

Chapter 10

Numerical Methods

As you know, most differential equations are far too complicated to be solved by an explicit analytic formula. Thus, the development of accurate numerical approximation schemes is essential for both extracting quantitative information as well as achieving a qualitative understanding of the behavior of their solutions. Even in cases, such as the heat and wave equations, where explicit solution formulas (either closed form or infinite series) exist, numerical methods still can be profitably employed. Indeed, one is able to accurately test a proposed numerical algorithm by running it on a known solution. Furthermore, the lessons learned in the design of numerical algorithms for “solved” examples are of inestimable value when confronting more challenging problems.

Basic numerical solution schemes for partial differential equations fall into two broad categories. The first are the *finite difference methods*, obtained by replacing the derivatives in the equation by the appropriate numerical differentiation formulae. We thus start with a brief discussion of the most basic finite difference formulae for numerically approximating first and second order derivatives of functions. The ensuing sections establish and analyze some of the most basic finite difference schemes for the heat equation, first order transport equations, and the second order wave equation. As we will see, not all finite difference approximations lead to accurate numerical schemes, and the issues of stability and convergence must be dealt with in order to distinguish valid from worthless methods. In fact, inspired by Fourier analysis, the crucial stability criterion that serves to distinguish accurate finite difference schemes is based on how it handles complex exponentials.

The second broad category of numerical solution techniques are the *finite element methods*, designed for the differential equations describing equilibrium configurations. The finite element approach relies on the characterization of the solution to the boundary value problem by a minimization principle. Restricting the minimizing functional to an appropriately chosen finite-dimensional subspace of functions leads to a finite-dimensional minimization problem that can be solved by linear algebra. Alternatively, one can base the method on the notion of a weak solution, and restrict the class of test functions to the same finite element subspace. We first present the basic ideas in the context of boundary value problems for ordinary differential equations. The final section develops the finite element analysis of the two-dimensional Laplace and Poisson equations and related positive definite boundary value problems.

This chapter serves as a preliminary foray into this vast and active area of contemporary research. More sophisticated variations and extensions can be found in specialized numerical analysis texts, e.g., [70, 95].

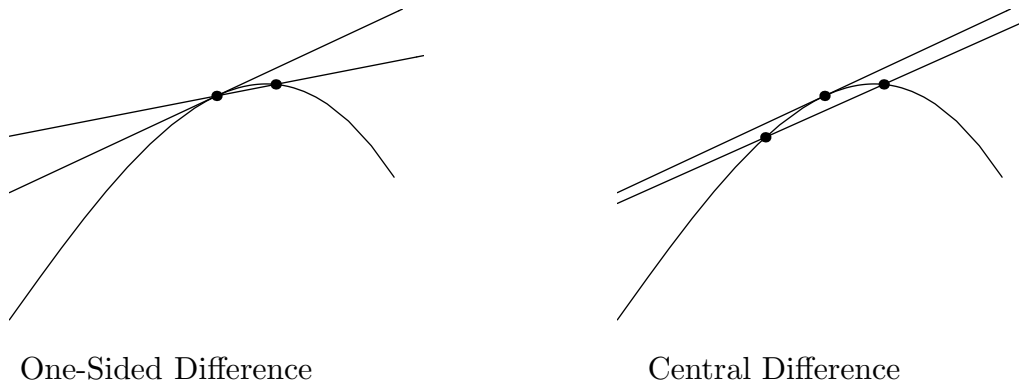


Figure 10.1. Finite Difference Approximations.

10.1. Finite Differences.

In general, a *finite difference* approximate to the value of some derivative of a function at a point, say $u'(x)$ or $u''(x)$, relies on a suitable combination of sampled function values at nearby points. The underlying formalism used to construct these approximation formulae is known as the *calculus of finite differences*. Its development has a long and influential history, dating back to Newton.

We begin with the first order derivative of a function $u(x)$. The simplest finite difference approximation is the ordinary *difference quotient*

$$\frac{u(x+h) - u(x)}{h} \approx u'(x) \quad (10.1)$$

appearing in the original calculus definition of the derivative. Indeed, if u is differentiable at x , then $u'(x)$ is, by definition, the limit, as $h \rightarrow 0$ of the finite difference quotients. Geometrically, the difference quotient equals the slope of the secant line through the two points $(x, u(x))$ and $(x+h, u(x+h))$ on the graph of the function. For small h , this should be a reasonably good approximation to the slope of the tangent line, $u'(x)$, as illustrated in the first picture in Figure 10.1. Throughout our discussion, h , the *step size*, which may be either positive or negative, is assumed to be small: $|h| \ll 1$. When $h > 0$, (10.1) is referred to as a *forward difference*, while $h < 0$ gives a *backward difference*.

How close an approximation is the difference quotient? To answer this question, we assume that $u(x)$ is at least twice continuously differentiable, and examine the first order Taylor expansion

$$u(x+h) = u(x) + u'(x)h + \frac{1}{2}u''(\xi)h^2 \quad (10.2)$$

at the point x . We have used the Cauchy form for the remainder term, [7, 124], in which ξ , which depends on both x and h , represents some point lying between x and $x+h$. Rearranging (10.2), we find

$$\frac{u(x+h) - u(x)}{h} - u'(x) = \frac{1}{2}u''(\xi)h.$$

Thus, the *error* in the finite difference approximation (10.1) can be bounded by a multiple

of the step size:

$$\left| \frac{u(x+h) - u(x)}{h} - u'(x) \right| \leq C|h|,$$

where $C = \max \frac{1}{2} |u''(\xi)|$ is a bound on the magnitude of the second derivative of the function over the interval in question. Since the error is proportional to the first power of h , we say that the finite difference quotient (10.1) is a *first order* approximation. When the precise formula for the error is not so important, we will write

$$u'(x) = \frac{u(x+h) - u(x)}{h} + O(h). \quad (10.3)$$

The “big Oh” notation $O(h)$ refers to a term that is proportional to h , or, more rigorously, bounded by a constant multiple of h as $h \rightarrow 0$.

Example 10.1. Let $u(x) = \sin x$. Let us try to approximate

$$u'(1) = \cos 1 = .5403023 \dots$$

by computing finite difference quotients

$$\cos 1 \approx \frac{\sin(1+h) - \sin 1}{h}.$$

The result for different (positive) values of h is listed in the following table.

h	.1	.01	.001	.0001
approximation	.497364	.536086	.539881	.540260
error	-.042939	-.004216	-.000421	-.000042

We observe that reducing the step size by a factor of $\frac{1}{10}$ reduces the size of the error by approximately the same factor. Thus, to obtain 10 decimal digits of accuracy, we anticipate needing a step size of about $h = 10^{-11}$. The fact that the error is more or less proportional to the step size confirms that we are dealing with a first order numerical approximation.

To approximate higher order derivatives, we need to evaluate the function at more than two points. In general, an approximation to the n^{th} order derivative $u^{(n)}(x)$ requires at least $n + 1$ distinct sample points. For simplicity, we restrict our attention to equally spaced sample points, leaving the general case to the exercises.

For example, let us try to approximate $u''(x)$ by sampling u at the particular points x , $x + h$ and $x - h$. Which combination of the function values $u(x - h)$, $u(x)$, $u(x + h)$ should be used? The answer to such a question can be found by consideration of the relevant Taylor expansions

$$\begin{aligned} u(x+h) &= u(x) + u'(x)h + u''(x)\frac{h^2}{2} + u'''(x)\frac{h^3}{6} + O(h^4), \\ u(x-h) &= u(x) - u'(x)h + u''(x)\frac{h^2}{2} - u'''(x)\frac{h^3}{6} + O(h^4), \end{aligned} \quad (10.4)$$

where the error terms are proportional to h^4 . Adding the two formulae together gives

$$u(x+h) + u(x-h) = 2u(x) + u''(x)h^2 + O(h^4).$$

Dividing by h^2 and rearranging terms, we arrive at the *centered finite difference approximation* to the second derivative of a function:

$$u''(x) = \frac{u(x+h) - 2u(x) + u(x-h)}{h^2} + O(h^2). \quad (10.5)$$

Since the error is proportional to h^2 , this forms a second order approximation.

Example 10.2. Let $u(x) = e^{x^2}$, with $u''(x) = (4x^2 + 2)e^{x^2}$. Let us approximate

$$u''(1) = 6e = 16.30969097 \dots$$

by using the finite difference quotient (10.5):

$$u''(1) = 6e \approx \frac{e^{(1+h)^2} - 2e + e^{(1-h)^2}}{h^2}.$$

The results are listed in the following table.

h	.1	.01	.001	.0001
approximation	16.48289823	16.31141265	16.30970819	16.30969115
error	.17320726	.00172168	.00001722	.00000018

Each reduction in step size by a factor of $\frac{1}{10}$ reduces the size of the error by a factor of about $\frac{1}{100}$, and thus results in a gain of two new decimal digits of accuracy. This confirms that the centered finite difference approximation is of second order.

However, this prediction is not completely borne out in practice. If we take[†] $h = .00001$ then the formula produces the approximation 16.3097002570, with an error of .0000092863 — which is *less* accurate than the approximation with $h = .0001$. The problem is that round-off errors due to the finite precision of numbers stored in the computer have now begun to affect the computation. This highlights the inherent difficulty with numerical differentiation: Finite difference formulae inevitably require dividing very small quantities, and so round-off inaccuracies tends to induce large numerical errors. As a result, while finite difference formulae typically produce reasonably good approximations to the derivatives for moderately small step sizes, to achieve very high accuracy, one must switch to a higher precision computer arithmetic. Indeed, a similar comment applies to the previous computation in Example 10.1. Our expectations about the error were not, in fact, fully justified, as you may have discovered had you tried an extremely small step size.

Another way to improve the order of accuracy of finite difference approximations is to employ more sample points. For instance, if the first order approximation (10.3) based on

[†] This computation depends upon the computer's precision; here we employed single precision floating point arithmetic.

the two points x and $x + h$ is not sufficiently accurate, one can try combining the function values at three points, say x , $x + h$, and $x - h$. To find the appropriate combination of function values $u(x - h)$, $u(x)$, $u(x + h)$, we return to the Taylor expansions (10.4). To solve for $u'(x)$, we subtract[‡] the two formulae, and so

$$u(x + h) - u(x - h) = 2u'(x)h + u'''(x)\frac{h^3}{3} + O(h^4).$$

Rearranging the terms, we are led to the well-known *centered difference formula*

$$u'(x) = \frac{u(x + h) - u(x - h)}{2h} + O(h^2), \quad (10.6)$$

which is a second order approximation to the first derivative. Geometrically, the centered difference quotient represents the slope of the secant line through the two points $(x - h, u(x - h))$ and $(x + h, u(x + h))$ on the graph of u centered symmetrically about the point x . Figure 10.1 illustrates the two approximations; the advantages in accuracy in the centered difference version are graphically evident. Higher order approximations can be found by evaluating the function at yet more sample points, including, say, $x + 2h$, $x - 2h$, etc.

Example 10.3. Return to the function $u(x) = \sin x$ considered in Example 10.1. The centered difference approximation to its derivative $u'(1) = \cos 1 = .5403023 \dots$ is

$$\cos 1 \approx \frac{\sin(1 + h) - \sin(1 - h)}{2h}.$$

The results are tabulated as follows:

h	.1	.01	.001	.0001
approximation	.53940225217	.54029330087	.54030221582	.54030230497
error	-.00090005370	-.00000900499	-.00000009005	-.00000000090

As advertised, the results are much more accurate than the one-sided finite difference approximation used in Example 10.1 at the same step size. Since it is a second order approximation, each reduction in the step size by a factor of $\frac{1}{10}$ results in two more decimal places of accuracy — up until the point where round-off errors start to have an effect.

Many additional finite difference approximations can be constructed by similar manipulations of Taylor expansions, but these few very basic ones will suffice for our subsequent purposes. In the following sections, we will apply the finite difference formulae to develop numerical solution schemes for the heat and wave equations.

10.2. Numerical Algorithms for the Heat Equation.

Consider the heat equation

$$\frac{\partial u}{\partial t} = \gamma \frac{\partial^2 u}{\partial x^2}, \quad 0 < x < \ell, \quad t \geq 0, \quad (10.7)$$

[‡] The terms $O(h^4)$ do *not* cancel, since they represent potentially different multiples of h^4 .

on an interval of length ℓ , with constant thermal diffusivity $\gamma > 0$. To be concrete, we impose time-dependent Dirichlet boundary conditions

$$u(t, 0) = \alpha(t), \quad u(t, \ell) = \beta(t), \quad t \geq 0, \quad (10.8)$$

specifying the temperature at the ends of the interval, along with the initial conditions

$$u(0, x) = f(x), \quad 0 \leq x \leq \ell, \quad (10.9)$$

specifying the initial temperature distribution. In order to effect a numerical approximation to the solution to this initial-boundary value problem, we begin by introducing a *rectangular mesh* consisting of points (t_j, x_m) with

$$0 = t_0 < t_1 < t_2 < \cdots \quad \text{and} \quad 0 = x_0 < x_1 < \cdots < x_n = \ell.$$

For simplicity, we maintain a uniform mesh spacing in both directions, with

$$\Delta t = t_{j+1} - t_j, \quad \Delta x = x_{m+1} - x_m = \frac{\ell}{n},$$

representing, respectively, the time step size and the spatial mesh size. It will be essential that we do *not* a priori require the two to be the same. We shall use the notation

$$u_{j,m} \approx u(t_j, x_m) \quad \text{where} \quad t_j = j \Delta t, \quad x_m = m \Delta x, \quad (10.10)$$

to denote the numerical approximation to the solution value at the indicated mesh point.

As a first attempt at designing a numerical solution scheme, we shall employ the simplest finite difference approximations to the derivatives. The second order space derivative is approximated by the centered difference formula (10.5), and hence

$$\begin{aligned} \frac{\partial^2 u}{\partial x^2}(t_j, x_m) &\approx \frac{u(t_j, x_{m+1}) - 2u(t_j, x_m) + u(t_j, x_{m-1}))}{(\Delta x)^2} + O((\Delta x)^2) \\ &\approx \frac{u_{j,m+1} - 2u_{j,m} + u_{j,m-1}}{(\Delta x)^2} + O((\Delta x)^2), \end{aligned} \quad (10.11)$$

where the error in the approximation is proportional to $(\Delta x)^2$. Similarly, the one-sided finite difference approximation (10.3) is used to approximate the time derivative, and so

$$\frac{\partial u}{\partial t}(t_j, x_m) \approx \frac{u(t_{j+1}, x_m) - u(t_j, x_m)}{\Delta t} + O(\Delta t) \approx \frac{u_{j+1,m} - u_{j,m}}{\Delta t} + O(\Delta t), \quad (10.12)$$

where the error is proportion to Δt . In practice, one should try to ensure that the approximations have similar orders of accuracy, which leads us to choose

$$\Delta t \approx (\Delta x)^2.$$

Assuming $\Delta x < 1$, this requirement has the important consequence that the time steps must be *much* smaller than the space mesh size.

Remark: At this stage, the reader might be tempted to replace (10.12) by the second order central difference approximation (10.6). However, this produces significant complications, and the resulting numerical scheme is not practical; see Exercise ■.

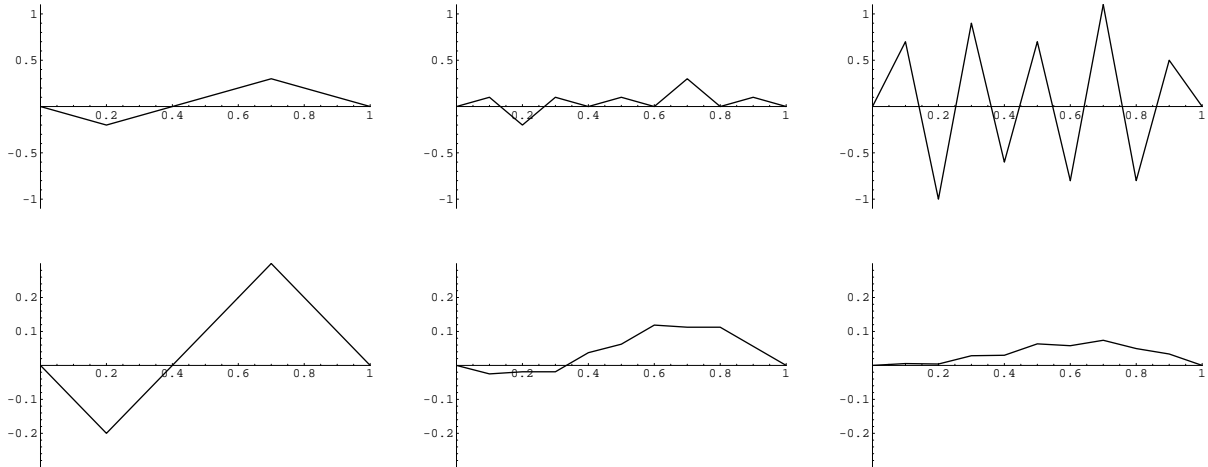


Figure 10.2. Numerical Solutions for the Heat Equation Based on the Explicit Scheme.

