Chapter 11

Partial Differential Equations

A wide variety of partial differential equations occurs in technical computing. We cannot begin to cover them all in this book. In this chapter, we limit ourselves to three model problems for second-order partial differential equations in one or two space dimensions.

11.1 Model Problems

All the problems we consider involve the Laplacian operator, which is

$$\triangle = \frac{\partial^2}{\partial x^2}$$

in one space dimension and

$$\triangle = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

in two space dimensions. We let $\vec{x}$ denote the single variable x in one dimension and the pair of variables (x, y) in two dimensions.

The first model problem is the Poisson equation. This *elliptic* equation does not involve a time variable, and so describes the steady state, quiescent behavior of a model variable:

$$\triangle u = f(\vec{x}).$$

There are no initial conditions.

The second model problem is the heat equation. This *parabolic* equation occurs in models involving diffusion and decay:

$$\frac{\partial u}{\partial t} = \triangle u - f(\vec{x}).$$

The initial condition is

$$u(\vec{x}, 0) = u_0(\vec{x}).$$

The third model problem is the wave equation. This *hyperbolic* equation describes how a disturbance travels through matter. If the units are chosen so that the wave propagation speed is equal to one, the amplitude of a wave satisfies

$$\frac{\partial^2 u}{\partial t^2} = \triangle u.$$

Typical initial conditions specify the initial amplitude and take the initial velocity to be zero:

$$u(\vec{x}, 0) = u_0(\vec{x}),\ \frac{\partial u}{\partial t}(\vec{x}, 0) = 0.$$

In one dimension, all the problems take place on a finite interval on the x-axis. In more than one space dimension, geometry plays a vital role. In two dimensions, all the problems take place in a bounded region Ω in the (x, y) plane. In all cases, $f(\vec{x})$ and $u_0(\vec{x})$ are given functions of $\vec{x}$. All the problems involve boundary conditions where the value of u or some partial derivative of u is specified on the boundary of Ω. Unless otherwise specified, we will take the boundary values to be zero.

11.2 Finite Difference Methods

Basic finite difference methods for approximating solutions to these problems use a uniform mesh with spacing h. In one dimension, for the interval $a \le x \le b$, the spacing is $h = (b - a)/(m + 1)$ and the mesh points are

$$x_i = a + ih,\ i = 0, \ldots, m + 1.$$

The second derivative with respect to x is approximated by the 3-point centered second difference:

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

In two dimensions, the mesh is the set of points

$$(x_i, y_j) = (ih, jh)$$

that lie within the region Ω. Approximating the partial derivatives with centered second differences gives the 5-point discrete Laplacian

$$\begin{aligned}\triangle_h u(x, y) = &\frac{u(x + h, y) - 2u(x, y) + u(x - h, y)}{h^2} \\ &+ \frac{u(x, y + h) - 2u(x, y) + u(x, y - h)}{h^2}.\end{aligned}$$

Alternative notation uses $P = (x, y)$ for a mesh point and $N = (x, y + h)$, $E = (x + h, y)$, $S = (x, y - h)$, and $W = (x - h, y)$ for its four neighbors in the four compass directions. The discrete Laplacian is

$$\triangle_h u(P) = \frac{u(N) + u(W) + u(E) + u(S) - 4u(P)}{h^2}.$$

The finite difference Poisson problem involves finding values of u so that

$$\triangle_h u(\vec{x}) = f(\vec{x})$$

for each point $\vec{x}$ on the mesh.

If the source term $f(\vec{x})$ is zero, Poisson's equation is called Laplace's equation:

$$\triangle_h u(x) = 0.$$

In one dimension, Laplace's equation has only trivial solutions. The value of u at a mesh point x is the average of the values of u at its left and right neighbors, so $u(x)$ must be a linear function of x. Taking the boundary conditions into consideration implies that $u(x)$ is the linear function connecting the two boundary values. If the boundary values are zero, then $u(x)$ is identically zero. In more than one dimension, solutions to Laplace's equation are called *harmonic* functions and are not simply linear functions of $\vec{x}$.

The finite difference heat and wave equations also make use of first and second differences in the t direction. Let δ denote the length of a time step. For the heat equation, we use a difference scheme that corresponds to Euler's method for ordinary differential equations:

$$\frac{u(\vec{x}, t+\delta) - u(\vec{x}, t)}{\delta} = \triangle_h u(\vec{x}).$$

Starting with the initial conditions $u(\vec{x}, 0) = u_0(\vec{x})$, we can step from any value of t to $t + \delta$ with

$$u(\vec{x}, t+\delta) = u(\vec{x}, t) + \delta \triangle_h u(\vec{x}, t)$$

for all of the mesh points $\vec{x}$ in the region. The boundary conditions supply values on the boundary or outside the region. This method is *explicit* because each new value of u can be computed directly from values of u at the previous time step. More complicated methods are *implicit* because they involve the solution of systems of equations at each step.

For the wave equation, we can use a centered second difference in t:

$$\frac{u(\vec{x}, t+\delta) - 2u(\vec{x}, t) + u(\vec{x}, t-\delta)}{\delta^2} = \triangle_h u(\vec{x}, t).$$

This requires two "layers" of values of the solution, one at $t - \delta$ and one at t. In our simple model problem, the initial condition

$$\frac{\partial u}{\partial t}(\vec{x}, 0) = 0$$

allows us to start with both $u(\vec{x}, 0) = u_0(\vec{x})$ and $u(\vec{x}, \delta) = u_0(\vec{x})$. We compute subsequent layers with

$$u(\vec{x}, t+\delta) = 2u(\vec{x}, t) - u(\vec{x}, t-\delta) + \delta^2 \triangle_h u(\vec{x}, t)$$

for all of the mesh points $\vec{x}$ in the region. The boundary conditions supply values on the boundary or outside the region. Like our scheme for the heat equation, this method for solving the wave equation is *explicit*.

