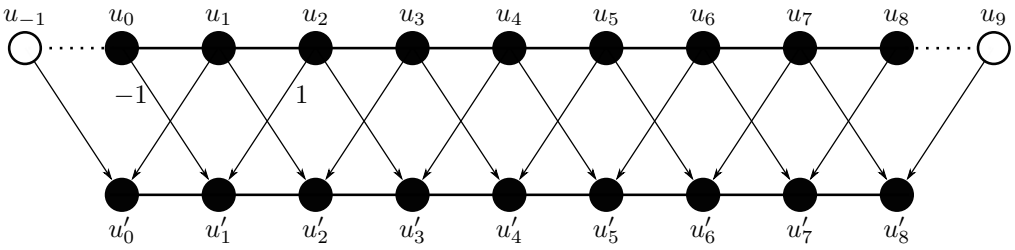


Figure 16.15 Centered differencing yields a symmetric approximation of $u'(x)$, but u'_k is not affected by the value of u_k using this formula.

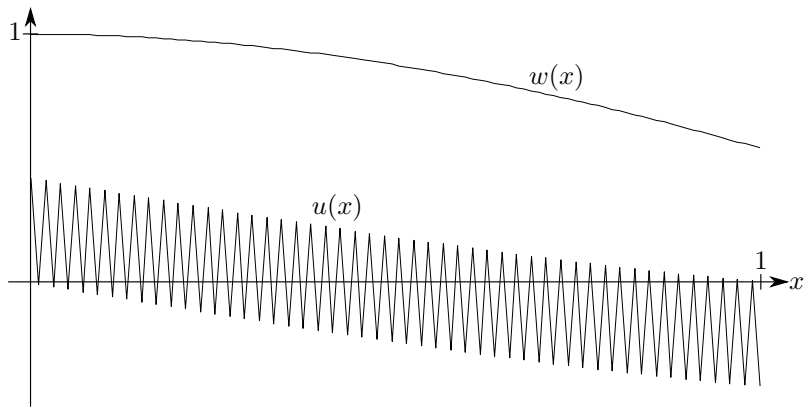


Figure 16.16 Solving $u'(x) = w(x)$ for $u(x)$ using a centered difference discretization suffers from the fencepost problem; odd- and even-indexed values of u have completely separate behavior. As more gridpoints are added in x, the resulting $u(x)$ does not converge to a smooth function, so $O(h^k)$ estimates of derivative quality do *not* apply.

from a more subtle *fencepost problem* illustrated in Figure 16.15. In particular, this version of u'_k ignores the value of u_k itself and only looks at its neighbors u_{k-1} and u_{k+1}. This oversight means that u_k and u_ℓ are treated differently depending on whether k and ℓ are even or odd. Figure 16.16 shows the result of attempting to solve a numerical problem with this discretization; the result is not differentiable.

As with the leapfrog integration algorithm in §15.4.2, one way to avoid these issues is to think of the derivatives as living on *half* gridpoints. In the one-dimensional case, this change corresponds to labeling the difference $\frac{1}{h}(y_{k+1} - y_k)$ as $y'_{k+1/2}$. This technique of placing different derivatives on vertices, edges, and centers of grid cells is particularly common in fluid simulation, which often maintains pressures, fluid velocities, and other physical quantities at locations suggested by the discretization.

16.4.2 Collocation

A challenge when working with finite differences is that we must justify that the end result "looks like" the theoretical solution we are seeking to approximate. That is, we have replaced a continuous unknown $u(\vec{x})$ with a sampled proxy on a grid but may inadvertently lose the connection to continuous mathematics in the process; Example 16.8 showed one example where a discretization is not convergent and hence yields unusable output. To avoid these issues, many numerical PDE methods attempt to make the connection between continuous and discrete less subtle.

One way to link continuous and discrete models is to write $u(\vec{x})$ in a basis $\phi_1, \ldots, \phi_k$:

$$u(\vec{x}) \approx \sum_{i=1}^{k} a_i \phi_i(\vec{x}).$$

This strategy should be familiar, as it underlies machinery for interpolation, quadrature, and differentiation. The philosophy here is to find coefficients $a_1, \ldots, a_k$ providing the *best possible approximation* of the solution to the continuous problem in the ϕ_i basis. As we add more functions ϕ_i to the basis, in many cases the approximation will converge to the theoretical solution, so long as the ϕ_i's eventually cover the relevant part of function space.

Perhaps the simplest method making use of this new construction is the *collocation* method. In the presence of k basis functions, this method samples k points $\vec{x}_1, \ldots, \vec{x}_k \in \Omega$ and requires that the PDE holds exactly at these locations. For example, if we wish to solve the Poisson equation $\nabla^2 u = w$, then for each $i \in \{1, \ldots, k\}$ we write

$$w(\vec{x}_i) = \nabla^2 u(\vec{x}_i) = \sum_{j=1}^{k} a_j \nabla^2 \phi_j(\vec{x}_i).$$

The only unknown quantities in this expression are the a_j's, so it can be used to write a square linear system for the vector $\vec{a} \in \mathbb{R}^k$ of coefficients. It can be replaced with a least-squares problem if more than k points are sampled in Ω.

Collocation requires a choice of basis functions $\phi_1, \ldots, \phi_k$ and a choice of *collocation points* $\vec{x}_1, \ldots, \vec{x}_k$. Typical basis functions include full or piecewise polynomial functions and trigonometric functions. When the ϕ_i's are compactly supported, that is, when $\phi_i(\vec{x}) = 0$ for most $\vec{x} \in \Omega$, the resulting system of equations is sparse. Collocation outputs a set of coefficients rather than a set of function values as in finite differences. Since the basis functions do not have to have any sort of grid structure, it is well-suited to non-rectangular domains, which can provide some challenge for finite differencing.

A drawback of collocation is that it does not regularize the behavior of the approximation $u(\vec{x})$ between the collocation points. Just as interpolating a polynomial through a set of sample points can lead to degenerate and in some cases highly oscillatory behavior between the samples, the collocation method must be used with caution to avoid degeneracies, for instance by optimizing the choice of basis functions and collocation points. Another option is to use a method like finite elements, considered below, which integrates behavior of an approximation over more than one sample point at a time.

16.4.3 Finite Elements

Finite element discretizations also make use of basis functions but do so by examining integrated quantities rather than pointwise values of the unknown function $u(\vec{x})$. This type of discretization is relevant to simulating a wide variety of phenomena and remains a popular choice in a diverse set of fields including mechanical engineering, digital geometry processing, and cloth simulation.