Example 10.4. Let us fix the diffusivity $\gamma = 1$ and the interval length $\ell = 1$. For illustrative purposes, we take a spatial step size of $\Delta x = .1$. In Figure 10.2 we compare two (slightly) different time step sizes on the same initial data as used in (4.25). The first sequence uses the time step $\Delta t = (\Delta x)^2 = .01$ and plots the solution at times $t = 0., .02, .04$. The numerical solution is already showing signs of instability, and indeed soon thereafter becomes completely wild. The second sequence takes $\Delta t = .005$ and plots the solution at times $t = 0., .025, .05$. (Note that the two sequences of plots have different vertical scales.) Even though we are employing a rather coarse mesh, the numerical solution is not too far away from the true solution to the initial value problem, which can be found in Figure 4.1.

In light of this calculation, we need to understand why our scheme sometimes gives reasonable answers but sometimes utterly fails. To this end, we investigate the effect of the numerical scheme on simple functions. As we know, the general solution to the heat equation can be decomposed into a sum over the various Fourier modes. Thus, we can concentrate on understanding what the numerical scheme does to an individual complex exponential[†] function, bearing in mind that we can then reconstruct its effect on more general data by taking suitable linear combinations of exponentials.

Suppose that, at a time $t = t_j$, the solution is a pure exponential

$$u(t_j, x) = e^{ikx}, \quad \text{and so} \quad u_{j,m} = u(t_j, x_m) = e^{ikx_m}. \quad (10.20)$$

Substituting the latter values into our numerical equations (10.13), we find the updated

[†] As usual, complex exponentials are much easier to work with than real trigonometric functions.

value at time t_{i+1} is also a sampled exponential:

$$\begin{aligned} u_{j+1,m} &= \mu u_{j,m+1} + (1 - 2\mu)u_{j,m} + \mu u_{j,m-1} \\ &= \mu e^{ikx_{m+1}} + (1 - 2\mu)e^{ikx_m} + \mu e^{ikx_{m-1}} \\ &= \mu e^{ik(x_m+\Delta x)} + (1 - 2\mu)e^{ikx_m} + \mu e^{ik(x_m-\Delta x)} = \lambda e^{ikx_m}, \end{aligned} \quad (10.21)$$

where

$$\lambda = \mu e^{ik\Delta x} + (1 - 2\mu) + \mu e^{-ik\Delta x} = 1 - 2\mu(1 - \cos k\Delta x) = 1 - 4\mu \sin^2 \frac{1}{2} k \Delta x. \quad (10.22)$$

Thus, the effect of a single step of the numerical scheme is to multiply the complex exponential (10.20) by a *magnification factor* λ :

$$u(t_{j+1}, x) = \lambda e^{ikx}. \quad (10.23)$$

In other words, e^{ikx} plays the role of an *eigenfunction*, with the magnification factor λ the corresponding *eigenvalue*, of the linear operator governing each step of the numerical scheme. Continuing in this fashion, we find that the effect of p further iterations of the scheme is to multiply the exponential by the p^{th} power of the magnification factor:

$$u(t_{j+p}, x) = \lambda^p e^{ikx}. \quad (10.24)$$

As a result, the stability of the scheme is governed by the size of the magnification factor: If $|\lambda| > 1$, then λ^p grows exponentially, and so the numerical solutions (10.24) become unbounded as $p \rightarrow \infty$. This is clearly incompatible with the analytical behavior of solutions to the heat equation. Therefore, an evident necessary condition for the stability of our numerical scheme is that its magnification factor satisfy

$$|\lambda| \leq 1. \quad (10.25)$$

This method of stability analysis was developed by the mid-twentieth century Hungarian/American mathematician — and father of the electronic computer — John von Neumann. The *stability criterion* (10.25) effectively distinguishes the stable, and hence valid numerical algorithms from the unstable, and hence worthless schemes. For the particular case (10.22), the von Neumann stability criterion (10.25) requires

$$-1 \leq 1 - 4\mu \sin^2 \frac{1}{2} k \Delta x \leq 1, \quad \text{or, equivalently,} \quad 0 \leq \mu \sin^2 \frac{1}{2} k \Delta x \leq \frac{1}{2}.$$

As this is required to hold for all possible wave numbers k , we must have

$$0 \leq \mu = \frac{\gamma \Delta t}{(\Delta x)^2} \leq \frac{1}{2}, \quad \text{and hence} \quad \Delta t \leq \frac{(\Delta x)^2}{2\gamma}, \quad (10.26)$$

since $\gamma > 0$. Thus, stability of the numerical scheme places a restriction on the allowable time step size. For instance, if $\gamma = 1$, and the space mesh size $\Delta x = .01$, then we must adopt a minuscule time step size $\Delta t \leq .00005$. It would take an inordinately large number of time steps to compute the value of the solution at even a moderate times, e.g., $t = 1$. Moreover, the propagation of round-off errors might then cause a significant reduction in

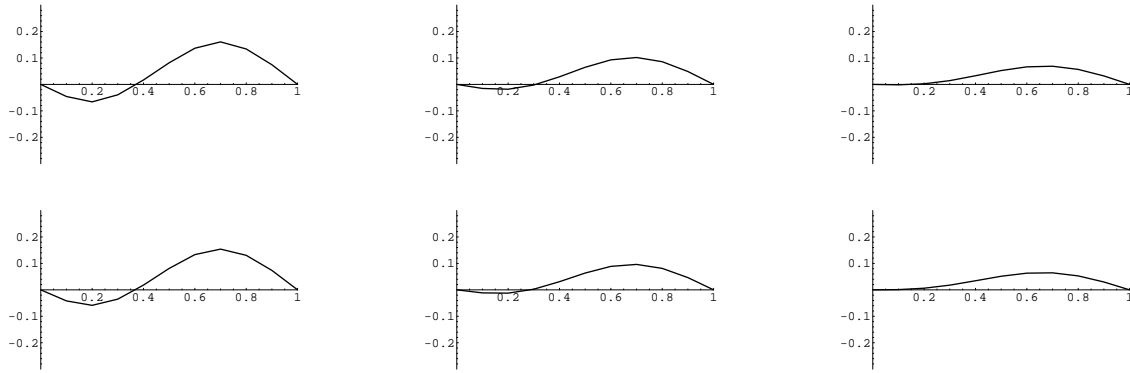


Figure 10.3. Numerical Solutions for the Heat Equation Based on the Implicit Scheme.

the overall accuracy of the final solution values. Since not all choices of space and time steps lead to a convergent scheme, the explicit scheme (10.13) is called *conditionally stable*.

An unconditionally stable method — one that does not restrict the time step — can be constructed by replacing the finite difference formula (10.12) used to approximate the time derivative by the backwards difference formula

$$\frac{\partial u}{\partial t}(t_j, x_m) \approx \frac{u(t_j, x_m) - u(t_{j-1}, x_m)}{\Delta t} + O((\Delta t)^2). \quad (10.27)$$

Substituting (10.27) and the same centered difference approximation (10.11) for u_{xx} into the heat equation, and then replacing j by $j + 1$, leads to the iterative system

$$-\mu u_{j+1, m+1} + (1 + 2\mu)u_{j+1, m} - \mu u_{j+1, m-1} = u_{j, m}, \quad \begin{array}{l} j = 0, 1, 2, \dots, \\ m = 1, \dots, n - 1, \end{array} \quad (10.28)$$

where the parameter $\mu = \gamma \Delta t / (\Delta x)^2$ is as above. The initial and boundary conditions have the same form (10.15, 16). The latter system can be written in the matrix form

$$\widehat{A} \mathbf{u}^{(j+1)} = \mathbf{u}^{(j)} + \mathbf{b}^{(j+1)}, \quad (10.29)$$

where \widehat{A} is obtained from the matrix A in (10.19) by replacing μ by $-\mu$. This serves to define an *implicit scheme*, since we have to solve a linear system of algebraic equations at each step in order to compute the next iterate $\mathbf{u}^{(j+1)}$. However, as the coefficient matrix \widehat{A} is tridiagonal, the system can be solved very rapidly, [104], and so speed is not a significant issue in the practical implementation of this implicit scheme.

Example 10.5. Consider the same initial-boundary value problem considered in Example 10.4. In Figure 10.3, we plot the numerical solutions obtained using the implicit scheme. The initial data is not displayed, but we graph the numerical solutions at times $t = .2, .4, .6$ with a mesh size of $\Delta x = .1$. On the top line, we use a time step of $\Delta t = .01$, while on the bottom $\Delta t = .005$. Unlike the explicit scheme, there is very little difference between the two — both come much closer to the actual solution than the explicit scheme. Indeed, even significantly larger time steps give reasonable numerical approximations to the solution.

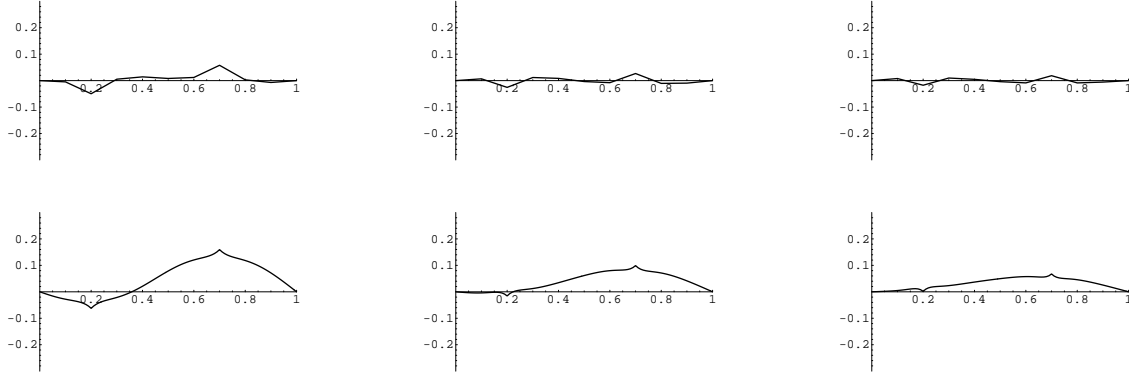


Figure 10.4. Numerical Solutions for the Heat Equation Based on the Crank–Nicolson Scheme.

Let us apply the von Neumann analysis to investigate the stability of the implicit scheme. Again, we need only look at the effect of the scheme on a complex exponential. Substituting (10.20, 23) into (10.28) and canceling the common exponential factor leads to the equation

$$\lambda(-\mu e^{ik\Delta x} + 1 + 2\mu - \mu e^{-ik\Delta x}) = 1.$$

We solve for the magnification factor

$$\lambda = \frac{1}{1 + 2\mu(1 - \cos k\Delta x)} = \frac{1}{1 + 4\mu \sin^2 \frac{1}{2} k\Delta x}. \quad (10.30)$$

Since $\mu > 0$, the magnification factor is *always* less than 1 in absolute value, and so the stability criterion (10.25) is satisfied *for any choice of step sizes*. We conclude that the implicit scheme (10.13) is *unconditionally stable*.

Another popular numerical scheme for solving the heat equation is the *Crank–Nicolson method*

$$u_{j+1,m} - u_{j,m} = \frac{1}{2}\mu(u_{j+1,m+1} - 2u_{j+1,m} + u_{j+1,m-1} + u_{j,m+1} - 2u_{j,m} + u_{j,m-1}), \quad (10.31)$$

which can be obtained by averaging the explicit and implicit schemes (10.13, 28). We can write (10.31) in vectorial form

$$B \mathbf{u}^{(j+1)} = C \mathbf{u}^{(j)} + \frac{1}{2}(\mathbf{b}^{(j)} + \mathbf{b}^{(j+1)}),$$

where

$$B = \begin{pmatrix} 1 + \mu & -\frac{1}{2}\mu & & & \\ -\frac{1}{2}\mu & 1 + \mu & -\frac{1}{2}\mu & & \\ & -\frac{1}{2}\mu & \ddots & \ddots & \\ & & \ddots & \ddots & \\ & & & \ddots & \ddots \end{pmatrix}, \quad C = \begin{pmatrix} 1 - \mu & \frac{1}{2}\mu & & & \\ \frac{1}{2}\mu & 1 - \mu & \frac{1}{2}\mu & & \\ & \frac{1}{2}\mu & \ddots & \ddots & \\ & & \ddots & \ddots & \\ & & & \ddots & \ddots \end{pmatrix}. \quad (10.32)$$

Applying the von Neumann analysis as before, we deduce that the magnification factor has the form

$$\lambda = \frac{1 - 2\mu \sin^2 \frac{1}{2} k \Delta x}{1 + 2\mu \sin^2 \frac{1}{2} k \Delta x}. \quad (10.33)$$

Since $\mu > 0$, we see that $|\lambda| \leq 1$ for all choices of step size, and so the Crank–Nicolson scheme is also unconditionally stable. A detailed analysis based on a Taylor expansion of the solution reveals that the error is of the order of $(\Delta t)^2$ and $(\Delta x)^2$, and so it is reasonable to choose the time step to have the same order of magnitude as the space step, $\Delta t \approx \Delta x$. This gives the Crank–Nicolson scheme one advantage over the previous two methods. However, applying it to the initial value problem considered earlier points out a significant weakness. Figure 10.4 shows the result of running the scheme on the initial data (4.25). The top row has space and time step sizes $\Delta t = \Delta x = .1$, and does a rather poor job replicating the solution. The bottom row uses $\Delta t = \Delta x = .01$, and performs better, except near the corners where an annoying and incorrect local time oscillation persists as the solution decays. Indeed, unlike the implicit scheme, the Crank–Nicolson method fails to rapidly damp out the high frequency modes associated with small scale features such as discontinuities and corners in the initial data. In such situations, a good strategy is to first evolve using the implicit scheme until the small scale noise is dissipated away, and then switch to Crank–Nicolson with a much larger time step for the final large scale changes.

Finally, we remark that the finite difference schemes developed above for the heat equation can all be readily adapted to more general parabolic partial differential equations. The stability criteria and observed behaviors follow similar lines. Some examples can be found in the exercises.

10.3. Numerical Methods for

First Order Partial Differential Equations.

Let us next use the method of finite differences to construct some basic numerical methods for first order partial differential equations. As noted in Section 4.4, first order partial differential equations can be regarded as prototypes for general hyperbolic partial differential equations, and so many of the lessons learned here carry over to the general hyperbolic regime, including the second order wave equation that we analyze in detail in the following section.

Consider the initial value problem for the elementary constant coefficient transport equation

$$\frac{\partial u}{\partial t} + c \frac{\partial u}{\partial x} = 0, \quad u(0, x) = f(x). \quad (10.34)$$

Of course, as we learned in Section 2.2 the solution is a simple traveling wave

$$u(t, x) = f(x - ct), \quad (10.35)$$

that is constant along the characteristic lines of slope c in the tx plane. Although the analytical solution is completely elementary, there will be valuable lessons to be learned from our attempt to reproduce it by numerical approximation. Indeed, each of the numerical

schemes developed below has an evident adaptation to transport equations with variable wave speeds $c(t, x)$, and even to nonlinear transport equations whose wave speed depends on the solution u , and so may admit shock wave solutions.