11.3 Matrix Representation

If a one-dimensional mesh function is represented as a vector, the one-dimensional difference operator $\triangle_h$ becomes the tridiagonal matrix

$$\frac{1}{h^2}\begin{pmatrix} -2 & 1 & & & & & \\ 1 & -2 & 1 & & & & \\ & 1 & -2 & 1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & 1 & -2 & 1 & \\ & & & & 1 & -2 & \end{pmatrix}.$$

This matrix is symmetric. (It is also *negative definite*.) Most importantly, even if there are thousands of interior mesh points, there are at most three nonzero elements in each row and column. Such matrices are the prime examples of *sparse* matrices. When computing with sparse matrices, it is important to use data structures that store only the locations and values of the nonzero elements.

With u represented as a vector and $h^2\triangle_h$ as a matrix A, the Poisson problem becomes

$$Au = b,$$

where b is a vector (the same size as u) containing the values of $h^2 f(x)$ at the interior mesh points. The first and last components of b would also include any nonzero boundary values.

In MATLAB, the solution to the discrete Poisson problem is computed using sparse backslash, which takes advantage of the sparsity in A:

```
u = A\b
```

The situation for meshes in two dimensions is more complicated. Let's number the interior mesh points in Ω from top to bottom and from left to right. For example, the numbering of an L-shaped region would be

```
L =
     0     0     0     0     0     0     0     0     0     0     0
     0     1     5     9    13    17    21    30    39    48     0
     0     2     6    10    14    18    22    31    40    49     0
     0     3     7    11    15    19    23    32    41    50     0
     0     4     8    12    16    20    24    33    42    51     0
     0     0     0     0     0     0    25    34    43    52     0
     0     0     0     0     0     0    26    35    44    53     0
     0     0     0     0     0     0    27    36    45    54     0
     0     0     0     0     0     0    28    37    46    55     0
     0     0     0     0     0     0    29    38    47    56     0
     0     0     0     0     0     0     0     0     0     0     0
```

The zeros are points on the boundary or outside the region. With this numbering, the values of any function defined on the interior of the region can be reshaped into a long column vector. In this example, the length of the vector is 56.

If a two-dimensional mesh function is represented as a vector, the finite difference Laplacian becomes a matrix. For example, at point number 43,

$$h^2 \triangle_h u(43) = u(34) + u(42) + u(44) + u(52) - 4u(43).$$

If A is the corresponding matrix, then its 43rd row would have five nonzero elements:

$$a_{43,34} = a_{43,42} = a_{43,44} = a_{43,52} = 1, \text{ and } a_{43,43} = -4.$$

A mesh point near the boundary has only two or three interior neighbors, so the corresponding row of A has only three or four nonzero entries.

The complete matrix A has -4's on its diagonal, four 1's off the diagonal in most of its rows, two or three 1's off the diagonal in some of its rows, and zeros elsewhere. For the example region above, A would be 56 by 56. Here is A if there are only 16 interior points.

```
A =
   -4    1    1    0    0    0    0    0    0    0    0    0    0    0    0    0
    1   -4    0    1    0    0    0    0    0    0    0    0    0    0    0    0
    1    0   -4    1    1    0    0    0    0    0    0    0    0    0    0    0
    0    1    1   -4    0    1    0    0    0    0    0    0    0    0    0    0
    0    0    1    0   -4    1    1    0    0    0    0    0    0    0    0    0
    0    0    0    1    1   -4    0    1    0    0    0    0    0    0    0    0
    0    0    0    0    1    0   -4    1    0    0    0    1    0    0    0    0
    0    0    0    0    0    1    1   -4    1    0    0    0    1    0    0    0
    0    0    0    0    0    0    0    1   -4    1    0    0    0    1    0    0
    0    0    0    0    0    0    0    0    1   -4    1    0    0    0    1    0
    0    0    0    0    0    0    0    0    0    1   -4    0    0    0    0    1
    0    0    0    0    0    0    1    0    0    0    0   -4    1    0    0    0
    0    0    0    0    0    0    0    1    0    0    0    1   -4    1    0    0
    0    0    0    0    0    0    0    0    1    0    0    0    1   -4    1    0
    0    0    0    0    0    0    0    0    0    1    0    0    0    1   -4    1
    0    0    0    0    0    0    0    0    0    0    1    0    0    0    1   -4
```

This matrix is symmetric, negative definite, and sparse. There are at most five nonzero elements in each row and column.

MATLAB has two functions that involve the discrete Laplacian, `del2` and `delsq`. If `u` is a two-dimensional array representing a function $u(x, y)$, then `del2(u)` computes $\triangle_h u$, scaled by $h^2/4$, at interior points, and uses one-sided formulas at points near the boundary. For example, the function $u(x, y) = x^2 + y^2$ has $\triangle u = 4$. The statements

```
h = 1/20;
[x,y] = meshgrid(-1:h:1);
u = x.^2 + y.^2;
d = (4/h^2) * del2(u);
```

produce an array `d`, the same size as `x` and `y`, with all the elements equal to 4.

If `G` is a two-dimensional array specifying the numbering of a mesh, then `A = -delsq(G)` is the matrix representation of the operator $h^2\triangle_h$ on that mesh. The mesh numbering for several specific regions is generated by `numgrid`. For example,

```
m = 5
L = numgrid('L',2*m+1)
```

generates the L-shaped mesh with 56 interior points shown above. And

```
m = 3
A = -delsq(numgrid('L',2*m+1))
```

generates the 16-by-16 matrix `A` shown above.