As an example, suppose that $\Omega \subseteq \mathbb{R}^2$ is a region on the plane and that we wish to solve the Dirichlet equation $\nabla^2 u = 0$ in its interior. Take any other function $v(\vec{x})$ satisfying $v(\vec{y}) = 0$ for all $\vec{y} \in \partial\Omega$. If we solve the PDE for u successfully, then the function $u(\vec{x})$ will satisfy the relationship

$$\int_\Omega v(\vec{x})\nabla^2 u(\vec{x})\,d\vec{x} = \int_\Omega v(\vec{x}) \cdot 0\,d\vec{x} = 0,$$

regardless of the choice of $v(\vec{x})$.

We can define a bilinear operator $\langle u, v\rangle_{\nabla^2}$ as the integral

$$\langle u, v\rangle_{\nabla^2} \equiv \int_\Omega v(\vec{x})\nabla^2 u(\vec{x})\,d\vec{x}.$$

Any function $u(\vec{x})$ for which $\langle u, v\rangle_{\nabla^2} = 0$ for all reasonable $v : \Omega \to \mathbb{R}$ defined above is called a *weak solution* to the Dirichlet equation. The functions v are known as *test functions*.

A remarkable observation suggests that weak solutions to PDEs may exist even when a strong solution does not. When $v(\vec{x})$ vanishes on $\partial\Omega$, the *divergence theorem* from multivariable calculus implies the following alternative form for $\langle u, v\rangle_{\nabla^2}$:

$$\langle u, v\rangle_{\nabla^2} = -\int_\Omega \nabla u(\vec{x}) \cdot \nabla v(\vec{x})\,d\vec{x}.$$

We used a similar step in Example 16.3 to derive Laplace's equation. Whereas the Laplacian ∇^2 in the Dirichlet equation requires the second derivative of u, this expression only requires u to be *once* differentiable. In other words, we have expressed a second-order PDE in first-order language. Furthermore, this form of $\langle \cdot, \cdot\rangle_{\nabla^2}$ is symmetric and negative semidefinite, in the sense that

$$\langle u, u\rangle_{\nabla^2} = -\int_\Omega \|\nabla u(\vec{x})\|_2^2\,d\vec{x} \leq 0.$$

Our definition of weak PDE solutions above is far from formal, since we were somewhat cavalier about the *space* of functions we should consider for u and v. Asking that $\langle u, v\rangle_{\nabla^2} = 0$ for all possible functions $v(\vec{x})$ is an unreasonable condition, since the space of all functions includes many degenerate functions that may not even be integrable. For the theoretical study of PDEs, it is usually sufficient to assume v is sufficiently smooth and has small support. Even with this restriction, however, the space of functions is far too large to be discretized in any reasonable way.

The *finite elements method* (FEM), however, makes the construction above tractable by restricting functions to a finite basis. Suppose we approximate u in a basis $\phi_1(\vec{x}), \ldots, \phi_k(\vec{x})$ by writing $u(\vec{x}) \approx \sum_{i=1}^{k} a_i \phi_i(\vec{x})$ for unknown coefficients $a_1, \ldots, a_k$. Since the actual solution $u(\vec{x})$ of the PDE is unlikely to be expressible in this form, we cannot expect $\langle \sum_i a_i \phi_i, v \rangle_{\nabla^2} = 0$ for all test functions $v(\vec{x})$. Hence, we not only approximate $u(\vec{x})$ but also restrict the class of test functions $v(\vec{x})$ to one in which we are more likely to be successful.

The best-known finite element approximation is the *Galerkin method*. In this method, we require that $\langle u, v \rangle_{\nabla^2} = 0$ for all test functions v that also can be written in the ϕ_i basis. By linearity of $\langle \cdot, \cdot \rangle_{\nabla^2}$, this method amounts to requiring that $\langle u, \phi_i \rangle_{\nabla^2} = 0$ for all $i \in \{1, \ldots, k\}$. Expanding this relationship shows

$$\begin{aligned} \langle u, \phi_i \rangle_{\nabla^2} &= \left\langle \sum_j a_j \phi_j, \phi_i \right\rangle_{\nabla^2} \text{ by our approximation of } u \\ &= \sum_j a_j \langle \phi_i, \phi_j \rangle_{\nabla^2} \text{ by linearity and symmetry of } \langle \cdot, \cdot \rangle_{\nabla^2}. \end{aligned}$$

Using this final expression, we can recover the vector $\vec{a} \in \mathbb{R}^k$ of coefficients by solving the following linear system of equations:

$$\begin{pmatrix} \langle \phi_1, \phi_1 \rangle_{\nabla^2} & \langle \phi_1, \phi_2 \rangle_{\nabla^2} & \cdots & \langle \phi_1, \phi_k \rangle_{\nabla^2} \\ \langle \phi_2, \phi_1 \rangle_{\nabla^2} & \langle \phi_2, \phi_2 \rangle_{\nabla^2} & \cdots & \langle \phi_2, \phi_k \rangle_{\nabla^2} \\ \vdots & \vdots & \ddots & \vdots \\ \langle \phi_k, \phi_1 \rangle_{\nabla^2} & \langle \phi_k, \phi_2 \rangle_{\nabla^2} & \cdots & \langle \phi_k, \phi_k \rangle_{\nabla^2} \end{pmatrix} \vec{a} = \vec{0},$$

subject to the proper boundary conditions. For example, to impose nonzero Dirichlet boundary conditions, we can fix those values a_i corresponding to elements on the boundary $\partial\Omega$.