As usual, we restrict our attention to a regular mesh (t_j, x_m) with uniform time and space step sizes: $\Delta t = t_{j+1} - t_j$ and $\Delta x = x_{m+1} - x_m$ for all j, m . We use $u_{j,m} \approx u(t_j, x_m)$ to denote our numerical approximation to the solution $u(t, x)$ at the indicated mesh point. The most elementary numerical solution scheme is obtained by replacing the time and space derivatives by their first order finite difference approximations (10.1):

$$\frac{\partial u}{\partial t}(t_j, x_m) \approx \frac{u_{j+1,m} - u_{j,m}}{\Delta t} + O(\Delta t), \quad \frac{\partial u}{\partial x}(t_j, x_m) \approx \frac{u_{j,m+1} - u_{j,m}}{\Delta x} + O(\Delta x). \quad (10.36)$$

Substituting these expressions into the transport equation (10.34) leads to the explicit numerical scheme

$$u_{j+1,m} = -\sigma u_{j,m+1} + (\sigma + 1)u_{j,m}, \quad (10.37)$$

in which the parameter

$$\sigma = \frac{c \Delta t}{\Delta x} \quad (10.38)$$

depends upon the wave speed and the ratio of space and time step sizes. Since we are employing first order approximations to both derivatives, we should choose the time and space step sizes to be of comparable size $\Delta t \approx \Delta x$. When working on a bounded interval, say $0 \leq x \leq \ell$, we will need to specify a value for the numerical solution at the right end, e.g., setting $u_{j,n} = 0$, which corresponds to imposing the boundary condition $u(t, \ell) = 0$.

In Figure 10.5, we plot the solutions arising from the following initial conditions

$$u(0, x) = f(x) = .4 e^{-300(x-.5)^2} + .1 e^{-300(x-.65)^2}, \quad (10.39)$$

at times $t = 0, .15$ and $.3$. We use step sizes $\Delta t = \Delta x = .005$, and try four different values of the wave speed. The cases $c = .5$ and $c = -1.5$ are clearly exhibiting some form of numerical instability. The numerical solution $c = -.5$ is a bit more reasonable, although one can already observe some degradation due to the relatively low accuracy of the scheme. This can be overcome by selecting a smaller step size. The case $c = -1$ looks particularly good — but this is an accident. The analytic solution (10.35) happens to be an *exact* solution to the numerical equations, and so the only possible source of error is round-off.

There are two ways to understand the observed numerical instability. First, we recall the exact solution (10.35) is constant along the characteristic lines $x = ct + \xi$, and hence the value of $u(t, x)$ only depends on the initial value $f(\xi)$ at the point $\xi = x - ct$. On the other hand, at a time $t = t_k$, the numerical solution $u_{k,m} \approx u(t_k, x_m)$ computed using (10.37) depends on the values of $u_{k-1,m}$ and $u_{k-1,m+1}$. The latter two values have been computed from the previous approximations $u_{k-2,m}, u_{k-2,m+1}, u_{k-2,m+2}$. And so on. Going all the way back to the initial time $t_0 = 0$, we find that $u_{n,m}$ depends on the initial values $u_{0,m} = f(x_m), \dots, u_{0,m+k} = f(x_m + k \Delta x)$ at the mesh points lying in the interval $x_m \leq x \leq x_m + k \Delta x$. On the other hand, the actual solution $u(t_k, x_m)$ depends only on the value of $f(\xi)$ where

$$\xi = x_m - ct_k = x_m - ck \Delta t.$$

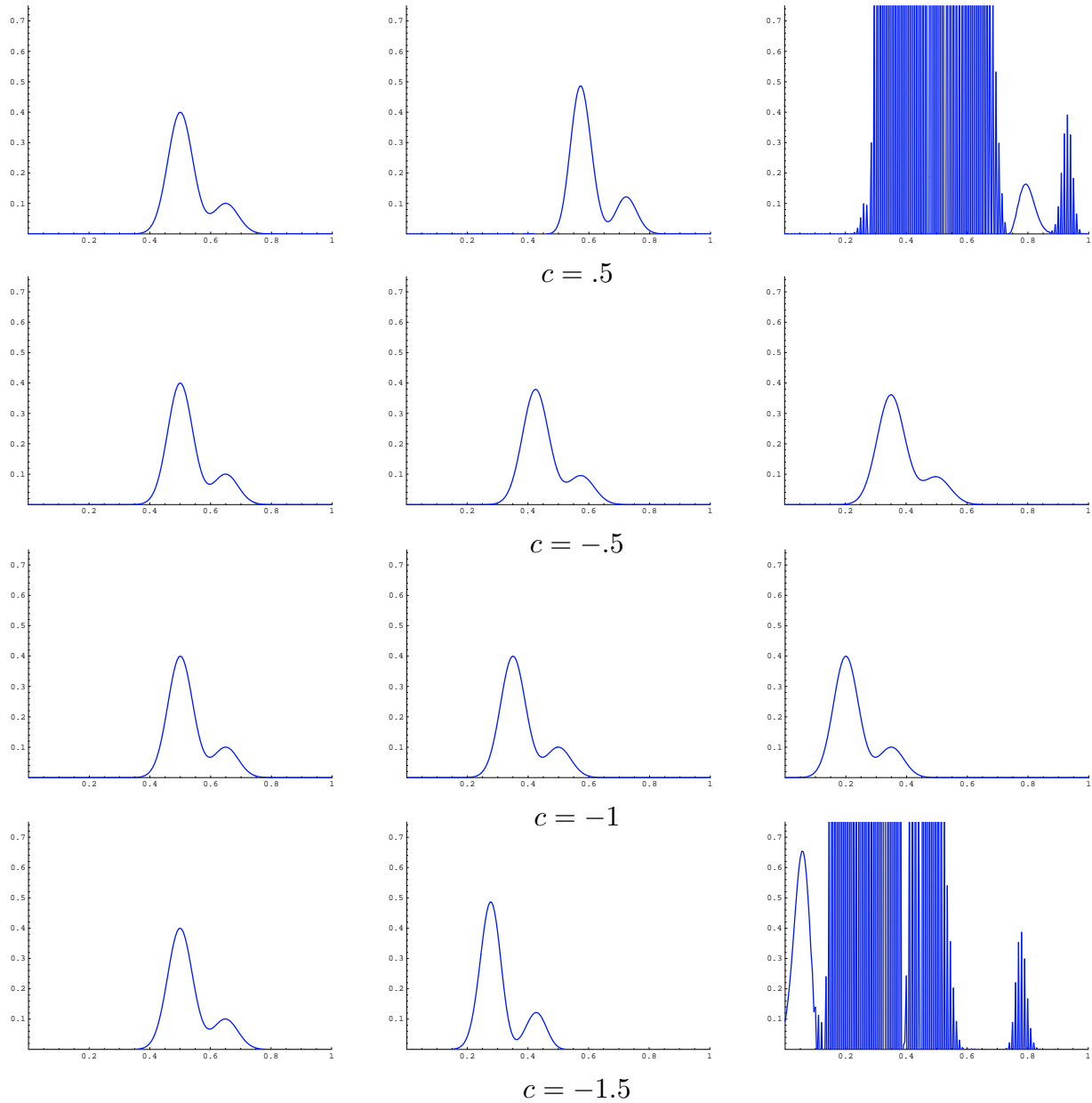


Figure 10.5. Numerical Solutions to the Transport Equation.

Thus, if ξ lies outside the interval $[x_m, x_m + k \Delta x]$, then varying the initial condition $f(x)$ nearby $x = \xi$ will change the solution value $u(t_k, x_m)$ without affecting its numerical approximation $u_{k,m}$ at all! So the numerical scheme cannot possibly provide an accurate approximation to the solution value. As a result, we must require

$$x_m \leq \xi = x_m - cn \Delta t \leq x_m + n \Delta x, \quad \text{and hence} \quad 0 \leq -ck \Delta t \leq k \Delta x,$$

which we rewrite as

$$0 \geq \sigma = \frac{c \Delta t}{\Delta x} \geq -1, \quad \text{or, equivalently,} \quad -\frac{\Delta x}{\Delta t} \leq c \leq 0. \quad (10.40)$$

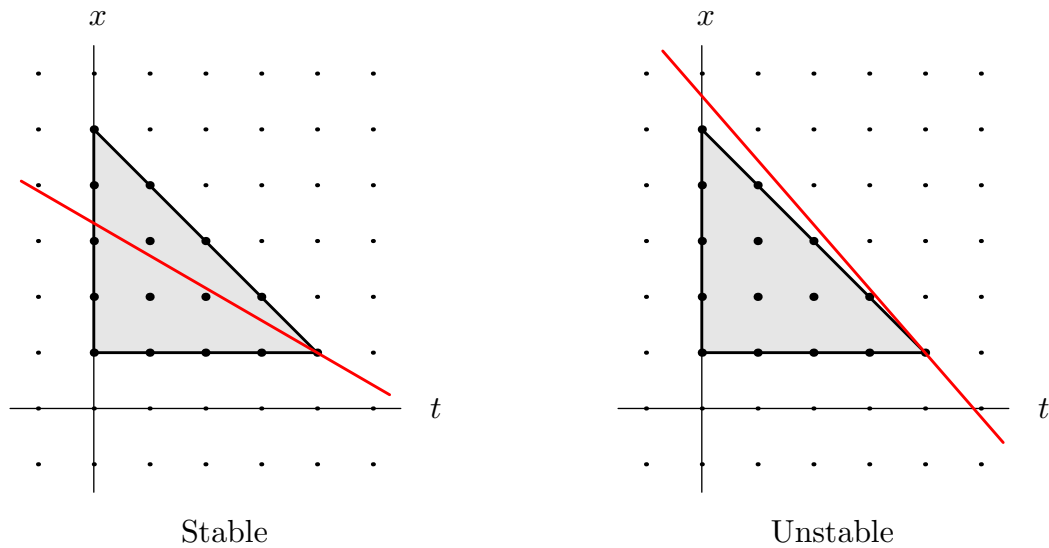


Figure 10.6. The CFL Condition.

This is the simplest manifestation of the *Courant–Friedrichs–Lewy condition*, or *CFL condition* for short, established in the 1950’s by three of the pioneers in the development of numerical solution schemes for hyperbolic partial differential equations. Note that the CFL condition requires that the wave speed be *negative*, but not too negative. For allowable wave speeds, the method is conditionally stable, since stability restricts the possible time step sizes.

The CFL condition can be recast in a more geometrically transparent manner as follows. For the finite difference scheme (10.37), the *numerical domain of dependence* of a point (t_k, x_m) is the triangle

$$T_{(t_k, x_m)} = \{ (t, x) \mid 0 \leq t \leq t_k, x_m \leq x \leq x_m + t_k - t \}. \quad (10.41)$$

The reason for this term is that the numerical approximation to the solution at the mesh point (t_k, x_m) depends on the computed values at the mesh points lying within its numerical domain of dependence; see Figure 10.6. The *CFL condition* (10.40) requires that, for all $0 \leq t \leq t_k$, the characteristic passing through the point (t_k, x_m) lies entirely within the numerical domain of dependence (10.41). If the characteristic ventures outside the domain, then the scheme will be numerically unstable. With this geometric reformulation, the CFL criterion can be applied to both linear and nonlinear transport equations that have non-uniform wave speeds.

The CFL criterion (10.40) is reconfirmed by a von Neumann stability analysis. As before, we test the numerical scheme on an exponential function: Substituting

$$u_{j,m} = e^{ikx_m}, \quad u_{j+1,m} = \lambda e^{ikx_m}, \quad (10.42)$$

into (10.37) leads to

$$\lambda e^{ikx_m} = -\sigma e^{ikx_{m+1}} + (\sigma + 1)e^{ikx_m} = (-\sigma e^{ik\Delta x} + \sigma + 1)e^{ikx_m}.$$

The resulting magnification factor

$$\lambda = 1 + \sigma(1 - e^{ik\Delta x}) = (1 + \sigma - \sigma \cos(k\Delta x)) - i\sigma \sin(k\Delta x)$$

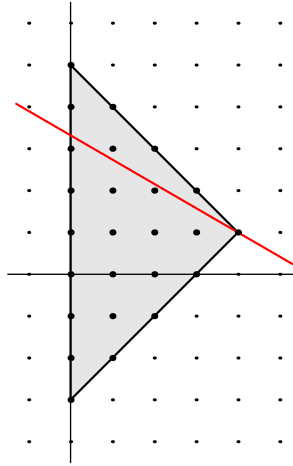


Figure 10.7. The CFL Condition for the Centered Difference Scheme.

satisfies the stability criterion (10.25) if and only if

$$|\lambda|^2 = (1 + \sigma - \sigma \cos(k \Delta x))^2 + (\sigma \sin(k \Delta x))^2 = 1 + 2\sigma(\sigma + 1)(1 - \cos(k \Delta x)) \leq 1$$

for all k . Thus, stability requires that $\sigma(\sigma + 1) \leq 0$, and thus $-1 \leq \sigma \leq 0$, in complete accord with the CFL condition (10.40).

To obtain a finite difference scheme that can be used for positive wave speeds, we replace the forward finite difference approximation to $\partial u / \partial x$ by the corresponding backwards difference quotient, namely, (10.1) with $h = -\Delta x$, leading to the alternative first order numerical scheme

$$u_{j+1,m} = (1 - \sigma)u_{j,m} + \sigma u_{j,m-1}, \quad (10.43)$$

where $\sigma = c\Delta t / \Delta x$ is as above. A similar analysis, left to the reader, produces the corresponding CFL stability criterion

$$0 \leq \sigma = \frac{c\Delta t}{\Delta x} \leq 1,$$

and so this scheme can be applied to suitable positive wave speeds.

In this manner, we have produced one numerical scheme that works for negative wave speeds, and an alternative scheme for positive speeds. The question arises — particularly when one is dealing with equations with variable wave speeds — whether one can devise a scheme that is (conditionally) stable for *both* positive and negative wave speeds. One might be tempted to use the centered difference approximation

$$\frac{\partial u}{\partial x}(t_j, x_m) \approx \frac{u_{j,m+1} - u_{j,m-1}}{\Delta x} + O((\Delta x)^2), \quad (10.44)$$

cf. (10.6). Substituting (10.44) and the previous approximation to the time derivative (10.36) into (10.34) leads to the numerical scheme

$$u_{j+1,m} = -\frac{1}{2}\sigma u_{j,m+1} + u_{j,m} + \frac{1}{2}\sigma u_{j,m-1}, \quad (10.45)$$

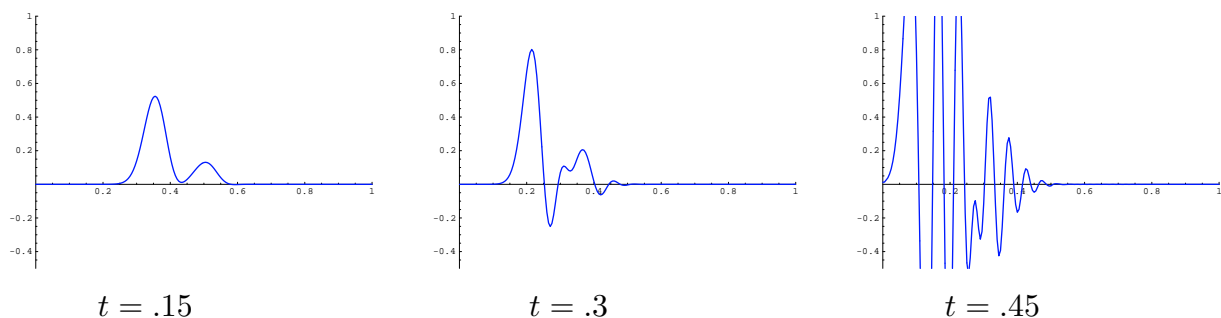


Figure 10.8. Centered Difference Numerical Solution to the Transport Equation.

where $\sigma = c \Delta t / \Delta x$ is the same as above. In this case, the *numerical domain of dependence* of the mesh point (t_n, x_m) consists of the mesh points in the triangle

$$\tilde{T}_{(t_k, x_m)} = \{ (t, x) \mid 0 \leq t \leq t_k, x_m - t_k + t \leq x \leq x_m + t_k - t \}. \quad (10.46)$$

The CFL condition requires that, for $0 \leq t \leq t_k$, the characteristic going through (t_k, x_m) lie within this triangle, as in Figure 10.7, which imposes the condition

$$|\sigma| = \left| \frac{c \Delta t}{\Delta x} \right| \leq 1, \quad \text{or, equivalently,} \quad |c| \leq \frac{\Delta x}{\Delta t}. \quad (10.47)$$

Unfortunately, although it satisfies the CFL condition over this range of wave speeds, the centered difference scheme is, in fact, always *unstable*! For instance, the instability of the numerical solution to the preceding initial value problem (10.39) for $c = 1$ can be observed in Figure 10.8. This is confirmed by applying the von Neumann analysis: We substitute (10.42) into (10.45), and cancel the common exponential factors. Provided $\sigma \neq 0$, which means that $c \neq 0$, the resulting magnification factor

$$\lambda = 1 - i \sigma \sin(k \Delta x)$$

satisfies $|\lambda| > 1$ for all k with $\sin(k \Delta x) \neq 0$. Thus, for $c \neq 0$, the centered difference scheme (10.45) is unstable for all (nonzero) wave speeds!