The function `inregion` can also generate mesh numberings. For example, the coordinates of the vertices of the L-shaped domain are

```
xv = [0  0  1  1 -1 -1  0];
yv = [0 -1 -1  1  1  0  0];
```

The statement

```
[x,y] = meshgrid(-1:h:1);
```

generates a square grid of width `h`. The statement

```
[in,on] = inregion(x,y,xv,yv);
```

generates arrays of zeros and ones that mark the points that are contained in the domain, including the boundary, as well as those that are strictly on the boundary. The statements

```
p = find(in-on);
n = length(p);
L = zeros(size(x));
L(p) = 1:n;
```

number the n interior points from top to bottom and left to right. The statement

```
A = -delsq(L);
```

generates the n-by-n sparse matrix representation of the discrete Laplacian on the mesh.

With u represented as a vector with n components, the Poisson problem becomes

$$Au = b,$$

where b is a vector (the same size as u) containing the values of $h^2 f(x, y)$ at the interior mesh points. The components of b that correspond to mesh points with neighbors on the boundary or outside the region also include any nonzero boundary values.

As in one dimension, the solution to the discrete Poisson problem is computed using sparse backslash.

```
u = A\b
```

11.4 Numerical Stability

The time-dependent heat and wave equations generate a sequence of vectors, $u^{(k)}$, where the k denotes the kth time step. For the heat equation, the recurrence is

$$u^{(k+1)} = u^{(k)} + \sigma A u^{(k)},$$

where

$$\sigma = \frac{\delta}{h^2}.$$

This can be written

$$u^{(k+1)} = M u^{(k)},$$

where

$$M = I + \sigma A.$$

In one dimension, the iteration matrix M has $1 - 2\sigma$ on the diagonal and one or two σ's off the diagonal in each row. In two dimensions, M has $1 - 4\sigma$ on the diagonal and two, three, or four σ's off the diagonal in each row. Most of the row sums in M are equal to 1; a few are less than 1. Each element of $u^{(k+1)}$ is a linear combination of elements of $u^{(k)}$ with coefficients that add up to 1 or less. Now here is the key observation. If the elements of M are nonnegative, then the recurrence is stable. In fact, it is dissipative. Any error or noise in $u^{(k)}$ is not magnified in $u^{(k+1)}$. But if the diagonal elements of M are negative, then the recurrence can be unstable. Error and noise, including roundoff error and noise in the initial conditions, can be magnified with each time step. Requiring $1 - 2\sigma$ or $1 - 4\sigma$ to be positive leads to a very important *stability condition* for this explicit method for the heat equation. In one dimension,

$$\sigma \le \frac{1}{2}.$$

And, in two dimensions,

$$\sigma \le \frac{1}{4}.$$

If this condition is satisfied, the iteration matrix has positive diagonal elements and the method is stable.

Analysis of the wave equation is a little more complicated because it involves three levels, $u^{(k+1)}$, $u^{(k)}$, and $u^{(k-1)}$. The recurrence is

$$u^{(k+1)} = 2u^{(k)} - u^{(k-1)} + \sigma A u^{(k)},$$

where

$$\sigma = \frac{\delta^2}{h^2}.$$

The diagonal elements of the iteration matrix are now $2 - 2\sigma$, or $2 - 4\sigma$. In one dimension, the stability condition is

$$\sigma \le 1.$$

And, in two dimensions,

$$\sigma \leq \frac{1}{2}.$$

These stability conditions are known as the *CFL conditions*, after Courant, Friedrichs and Lewy, who wrote a paper in 1928 that used finite difference methods to prove existence of solutions to the partial differential equations of mathematical physics. Stability conditions are restrictions on the size of the time step, δ. Any attempt to speed up the computation by taking larger time steps is likely to be disastrous. For the heat equation, the stability condition is particularly severe—the time step must be smaller than the *square* of the space mesh width. More sophisticated methods, often involving some implicit equation solving at each step, have less restrictive or unlimited stability conditions.

The M-file `pdegui` illustrates the concepts discussed in this chapter by offering the choice among several domains and among the model partial differential equations. For Poisson's equation, `pdegui` uses sparse backslash to solve

$$\triangle_h u = 1$$

in the chosen domain. For the heat and wave equations, the stability parameter σ can be varied. If the critical value, 0.25 for the heat equation and 0.50 for the wave equation, is exceeded by even a small amount, the instability rapidly becomes apparent.

You will find much more powerful capabilities in the MATLAB Partial Differential Equation Toolbox.

11.5 The L-Shaped Membrane

Separating out periodic time behavior in the wave equation leads to solutions of the form

$$u(\vec{x}, t) = \cos(\sqrt{\lambda}\, t)\ v(\vec{x}).$$

The functions $v(\vec{x})$ depend upon λ. They satisfy

$$\triangle v + \lambda v = 0$$

and are zero on the boundary. The quantities λ that lead to nonzero solutions are the *eigenvalues*, and the corresponding functions $v(\vec{x})$ are the *eigenfunctions* or *modes*. They are determined by the physical properties and the geometry of each particular situation. The square roots of the eigenvalues are resonant frequencies. A periodic external driving force at one of these frequencies generates an unboundedly strong response in the medium.

Any solution of the wave equation can be expressed as a linear combination of these eigenfunctions. The coefficients in the linear combination are obtained from the initial conditions.