Approximating solutions to the Poisson equation $\nabla^2 u = w$ can be carried out in a similar fashion. If we write $w = \sum_i b_i \phi_i$, then Galerkin's method amounts to writing a slightly modified linear system of equations. The weak form of the Poisson equation has the same left-hand side but now has a nonzero right-hand side:

$$\int_\Omega v(\vec{x}) \nabla^2 u(\vec{x})\, d\vec{x} = \int_\Omega v(\vec{x}) w(\vec{x})\, d\vec{x},$$

for all test functions $v(\vec{x})$. To apply Galerkin's method in this case, we not only approximate $u(\vec{x}) = \sum_i a_i \phi_i(\vec{x})$ but also assume the right-hand side $w(\vec{x})$ can be written $w(\vec{x}) = \sum_i b_i \phi_i(\vec{x})$. Then, solving the weak Poisson equation in the ϕ_i basis amounts to solving:

$$\begin{pmatrix} \langle \phi_1, \phi_1 \rangle_{\nabla^2} & \langle \phi_1, \phi_2 \rangle_{\nabla^2} & \cdots & \langle \phi_1, \phi_k \rangle_{\nabla^2} \\ \langle \phi_2, \phi_1 \rangle_{\nabla^2} & \langle \phi_2, \phi_2 \rangle_{\nabla^2} & \cdots & \langle \phi_2, \phi_k \rangle_{\nabla^2} \\ \vdots & \vdots & \ddots & \vdots \\ \langle \phi_k, \phi_1 \rangle_{\nabla^2} & \langle \phi_k, \phi_2 \rangle_{\nabla^2} & \cdots & \langle \phi_k, \phi_k \rangle_{\nabla^2} \end{pmatrix} \vec{a} = \begin{pmatrix} \langle \phi_1, \phi_1 \rangle & \langle \phi_1, \phi_2 \rangle & \cdots & \langle \phi_1, \phi_k \rangle \\ \langle \phi_2, \phi_1 \rangle & \langle \phi_2, \phi_2 \rangle & \cdots & \langle \phi_2, \phi_k \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle \phi_k, \phi_1 \rangle & \langle \phi_k, \phi_2 \rangle & \cdots & \langle \phi_k, \phi_k \rangle \end{pmatrix} \vec{b},$$

where $\langle f, g \rangle \equiv \int_\Omega f(\vec{x}) g(\vec{x})\, d\vec{x}$, the usual inner product of functions. The matrix next to $\vec{a}$ is known as the *stiffness matrix*, and the matrix next to $\vec{b}$ is known as the *mass matrix*. This is still a linear system of equations, since $\vec{b}$ is a fixed input to the Poisson equation.

Finite element discretizations like Galerkin's method boil down to choosing appropriate spaces for approximation solutions u and test functions v. Once these spaces are chosen,

the mass and stiffness matrices can be worked out offline, either in closed form or by using a quadrature method as explained in Chapter 14. These matrices are computable from the choice of basis functions. A few common choices are documented below:

- In two dimensions, the most typical use case for FEM makes use of a triangulation of the domain $\Omega \subset \mathbb{R}^2$ and takes the ϕ_i basis to be localized small neighborhoods of triangles. For example, for the Poisson equation it is sufficient to use piecewise-linear "hat" basis functions as discussed in §13.2.2 and illustrated in Figure 13.9. In this case, the mass and stiffness matrices are very sparse, because most of the basis functions ϕ_i have no overlap. Exercise 16.2 works out the details of one such approach on the plane. Volumes in $\mathbb{R}^3$ admit similar formulations with triangles replaced by tetrahedra.
- Spectral methods use bases constructed out of cosine and sine, which have the advantage of being orthogonal with respect to $\langle \cdot, \cdot \rangle$; in particularly favorable situations, this orthogonality can make the mass or stiffness matrices diagonal. Furthermore, the fast Fourier transform and related algorithms accelerate computations in this case.
- Adaptive finite element methods analyze the output of a FEM solver to identify regions of Ω in which the solution has poor quality. Additional basis functions ϕ_i are added to refine the output in those regions.

Example 16.9 (Piecewise-linear FEM)**.** Suppose we wish to solve the Poisson equation $u''(x) = w(x)$ for $u(x)$ on the unit interval $x \in [0, 1]$ subject to Dirichlet boundary conditions $u(0) = c$ and $u(1) = d$. We will use the piecewise linear basis functions introduced in §13.1.3. Define

$$\phi(x) \equiv \begin{cases} 1 + x & \text{when } x \in [-1, 0] \\ 1 - x & \text{when } x \in [0, 1] \\ 0 & \text{otherwise.} \end{cases}$$

We define $k + 1$ basis elements using the formula $\phi_i(x) \equiv \phi(kx - i)$ for $i \in \{0, \dots, k\}$.

For convenience, we begin by computing the following integrals:

$$\int_{-1}^{1} \phi(x)^2 \, dx = \int_{-1}^{0} (1 + x)^2 \, dx + \int_{0}^{1} (1 - x)^2 \, dx = \frac{2}{3}$$

$$\int_{-1}^{1} \phi(x)\phi(x - 1) \, dx = \int_{0}^{1} x(1 - x) \, dx = \frac{1}{6}.$$

After applying change of coordinates, these integrals show

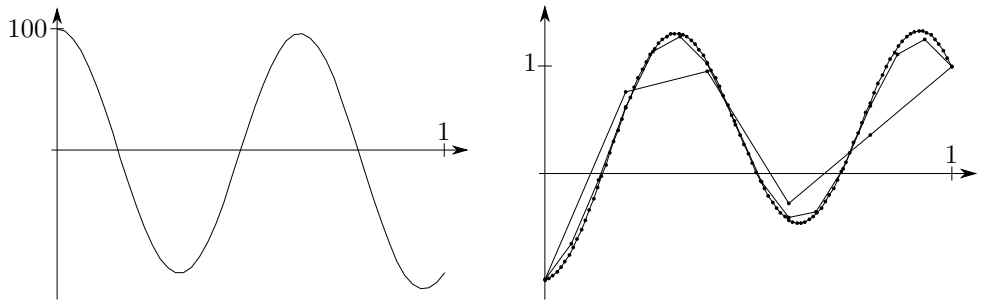
$$\langle \phi_i, \phi_j \rangle = \frac{1}{6k} \cdot \begin{cases} 4 & \text{when } i = j \\ 1 & \text{when } |i - j| = 1 \\ 0 & \text{otherwise.} \end{cases}$$

Furthermore, the derivative $\phi'(x)$ satisfies

$$\phi'(x) \equiv \begin{cases} 1 & \text{when } x \in [-1, 0] \\ -1 & \text{when } x \in [0, 1] \\ 0 & \text{otherwise.} \end{cases}$$

Hence, after change-of-variables we have

$$\langle \phi_i, \phi_j \rangle_{d^2/dx^2} = -\langle \phi_i', \phi_j' \rangle_2 = k \cdot \begin{cases} -2 & \text{when } i = j \\ 1 & \text{when } |i - j| = 1 \\ 0 & \text{otherwise.} \end{cases}$$



$w(x)$ $u(x)$ (approx.)