One elementary means of overcoming the sign restriction on the wave speed is to use the forward difference scheme (10.37) when the wave speed is negative and the backwards scheme (10.43) when it is positive. The resulting scheme, valid for varying wave speeds $c(t, x)$, takes the form

$$u_{j+1, m} = \begin{cases} -\sigma_{j, m} u_{j, m+1} + (\sigma_{j, m} + 1) u_{j, m}, & c_{j, m} \leq 0, \\ (1 - \sigma_{j, m}) u_{j, m} + \sigma_{j, m} u_{j, m-1}, & c_{j, m} > 0, \end{cases} \quad \text{where} \quad \begin{cases} \sigma_{j, m} = c_{j, m} \frac{\Delta t}{\Delta x}, \\ c_{j, m} = c(t_j, x_m). \end{cases} \quad (10.48)$$

In the literature, this is referred to as an *upwind scheme*, since the second mesh point always lies “upwind” — that is away from the direction of motion — from the reference point (t_j, x_m) . The upwind scheme works reasonably well over short time intervals assuming the space step size is sufficiently small and the time step satisfies the CFL condition $\Delta x / \Delta t \leq |c_{j, m}|$ at each mesh point, cf. (10.40). However, over longer time intervals, as

we already observed in Figure 10.5, it tends to exhibit an unacceptable rate of damping of waves or, alternatively, require an unacceptably small step size. One way of overcoming this defect is to use the popular *Lax–Wendroff scheme*

$$u_{j+1,m} = \frac{1}{2}\sigma(1+\sigma)u_{j,m+1} + (1-\sigma^2)u_{j,m} - \frac{1}{2}\sigma(1-\sigma)u_{j,m-1}, \quad (10.49)$$

which is based on second order approximations to the derivatives, [95]. The stability analysis of the Lax–Wendroff scheme is relegated to the exercises.

10.4. Numerical Algorithms for the Wave Equation.

Let us now turn to basic numerical solution techniques for the second order wave equation. As above, although we are in possession of the explicit d’Alembert solution formula (2.80), the lessons learned in designing stable schemes in the simplest case will carry over to more complicated situations, including inhomogeneous media and higher dimensional problems, where analytic solution formulas are no longer readily available.

Consider the second order wave equation

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}, \quad 0 < x < \ell, \quad t \geq 0, \quad (10.50)$$

in one space dimension. For specificity, we impose (possibly time dependent) Dirichlet boundary conditions

$$u(t, 0) = \alpha(t), \quad u(t, \ell) = \beta(t), \quad t \geq 0, \quad (10.51)$$

along with the usual initial conditions

$$u(0, x) = f(x), \quad \frac{\partial u}{\partial t}(0, x) = g(x), \quad 0 \leq x \leq \ell. \quad (10.52)$$

As before, we adopt a uniformly spaced mesh

$$t_j = j \Delta t, \quad x_m = m \Delta x, \quad \text{where} \quad \Delta x = \frac{\ell}{n}.$$

In order to discretize the wave equation, we replace the second order derivatives by their standard finite difference approximations (10.5):

$$\begin{aligned} \frac{\partial^2 u}{\partial t^2}(t_j, x_m) &\approx \frac{u(t_{j+1}, x_m) - 2u(t_j, x_m) + u(t_{j-1}, x_m)}{(\Delta t)^2} + O((\Delta t)^2), \\ \frac{\partial^2 u}{\partial x^2}(t_j, x_m) &\approx \frac{u(t_j, x_{m+1}) - 2u(t_j, x_m) + u(t_j, x_{m-1}))}{(\Delta x)^2} + O((\Delta x)^2). \end{aligned} \quad (10.53)$$

Since the errors are of orders of $(\Delta t)^2$ and $(\Delta x)^2$, we anticipate being able to choose the space and time step sizes to have comparable magnitude:

$$\Delta t \approx \Delta x.$$

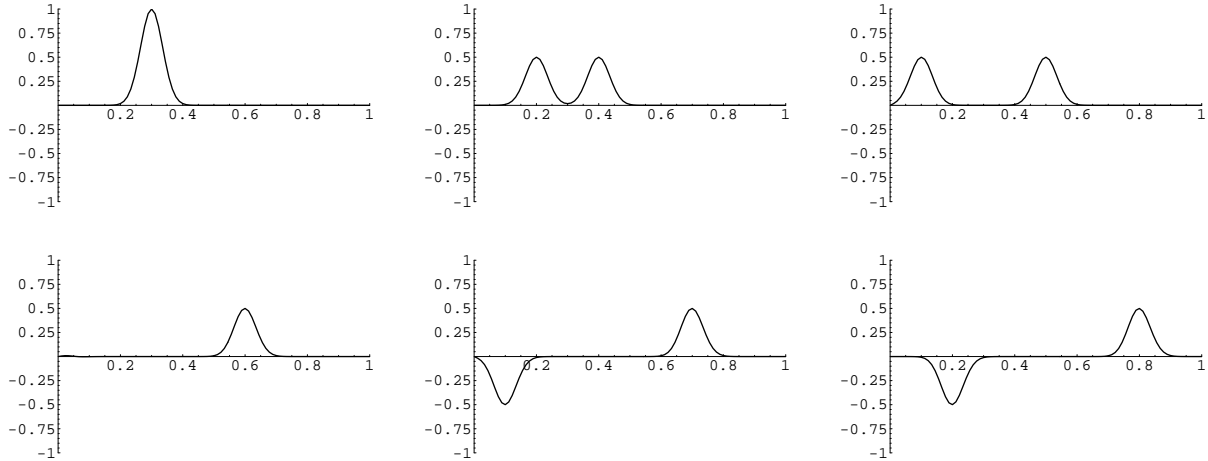


Figure 10.9. Numerically Stable Waves.

To construct an initial approximation to $\mathbf{u}^{(1)}$ with error on the order of $(\Delta t)^2$, we need to analyze the error in the approximation (10.59) in more detail. Note that, by Taylor's theorem,

$$\begin{aligned} \frac{u(\Delta t, x_m) - u(0, x_m)}{\Delta t} &= \frac{\partial u}{\partial t}(0, x_m) + \frac{\Delta t}{2} \frac{\partial^2 u}{\partial t^2}(0, x_m) + O((\Delta t)^2) \\ &= \frac{\partial u}{\partial t}(0, x_m) + \frac{c^2 \Delta t}{2} \frac{\partial^2 u}{\partial x^2}(0, x_m) + O((\Delta t)^2), \end{aligned}$$

where, in the final equality, we have used the fact that $u(t, x)$ solves the wave equation. Therefore,

$$\begin{aligned} u_{1,m} = u(\Delta t, x_m) &\approx u(0, x_m) + \Delta t \frac{\partial u}{\partial t}(0, x_m) + \frac{c^2 (\Delta t)^2}{2} \frac{\partial^2 u}{\partial x^2}(0, x_m) \\ &= f(x_m) + \Delta t g(x_m) + \frac{c^2 (\Delta t)^2}{2} f''(x_m) \approx f_m + \Delta t g_m + \frac{c^2 (\Delta t)^2}{2 (\Delta x)^2} (f_{m+1} - 2f_m + f_{m-1}), \end{aligned}$$

where we can use the finite difference approximation (10.5) for the second derivative of $f(x)$ if the explicit formula is either not known or too complicated to program. Therefore, we initiate the scheme by setting

$$u_{1,m} = \frac{1}{2} \sigma^2 f_{m+1} + (1 - \sigma^2) f_m + \frac{1}{2} \sigma^2 f_{m-1} + \Delta t g_m, \quad (10.60)$$

or, in matrix form,

$$\mathbf{u}^{(0)} = \mathbf{f}, \quad \mathbf{u}^{(1)} = \frac{1}{2} B \mathbf{u}^{(0)} + \Delta t \mathbf{g} + \frac{1}{2} \mathbf{b}^{(0)}. \quad (10.61)$$

This maintains the desired order $(\Delta t)^2$ (and $(\Delta x)^2$) of accuracy.

Example 10.6. Consider the particular initial value problem

$$\begin{aligned} u_{tt} = u_{xx}, \quad u(0, x) &= e^{-400(x-.3)^2}, \quad u_t(0, x) = 0, \quad 0 \leq x \leq 1, \\ u(t, 0) = u(t, 1) &= 0, \quad t \geq 0, \end{aligned}$$

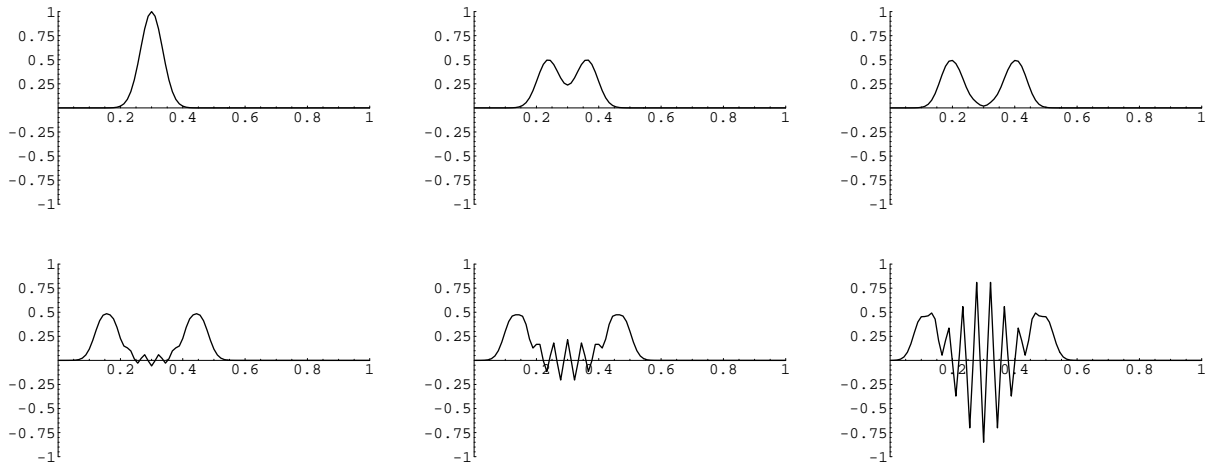


Figure 10.10. Numerically Unstable Waves.

subject to homogeneous Dirichlet boundary conditions on the interval $[0, 1]$. The initial data is a fairly concentrated hump centered at $x = .3$. As time progresses, we expect the initial hump to split into two half sized humps, which then collide with the ends of the interval, in accordance with the d'Alembert solution.

For our numerical approximation, let's use a space discretization consisting of 90 equally spaced points, and so $\Delta x = \frac{1}{90} = .0111\dots$. If we choose a time step of $\Delta t = .01$, whereby $\sigma = .9$, then we get reasonably accurate solution over a fairly long time range, as plotted in Figure 10.9 at times $t = 0, .1, .2, \dots, .5$. On the other hand, if we double the time step, setting $\Delta t = .02$, so $\sigma = 1.8$, then, as plotted in Figure 10.10 at times $t = 0, .05, .1, .14, .16, .18$, we observe an instability that eventually overwhelms the numerical solution. Thus, the numerical scheme appears to only be conditionally stable.

The stability analysis of this numerical scheme proceeds along the same lines as in the first order case. The CFL condition requires that the characteristics emanating from a mesh point (t_k, x_m) must, for $0 \leq t \leq t_k$, remain in its numerical domain of dependence, which, for our particular numerical scheme, is the same triangle

$$\tilde{T}_{(t_k, x_m)} = \{ (t, x) \mid 0 \leq t \leq t_k, x_m - t_k + t \leq x \leq x_m + t_k - t \},$$

that we plotted Figure 10.11. Since the characteristics are the lines of slope $\pm c$, the CFL condition is the same as in (10.47):

$$\sigma = \frac{c \Delta t}{\Delta x} \leq 1, \quad \text{or, equivalently,} \quad 0 < c \leq \frac{\Delta x}{\Delta t}. \quad (10.62)$$

This explains the difference between the numerically stable and unstable cases exhibited above.

However, as we noted above, the CFL condition is, in general, only necessary for stability of the numerical scheme; sufficiency requires that we perform a von Neumann stability analysis. As before, we specialize the calculation to a single complex exponential e^{ikx} . After one time step, the scheme will have the effect of multiplying it by the (possibly

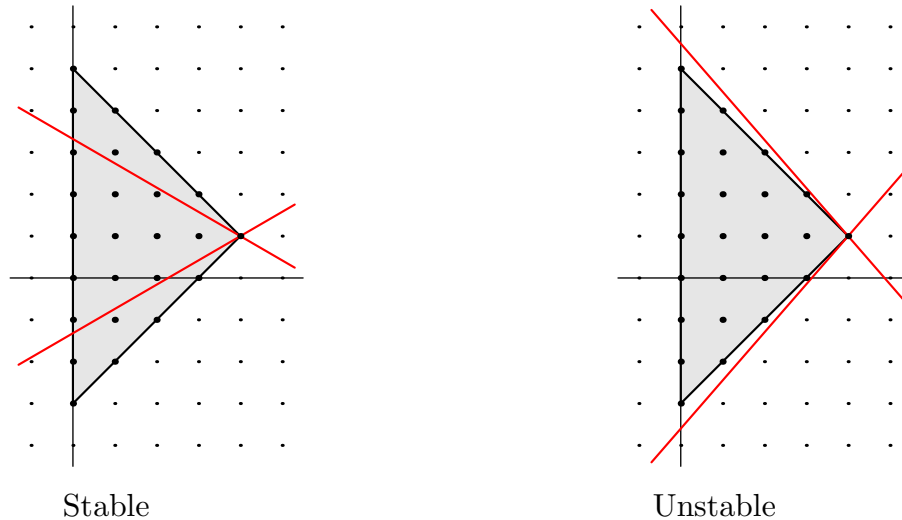


Figure 10.11. The CFL Condition for the Wave Equation.

complex) *magnification factor* $\lambda = \lambda(k)$, after another time step by λ^2 , and so on. Stability requires that all such magnification factors satisfy $|\lambda| \leq 1$. To determine the magnification factors, we substitute the relevant sampled exponential values

$$u_{j-1,m} = e^{ikx_m}, \quad u_{j,m} = \lambda e^{ikx_m}, \quad u_{j+1,m} = \lambda^2 e^{ikx_m}, \quad (10.63)$$

into the scheme (10.54). After canceling the common exponential, we find that any magnification factor will satisfy the following quadratic equation:

$$\lambda^2 = (2 - 4\sigma^2 \sin^2 \frac{1}{2} k \Delta x) \lambda + 1,$$

whence

$$\lambda = \alpha \pm \sqrt{\alpha^2 - 1}, \quad \text{where} \quad \alpha = 1 - 2\sigma^2 \sin^2 \frac{1}{2} k \Delta x. \quad (10.64)$$

Thus, there are *two* different magnification factors associated with each complex exponential — which is a consequence of the scheme being of second order. Stability requires that both be ≤ 1 in modulus. Now, if the CFL condition (10.62) holds, then $|\alpha| \leq 1$, which implies that the magnification factors (10.64) are complex numbers of modulus $|\lambda| = 1$, and thus the numerical scheme satisfies the stability criterion (10.25). On the other hand, if $\sigma > 1$, then, for a range of values of k , we have $\alpha < -1$. This implies that the two magnification factors (10.64) are both real, and one of them is < -1 , which thus violates the stability criterion. Thus, the CFL condition (10.62) does indeed distinguish between the (conditionally) stable and unstable numerical schemes for the wave equation.

10.5. Finite Elements.

The second of the two major numerical paradigms for partial differential equations is the finite element method. Finite elements are of more recent vintage than finite differences, having been developed after the second world war; see [129] for historical details. Finite elements have, for the most part, become the method of choice for solving elliptic partial differential equations, owing to their flexibility for handling complicated geometries.

Finite elements rely on a more sophisticated understanding of the partial differential equation, and are not obtained by simply replacing derivatives by numerical approximations. There are two ways to motivate the underlying construction. The first is based on minimization principles which, as we learned in Chapter 9, can be used to characterize the analytic solution. The key idea is to restrict the infinite-dimensional minimization principle characterizing the exact solution to a suitably chosen finite-dimensional subspace of the function space. When properly formulated, the solution to the resulting finite-dimensional minimization problem approximates the true minimizer.