In one dimension, the eigenvalues and eigenfunctions are easily determined. The simplest example is a violin string, held fixed at the ends of the interval of length π. The eigenfunctions are

$$v_k(x) = \sin(kx).$$

The eigenvalues are determined by the boundary condition, $v_k(\pi) = 0$. Hence, k must be an integer and

$$\lambda_k = k^2.$$

If the initial condition, $u_0(x)$, is expanded in a Fourier sine series,

$$u_0(x) = \sum_k a_k \sin(kx),$$

then the solution to the wave equation is

$$\begin{aligned} u(x,t) &= \sum_k a_k \cos(kt) \sin(kx) \\ &= \sum_k a_k \cos(\sqrt{\lambda_k}\, t)\ v_k(x). \end{aligned}$$

In two dimensions, an L-shaped region formed from three unit squares is interesting for several reasons. It is one of the simplest geometries for which solutions to the wave equation cannot be expressed analytically, so numerical computation is necessary. Furthermore, the 270° nonconvex corner causes a singularity in the solution. Mathematically, the gradient of the first eigenfunction is unbounded near the corner. Physically, a membrane stretched over such a region would rip at the corner. This singularity limits the accuracy of finite difference methods with uniform grids. The MathWorks has adopted a surface plot of the first eigenfunction of the L-shaped region as the company logo. The computation of this eigenfunction involves several of the numerical techniques we have described in this book.

Simple model problems involving waves on an L-shaped region include an L-shaped membrane, or L-shaped tambourine, and a beach towel blowing in the wind, constrained by a picnic basket on one fourth of the towel. A more practical example involves ridged microwave waveguides. One such device, shown in Figure 11.1, is a waveguide-to-coax adapter. The active region is the channel with the H-shaped cross section visible at the end of the adapter. The ridges increase the bandwidth of the guide at the expense of higher attenuation and lower power-handling capability. Symmetry of the H about the dotted lines shown in the contour plot of the electric field implies that only one quarter of the domain needs to be considered and that the resulting geometry is our L-shaped region. The boundary conditions are different than our membrane problem, but the differential equation and the solution techniques are the same.

Eigenvalues and eigenfunctions of the L-shaped domain can be computed by finite difference methods. The MATLAB statements

```
m = 200
h = 1/m
A = delsq(numgrid('L',2*m+1))/h^2
```

set up the 5-point finite difference approximation to the Laplacian on an 200-by-200 mesh in each of the three squares that make up the domain. The resulting sparse matrix `A` has order 119201 and 594409 nonzero entries. The statement

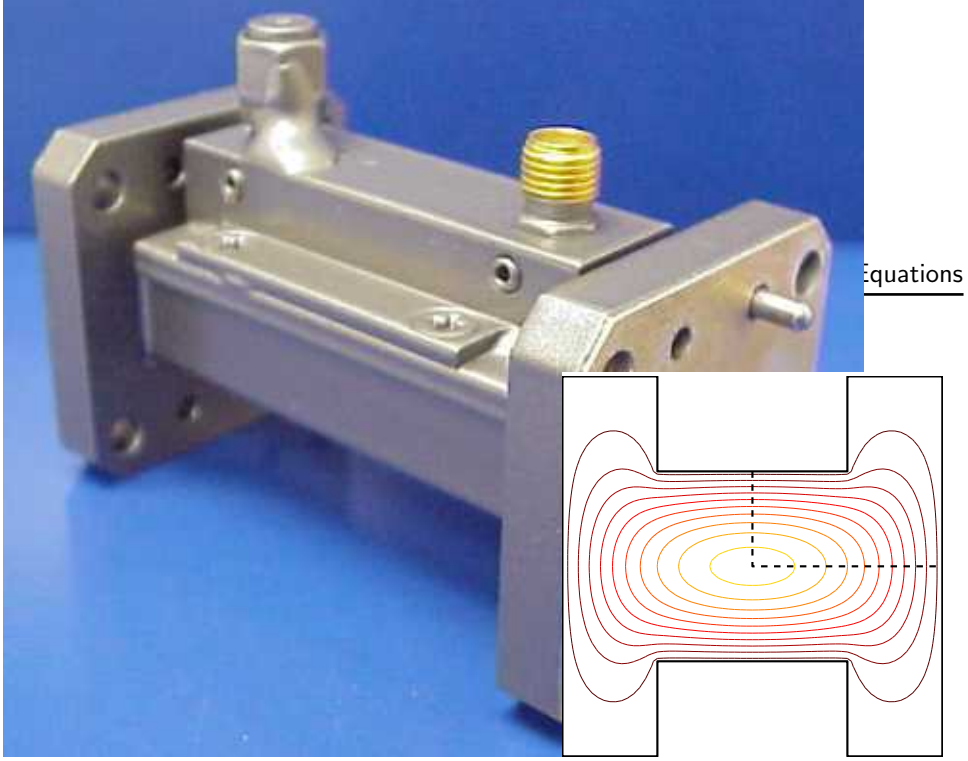

Figure 11.1. *A double-ridge microwave-to-coax adapter and its H-shaped region. Photo courtesy Advanced Technical Materials, Inc.* [1].

```
lambda = eigs(A,6,0)
```

uses Arnoldi's method from the MATLAB implementation of ARPACK to compute the first six eigenvalues. It takes less than 2 min on a 1.4 GHz Pentium laptop to produce

```
lambda =
    9.64147
   15.19694
   19.73880
   29.52033
   31.91583
   41.47510
```

The exact values are

```
    9.63972
   15.19725
   19.73921
   29.52148
   31.91264
   41.47451
```

You can see that even with this fine mesh and large matrix calculation, the computed eigenvalues are accurate to only three or four significant digits. If you try to get more accuracy by using a finer mesh and hence a larger matrix, the computation requires so much memory that the total execution time is excessive.