Figure 16.17 Approximated piecewise linear solutions of $u''(x) = w(x)$ computed using finite elements as derived in Example 16.9; in these examples, we take $c = -1$, $d = 1$, and $k \in \{5, 15, 100\}$.

Up to the constant k, these values coincide with the divided difference second-derivative from §16.4.1.

We will apply the Galerkin method to discretize $u(x) \approx \sum_i a_i \phi_i(x)$. Assume we sample $b_i = w(i/k)$. Then, based on the integrals above, we should solve:

$$k \begin{pmatrix} 1/k & & & & & & \\ 1 & -2 & 1 & & & & \\ & 1 & -2 & 1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & 1 & -2 & 1 & \\ & & & & & & 1/k \end{pmatrix} \vec{a} = \frac{1}{6k} \begin{pmatrix} 6k & & & & & & \\ 1 & 4 & 1 & & & & \\ & 1 & 4 & 1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & 1 & 4 & 1 & \\ & & & & & & 6k \end{pmatrix} \begin{pmatrix} c \\ b_1 \\ \vdots \\ b_{k-1} \\ d \end{pmatrix}.$$

The first and last rows of this equation encode the boundary conditions, and the remaining rows come from the finite elements discretization. Figure 16.17 shows an example of this discretization in practice.

16.4.4 Finite Volumes

The *finite volume* method might be considered somewhere on the spectrum between finite elements and collocation. Like collocation, this method starts from the pointwise formulation of a PDE. Rather than asking that the PDE holds at a particular set of points in the domain Ω, however, finite volumes requires that the PDE is satisfied *on average* by integrating within the cells of a partition of Ω.

Suppose $\Gamma \subseteq \Omega$ is a region contained within the domain Ω and that we once again wish to solve the Laplace equation $\nabla^2 u = 0$. A key tool for the finite volume method is the divergence theorem, which states that the divergence of a smooth vector field $\vec{v}(x)$ can be integrated over Γ two different ways:

$$\int_\Gamma \nabla \cdot \vec{v}(\vec{x})\, d\vec{x} = \int_{\partial\Gamma} \vec{v}(\vec{x}) \cdot \vec{n}(\vec{x})\, d\vec{x}.$$

Here, $\vec{n}$ is the normal to the boundary $\partial\Gamma$. In words, the divergence theorem states that the total divergence of a vector field $\vec{v}(x)$ in the interior of Γ is the same as summing the amount of $\vec{v}$ "leaving" the boundary $\partial\Gamma$.

Suppose we solve the Poisson equation $\nabla^2 u = w$ in Ω. Integrating over Γ shows

$$\begin{aligned}\int_\Gamma w(\vec{x})\,d\vec{x} &= \int_\Gamma \nabla^2 u(\vec{x})\,d\vec{x} \text{ since we solved the Poisson equation} \\ &= \int_\Gamma \nabla\cdot(\nabla u(\vec{x}))\,d\vec{x} \text{ since the Laplacian is the divergence of the gradient} \\ &= \int_{\partial\Gamma} \nabla u(\vec{x})\cdot\vec{n}(\vec{x})\,d\vec{x} \text{ by the divergence theorem.}\end{aligned}$$

This final expression characterizes solutions to the Poisson equation when they are averaged over Γ.

To derive a finite-volume approximation, again write $u(\vec{x}) \approx \sum_{i=1}^k a_i\phi_i(\vec{x})$ and now divide Ω into k regions $\Omega = \cup_{i=1}^k \Omega_i$. For each Ω_i,

$$\int_{\Omega_i} w(\vec{x})\,d\vec{x} = \int_{\partial\Omega_i} \nabla\left(\sum_{j=1}^k a_j\phi_j(\vec{x})\right)\cdot\vec{n}(\vec{x})\,d\vec{x} = \sum_{j=1}^k a_j\left[\int_{\partial\Omega_i}\nabla\phi_j(\vec{x})\cdot\vec{n}(\vec{x})\,d\vec{x}\right].$$

This is a linear system of equations for the a_i's. A typical discretization in this case might take the ϕ_i's to be piecewise-linear hat functions and the Ω_i's to be the Voronoi cells associated with the triangle centers (see §13.2.1).

16.4.5 Other Methods

Countless techniques exist for discretizing PDEs, and we have only scraped the surface of a few common methods in our discussion. Texts such as [78] are dedicated to developing the theoretical and practical aspects of these tools. Briefly, a few other notable methods for discretization include the following:

- *Domain decomposition* methods solve small versions of a PDE in different subregions of the domain Ω, iterating from one to the next until a solution to the global problem is reached. The subproblems can be made independent, in which case they are solvable via parallel processors. A single iteration of these methods can be used to approximate the global solution of a PDE to precondition iterative solvers like conjugate gradients.
- The *boundary element* and *analytic element* methods solve certain PDEs using basis functions associated with points on the boundary $\partial\Omega$, reducing dependence on a triangulation or other discretization of the interior of Ω.
- *Mesh-free methods* simulate dynamical phenomena by tracking particles rather than meshing the domain. For example, the *smoothed-particle hydrodynamics* (SPH) technique in fluid simulation approximates a fluid as a collection of particles moving in space; particles can be added where additional detail is needed, and relatively few particles can be used to get realistic effects with limited computational capacity.
- *Level set methods*, used in image processing and fluid simulation, discretize PDEs governing the evolution and construction of curves and surfaces by representing those objects as level sets $\{\vec{x}\in\mathbb{R}^n : \psi(\vec{x}) = 0\}$. Geometric changes are represented by evolution of the level set function ψ.

16.5 SOLVING PARABOLIC AND HYPERBOLIC EQUATIONS

In the previous section, we mostly dealt with the Poisson equation, which is an elliptic PDE. Parabolic and hyperbolic equations generally introduce a *time* variable into the formulation, which also is differentiated but potentially to lower order or with a different sign. Discretizing time in the same fashion as space may not make sense for a given problem, since the two play fundamentally different roles in most physical phenomena. In this section, we consider options for discretizing this variable independently of the others.

16.5.1 Semidiscrete Methods

Semidiscrete methods apply the discretizations from §16.4 to the spatial domain but not to time, leading to an ODE with a continuous time variable that can be solved using the methods of Chapter 15. This strategy is also known as the *method of lines*.