An alternative formulation of the finite element solution, that can be applied even in situations where there is no minimum principle available, is based on the idea of a weak solution to the boundary value problem. Rather than impose the weak solution criterion on the entire infinite-dimensional function space, one again restricts to a suitably chosen finite-dimensional subspace. For positive definite boundary value problems, which necessarily admit a minimization principle, the weak solution approach leads to the same finite element equations as the minimization approach. While the weak solution approach is of wider applicability, outside of boundary value problems with well-defined minimization principles, there is not necessarily a rigorous underpinning that guarantees that the numerical solution is close to the actual solution. Indeed, one can find boundary value problems without analytic solutions that have spurious finite element numerical solutions, and, vice versa, boundary value problems with solutions for which some finite element approximations do not exist.

To introduce the finite element method, it will help to begin with one-dimensional boundary value problems involving ordinary differential equations. After the basic constructions have been assimilated in this simpler context, the reader will be ready to handle boundary value problems governed by partial differential equations to be discussed in Section 10.6. A rigorous justification, under appropriate hypotheses, requires further analysis of the finite element method, and we refer the interested reader to [129, 142]. Here we shall concentrate on trying to understand how to apply the method in practice.

Minimization and Finite Elements

To set the stage, we return to the abstract framework developed in Chapter 9. The key fact is that we are able to characterize the solution to a positive definite boundary value problem as the function $u_\star \in U$, belonging to a certain infinite-dimensional function space, that minimizes an associated quadratic functional $\mathcal{P}: U \rightarrow \mathbb{R}$. Now comes the first key idea of the finite element method. Instead of trying to minimize the functional over the entire infinite-dimensional function space, we will minimize it over a *finite-dimensional subspace* $W \subset U$. The effect is to reduce a problem in analysis — solving a differential equation plus boundary conditions — to a problem in linear algebra, and hence one that a computer can solve. On the surface, the idea seems slightly crazy: how could one expect to come close to finding the minimizer in a gigantic infinite-dimensional function space by restricting the search to a measly finite-dimensional subspace. But this is where the magic of infinite dimensions comes into play. One can, in fact, approximate all (reasonable) functions arbitrarily closely by functions belonging to finite-dimensional subspaces. Indeed, you are already familiar with two examples: Fourier series, where one approximates rather

general periodic functions by trigonometric polynomials, and polynomial interpolation, in which one approximates functions by polynomials. Thus, the finite element idea is not as crazy as it initially seems.

To be a bit more explicit, let us begin with a linear (differential) operator $L:U \rightarrow V$ between inner product spaces, with trivial kernel: $\ker L = \{0\}$. As we learned in Theorem 9.25, the element $u_* \in U$ that minimizes

$$\mathcal{P}[u] = \frac{1}{2} \|L[u]\|^2 - \langle f; u \rangle \quad (10.65)$$

is the solution to the linear system

$$K[u] = f, \quad \text{where} \quad K = L^* \circ L. \quad (10.66)$$

The hypothesis on L implies that $K > 0$ is a self-adjoint, positive definite linear operator, and the solution to (10.66) is unique. In our applications, L is a linear differential operator between function spaces, e.g., the gradient, $\mathcal{P}[u]$ represents a quadratic functional, e.g., the Dirichlet principle, and the associated positive definite linear system (10.66) a boundary value problem, e.g., the Poisson equation along with suitable self-adjoint boundary conditions.

Rather than try to minimize $\mathcal{P}[u]$ on the entire function space U , we now seek to minimize it on a suitably chosen finite-dimensional subspace $W \subset U$. We will specify W by selecting a basis $\varphi_1, \dots, \varphi_n \in U$. Then the general element of W is a (uniquely determined) linear combination

$$\varphi(x) = c_1\varphi_1(x) + \dots + c_n\varphi_n(x) \quad (10.67)$$

of the basis functions. Our goal is to minimize $\mathcal{P}[\varphi]$ over all possible elements (10.67); in other words, we need to determine the coefficients c_1, \dots, c_n such that

$$\mathcal{P}[\varphi] = \mathcal{P}[c_1\varphi_1 + \dots + c_n\varphi_n] = Q(c)$$

is as small as possible. Substituting (10.67) back into (10.65), and then expanding, using the linearity of L and then the bilinearity of the inner product, we find

$$Q(c) = \frac{1}{2} \sum_{i,j=1}^n m_{ij} c_i c_j - \sum_{i=1}^n b_i c_i = \frac{1}{2} \mathbf{c}^T M \mathbf{c} - \mathbf{c}^T \mathbf{b}, \quad (10.68)$$

where

- (a) $\mathbf{c} = (c_1, c_2, \dots, c_n)^T$ is the vector of unknown coefficients in (10.67),
- (b) $M = (m_{ij})$ is the symmetric $n \times n$ matrix with entries

$$m_{ij} = \langle L[\varphi_i]; L[\varphi_j] \rangle, \quad i, j = 1, \dots, n, \quad (10.69)$$

- (c) $\mathbf{b} = (b_1, b_2, \dots, b_n)^T$ is the vector with entries

$$b_i = \langle f; \varphi_i \rangle, \quad i = 1, \dots, n. \quad (10.70)$$

Remark: Note that formula (10.69) uses the inner product on the target space V , whereas (10.70) relies on the inner product on the domain space U .

Thus, once we specify the basis functions φ_i , the coefficients m_{ij} and b_i are all known quantities. Therefore, we have reduced our original problem to a finite-dimensional problem of minimizing the quadratic function (10.68) over all possible vectors $\mathbf{c} \in \mathbb{R}^n$. The coefficient matrix M is, in fact, positive definite, since, by the preceding computation,

$$\mathbf{c}^T M \mathbf{c} = \sum_{i,j=1}^n m_{ij} c_i c_j = \|L[c_1 \varphi_1(x) + \cdots + c_n \varphi_n]\|^2 = \|L[\varphi]\|^2 > 0 \quad (10.71)$$

as long as $L[\varphi] \neq 0$. Moreover, our positivity assumption implies that $L[\varphi] = 0$ if and only if $\varphi \equiv 0$, and hence (10.71) is indeed positive for all $\mathbf{c} \neq \mathbf{0}$. We can now invoke the finite-dimensional minimization result of Example 9.24 to conclude that the unique minimizer to (10.68) is obtained by solving the associated linear system

$$M \mathbf{c} = \mathbf{b}, \quad \text{and hence} \quad \mathbf{c} = M^{-1} \mathbf{b}. \quad (10.72)$$

Remark: To solve the linear system (10.72), we can, when not too large, rely on basic Gaussian Elimination. When the size of the system (i.e., the dimension of the subspace W) becomes too large, as is often the case in dealing with partial differential equations, it is better to rely on an iterative linear system solver, e.g., the Gauss–Seidel method or SOR, [104].

This constitutes the basic abstract setting for the finite element method. The key issue, then, is how to effectively choose the finite-dimensional subspace W . Two candidates that might spring to mind are the space of polynomials of degree $\leq n$, or the space of trigonometric polynomials (truncated Fourier series) of degree $\leq n$. However, for a variety of reasons, neither is well suited to the finite element method. One criterion is that the functions in W must satisfy the relevant boundary conditions — otherwise W would not be a subspace of U . More importantly, in order to obtain sufficient accuracy, the linear algebraic system (10.72) will typically be rather large, and so, to be efficiently solved, the coefficient matrix M should be as sparse as possible, i.e., have lots of zero entries. Otherwise, computing the solution will be too time-consuming to be of much practical value. Such considerations prove to be of absolutely crucial importance when applying the method to solve boundary value problems for partial differential equations in higher dimensions.

The second innovative contribution of the finite element method is to first (paradoxically) *enlarge* the space U of allowable functions upon which to minimize the quadratic functional $\mathcal{P}[u]$. The governing differential equation requires its (classical) solutions to have a certain degree of smoothness, whereas the associated minimization principle typically requires only half as many continuous derivatives. Thus, for second order boundary value problems, the differential equation requires continuous second order derivatives, while the quadratic functional $\mathcal{P}[u]$ only involves first order derivatives. It can be rigorously shown that, under rather mild hypotheses, the functional has the *same* minimizing solution, even if one allows functions that fail to qualify as classical solutions to the differential equation.

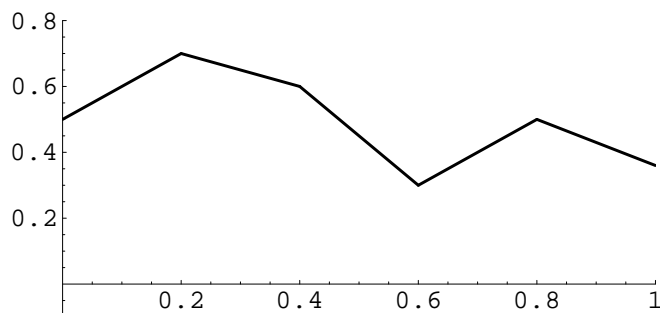


Figure 10.12. A Continuous Piecewise Affine Function.

Finite Elements for Ordinary Differential Equations

To make the preceding discussion concrete, let us focus our attention on a boundary value problem governed by a second order ordinary differential equation. For example, we might consider a Sturm–Liouville problem with homogeneous Dirichlet (fixed) boundary conditions. The basic ideas carry over in their essence to more general contexts.

For such boundary value problems, a popular and effective choice of the finite-dimensional subspace W is to use continuous, piecewise affine functions. Recall that a function is affine, $f(x) = ax + b$, if and only if its graph is a straight line. The function is *piecewise affine* if its graph consists of a finite number of straight line segments; a typical example is plotted in Figure 10.12. Continuity requires that the individual line segments be connected together end to end.

Given a boundary value problem on a bounded interval $[a, b]$, let us fix a finite collection of *mesh points*

$$a = x_0 < x_1 < x_2 < \cdots < x_{n-1} < x_n = b.$$

The formulas simplify if one uses equally spaced mesh points, but this is not necessary for the method to apply. Let W denote the vector space consisting of all continuous, piecewise affine functions, with corners at the nodes, that satisfy the homogeneous boundary conditions. Thus, for Dirichlet boundary conditions, we require that

$$\varphi(a) = \varphi(b) = 0. \tag{10.73}$$

Thus, on each subinterval

$$\varphi(x) = c_j + b_j(x - x_j), \quad \text{for } x_j \leq x \leq x_{j+1}, \quad j = 0, \dots, n - 1.$$

Continuity of $\varphi(x)$ requires

$$c_j = \varphi(x_j^+) = \varphi(x_j^-) = c_{j-1} + b_{j-1}h_{j-1}, \quad j = 1, \dots, n - 1, \tag{10.74}$$

where $h_{j-1} = x_j - x_{j-1}$ denotes the length of the j^{th} subinterval. The boundary conditions (10.73) require

$$\varphi(a) = c_0 = 0, \quad \varphi(b) = c_{n-1} + h_{n-1}b_{n-1} = 0. \tag{10.75}$$

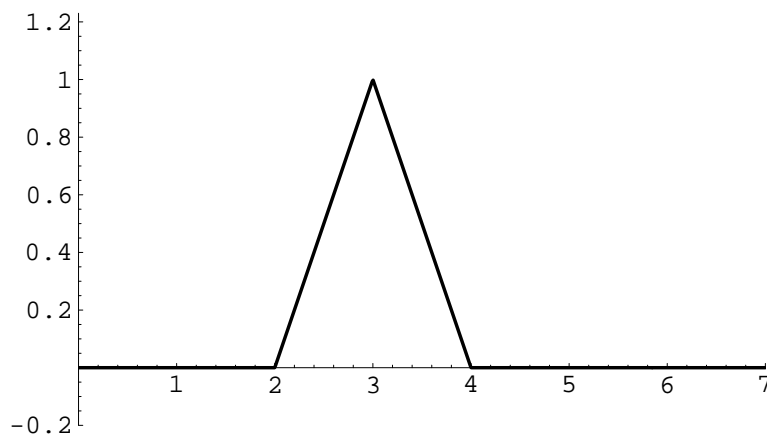


Figure 10.13. A Hat Function.

The function $\varphi(x)$ involves a total of $2n$ unspecified coefficients $c_0, \dots, c_{n-1}, b_0, \dots, b_{n-1}$. The continuity conditions (10.74) and the second boundary condition (10.75) uniquely determine the b_j . The first boundary condition specifies c_0 , while the remaining $n - 1$ coefficients $c_1 = \varphi(x_1), \dots, c_{n-1} = \varphi(x_{n-1})$ are arbitrary. We conclude that the finite element subspace W has dimension $n - 1$, which is the number of interior mesh points.

Remark: Every function $\varphi(x)$ in our subspace has piecewise constant first derivative $w'(x)$. However, the jump discontinuities in $\varphi'(x)$ imply that its second derivative $\varphi''(x)$ has a delta function impulse at each mesh point, and is therefore far from being a solution to the differential equation. Nevertheless, the finite element minimizer $\varphi_*(x)$ will, in practice, provide a reasonable approximation to the actual solution $u_*(x)$.

The most convenient basis for W consists of the *hat functions*, which are continuous, piecewise affine functions that interpolate the data

$$\varphi_j(x_k) = \begin{cases} 1, & j = k, \\ 0, & j \neq k, \end{cases} \quad \text{for } j = 1, \dots, n - 1, \quad k = 0, \dots, n.$$

The graph of a typical hat function appears in Figure 10.13. The explicit formula is easily established:

$$\varphi_j(x) = \begin{cases} \frac{x - x_{j-1}}{x_j - x_{j-1}}, & x_{j-1} \leq x \leq x_j, \\ \frac{x_{j+1} - x}{x_{j+1} - x_j}, & x_j \leq x \leq x_{j+1}, \\ 0, & x \leq x_{j-1} \text{ or } x \geq x_{j+1}, \end{cases} \quad j = 1, \dots, n - 1. \quad (10.76)$$

An advantage of using these basis elements is that the resulting coefficient matrix (10.69) turns out to be tridiagonal. Therefore, the tridiagonal Gaussian Elimination algorithm, [104], will rapidly produce the solution to the linear system (10.72). Since the accuracy of the finite element solution increases with the number of mesh points, this numerical scheme allows us to easily compute very accurate approximations to the solution to the boundary value problem.

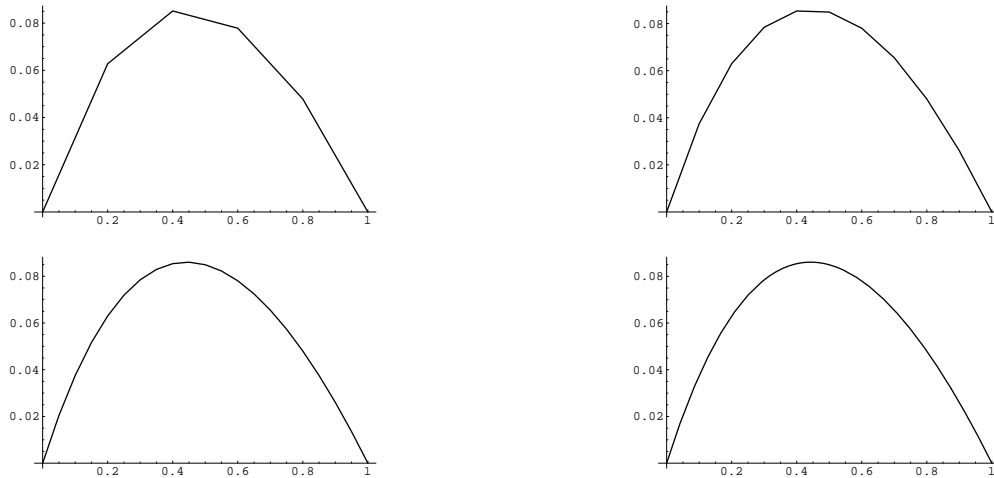


Figure 10.14. Finite Element Solution to (10.83).

where

$$s_j = \int_{x_j}^{x_{j+1}} c(x) dx \quad (10.79)$$

is the total stiffness of the j^{th} subinterval. The corresponding right hand side has entries

$$\begin{aligned} b_j &= \langle f; \varphi_j \rangle = \int_0^\ell f(x) \varphi_j(x) dx \\ &= \frac{1}{h} \left[\int_{x_{j-1}}^{x_j} (x - x_{j-1}) f(x) dx + \int_{x_j}^{x_{j+1}} (x_{j+1} - x) f(x) dx \right]. \end{aligned} \quad (10.80)$$

In this manner, we have assembled the basic ingredients for determining the finite element approximation to the solution.