For the L-shaped domain and similar problems, a technique using analytic solutions to the underlying differential equation is much more efficient and accurate than finite difference methods. The technique involves polar coordinates and

fractional-order Bessel functions. With parameters α and λ, the functions

$$v(r,\theta) = J_\alpha(\sqrt{\lambda}\, r)\sin(\alpha\,\theta)$$

are exact solutions to the polar coordinate version of the eigenfunction equation

$$\frac{\partial^2 v}{\partial r^2} + \frac{1}{r}\frac{\partial v}{\partial r} + \frac{1}{r^2}\frac{\partial^2 v}{\partial \theta^2} + \lambda v = 0.$$

For any value of λ, the functions $v(r,\theta)$ satisfy the boundary conditions

$$v(r,0) = 0 \text{ and } v(r,\pi/\alpha) = 0$$

on the two straight edges of a circular sector with angle π/α. If $\sqrt{\lambda}$ is chosen to be a zero of the Bessel function, $J_\alpha(\sqrt{\lambda}) = 0$, then $v(r,\theta)$ is also zero on the circle, $r = 1$. Figure 11.2 shows a few of the eigenfunctions of the circular sector with angle $3\pi/2$. The eigenfunctions have been chosen to illustrate symmetry about $3\pi/4$ and $\pi/2$.

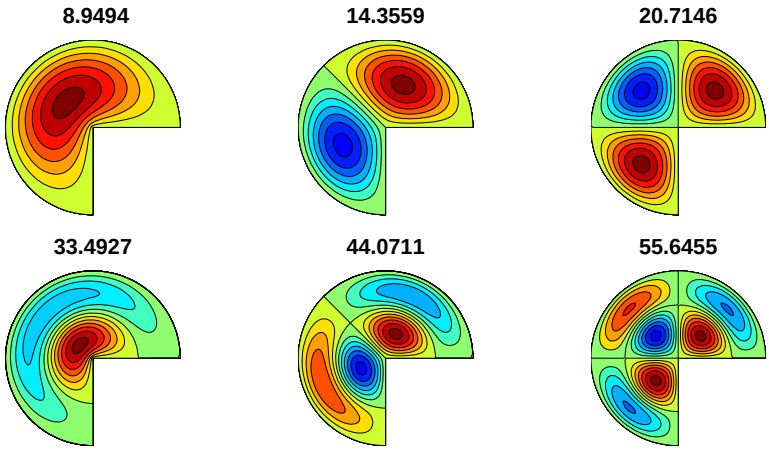

Figure 11.2. *Eigenfunctions of the three-quarter disc.*

We approximate the eigenfunctions of the L-shaped domain and other regions with corners by linear combinations of the circular sector solutions:

$$v(r,\theta) = \sum_j c_j J_{\alpha_j}(\sqrt{\lambda}\, r)\sin(\alpha_j\,\theta).$$

The angle of the reentrant 270° corner in the L-shaped region is $3\pi/2$, or $\pi/(2/3)$, so the values of α are integer multiples of $2/3$:

$$\alpha_j = \frac{2j}{3}.$$

These functions $v(r,\theta)$ are exact solutions to the eigenfunction differential equation. There is no finite difference mesh involved. The functions also satisfy the boundary

conditions on the two edges that meet at the reentrant corner. All that remains is to pick the parameter λ and the coefficients c_j so that the boundary conditions on the remaining edges are satisfied.

A least squares approach involving the SVD is used to determine λ and the c_j. Pick m points, (r_i, θ_i), on the remaining edges of the boundary. Let n be the number of fundamental solutions to be used. Generate an m-by-n matrix A with elements that depend upon λ:

$$A_{i,j}(\lambda) = J_{\alpha_j}(\sqrt{\lambda}\, r_i) \sin(\alpha_j\, \theta_i), \;\; i = 1, \ldots, m, \;\; j = 1, \ldots, n.$$

Then, for any vector c, the vector Ac is the vector of boundary values, $v(r_i, \theta_i)$. We want to make $||Ac||$ small without taking $||c||$ small. The SVD provides the solution.

Let $\sigma_n(A(\lambda))$ denote the smallest singular value of the matrix $A(\lambda)$. and let λ_k denote a value of λ that produces a local minimum of the smallest singular value:

$$\lambda_k = k\text{th minimizer}(\sigma_n(A(\lambda))).$$

Each λ_k approximates an eigenvalue of the region. The corresponding right singular vector provides the coefficients for the linear combination `c = V(:,n)`.

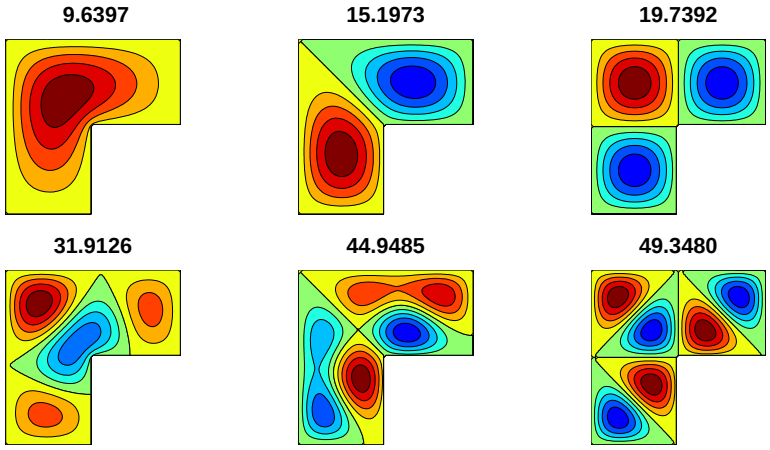

Figure 11.3. *Eigenfunctions of the L-shaped region.*

It is worthwhile to take advantage of symmetries. It turns out that the eigenfunctions fall into three symmetry classes:

- symmetric about the center line at $\theta = 3\pi/4$, so $v(r, \theta) = v(r, 3\pi/2 - \theta)$;
- antisymmetric about the center line at $\theta = 3\pi/4$, so $v(r, \theta) = -v(r, 3\pi/2 - \theta)$;
- eigenfunction of the square, so $v(r, \pi/2) = 0$ and $v(r, \pi) = 0$.