Example 16.10 (Semidiscrete heat equation)**.** Consider the heat equation in one variable, given by $u_t = u_{xx}$, where $u(t;x)$ represents the heat of a wire at position $x \in [0,1]$ and time t. As boundary data, the user provides a function $u_0(x)$ such that $u(0;x) \equiv u_0(x)$; we also attach the boundary $x \in \{0,1\}$ to a refrigerator and enforce Dirichlet conditions $u(t;0) = u(t;1) = 0$.

Suppose we discretize x using evenly spaced samples but leave t as a continuous variable. If we use the finite differences technique from §16.4.1, this discretization results in functions $u_0(t), u_1(t), \ldots, u_n(t)$, where $u_i(t)$ represents the heat at position i as a function of time. Take L to be the corresponding second derivative matrix in the x samples with Dirichlet conditions. Then, the semidiscrete heat equation can be written $h^2\vec{u}'(t) = L\vec{u}(t)$, where $h = 1/n$ is the spacing between samples. This is an ODE for $\vec{u}(t)$ that could be time-stepped using backward Euler integration:

$$\vec{u}(t_{k+1}) \approx \left(I_{(n+1)\times(n+1)} - \frac{1}{h}L\right)^{-1} \vec{u}(t_k).$$

The previous example is an instance of a general pattern for parabolic equations. PDEs for *diffusive* phenomena like heat moving across a domain or chemicals moving through a membrane usually have one lower-order time variable and several spatial variables that are differentiated in an elliptic way. When we discretize the spatial variables using finite differences, finite elements, or another technique, the resulting semidiscrete formulation $\vec{u}' = A\vec{u}$ usually contains a negative definite matrix A. This makes the resulting ODE unconditionally stable.

As outlined in the previous chapter, there are many choices for solving the ODE after spatial discretization. If time steps are small, explicit methods may be acceptable. Implicit solvers, however, often are applied to solving parabolic PDEs; diffusive behavior of implicit Euler agrees behaviorally with diffusion from the heat equation and may be acceptable even with fairly large time steps. Hyperbolic PDEs, on the other hand, may require implicit steps for stability, but advanced integrators can combine implicit and explicit terms to prevent oversmoothing of non-diffusive phenomena.

When A does not change with time, one contrasting approach is to write solutions of semidiscrete systems $\vec{u}' = A\vec{u}$ in terms of eigenvectors of A. Suppose $\vec{v}_1, \ldots, \vec{v}_n$ are eigenvectors of A with eigenvalues $\lambda_1, \ldots, \lambda_n$ and that $\vec{u}(0) = c_1\vec{v}_1 + \cdots + c_n\vec{v}_n$. As we showed in §6.1.2, the solution of $\vec{u}' = A\vec{u}$ is

$$\vec{u}(t) = \sum_i c_i e^{\lambda_i t}\vec{v}_i.$$

The eigenvectors and eigenvalues of A may have physical interpretations in the case of a semidiscrete PDE. Most commonly, the eigenvalues of the Laplacian ∇^2 on a domain Ω correspond to resonant frequencies of a domain, that is, the frequencies that sound when hitting the domain with a hammer. The eigenvectors provide closed-form "low-frequency approximations" of solutions to common PDEs after truncating the sum above.

16.5.2 Fully Discrete Methods

Rather than discretizing time and then space, we might treat the space and time variables more democratically and discretize them both simultaneously. This one-shot discretization is in some sense a more direct application of the methods we considered in §16.4, just by including t as a dimension in the domain Ω under consideration. Because we now multiply the number of variables needed to represent Ω by the number of time steps, the resulting linear systems of equations can be large if dependence between time steps has global reach.

Example 16.11 (Fully discrete heat diffusion, [58])**.** Consider the heat equation $u_t = u_{xx}$. Discretizing x and t simultaneously via finite differences yields a matrix of u values, which we can index as u_i^j, representing the heat at position i and time j. Take Δx and Δt to be the spacing of x and t in the grid, respectively. Choosing where to evaluate the different derivatives leads to different discretization schemes stepping from time j to time $j+1$.

For example, evaluating the x derivative at time j produces an *explicit* formula:

$$\frac{u_i^{j+1} - u_i^j}{\Delta t} = \frac{u_{i+1}^j - 2u_i^j + u_{i-1}^j}{(\Delta x)^2}.$$

Isolating u_i^{j+1} gives a formula for obtaining u at time $j+1$ without a linear solve.

Alternatively, we can evaluate the x derivative at time $j+1$ for an *implicit* integrator:

$$\frac{u_i^{j+1} - u_i^j}{\Delta t} = \frac{u_{i+1}^{j+1} - 2u_i^{j+1} + u_{i-1}^{j+1}}{(\Delta x)^2}.$$

This integrator is unconditionally stable but requires a linear solve to obtain the u values at time $j+1$ from those at time j.

These implicit and explicit integrators inherit their accuracy from the quality of the finite difference formulas, and hence—stability aside—both are first-order accurate in time and second-order accurate in space. To improve the accuracy of the time discretization, we can use the *Crank-Nicolson* method, which applies a trapezoidal time integrator:

$$\frac{u_i^{j+1} - u_i^j}{\Delta t} = \frac{1}{2}\left[\frac{u_{i+1}^j - 2u_i^j + u_{i-1}^j}{(\Delta x)^2} + \frac{u_{i+1}^{j+1} - 2u_i^{j+1} + u_{i-1}^{j+1}}{(\Delta x)^2}\right].$$

This method inherits the unconditional stability of trapezoidal integration and is second-order accurate in time and space. Despite this stability, however, as explained in §15.3.3, taking time steps that are too large can produce unrealistic oscillatory behavior.

In the end, even semidiscrete methods can be considered fully discrete in that the time-stepping ODE method still discretizes the t variable; the difference between semidiscrete and fully discrete is mostly for classification of how methods were derived. One advantage of semidiscrete techniques, however, is that they can adjust the time step for t depending on the current iterate, e.g., if objects are moving quickly in a physical simulation, it might make sense to take more dense time steps and resolve this motion. Some methods also

adjust the discretization of the domain of x values in case more resolution is needed near local discontinuities such as shock waves.

16.6 NUMERICAL CONSIDERATIONS

We have considered several options for discretizing PDEs. As with choosing time integrators for ODEs, the trade-offs between these options are intricate, representing different compromises between computational efficiency, numerical quality, stability, and so on. We conclude our treatment of numerical methods for PDE by outlining a few considerations when choosing a PDE discretization.