In practice, we do not have to explicitly evaluate the integrals (10.79, 80), but may replace them by a suitably close numerical approximation. When the step size $h \ll 1$ is small, then the integrals are taken over small intervals, and we can use the trapezoid rule, [27, 124], to approximate them:

$$s_j \approx \frac{h}{2} [c(x_j) + c(x_{j+1})], \quad b_j \approx h f(x_j). \quad (10.81)$$

Observe that the j^{th} entry of the resulting finite element system $M\mathbf{c} = \mathbf{b}$ is, upon dividing by h , given by

$$-\frac{c_{j+1} - 2c_j + c_{j-1}}{h^2} = -\frac{u(x_{j+1}) - 2u(x_j) + u(x_{j-1}))}{h^2} = -f(x_j). \quad (10.82)$$

The left hand side coincides with the standard finite difference approximation (10.5) to minus the second derivative $-u''(x_j)$ at the mesh point x_j . As a result, for this particular case, the finite element and finite difference numerical solution methods happen to coincide.

Example 10.9. Consider the boundary value problem

$$-\frac{d}{dx}(x+1)\frac{du}{dx} = 1, \quad u(0) = 0, \quad u(1) = 0. \quad (10.83)$$

The explicit solution is easily found by direct integration:

$$u(x) = -x + \frac{\log(x+1)}{\log 2}. \quad (10.84)$$

It minimizes the associated quadratic functional

$$\mathcal{P}[u] = \int_0^\ell \left[\frac{1}{2}(x+1)u'(x)^2 - u(x) \right] dx \quad (10.85)$$

over all possible functions $u \in C^1$ that satisfy the given boundary conditions. The finite element system (10.72) has coefficient matrix given by (10.78) and right hand side (10.80), where

$$s_j = \int_{x_j}^{x_{j+1}} (1+x) dx = h(1+x_j) + \frac{1}{2}h^2 = h + h^2 \left(j + \frac{1}{2} \right), \quad b_j = \int_{x_j}^{x_{j+1}} 1 dx = h.$$

The resulting solution is plotted in Figure 10.14. The first three graphs contain, respectively, 5, 10, 20 mesh points, so that $h = .2, .1, .05$, while the last plots the exact solution (10.84). Even when computed on rather coarse meshes, the finite element approximation is quite respectable.

10.6. Finite Elements in Two Dimensions.

As the reader has no doubt already surmised, explicit solutions to boundary value problems for the Laplace and Poisson equations are few and far between. In most cases, exact solution formulae are not available, or are so complicated as to be of scant utility. To proceed further, one is forced to design suitable numerical approximation schemes that can accurately evaluate the desired solution.

An especially powerful class of numerical algorithms for solving elliptic boundary value problems are the finite element methods. We have already learned, in Section 10.5, the key underlying idea. One begins with a minimization principle, prescribed by a quadratic functional defined on a suitable vector space of functions U that serves to incorporate the (homogeneous) boundary conditions. The desired solution is characterized as the unique minimizer $u_\star \in U$. One then restricts the functional to a suitably chosen finite-dimensional subspace $W \subset U$, and seeks a minimizer $w_\star \in W$. Finite-dimensionality of W has the effect of reducing the infinite-dimensional minimization problem to a finite-dimensional problem, which can then be solved by numerical linear algebra. The resulting minimizer w_\star will — provided the subspace W has been cleverly chosen — provide a good approximation to the true minimizer u_\star on the entire domain. Here we concentrate on the practical design of the finite element procedure, and refer the reader to a more advanced text, e.g., [129], for the analytical details and proofs of convergence. Most of the multi-dimensional complications are not in the underlying theory, but rather in the realms of data management and organizational details.

In this section, we first concentrate on applying these ideas to the two-dimensional Poisson equation. For specificity, we concentrate on the homogeneous Dirichlet boundary value problem

$$-\Delta u = f \quad \text{in } \Omega \qquad u = 0 \quad \text{on } \partial\Omega. \quad (10.86)$$

According to Theorem 9.30, the solution $u = u_*$ is characterized as the unique minimizing function for the Dirichlet functional (9.79) among all smooth functions $u(x, y)$ that satisfy the prescribed boundary conditions. In the finite element approximation, we restrict the Dirichlet functional to a suitably chosen finite-dimensional subspace. As in the one-dimensional situation, the most convenient finite-dimensional subspaces consist of functions that may lack the requisite degree of smoothness that qualifies them as possible solutions to the partial differential equation. Nevertheless, they do provide good approximations to the actual solution. An important practical consideration, impacting the speed of the calculation, is to employ functions with small support, as in Definition 10.7. The resulting finite element matrix will then be sparse and the solution to the linear system can be relatively rapidly calculated, usually by application of an iterative numerical scheme such as the Gauss–Seidel or SOR methods discussed in [104].

Triangulation

For one-dimensional boundary value problems, the finite element construction rests on the introduction of a mesh $a = x_0 < x_1 < \cdots < x_n = b$ on the interval of definition. The mesh nodes x_k break the interval into a collection of small subintervals. In two-dimensional problems, a *mesh* consists of a finite number of points $\mathbf{x}_k = (x_k, y_k)$, $k = 1, \dots, m$, known as *nodes*, usually lying inside the domain $\Omega \subset \mathbb{R}^2$. As such, there is considerable freedom in the choice of mesh nodes, and completely uniform spacing is often not possible. We regard the nodes as forming the vertices of a *triangulation* of the domain Ω , consisting of a finite number of small triangles, which we denote by T_1, \dots, T_N . The nodes are split into two categories — *interior nodes* and *boundary nodes*, the latter lying on or close to the boundary of the domain. A curved boundary is approximated by the polygon through the boundary nodes formed by the sides of the triangles lying on the edge of the domain; see Figure 10.15 for a typical example. Thus, in computer implementations of the finite element method, the first module is a routine that will automatically triangulate a specified domain in some reasonable manner; see below for details on what “reasonable” entails.

As in our one-dimensional finite element construction, the functions $w(x, y)$ in the finite-dimensional subspace W will be continuous and *piecewise affine*. “Piecewise affine” means that, on each triangle, the graph of w is flat, and so has the formula[†]

$$w(x, y) = \alpha^\nu + \beta^\nu x + \gamma^\nu y, \quad \text{for } (x, y) \in T_\nu. \quad (10.87)$$

Continuity of w requires that its values on a common edge between two triangles must agree, and this will impose certain compatibility conditions on the coefficients $\alpha^\mu, \beta^\mu, \gamma^\mu$ and $\alpha^\nu, \beta^\nu, \gamma^\nu$ associated with adjacent pairs of triangles T_μ, T_ν . The graph of $z = w(x, y)$

[†] Here and subsequently, the index ν is a superscript, not a power!

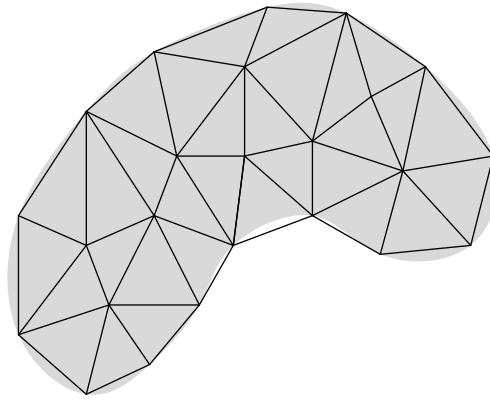


Figure 10.15. Triangulation of a Planar Domain.

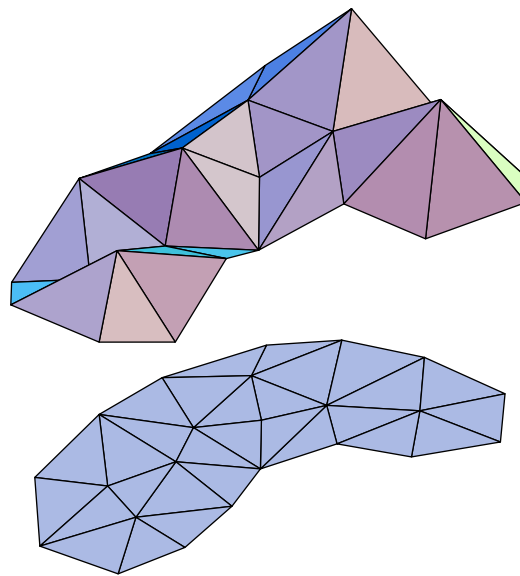


Figure 10.16. Piecewise Affine Function.

forms a connected polyhedral surface whose triangular faces lie above the triangles in the domain; see Figure 10.16 for an illustration.

The next step is to choose a basis of the subspace of piecewise affine functions for the given triangulation. As in the one-dimensional version, the most convenient basis consists of *pyramid functions* $\varphi_k(x, y)$ which assume the value 1 at a single node \mathbf{x}_k , and are zero at all the other nodes; thus

$$\varphi_k(x_i, y_i) = \begin{cases} 1, & i = k, \\ 0, & i \neq k. \end{cases} \quad (10.88)$$

Note that φ_k will be nonzero only on those triangles which have the node \mathbf{x}_k as one of their vertices, and hence the graph of φ_k looks like a pyramid of unit height sitting on a flat plane, as illustrated in Figure 10.17.

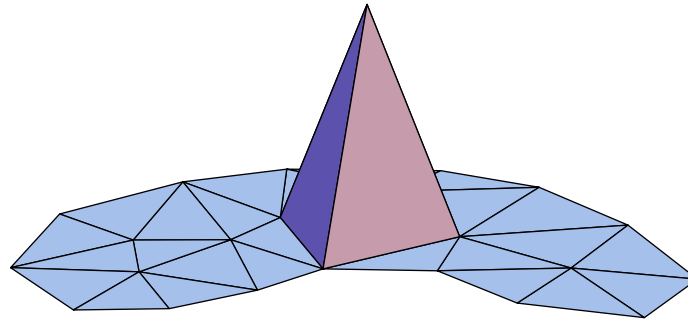


Figure 10.17. Finite Element Pyramid Function.

The pyramid functions $\varphi_k(x, y)$ corresponding to the *interior nodes* \mathbf{x}_k automatically satisfy the homogeneous Dirichlet boundary conditions on the boundary of the domain — or, more correctly, on the polygonal boundary of the triangulated domain, which is supposed to be a good approximation to the curved boundary of the original domain Ω . Thus, the finite-dimensional finite element subspace W is the span of the interior node pyramid functions, and so a general element $w \in W$ is a linear combination thereof:

$$w(x, y) = \sum_{k=1}^n c_k \varphi_k(x, y), \quad (10.89)$$

where the sum ranges over the n interior nodes of the triangulation. Owing to the original specification (10.88) of the pyramid functions, the coefficients

$$c_k = w(x_k, y_k) \approx u(x_k, y_k), \quad k = 1, \dots, n, \quad (10.90)$$

are the *same* as the values of the finite element approximation $w(x, y)$ at the interior nodes. This immediately implies linear independence of the pyramid functions, since the only linear combination that vanishes at all nodes is the trivial one $c_1 = \dots = c_n = 0$. Thus, the interior node pyramid functions $\varphi_1, \dots, \varphi_n$ form a basis for the finite element subspace W , which therefore has dimension equal to n , the number of interior nodes.

Determining the explicit formulae for the finite element basis functions is not difficult. On one of the triangles T_ν that has \mathbf{x}_k as a vertex, $\varphi_k(x, y)$ will be the unique affine function (10.87) that takes the value 1 at the vertex \mathbf{x}_k and 0 at its other two vertices \mathbf{x}_l and \mathbf{x}_m . Thus, we are in need of a formula for an affine function or *element*

$$\omega_k^\nu(x, y) = \alpha_k^\nu + \beta_k^\nu x + \gamma_k^\nu y, \quad (x, y) \in T_\nu, \quad (10.91)$$

that takes the prescribed values

$$\omega_k^\nu(x_k, y_k) = 1, \quad \omega_k^\nu(x_l, y_l) = \omega_k^\nu(x_m, y_m) = 0,$$

at three distinct points. These three conditions lead to the linear system

$$\begin{aligned} \omega_k^\nu(x_k, y_k) &= \alpha_k^\nu + \beta_k^\nu x_k + \gamma_k^\nu y_k = 1, \\ \omega_k^\nu(x_l, y_l) &= \alpha_k^\nu + \beta_k^\nu x_l + \gamma_k^\nu y_l = 0, \\ \omega_k^\nu(x_m, y_m) &= \alpha_k^\nu + \beta_k^\nu x_m + \gamma_k^\nu y_m = 0. \end{aligned} \quad (10.92)$$

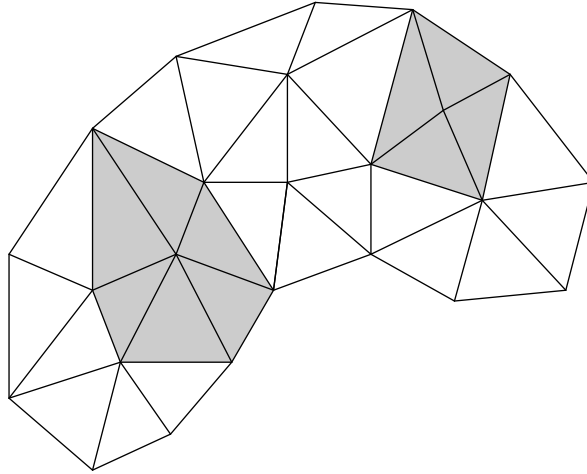


Figure 10.18. Vertex Polygons.

The solution produces the explicit formulae

$$\alpha_k^\nu = \frac{x_l y_m - x_m y_l}{\Delta_\nu}, \quad \beta_k^\nu = \frac{y_l - y_m}{\Delta_\nu}, \quad \gamma_k^\nu = \frac{x_m - x_l}{\Delta_\nu}, \quad (10.93)$$

for the coefficients; the denominator

$$\Delta_\nu = \det \begin{pmatrix} 1 & x_k & y_k \\ 1 & x_l & y_l \\ 1 & x_m & y_m \end{pmatrix} = \pm 2 \text{ area } T_\nu \quad (10.94)$$

is, up to sign, twice the area of the triangle T_ν ; see Exercise ■.

Example 10.10. Consider an isosceles right triangle T with vertices

$$\mathbf{x}_1 = (0, 0), \quad \mathbf{x}_2 = (1, 0), \quad \mathbf{x}_3 = (0, 1).$$

Using (10.93–94) (or solving the linear system (10.92) directly), we immediately produce the three affine elements

$$\omega_1(x, y) = 1 - x - y, \quad \omega_2(x, y) = x, \quad \omega_3(x, y) = y. \quad (10.95)$$

As required, each ω_k equals 1 at the vertex \mathbf{x}_k and is zero at the other two vertices.

The finite element pyramid function is then obtained by piecing together the individual affine elements, whence

$$\varphi_k(x, y) = \begin{cases} \omega_k^\nu(x, y), & \text{if } (x, y) \in T_\nu \text{ which has } \mathbf{x}_k \text{ as a vertex,} \\ 0, & \text{otherwise.} \end{cases} \quad (10.96)$$

Continuity of $\varphi_k(x, y)$ is assured since the constituent affine elements have the same values at common vertices. The support of the pyramid function (10.96) is the polygon

$$\text{supp } \varphi_k = P_k = \bigcup_{\nu} T_\nu \quad (10.97)$$

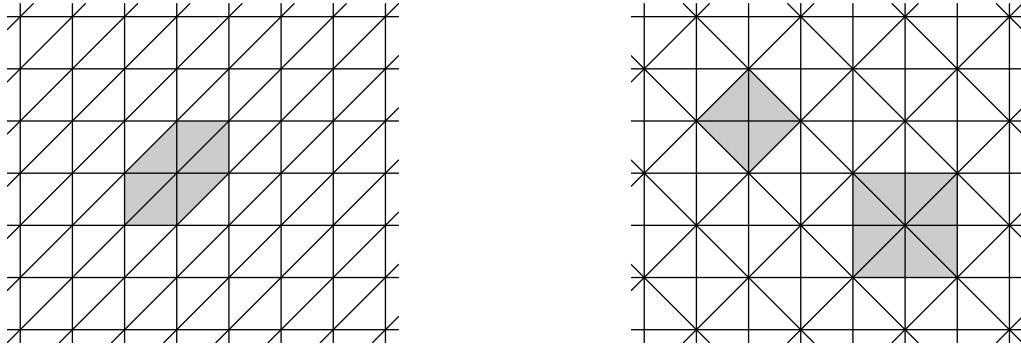


Figure 10.19. Square Mesh Triangulations.

consisting of all the triangles T_ν that have the node \mathbf{x}_k as a vertex. In other words, $\varphi_k(x, y) = 0$ whenever $(x, y) \notin P_k$. We will call P_k the k^{th} *vertex polygon*. The node \mathbf{x}_k lies on the interior of its vertex polygon P_k , while the vertices of P_k are all those that are connected to \mathbf{x}_k by a single edge of the triangulation. In Figure 10.18, the shaded regions indicate two of the vertex polygons for the triangulation in Figure 10.15.