These symmetries allow us to restrict the values of α_j used in each expansion:

- $\alpha_j = \frac{2j}{3}$, j odd and not a multiple of 3;
- $\alpha_j = \frac{2j}{3}$, j even and not a multiple of 3;
- $\alpha_j = \frac{2j}{3}$, j a multiple of 3.

The M-file `membranetx` in the NCM directory computes eigenvalues and eigenfunctions of the L-membrane using these symmetries and a search for local minima of $\sigma_n(A(\lambda))$. The M-file `membrane`, distributed with MATLAB in the `demos` directory, uses an older version of the algorithm based on the QR decomposition instead of the SVD. Figure 11.3 shows six eigenfunctions of the L-shaped region, with two from each of the three symmetry classes. They can be compared with the eigenfunctions of the sector shown in Figure 11.2. By taking the radius of the sector to be $2/\sqrt{\pi}$, both regions have the same area and the eigenvalues are comparable.

The `demo` M-file `logo` makes a `surf` plot of the first eigenfunction, then adds lighting and shading to create The MathWorks logo. After being so careful to satisfy the boundary conditions, the logo uses only the first two terms in the circular sector expansion. This artistic license gives the edge of the logo a more interesting curved shape.

Exercises

11.1. Let n be an integer and generate n-by-n matrices A, D, and I with the statements

```
e = ones(n,1);
I = spdiags(e,0,n,n);
D = spdiags([-e e],[0 1],n,n);
A = spdiags([e -2*e e],[-1 0 1],n,n);
```

(a) For an appropriate value of h, the matrix $(1/h^2)A$ approximates $\triangle_h$ for the interval $0 \le x \le 1$. Is that value of h equal to $1/(n-1)$, $1/n$, or $1/(n+1)$?
(b) What does $(1/h)D$ approximate?
(c) What are D^TD and DD^T?
(d) What is A^2?
(e) What is `kron(A,I)+kron(I,A)`?
(f) Describe the output produced by `plot(inv(full(-A)))`.

11.2. (a) Use finite differences to compute a numerical approximation to the solution $u(x)$ to the one-dimensional Poisson problem

$$\frac{d^2u}{dx^2} = \exp(-x^2)$$

on the interval $-1 \le x \le 1$. The boundary conditions are $u(-1) = 0$ and $u(1) = 0$. Plot your solution.
(b) If you have access to `dsolve` in the Symbolic Toolbox, or if you are very good at calculus, find the analytic solution of the same problem and compare it with your numerical approximation.

11.3. Reproduce the contour plot in Figure 11.1 of the first eigenfunction of the H-shaped ridge waveguide formed from four L-shaped regions.

11.4. Let $h(x)$ be the function defined by the M-file `humps(x)`. Solve four different problems involving $h(x)$ on the interval $0 \le x \le 1$.
(a) One-dimensional Poisson problem with `humps` as the source term:

$$\frac{d^2u}{dx^2} = -h(x),$$

with boundary conditions

$$u(0) = 0,\ u(1) = 0.$$

Make plots, similar to Figure 11.4, of $h(x)$ and $u(x)$. Compare `diff(u,2)` with `humps(x)`.

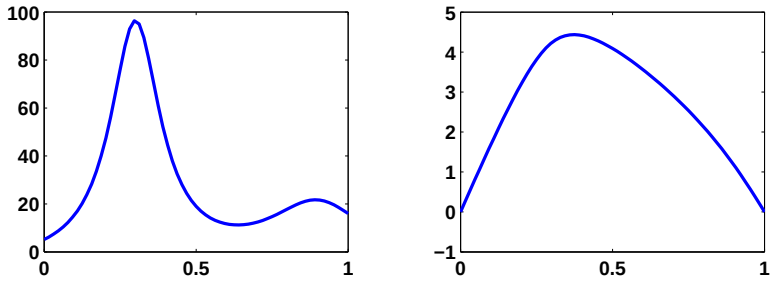

Figure 11.4. *$h(x)$ and $u(x)$.*

(b) One-dimensional heat equation with `humps` as the source term:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + h(x),$$

with initial value

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

and boundary conditions

$$u(0, t) = 0,\ u(1, t) = 0.$$

Create an animated plot of the solution as a function of time. What is the limit as $t \to \infty$ of $u(x, t)$?
(c) One-dimensional heat equation with `humps` as the initial value:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2},$$

with initial value

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

and boundary conditions

$$u(0,t) = h(0),\ u(1,t) = h(1).$$

Create an animated plot of the solution as a function of time. What is the limit as $t \to \infty$ of $u(x,t)$?
(d) One-dimensional wave equation with `humps` as the initial value:

$$\frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2},$$

with initial values

$$\begin{aligned} u(x,0) &= h(x), \\ \frac{\partial u}{\partial t}(x,0) &= 0, \end{aligned}$$

and boundary conditions

$$u(0,t) = h(0),\ u(1,t) = h(1).$$

Create an animated plot of the solution as a function of time. For what values of t does $u(x,t)$ return to its initial value $h(x)$?

11.5. Let $p(x,y)$ be the function defined by the M-file `peaks(x,y)`. Solve four different problems involving $p(x,y)$ on the square $-3 \le x \le 3$, $-3 \le y \le 3$.