16.6.1 Consistency, Convergence, and Stability

A key consideration when choosing ODE integrators was *stability*, which guaranteed that errors in specifying initial conditions would not be amplified over time. Stability remains a consideration in PDE integration, but it also can interact with other key properties:

- A method is *convergent* if solutions to the discretized problem converge to the theoretical solution of the PDE as spacing between discrete samples approaches zero.
- A method is *consistent* if the accompanying discretization of the differential operators better approximates the derivatives taken in the PDE as spacing approaches zero.

For finite differencing schemes, the Lax-Richtmyer Equivalence Theorem states that if a linear problem is well-posed, *consistency* and *stability* together are necessary and sufficient for *convergence* [79]. Consistency and stability tend to be easier to check than convergence. Consistency arguments usually come from Taylor series. A number of well-established methods establish stability or lack thereof; for example, the well-known *CFL condition* states that the ratio of time spacing to spatial spacing of samples should exceed the speed at which waves propagate in the case of hyperbolic PDE [29]. Even more caution must be taken when simulating advective phenomena and PDEs that can develop fronts and shocks; specialized *upwinding* schemes attempt to detect the formation of these features to ensure that they move in the right direction and at the proper speed.

Even when a time variable is not involved, some care must be taken to ensure that a PDE approximation scheme reduces error as sampling becomes more dense. For example, in elliptic PDE, convergence of finite elements methods depends on the choice of basis functions, which must be sufficiently smooth to represent the theoretical solution and must span the function space in the limit [16].

The subtleties of consistency, convergence, and stability underlie the theory of numerical PDE, and the importance of these concepts cannot be overstated. Without convergence guarantees, the output of a numerical PDE solver cannot be trusted. Standard PDE integration packages often incorporate checks for assorted stability conditions or degenerate behavior to guide clients whose expertise is in modeling rather than numerics.

16.6.2 Linear Solvers for PDE

The matrices resulting from PDE discretizations have many favorable properties that make them ideal inputs for the methods we have considered in previous chapters. For instance, as motivated in §16.3.1, elliptic PDEs are closely related to positive definite matrices, and typical discretizations require solution of a positive definite linear system. The same derivative operators appear in parabolic PDEs, which hence have well-posed semidiscretizations. For

this reason, methods like Cholesky decomposition and conjugate gradients can be applied to these problems. Furthermore, derivative matrices tend to be sparse, inducing additional memory and time savings. Any reasonable implementation of a PDE solver should include these sorts of optimizations, which make them scalable to large problems.

Example 16.12 (Elliptic operators as matrices)**.** Consider the one-dimensional second derivative matrix L with Dirichlet boundary conditions from §16.4.1. L is sparse and negative definite. To show the latter property, we can write $L = -D^\top D$ for the matrix $D \in \mathbb{R}^{(n+1)\times n}$ given by

$$D = \begin{pmatrix} 1 & & & & & \\ -1 & 1 & & & & \\ & -1 & 1 & & & \\ & & \ddots & \ddots & & \\ & & & -1 & 1 & \\ & & & & -1 & \end{pmatrix}.$$

This matrix is a finite-differenced *first* derivative, so this observation parallels the fact that ${}^{d^2y}/_{dx^2} = {}^{d}/_{dx}({}^{dy}/_{dx})$. For any $\vec{x} \in \mathbb{R}^n$, $\vec{x}^\top L\vec{x} = -\vec{x}^\top D^\top D\vec{x} = -\|D\vec{x}\|_2^2 \leq 0$, showing L is negative semidefinite. Furthermore, $D\vec{x} = 0$ only when $\vec{x} = 0$, completing the proof that L is negative definite.

Example 16.13 (Stiffness matrix is positive semidefinite)**.** Regardless of the basis $\phi_1, \ldots, \phi_k$, the stiffness matrix from discretizing the Poisson equation via finite elements (see §16.4.3) is negative semidefinite. Taking M_{∇^2} to be the stiffness matrix and $\vec{a} \in \mathbb{R}^k$,

$$\begin{aligned}
\vec{a}^\top M_{\nabla^2}\vec{a} &= \sum_{ij} a_i a_j \langle \phi_i, \phi_j \rangle_{\nabla^2} \text{ by definition of } M_{\nabla^2} \\
&= \left\langle \sum_i a_i\phi_i, \sum_j a_j\phi_j \right\rangle_{\nabla^2} \text{ by bilinearity of } \langle\cdot,\cdot\rangle_{\nabla^2} \\
&= \langle \psi, \psi \rangle_{\nabla^2} \text{ if we define } \psi \equiv \sum_i a_i\phi_i \\
&= -\int_\Omega \|\nabla\psi(\vec{x})\|_2^2 \, d\vec{x} \text{ by definition of } \langle\cdot,\cdot\rangle_{\nabla^2} \\
&\leq 0 \text{ since the integrand is nonnegative.}
\end{aligned}$$

16.7 EXERCISES

16.1 ("Shooting method," [58]) The *two-point boundary value problem* inherits some structure from ODE and PDE problems alike. In this problem, we wish to solve the ODE $\vec{y}' = F[\vec{y}]$ for a function $\vec{y}(t) : [0, 1] \to \mathbb{R}^n$. Rather than specifying initial conditions, however, we specify some relationship $g(\vec{y}(0), \vec{y}(1)) = \vec{0}$.

(a) Give an example of a two-point boundary value problem that does not admit a solution.

(b) Assume we have checked the conditions of an existence/uniqueness theorem, so given $\vec{y}_0 = \vec{y}(0)$ we can generate $\vec{y}(t)$ for all $t > 0$ satisfying $\vec{y}'(t) = F[\vec{y}(t)]$.

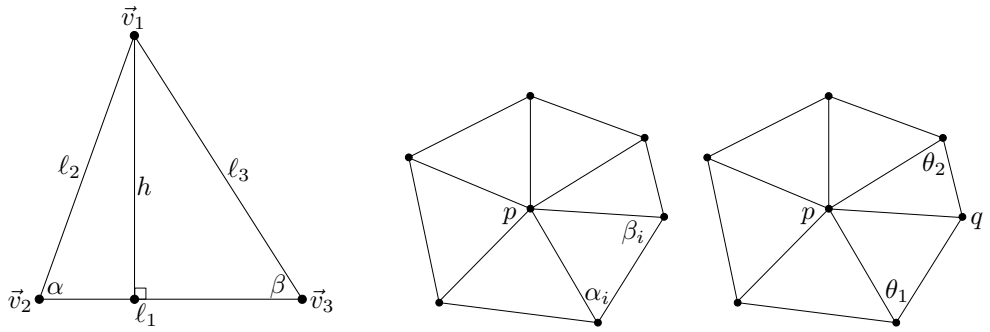


Figure 16.18 Notation for Exercise 16.2.