Example 10.11. The simplest, and most common triangulations are based on regular meshes. Suppose that the nodes lie on a square grid, and so are of the form $\mathbf{x}_{i,j} = (ih + a, jh + b)$ where $h > 0$ is the inter-node spacing, and (a, b) represents an overall offset. If we choose the triangles to all have the same orientation, as in the first picture in Figure 10.19, then the vertex polygons all have the same shape, consisting of 6 triangles of total area $3h^2$ — the shaded region. On the other hand, if we choose an alternating, perhaps more aesthetically pleasing triangulation as in the second picture, then there are two types of vertex polygons. The first, consisting of four triangles, has area $2h^2$, while the second, containing 8 triangles, has twice the area, $4h^2$. In practice, there are good reasons to prefer the former triangulation.

In general, in order to ensure convergence of the finite element solution to the true minimizer, one should choose a triangulation with the following properties:

- (a) The triangles are not too long and skinny. In other words, their sides should have comparable lengths. In particular, obtuse triangles should be avoided.
- (b) The areas of nearby triangles T_ν should not vary too much.
- (c) The areas of nearby vertex polygons P_k should also not vary too much.

For adaptive or variable meshes, one might very well have wide variations in area over the entire grid, with small triangles in regions of rapid change in the solution, and large ones in less interesting regions. But, overall, the sizes of the triangles and vertex polygons should not dramatically vary as one moves across the domain.

The Finite Element Equations

We now seek to approximate the solution to the homogeneous Dirichlet boundary value problem by restricting the Dirichlet functional to the selected finite element subspace W .

Substituting the formula (10.89) for a general element of W into the quadratic Dirichlet functional (9.79) and expanding, we find

$$\begin{aligned} \mathcal{P}[w] &= \mathcal{P} \left[\sum_{i=1}^n c_i \varphi_i \right] = \iint_{\Omega} \left[\left(\sum_{i=1}^n c_i \nabla \varphi_i \right)^2 - f(x, y) \left(\sum_{i=1}^n c_i \varphi_i \right) \right] dx dy \\ &= \frac{1}{2} \sum_{i,j=1}^n k_{ij} c_i c_j - \sum_{i=1}^n b_i c_i = \frac{1}{2} \mathbf{c}^T K \mathbf{c} - \mathbf{b}^T \mathbf{c}. \end{aligned}$$

Here, $K = (k_{ij})$ is the symmetric $n \times n$ matrix, while $\mathbf{b} = (b_1, b_2, \dots, b_n)^T$ is the vector that have the respective entries

$$\begin{aligned} k_{ij} &= \langle \nabla \varphi_i; \nabla \varphi_j \rangle = \iint_{\Omega} \nabla \varphi_i \cdot \nabla \varphi_j dx dy, \\ b_i &= \langle f; \varphi_i \rangle = \iint_{\Omega} f \varphi_i dx dy. \end{aligned} \tag{10.98}$$

Thus, to determine the finite element approximation, we need to minimize the quadratic function

$$P(\mathbf{c}) = \frac{1}{2} \mathbf{c}^T K \mathbf{c} - \mathbf{b}^T \mathbf{c} \tag{10.99}$$

over all possible choices of coefficients $\mathbf{c} = (c_1, c_2, \dots, c_n)^T \in \mathbb{R}^n$, i.e., over all possible function values at the interior nodes. Restricting to the finite element subspace has reduced us to a standard finite-dimensional quadratic minimization problem. First, the coefficient matrix $K > 0$ is positive definite due to the positive definiteness of the original functional; the proof in Section 10.5 is easily adapted to the present situation. Example 9.24 tells us that the minimizer is obtained by solving the associated linear system

$$K \mathbf{c} = \mathbf{b}. \tag{10.100}$$

The solution to (10.100) can be effected by either Gaussian elimination or an iterative technique.

To find explicit formulae for the matrix coefficients k_{ij} in (10.98), we begin by noting that the gradient of the affine element (10.91) is equal to

$$\nabla \omega_k^\nu(x, y) = \mathbf{a}_k^\nu = \begin{pmatrix} \beta_k^\nu \\ \gamma_k^\nu \end{pmatrix} = \frac{1}{\Delta_\nu} \begin{pmatrix} y_i - y_j \\ x_j - x_i \end{pmatrix}, \quad (x, y) \in T_\nu, \tag{10.101}$$

which is a constant vector inside the triangle T_ν , while outside $\nabla \omega_k^\nu = \mathbf{0}$. Therefore,

$$\nabla \varphi_k(x, y) = \begin{cases} \nabla \omega_k^\nu = \mathbf{a}_k^\nu, & \text{if } (x, y) \in T_\nu \text{ which has } \mathbf{x}_k \text{ as a vertex,} \\ \mathbf{0}, & \text{otherwise,} \end{cases} \tag{10.102}$$

reduces to a piecewise constant function on the triangulation. Actually, (10.102) is not quite correct since if (x, y) lies on the boundary of a triangle T_ν , then the gradient does not exist. However, this technicality will not cause any difficulty in evaluating the ensuing integral.



Figure 10.20. Right and Equilateral Triangles.

We will approximate integrals over the domain Ω by integrals over the triangles, which relies on our assumption that the polygonal boundary of the triangulation is a reasonably close approximation to the true boundary $\partial\Omega$. In particular,

$$k_{ij} \approx \sum_{\nu} \iint_{T_{\nu}} \nabla\varphi_i \cdot \nabla\varphi_j \, dx \, dy \equiv \sum_{\nu} k_{ij}^{\nu}. \quad (10.103)$$

Now, according to (10.102), one or the other gradient in the integrand will vanish on the entire triangle T_{ν} unless both \mathbf{x}_i and \mathbf{x}_j are vertices. Therefore, the only terms contributing to the sum are those triangles T_{ν} that have both \mathbf{x}_i and \mathbf{x}_j as vertices. If $i \neq j$ there are only two such triangles, while if $i = j$ every triangle in the i^{th} vertex polygon P_i contributes. The individual summands are easily evaluated, since the gradients are constant on the triangles, and so, by (10.102),

$$k_{ij}^{\nu} = \iint_{T_{\nu}} \mathbf{a}_i^{\nu} \cdot \mathbf{a}_j^{\nu} \, dx \, dy = \mathbf{a}_i^{\nu} \cdot \mathbf{a}_j^{\nu} \text{ area } T_{\nu} = \frac{1}{2} \mathbf{a}_i^{\nu} \cdot \mathbf{a}_j^{\nu} |\Delta_{\nu}|.$$

Let T_{ν} have vertices $\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k$. Then, by (10.101, 102, 94),

$$\begin{aligned} k_{ij}^{\nu} &= \frac{1}{2} \frac{(y_j - y_k)(y_k - y_i) + (x_k - x_j)(x_i - x_k)}{(\Delta_{\nu})^2} |\Delta_{\nu}| = -\frac{(\mathbf{x}_i - \mathbf{x}_k) \cdot (\mathbf{x}_j - \mathbf{x}_k)}{2 |\Delta_{\nu}|}, \quad i \neq j, \\ k_{ii}^{\nu} &= \frac{1}{2} \frac{(y_j - y_k)^2 + (x_k - x_j)^2}{(\Delta_{\nu})^2} |\Delta_{\nu}| = \frac{\|\mathbf{x}_j - \mathbf{x}_k\|^2}{2 |\Delta_{\nu}|}. \end{aligned} \quad (10.104)$$

In this manner, each triangle T_{ν} specifies a collection of 6 different coefficients, $k_{ij}^{\nu} = k_{ji}^{\nu}$, indexed by its vertices, and known as the *elemental stiffnesses* of T_{ν} . Interestingly, the elemental stiffnesses depend only on the three vertex *angles* in the triangle and not on its size. Thus, similar triangles have the *same* elemental stiffnesses. Indeed, if $\theta_i^{\nu}, \theta_j^{\nu}, \theta_k^{\nu}$ denote the angles in T_{ν} at the respective vertices $\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k$, then, according to Exercise ■,

$$k_{ii}^{\nu} = \frac{1}{2} (\cot \theta_k^{\nu} + \cot \theta_j^{\nu}), \quad \text{while} \quad k_{ij}^{\nu} = k_{ji}^{\nu} = -\frac{1}{2} \cot \theta_k^{\nu}, \quad i \neq j. \quad (10.105)$$

Example 10.12. The right triangle with vertices $\mathbf{x}_1 = (0, 0)$, $\mathbf{x}_2 = (1, 0)$, $\mathbf{x}_3 = (0, 1)$ has elemental stiffnesses

$$k_{11} = 1, \quad k_{22} = k_{33} = \frac{1}{2}, \quad k_{12} = k_{21} = k_{13} = k_{31} = -\frac{1}{2}, \quad k_{23} = k_{32} = 0. \quad (10.106)$$

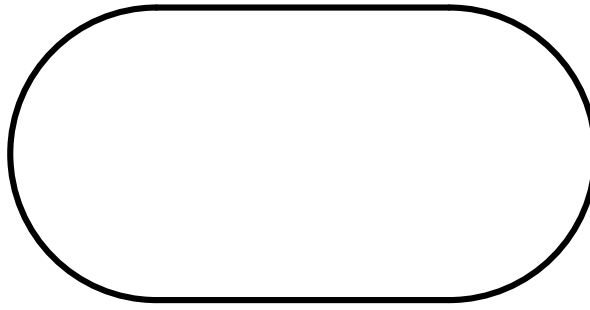


Figure 10.21. The Oval Plate.

The same holds for any other isosceles right triangle, as long as we chose the first vertex to be at the right angle. Similarly, an equilateral triangle has all 60° angles, and so its elemental stiffnesses are

$$\begin{aligned} k_{11} = k_{22} = k_{33} &= \frac{1}{\sqrt{3}} \approx .577350, \\ k_{12} = k_{21} = k_{13} = k_{31} = k_{23} = k_{32} &= -\frac{1}{2\sqrt{3}} \approx -.288675. \end{aligned} \quad (10.107)$$

Assembling the Elements

The elemental stiffnesses of each triangle will contribute, through the summation (10.103), to the finite element coefficient matrix K . We begin by constructing a larger matrix K^* , which we call the *full finite element matrix*, of size $m \times m$ where m is the total number of nodes in our triangulation, including both interior and boundary nodes. The rows and columns of K^* are labeled by the nodes \mathbf{x}_i . Let $K_\nu = (k_{ij}^\nu)$ be the corresponding $m \times m$ matrix containing the elemental stiffnesses k_{ij}^ν of T_ν in the rows and columns indexed by its vertices, and all other entries equal to 0. Thus, K_ν will have (at most) 9 nonzero entries. The resulting $m \times m$ matrices are all summed together over all the triangles,

$$K^* = \sum_{\nu=1}^N K_\nu, \quad (10.108)$$

to produce the full finite element matrix, in accordance with (10.103).

The full finite element matrix K^* is too large, since its rows and columns include all the nodes, whereas the finite element matrix K appearing in (10.100) only refers to the n interior nodes. The *reduced $n \times n$ finite element matrix* K is simply obtained from K^* by deleting all rows and columns indexed by boundary nodes, retaining only the elements k_{ij} when both \mathbf{x}_i and \mathbf{x}_j are interior nodes. For the homogeneous boundary value problem, this is all we require. As we shall see, inhomogeneous boundary conditions are most easily handled by retaining (part of) the full matrix K^* .

The easiest way to digest the construction is by working through a particular example.

Example 10.13. A metal plate has the shape of an oval running track, consisting of a rectangle, with side lengths 1 m by 2 m, and two semicircular disks glued onto its shorter ends, as sketched in Figure 10.21. The plate is subject to a heat source while its

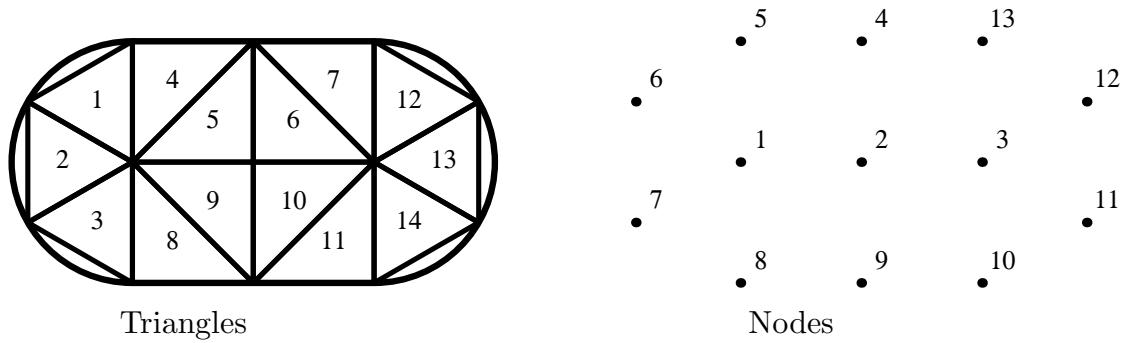


Figure 10.22. A Coarse Triangulation of the Oval Plate.

edges are held at a fixed temperature. The problem is to find the equilibrium temperature distribution within the plate. Mathematically, we must solve the Poisson equation with Dirichlet boundary conditions, for the equilibrium temperature $u(x, y)$.

Let us describe how to set up the finite element approximation to such a boundary value problem. We begin with a very coarse triangulation of the plate, which will not give particularly accurate results, but does serve to illustrate how to go about assembling the finite element matrix. We divide the rectangular part of the plate into 8 right triangles, while each semicircular end will be approximated by three equilateral triangles. The triangles are numbered from 1 to 14 as indicated in Figure 10.22. There are 13 nodes in all, numbered as in the second figure. Only nodes 1, 2, 3 are interior, while the boundary nodes are labeled 4 through 13, going counterclockwise around the boundary starting at the top. The full finite element matrix K^* will have size 13×13 , its rows and columns labeled by all the nodes, while the reduced matrix K appearing in the finite element equations (10.100) consists of the upper left 3×3 submatrix of K^* corresponding to the three interior nodes.