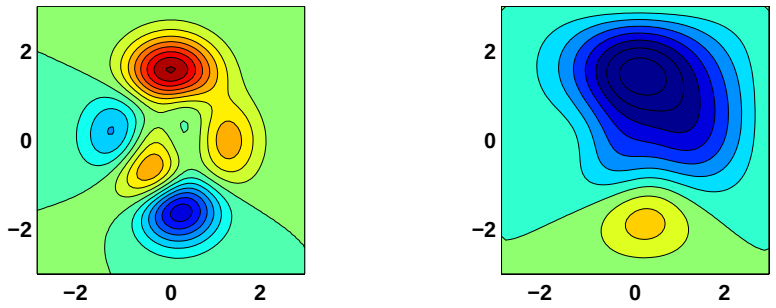

Figure 11.5. *$p(x,y)$ and $u(x,y)$.*

(a) Two-dimensional Poisson problem with `peaks` as the source term:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = p(x,y),$$

with boundary conditions

$$u(x,y) = 0 \text{ if } |x| = 3 \text{ or } |y| = 3.$$

Make contour plots, similar to Figure 11.5, of $p(x,y)$ and $u(x,y)$.

(b) Two-dimensional heat equation with `peaks` as the source term:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} - p(x,y),$$

with initial value

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

and boundary conditions

$$u(x,y,t) = 0 \text{ if } |x| = 3 \text{ or } |y| = 3.$$

Create an animated contour plot of the solution as a function of time. What is the limit as $t \to \infty$ of $u(x,t)$?
(c) Two-dimensional heat equation with `peaks` as the initial value:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2},$$

with initial value

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

and boundary conditions

$$u(x,y,t) = p(x,y) \text{ if } |x| = 3 \text{ or } |y| = 3.$$

Create an animated contour plot of the solution as a function of time. What is the limit as $t \to \infty$ of $u(x,t)$?
(d) Two-dimensional wave equation with `peaks` as the initial value:

$$\frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2},$$

with initial values

$$\begin{aligned} u(x,y,0) &= p(x,y), \\ \frac{\partial u}{\partial t}(x,y,0) &= 0 \end{aligned}$$

and boundary conditions

$$u(x,y,t) = p(x,y) \text{ if } |x| = 3 \text{ or } |y| = 3.$$

Create an animated contour plot of the solution as a function of time. Does the limit as $t \to \infty$ of $u(x,t)$ exist?

11.6. The *method of lines* is a convenient technique for solving time-dependent partial differential equations. Replace all the spatial derivatives with finite differences, but leave the time derivatives intact. Then use a stiff ordinary differential equation solver on the resulting system. In effect, this is an implicit time-stepping finite difference algorithm with the time step determined

automatically and adaptively by the ODE solver. For our model heat and wave equations, the ODE systems are simply

$$\dot{\mathbf{u}} = (1/h^2)A\mathbf{u}$$

and

$$\ddot{\mathbf{u}} = (1/h^2)A\mathbf{u}.$$

The matrix $(1/h^2)A$ represents $\triangle_h$, and $\mathbf{u}$ is the vector-valued function of t formed from all the elements $u(x_i, t)$ or $u(x_i, y_j, t)$ at the mesh points.
(a) The MATLAB function `pdepe` implements the method of lines in a general setting. Investigate its use for our one- and two-dimensional model heat equations.
(b) If you have access to the Partial Differential Equation Toolbox, investigate its use for our two-dimensional model heat and wave equations.
(c) Implement your own method of lines solutions for our model equations.

11.7. Answer the following questions about `pdegui`.
(a) How does the number of points n in the grid depend upon the grid size h for the various regions?
(b) How does the time step for the heat equation and for the wave equation depend upon the grid size h?
(c) Why are the contour plots of the solution to the Poisson problem and the eigenvalue problem with `index = 1` similar?
(d) How do the contour plots produced by `pdegui` of the eigenfunctions of the L-shaped domain compare with those produced by

```
contourf(membranetx(index))
```

(e) Why are the regions `Drum1` and `Drum2` interesting? Search the Web for "isospectral" and "Can you hear the shape of a drum?" You should find many articles and papers, including ones by Gordon, Webb, and Wolpert [3] and Driscoll [2].

11.8. Add the outline of your hand that you obtained in exercise 3.4 as another region to `pdegui`. Figure 11.6 shows one of the eigenfunctions of my hand.

11.9. The electrostatic capacity of a region Ω is the quantity

$$\int\int_{\Omega} u(x, y)dxdy,$$

where $u(x, y)$ is the solution to the Poisson problem

$$\triangle u = -1 \text{ in } \Omega$$

and $u(x, y) = 0$ on the boundary of Ω.
(a) What is the capacity of the unit square?
(b) What is the capacity of the L-shaped domain?
(c) What is the capacity of your hand?

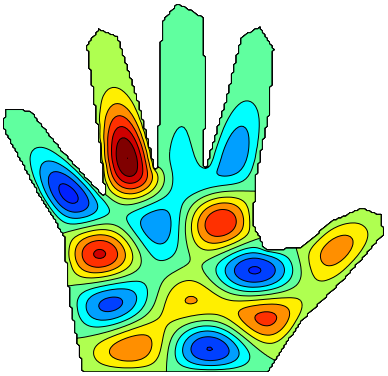

Figure 11.6. *An eigenfunction of a hand.*

11.10. The statements

```
load penny
P = flipud(P)
contour(P,1:12:255)
colormap(copper)
axis square
```

access a file in the MATLAB `demos` directory and produce Figure 11.7. The data were obtained in 1984 at what was then the National Bureau of Standards by an instrument that makes precise measurements of the depth of a mold used to mint the U.S. one cent coin.