Denote $\vec{y}(t;\vec{y}_0) : \mathbb{R}^+ \times \mathbb{R}^n \to \mathbb{R}$ as the function returning $\vec{y}$ at time t given $\vec{y}(0) = \vec{y}_0$. In this notation, pose the two-point boundary value problem as a root-finding problem.

(c) Use the ODE integration methods from Chapter 15 to propose a computationally feasible root-finding problem for approximating a solution $\vec{y}(t)$ of the two-point boundary value problem.

(d) As discussed in Chapter 8, most root-finding algorithms require the Jacobian of the objective function. Suggest a technique for finding the Jacobian of your objective from Exercise 16.1c.

16.2 In this problem, we use first-order finite elements to derive the famous *cotangent Laplacian* formula used in geometry processing. Refer to Figure 16.18 for notation.

(a) Suppose we construct a planar triangle T with vertices $\vec{v}_1, \vec{v}_2, \vec{v}_3 \in \mathbb{R}^2$ in counterclockwise order. Take $f_1(\vec{x})$ to be the affine *hat function* $f_1(\vec{x}) \equiv c + \vec{d} \cdot \vec{x}$ satisfying $f_1(\vec{v}_1) = 1$, $f_1(\vec{v}_2) = 0$, and $f_1(\vec{v}_3) = 0$. Show that ∇f_1 is a constant vector satisfying:

$$\begin{aligned}\nabla f_1 \cdot (\vec{v}_1 - \vec{v}_2) &= 1\\ \nabla f_1 \cdot (\vec{v}_1 - \vec{v}_3) &= 1\\ \nabla f_1 \cdot (\vec{v}_2 - \vec{v}_3) &= 0.\end{aligned}$$

The third relationship shows that ∇f_1 is perpendicular to the edge from $\vec{v}_2$ to $\vec{v}_3$.

(b) Show that $\|\nabla f_1\|_2 = \frac{1}{h}$, where h is the height of the triangle as marked in Figure 16.18 (left).
Hint: Start by showing $\nabla f_1 \cdot (\vec{v}_1 - \vec{v}_3) = \|\nabla f_1\|_2 \ell_3 \cos\left(\frac{\pi}{2} - \beta\right)$.

(c) Integrate over the triangle T to show

$$\int_T \|\nabla f_1\|_2^2 \, dA = \frac{1}{2}(\cot\alpha + \cot\beta).$$

Hint: Since ∇f_1 is a constant vector, the integral equals $\|\nabla f_1\|_2^2 A$, where A is the area of T. From basic geometry, $A = \frac{1}{2}\ell_1 h$.

(d) Define $\theta \equiv \pi - \alpha - \beta$, and take f_2 and f_3 to be the hat functions associated with $\vec{v}_2$ and $\vec{v}_3$, respectively. Show that

$$\int_T \nabla f_2 \cdot \nabla f_3 \, dA = -\frac{1}{2} \cot \theta.$$

(e) Now, consider a vertex p of a triangle mesh (Figure 16.18, middle), and define $f_p : \mathbb{R}^2 \to [0,1]$ to be the *piecewise linear hat function* associated with p (see §13.2.2 and Figure 13.9). That is, restricted to any triangle adjacent to p, the function f_p behaves as constructed in Exercise 16.2a; $f_p \equiv 0$ outside the triangles adjacent to p. Based on the results you already have constructed, show:

$$\int_{\mathbb{R}^2} \|\nabla f_p\|_2^2 \, dA = \frac{1}{2} \sum_i (\cot \alpha_i + \cot \beta_i),$$

where $\{\alpha_i\}$ and $\{\beta_i\}$ are the angles opposite p in its neighboring triangles.

(f) Suppose p and q are adjacent vertices on the same mesh, and define θ_1 and θ_2 as shown in Figure 16.18 (right). Show

$$\int_{\mathbb{R}^2} \nabla f_p \cdot \nabla f_q \, dA = -\frac{1}{2} (\cot \theta_1 + \cot \theta_2).$$

(g) Conclude that in the basis of hat functions on a triangle mesh, the stiffness matrix for the Poisson equation has the following form:

$$L_{ij} \equiv -\frac{1}{2} \begin{cases} \sum_{i \sim j} (\cot \alpha_j + \cot \beta_j) & \text{if } i = j \\ -(\cot \alpha_j + \cot \beta_j) & \text{if } i \sim j \\ 0 & \text{otherwise.} \end{cases}$$

Here, $i \sim j$ denotes that vertices i and j are adjacent.

(h) Write a formula for the entries of the corresponding mass matrix, whose entries are

$$\int_{\mathbb{R}^2} f_p f_q \, dA.$$

Hint: This matrix can be written completely in terms of triangle areas. Divide into cases: (1) $p = q$, (2) p and q are adjacent vertices, and (3) p and q are not adjacent.

16.3 Suppose we wish to approximate Laplacian eigenfunctions $f(\vec{x})$, satisfying $\nabla^2 f = \lambda f$. Show that discretizing such a problem using FEM results in a generalized eigenvalue problem $A\vec{x} = \lambda B\vec{x}$.

16.4 Propose a semidiscrete form for the one-dimensional wave equation $u_{tt} = u_{xx}$, similar to the construction in Example 16.10. Is the resulting ODE well-posed (§15.2.3)?

16.5 *Graph-based semi-supervised learning* algorithms attempt to predict a quantity or label associated with the nodes of a graph given labels on a few of its vertices. For instance, under the (dubious) assumption that friends are likely to have similar incomes, it could be used to predict the annual incomes of all members of a social network given the incomes of a few of its members. We will focus on a variation of the method proposed in [132].

(a) Take $G = (V, E)$ to be a connected graph, and define $f_0 : V_0 \to \mathbb{R}$ to be a set of scalar-valued labels associated with the nodes of a subset $V_0 \subseteq V$. The *Dirichlet energy* of a full assignment of labels $f : V \to \mathbb{R}$ is given by

$$E[f] \equiv \sum_{(v_1,v_2)\in E} (f(v_2) - f(v_1))^2.$$

Explain why $E[f]$ can be minimized over f satisfying $f(v_0) = f_0(v_0)$ for all $v_0 \in V_0$ using a linear solve.