Each triangle T_ν will contribute the summand K_ν , whose values are its elemental stiffnesses, as indexed by its vertices. For example, the first triangle T_1 is equilateral, and so has elemental stiffnesses (10.107). Its vertices are labeled 1, 5, and 6, and therefore we place the stiffnesses (10.107) in the rows and columns numbered 1, 5, 6 to form the summand

$$K_1 = \begin{pmatrix} .577350 & 0 & 0 & 0 & -.288675 & -.288675 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ -.288675 & 0 & 0 & 0 & .577350 & -.288675 & 0 & 0 & \dots \\ -.288675 & 0 & 0 & 0 & -.288675 & .577350 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix},$$

where all the undisplayed entries in the full 13×13 matrix are 0. The next triangle T_2 has the same equilateral elemental stiffness matrix (10.107), but now its vertices are 1, 6, 7,

and so it will contribute

$$K_2 = \begin{pmatrix} .577350 & 0 & 0 & 0 & 0 & -0.288675 & -0.288675 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ -0.288675 & 0 & 0 & 0 & 0 & .577350 & -0.288675 & 0 & \dots \\ -0.288675 & 0 & 0 & 0 & 0 & -0.288675 & .577350 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}.$$

Similarly for K_3 , with vertices 1, 7, 8. On the other hand, triangle T_4 is an isosceles right triangle, and so has elemental stiffnesses (10.106). Its vertices are labeled 1, 4, and 5, with vertex 5 at the right angle. Therefore, its contribution is

$$K_4 = \begin{pmatrix} .5 & 0 & 0 & 0 & -.5 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & .5 & -.5 & 0 & 0 & 0 & \dots \\ -.5 & 0 & 0 & -.5 & 1.0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}.$$

Continuing in this manner, we assemble 14 contributions K_1, \dots, K_{14} , each with (at most) 9 nonzero entries. The full finite element matrix is the sum

$$K^* = K_1 + K_2 + \dots + K_{14} = \begin{pmatrix} 3.732 & -1 & 0 & 0 & -.7887 & -.5774 & -.5774 \\ -1 & 4 & -1 & -1 & 0 & 0 & 0 \\ 0 & -1 & 3.732 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 2 & -.5 & 0 & 0 \\ -.7887 & 0 & 0 & -.5 & 1.577 & -.2887 & 0 \\ -.5774 & 0 & 0 & 0 & -.2887 & 1.155 & -.2887 \\ -.5774 & 0 & 0 & 0 & 0 & -.2887 & 1.155 \\ -.7887 & 0 & 0 & 0 & 0 & 0 & -.2887 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -.7887 & 0 & 0 & 0 & 0 \\ 0 & 0 & -.5774 & 0 & 0 & 0 & 0 \\ 0 & 0 & -.5774 & 0 & 0 & 0 & 0 \\ 0 & 0 & -.7887 & -.5 & 0 & 0 & 0 \end{pmatrix} \quad (10.109)$$

$$\begin{pmatrix} -.7887 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -.7887 & -.5774 & -.5774 & -.7887 \\ 0 & 0 & 0 & 0 & 0 & -.5 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -.2887 & 0 & 0 & 0 & 0 & 0 \\ 1.577 & -.5 & 0 & 0 & 0 & 0 \\ -.5 & 2 & -.5 & 0 & 0 & 0 \\ 0 & -.5 & 1.577 & -.2887 & 0 & 0 \\ 0 & 0 & -.2887 & 1.155 & -.2887 & 0 \\ 0 & 0 & 0 & -.2887 & 1.155 & -.2887 \\ 0 & 0 & 0 & 0 & -.2887 & 1.577 \end{pmatrix}.$$

Since only nodes 1, 2, 3 are interior nodes, the reduced finite element matrix only uses the upper left 3×3 block of K^* , so

$$K = \begin{pmatrix} 3.732 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 3.732 \end{pmatrix}. \quad (10.110)$$

It is not difficult to directly construct K , bypassing K^* entirely.

For a finer triangulation, the construction is similar, but the matrices become much larger. The procedure can, of course, be automated. Fortunately, if we choose a very regular triangulation, then we do not need to be nearly as meticulous in assembling the stiffness matrices, since many of the entries are the same. The simplest case is when we use a uniform square mesh, and so triangulate the domain into isosceles right triangles. This is accomplished by laying out a relatively dense square grid over the domain $\Omega \subset \mathbb{R}^2$. The interior nodes are the grid points that fall inside the oval domain, while the boundary nodes are all those grid points lying adjacent to one or more of the interior nodes, and are near but not necessarily precisely on the boundary $\partial\Omega$. Figure 10.23 shows the nodes in a square grid with intermesh spacing $h = .2$. While a bit crude in its approximation of the boundary of the domain, this procedure does have the advantage of making the construction of the associated finite element matrix relatively painless.

For such a mesh, all the triangles are isosceles right triangles, with elemental stiffnesses (10.106). Summing the corresponding matrices K_ν over all the triangles, as in (10.108), the rows and columns of K^* corresponding to the interior nodes are seen to all have the same form. Namely, if i labels an interior node, then the corresponding diagonal entry is $k_{ii} = 4$, while the off-diagonal entries $k_{ij} = k_{ji}$, $i \neq j$, are equal to either -1 when node i is adjacent to node j on the grid, and is equal to 0 in all other cases. Node j is allowed to be a boundary node. (Interestingly, the result does not depend on how one orients the pair of triangles making up each square of the grid, which only plays a role in the computation of the right hand side of the finite element equation.) Observe that the same computation applies even to our coarse triangulation. The interior node 2 belongs to all right isosceles triangles, and the corresponding entries in (10.109) are $k_{22} = 4$, and $k_{2j} = -1$ for the four adjacent nodes $j = 1, 3, 4, 9$.

Remark: Interestingly, the coefficient matrix arising from the finite element method on a square (or even rectangular) grid is the same as the coefficient matrix arising from a

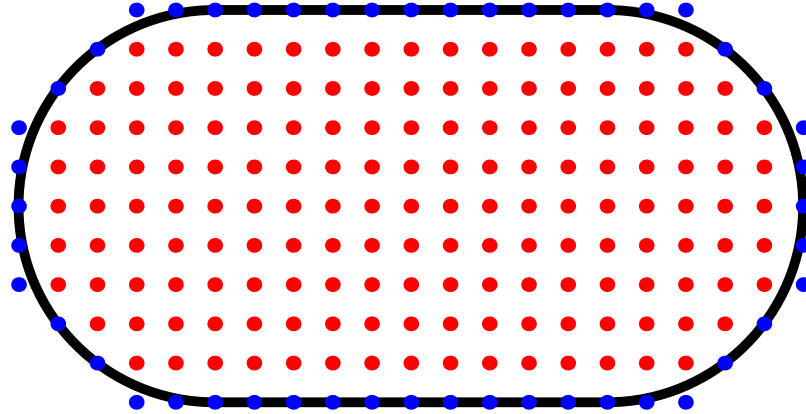


Figure 10.23. A Square Mesh for the Oval Plate.

finite difference solution to the Laplace or Poisson equation, as described in Exercise ■. The finite element approach has the advantage of applying to much more general triangulations.

In general, while the finite element matrix K for a two-dimensional boundary value problem is not as nice as the tridiagonal matrices we obtained in our one-dimensional problems, it is still very sparse and, on regular grids, highly structured. This makes solution of the resulting linear system particularly amenable to an iterative matrix solver such as Gauss–Seidel, Jacobi, or, for even faster convergence, successive over-relaxation (SOR).

The Coefficient Vector and the Boundary Conditions

So far, we have been concentrating on assembling the finite element coefficient matrix K . We also need to compute the forcing vector $\mathbf{b} = (b_1, b_2, \dots, b_n)^T$ appearing on the right hand side of the fundamental linear equation (10.100). According to (10.98), the entries b_i are found by integrating the product of the forcing function and the finite element basis function. As before, we will approximate the integral over the domain Ω by an integral over the triangles, and so

$$b_i = \iint_{\Omega} f \varphi_i \, dx \, dy \approx \sum_{\nu} \iint_{T_{\nu}} f \omega_i^{\nu} \, dx \, dy \equiv \sum_{\nu} b_i^{\nu}. \quad (10.111)$$

Typically, the exact computation of the various triangular integrals is not convenient, and so we resort to a numerical approximation. Since we are assuming that the individual triangles are small, we can adopt a very crude numerical integration scheme. If the function $f(x, y)$ does not vary much over the triangle T_{ν} — which will certainly be the case if T_{ν} is sufficiently small — we may approximate $f(x, y) \approx c_i^{\nu}$ for $(x, y) \in T_{\nu}$ by a constant. The integral (10.111) is then approximated by

$$b_i^{\nu} = \iint_{T_{\nu}} f \omega_i^{\nu} \, dx \, dy \approx c_i^{\nu} \iint_{T_{\nu}} \omega_i^{\nu}(x, y) \, dx \, dy = \frac{1}{3} c_i^{\nu} \text{area } T_{\nu} = \frac{1}{6} c_i^{\nu} |\Delta_{\nu}|. \quad (10.112)$$

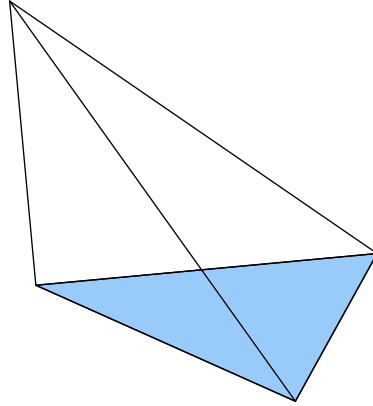


Figure 10.24. Finite Element Tetrahedron.

The formula for the integral of the affine element $\omega_i^\nu(x, y)$ follows from solid geometry. Indeed, it equals the volume under its graph, a tetrahedron of height 1 and base T_ν , as illustrated in Figure 10.24.

How to choose the constant c_i^ν ? In practice, the simplest choice is to let $c_i^\nu = f(x_i, y_i)$ be the value of the function at the i^{th} vertex. With this choice, the sum in (10.111) becomes

$$b_i \approx \sum_\nu \frac{1}{3} f(x_i, y_i) \text{ area } T_\nu = \frac{1}{3} f(x_i, y_i) \text{ area } P_i, \quad (10.113)$$

where P_i is the vertex polygon (10.97) corresponding to the node \mathbf{x}_i . In particular, for the square mesh with the uniform choice of triangles, as in Example 10.11,

$$\text{area } P_i = 3h^2 \quad \text{for all } i, \text{ and so} \quad b_i \approx f(x_i, y_i) h^2 \quad (10.114)$$

is well approximated by just h^2 times the value of the forcing function at the node. This is the underlying reason to choose the uniform triangulation for the square mesh; the alternating version would give unequal values for the b_i over adjacent nodes, and this would introduce unnecessary errors into the final approximation.

Example 10.14. For the coarsely triangulated oval plate, the reduced stiffness matrix is (10.110). The Poisson equation

$$-\Delta u = 4$$

models a constant external heat source of magnitude 4° over the entire plate. If we keep the edges of the plate fixed at 0° , then we need to solve the finite element equation $K\mathbf{c} = \mathbf{b}$, where K is the coefficient matrix (10.110), while

$$\mathbf{b} = \frac{4}{3} \left(2 + \frac{3\sqrt{3}}{4}, 2, 2 + \frac{3\sqrt{3}}{4} \right)^T = (4.39872, 2.66667, 4.39872)^T.$$

The entries of \mathbf{b} are, by (10.113), equal to $4 = f(x_i, y_i)$ times one third the area of the corresponding vertex polygon, which for node 2 is the square consisting of 4 right triangles, each of area $\frac{1}{2}$, whereas for nodes 1 and 3 it consists of 4 right triangles of area $\frac{1}{2}$ plus three equilateral triangles, each of area $\frac{\sqrt{3}}{4}$; see Figure 10.22.

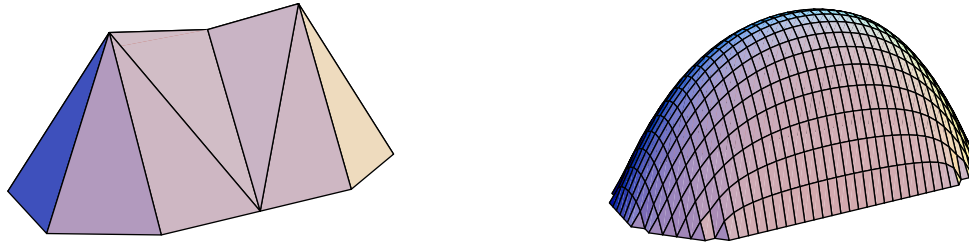


Figure 10.25. Finite Element Solutions to Poisson's Equation for an Oval Plate.

The solution to the final linear system is easily found:

$$\mathbf{c} = (1.56724, 1.45028, 1.56724)^T.$$

Its entries are the values of the finite element approximation at the three interior nodes. The finite element solution is plotted in the first illustration in Figure 10.25. A more accurate solution, based on a square grid triangulation of size $h = .1$ is plotted in the second figure.

Inhomogeneous Boundary Conditions

So far, we have restricted our attention to problems with homogeneous Dirichlet boundary conditions. According to Theorem 9.31, the solution to the inhomogeneous Dirichlet problem

$$-\Delta u = f \quad \text{in } \Omega, \quad u = h \quad \text{on } \partial\Omega,$$

is also obtained by minimizing the Dirichlet functional (9.79). However, now the minimization takes place over the affine subspace consisting of all functions that satisfy the inhomogeneous boundary conditions. It is not difficult to fit this problem into the finite element scheme.

The elements corresponding to the interior nodes of our triangulation remain as before, but now we need to include additional elements to ensure that our approximation satisfies the boundary conditions. Note that if \mathbf{x}_k is a boundary node, then the corresponding *boundary element* $\varphi_k(x, y)$ satisfies the interpolation condition (10.88), and so has the same piecewise affine form (10.96). The corresponding finite element approximation

$$w(x, y) = \sum_{i=1}^m c_i \varphi_i(x, y), \tag{10.115}$$

has the same form as before, (10.89), but now the sum is over all nodes, both interior and boundary. As before, the coefficients $c_i = w(x_i, y_i) \approx u(x_i, y_i)$ are the values of the finite element approximation at the nodes. Therefore, in order to satisfy the boundary conditions, we require

$$c_j = h(x_j, y_j) \quad \text{whenever} \quad \mathbf{x}_j = (x_j, y_j) \quad \text{is a boundary node.} \tag{10.116}$$

Remark: If the boundary node \mathbf{x}_j does not lie precisely on the boundary $\partial\Omega$, we need to approximate the value $h(x_j, y_j)$ appropriately, e.g., by using the value of $h(x, y)$ at the nearest boundary point $(x, y) \in \partial\Omega$.



Figure 10.26. Solution to the Dirichlet Problem for the Oval Plate.

The derivation of the finite element equations proceeds as before, but now there are additional terms arising from the nonzero boundary values. Leaving the intervening details to the reader, the final outcome can be written as follows. Let K^* denote the full $m \times m$ finite element matrix constructed as above. The reduced coefficient matrix K is obtained by retaining the rows and columns corresponding to only interior nodes, and so will have size $n \times n$, where n is the number of interior nodes. The *boundary coefficient matrix* \tilde{K} is the $n \times (m - n)$ matrix consisting of the entries of the interior rows that do not appear in K , i.e., those lying in the columns indexed by the boundary nodes. For instance, in the the coarse triangulation of the oval plate, the full finite element matrix is given in (10.109), and the upper 3×3 subblock is the reduced matrix (10.110). The remaining entries of the first three rows form the boundary coefficient matrix

$$\tilde{K} = \begin{pmatrix} 0 & -.7887 & -.5774 & -.5774 & -.7887 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -.7887 & -.5774 & -.5774 & -.7887 \end{pmatrix}. \quad (10.117)$$

We similarly split the coefficients c_i of the finite element function (10.115) into two groups. We let $\mathbf{c} \in \mathbb{R}^n$ denote the as yet unknown coefficients c_i corresponding to the values of the approximation at the interior nodes \mathbf{x}_i , while $\mathbf{h} \in \mathbb{R}^{m-n}$ will be the vector of boundary values (10.116). The solution to the finite element approximation (10.115) is obtained by solving the associated linear system

$$K\mathbf{c} + \tilde{K}\mathbf{h} = \mathbf{b}, \quad \text{or} \quad K\mathbf{c} = \mathbf{f} = \mathbf{b} - \tilde{K}\mathbf{h}. \quad (10.118)$$

Example 10.15. For the oval plate discussed in Example 10.13, suppose the right hand semicircular edge is held at 10° , the left hand semicircular edge at -10° , while the two straight edges have a linearly varying temperature distribution ranging from -10° at the left to 10° at the right, as illustrated in Figure 10.26. Our task is to compute its equilibrium temperature, assuming no internal heat source. Thus, for the coarse triangulation we have the boundary nodes values

$$\mathbf{h} = (h_4, \dots, h_{13})^T = (0, -10, -10, -10, -10, 0, 10, 10, 10, 10)^T.$$

Using the previously computed formulae (10.110, 117) for the interior coefficient matrix K and boundary coefficient matrix \tilde{K} , we approximate the solution to the Laplace equation

by solving (10.118). We are assuming that there is no external forcing function, $f(x, y) \equiv 0$, and so the right hand side is $\mathbf{b} = \mathbf{0}$, and so we must solve $K\mathbf{c} = \mathbf{f} = -\tilde{K}\mathbf{h} = (2.18564, 3.6, 7.64974)^T$. The finite element function corresponding to the solution $\mathbf{c} = (1.06795, 1.8, 2.53205)^T$ is plotted in the first illustration in Figure 10.26. Even on such a coarse mesh, the approximation is not too bad, as evidenced by the second illustration, which plots the finite element solution for a square mesh with spacing $h = .2$ between nodes.