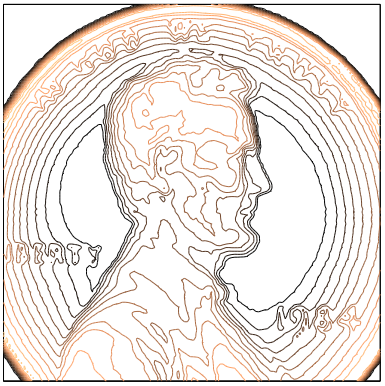

Figure 11.7. *The depth of a mold used to mint the U.S. one cent coin.*

The NCM function `pennymelt` uses this penny data as the initial condition, $u(x, y, 0)$, for the heat equation and produces an animated, evolving plot of the solution, $u(x, y, t)$. You can choose either a lighted, surface plot or

a contour plot. You can choose the time step δ with uicontrols or with `pennymelt(delta)`. You can also choose a time-stepping algorithm know as the *ADI* or *alternating direction implicit* method. Each time step involves two half-steps, one implicit in the x direction and the other implicit in the y direction.

$$\begin{aligned} -\sigma u^{(k+1/2)}(N) + (1+2\sigma)u^{(k+1/2)}(P) - \sigma u^{(k+1/2)}(S) \\ = \sigma u^{(k)}(E) + (1-2\sigma)u^{(k)}(P) + \sigma u^{(k)}(W), \\ -\sigma u^{(k+1)}(E) + (1+2\sigma)u^{(k+1)}(P) - \sigma u^{(k+1)}(W) \\ = \sigma u^{(k+1/2)}(N) + (1-2\sigma)u^{(k+1/2)}(P) + \sigma u^{(k+1/2)}(S). \end{aligned}$$

Solving these implicit equations on an m-by-n grid requires the solution of m tridiagonal systems of order n for the first half-step and then n tridiagonal systems of order m for the second half-step.
Answer these questions about `pennymelt`.
(a) What is the limiting behavior of $u(x, y, t)$ as $t \to \infty$?
(b) For what values of δ is the explicit time-stepping algorithm stable?
(c) Demonstrate that the ADI method is stable for any value of δ.

11.11. Let $p(x, y)$ be the function defined on a 128-by-128 square by the penny data described in the previous exercise.
(a) Make a contour plot of $p(x, y)$ and make a lighted surface plot using the section of code in `pennymelt.m`.
(b) Solve the discrete Poisson problem

$$\triangle_h u = p$$

with $u(x, y) = 0$ outside the square, and plot the solution $u(x, y)$.
(c) Use `del2` to compute

$$f = \triangle_h u,$$

and compare $f(x, y)$ with $p(x, y)$.

11.12. Modify `pennymelt.m` to solve the wave equation instead of the heat equation.

11.13. Modify `waves.m` to use nine eigenfunctions instead of four.

11.14. The eigenvalues and eigenfunctions of the unit square are

$$\begin{aligned} \lambda_{m,n} &= (m^2 + n^2)\pi^2, \\ u_{m,n} &= \sin mx \sin ny. \end{aligned}$$

If the $\lambda_{m,n}$ are indexed with one subscript and listed in increasing order, we have

$$\lambda_k = (2, 5, 5, 8, 10, 10, 13, 13, 17, 17, 18, 20, 20, \ldots)\pi^2.$$

We see that λ_1, λ_4, and λ_{11} are simple eigenvalues, but that most of the eigenvalues are double.
(a) What is the smallest triple eigenvalue of the unit square and what is its index? In other words, what is the smallest integer that can be written as the sum of two squares in three different ways?
(b) What is the smallest quadruple eigenvalue of the unit square?

11.15. By reflecting the eigenfunctions of the unit square twice, we obtain some of the eigenfunctions of the L-shaped domain. The indexing is different because the L also has eigenfunctions that are not derived from the square. For example, λ_3 of the L is $2\pi^2$ because it is equal to λ_1 of the square. And $\lambda_8 = \lambda_9$ of the L is a double eigenvalue, $5\pi^2$, corresponding to $\lambda_2 = \lambda_3$ of the square.
(a) Roughly what fraction of the eigenvalues of the L-shaped region are also eigenvalues of the square?
(b) What is the smallest triple eigenvalue of the L-shaped region and what is its index?
(c) What is the smallest quadruple eigenvalue of the L-shaped region?
(d) Neither `membranetx` nor `pdegui` uses the $\sin mx \sin ny$ representation of eigenfunctions of the square. This is OK because these eigenfunctions are not unique and can have other representations. How do `membranetx` and `pdegui` compute eigenfunctions? How do they get a set of linearly independent eigenfunctions for eigenvalues with multiplicity greater than one?

11.16. Enter the commands

```
ncmlogo
cameratoolbar
```

Or, just enter the command `ncmlogo` and then select **Camera Toolbar** from the **View** tab on the figure window. Experiment with the various icons available on the new toolbar. What do they all do?

11.17. Make your own copy of `ncmlogo` and modify it to create a logo for your own book or company.

Bibliography

[1] Advanced Technical Materials, Inc.
http://www.atmmicrowave.com

[2] T. A. Driscoll, *Eigenmodes of isospectral drums*, SIAM Review, 39 (1997), pp. 1–17.
http://www.math.udel.edu/~driscoll/pubs/drums.pdf

[3] C. Gordon, D. Webb, and S. Wolpert, *Isospectral plane domains and surfaces via Riemannian orbifolds*, Inventiones Mathematicae, 110 (1992), pp. 1–22.