(b) Explain the connection between the linear system from Exercise 16.5a and the 3×3 Laplacian stencil from §16.4.1.

(c) Suppose f is the result of the optimization from Exercise 16.5a. Prove the *discrete maximum principle*:

$$\max_{v\in V} f(v) = \max_{v_0\in V_0} f_0(v_0).$$

Relate this result to a physical interpretation of Laplace's equation.

16.6 Give an example where discretization of the Poisson equation via finite differences and via collocation lead to the same system of equations.

16.7 ("Von Neumann stability analysis," based on notes by D. Levy) Suppose we wish to approximate solutions to the PDE $u_t = au_x$ for some fixed $a \in \mathbb{R}$. We will use initial conditions $u(x, 0) = f(x)$ for some $f \in C^\infty([0, 2\pi])$ and periodic boundary conditions $u(0, t) = u(2\pi, t)$.

(a) What is the order of this PDE? Give conditions on a for it to be elliptic, hyperbolic, or parabolic.

(b) Show that the PDE is solved by $u(x, t) = f(x + at)$.

(c) The Fourier transform of $u(x, t)$ in x is

$$[\mathcal{F}_x u](\omega, t) \equiv \frac{1}{\sqrt{2\pi}} \int_0^{2\pi} u(x, t)e^{-i\omega x}\, dx,$$

where $i = \sqrt{-1}$ (see Exercise 4.15). It measures the frequency content of $u(\cdot, t)$. Define $v(x, t) \equiv u(x + \Delta x, t)$. If u satisfies the stated boundary conditions, show that $[\mathcal{F}_x v](\omega, t) = e^{i\omega\Delta x}[\mathcal{F}_x u](\omega, t)$.

(d) Suppose we use a forward Euler discretization:

$$\frac{u(x, t + \Delta t) - u(x, t)}{\Delta t} = a\frac{u(x + \Delta x, t) - u(x - \Delta x, t)}{2\Delta x}.$$

Show that this discretization satisfies

$$[\mathcal{F}_x u](\omega, t + \Delta t) = \left(1 + \frac{ai\Delta t}{\Delta x} \sin(\omega\Delta x)\right) [\mathcal{F}_x u](\omega, t).$$

(e) Define the *amplification factor*

$$\hat{Q} \equiv 1 + \frac{ai\Delta t}{\Delta x} \sin(\omega\Delta x).$$

Show that $|\hat{Q}| > 1$ for almost any choice of ω. This shows that the discretization amplifies frequency content over time and is *unconditionally unstable*.

(f) Carry out a similar analysis for the alternative discretization

$$u(x,t+\Delta t) = \frac{1}{2}\left(u(x-\Delta x,t) + u(x+\Delta x,t)\right) + \frac{a\Delta t}{2\Delta x}\left[u(x+\Delta x,t) - u(x-\Delta x,t)\right].$$

Derive an upper bound on the ratio $\Delta t/\Delta x$ for this discretization to be stable.

16.8 ("Fast marching," [19]) Nonlinear PDEs require specialized treatment. One nonlinear PDE relevant to computer graphics and medical imaging is the eikonal equation $\|\nabla d\|_2 = 1$ considered in §16.5. Here, we outline some aspects of the *fast marching* method for solving this equation on a triangulated domain $\Omega \subset \mathbb{R}^2$ (see Figure 13.9).

(a) We might approximate solutions of the eikonal equation as shortest-path distances along the edges of the triangulation. Provide a way to triangulate the unit square $[0,1]\times[0,1]$ with arbitrarily small triangle edge lengths and areas for which this approximation gives distance 2 rather than $\sqrt{2}$ from $(0,0)$ to $(1,1)$. Hence, can the edge-based approximation be considered convergent?

(b) Suppose we approximate $d(\vec{x})$ with a linear function $d(\vec{x}) \approx \vec{n}^\top\vec{x} + p$, where $\|\vec{n}\|_2 = 1$ by the eikonal equation. Given $d_1 = d(\vec{x}_1)$ and $d_2 = d(\vec{x}_2)$, show that p can be recovered by solving a quadratic equation and provide a geometric interpretation of the two roots. You can assume that $\vec{x}_1$ and $\vec{x}_2$ are linearly independent.

(c) What geometric assumption does the approximation in Exercise 16.8b make about the shape of the level sets $\{\vec{x} \in \mathbb{R}^2 : d(\vec{x}) = c\}$? Does this approximation make sense when d is large or small? See [91] for a contrasting circular approximation.

(d) Extend Dijkstra's algorithm for graph-based shortest paths to triangulated shapes using the approximation in Exercise 16.8b. What can go wrong with this approach?
Hint: Dijkstra's algorithm starts at the center vertex and builds the shortest path in breadth-first fashion. Change the update to use Exercise 16.8b, and consider when the approximation will make distances decrease unnaturally.

16.9 Constructing *higher-order elements* can be necessary for solving certain differential equations.

(a) Show that the parameters $a_0,\ldots,a_5$ of a function $f(x,y) = a_0 + a_1x + a_2y + a_3x^2 + a_4y^2 + a_5xy$ are uniquely determined by its values on the three vertices and three edge midpoints of a triangle.

(b) Show that if (x,y) is on an edge of the triangle, then $f(x,y)$ can be computed knowing only the values of f at the endpoints and midpoint of that edge.

(c) Use these facts to construct a basis of continuous, piecewise-quadratic functions on a triangle mesh, and explain why it may be useful for solving higher-order PDEs.

16.10 For matrices $A, B \in \mathbb{R}^{n\times n}$, the Lie-Trotter-Kato formula states

$$e^{A+B} = \lim_{n\to\infty}(e^{A/n}e^{B/n})^n,$$

where e^M denotes the matrix exponential of $M \in \mathbb{R}^{n\times n}$ (see §15.3.5).

Suppose we wish to solve a PDE $u_t = \mathcal{L}u$, where $\mathcal{L}$ is some differential operator that admits a *splitting* $\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2$. How can the Lie-Trotter-Kato formula be applied to designing PDE time-stepping machinery in this case?

Note: Such splittings are useful for breaking up integrators for complex PDEs like the Navier-Stokes equations into simpler steps.