24 Heat Flow

Problem: You are given an aluminum bar of length $L = 1$ m and width w aligned along the x axis (Fig. 24.1). It is insulated along its length but not its ends. Initially the bar is at a uniform temperature of 100°C, and then both ends are placed in contact with ice water at 0°C. Heat flows only out of the noninsulated ends. Your **problem** is to determine how the temperature will vary as we move along the length of the bar at later times.

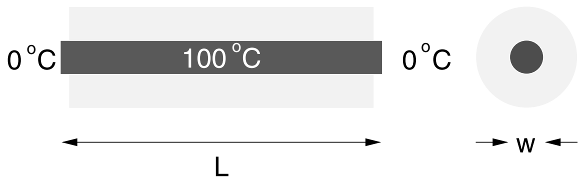

Fig. 24.1 A metallic bar insulated along its length with its ends kept at 0°C.

24.1 The Parabolic Heat Equation (Theory)

A basic fact of nature is that heat flows from hot to cold, that is, from regions of high temperature to regions of low temperature. We give these words mathematical expression by stating that the rate of heat flow **H** through a material is proportional to the gradient of the temperature T within that material:

$$\mathbf{H} = -K\,\nabla T(\mathbf{x}, t) \tag{24.1}$$

where K is the thermal conductivity of the material. The total amount of heat $Q(t)$ in the material at any one time is proportional to the integral of the temperature over the volume of the material:

$$Q(t) = \int d\mathbf{x}\, C\, \rho(\mathbf{x})\, T(\mathbf{x}, t) \tag{24.2}$$

where C is the specific heat of the material and ρ its density. Because energy is conserved, the rate of decrease of Q with time must equal the amount of heat flowing out of the material. When this energy balance is struck, and the divergence theorem applied, the *heat equation* results:

$$\frac{\partial T(\mathbf{x},t)}{\partial t} = \frac{K}{C\rho}\nabla^2 T(\mathbf{x},t) \tag{24.3}$$

The heat equation (24.3) is a parabolic PDE with space and time as independent variables. The specification of this problem implies that there is no temperature variation in directions perpendicular to the bar (y and z), and so we have only one spatial coordinate in our PDE:

$$\frac{\partial T(x,t)}{\partial t} = \frac{K}{C\rho}\frac{\partial^2 T(x,t)}{\partial x^2} \tag{24.4}$$

We are given the initial temperature of the bar, and the boundary conditions:

$$T(x,t=0) = 100°\mathrm{C} \qquad T(x=0,t) \;=\; T(x=L,t) \;=\; 0°\mathrm{C} \tag{24.5}$$

24.2 Solution: Analytic Expansion

As with Laplace's equation, the analytic "solution" starts with the assumption that the solution is the product of functions of space and time:

$$T(x,t) = X(x)\mathcal{T}(t) \tag{24.6}$$

When (24.6) is substituted into the heat equation (24.4), and the resulting equation is divided by the $X(x)\mathcal{T}(t)$, there results two noncoupled ODEs:

$$\frac{d^2X(x)}{dx^2} + k^2X(x) = 0 \qquad \frac{d\mathcal{T}(t)}{dt} + k^2\frac{C}{C\rho}\mathcal{T}(t) = 0 \tag{24.7}$$

where k is a constant to be determined. The boundary condition that the temperature equals zero at $x = 0$ requires a sine function for X:

$$X(x) = A\sin kx \tag{24.8}$$

The boundary condition that the temperature equals zero at $x = L$ requires the sine function to vanish there:

$$\sin kL \;=\; 0 \qquad \Rightarrow \qquad k = k_n = n\pi/L \qquad n = 1, 2, \ldots \tag{24.9}$$

The time function is a decaying exponential with k_n in the exponent:

$$\mathcal{T}(t) \;=\; e^{-k_n^2 t/C\rho} \qquad \Rightarrow \qquad T(x,t) = A_n \sin k_n x\, e^{-k_n^2 t/C\rho} \tag{24.10}$$

where n can be any integer, and A_n is an arbitrary constant.

Equation (24.10) is a particular solution. The most general solution is the linear superposition of all values of n:

$$T(x,t) = \sum_{n=1}^{\infty} A_n \sin k_n x \, e^{-k_n^2 t/C\rho} \tag{24.11}$$

The expansion coefficients A_n are determined by the initial condition that at time $t = 0$ the entire bar has temperature $T = 100°\text{C}$:

$$T(x,t=0) = 100°\text{C} \qquad \Rightarrow \qquad \sum_{n=1}^{\infty} A_n \sin k_n x = 100°\text{C} \tag{24.12}$$

As with Laplace's equation, projecting out the sine functions determines $A_n = 4T_0/n\pi$ for n odd:

$$T(x,t) = \sum_{n=1,3,\ldots}^{\infty} \frac{4T_0}{n\pi} \sin k_n x \, e^{-k_n^2 K t/(C\rho)} \tag{24.13}$$

24.3 Solution: Finite Time Stepping (Leap Frog)

As we did with Laplace's equation, the numerical solution is based on converting the differential equation into a finite-difference ("difference") equation. We discretize space and time on a lattice (Fig. 24.2), and look for a solution along the nodes. The horizontal nodes with white centers correspond to the known values of the temperature for the initial time, while the vertical white nodes correspond to the fixed temperature along the boundaries. If we *also* knew the temperature for times along the bottom row, then we could use a relaxation algorithm, as we did for Laplace's equation. However, with only the top row known, we shall end up with an algorithm that steps forward in time, one row at a time, as in the children's game *leapfrog*.

The algorithm is customized for the equation being solved and for the constraints imposed by the particular set of initial and boundary conditions. With only one row of times to start with, we use a forward-difference approximation for the time derivative of the temperature:

$$\frac{\partial T(x,t)}{\partial t} \simeq \frac{T(x,t+\Delta t) - T(x,t)}{\Delta t} \tag{24.14}$$

Because we know the spatial variation of the temperature along the entire top row, as well as along the left and right sides, we are not as constrained with the space derivative as with the time derivative. Consequently, as we did with the

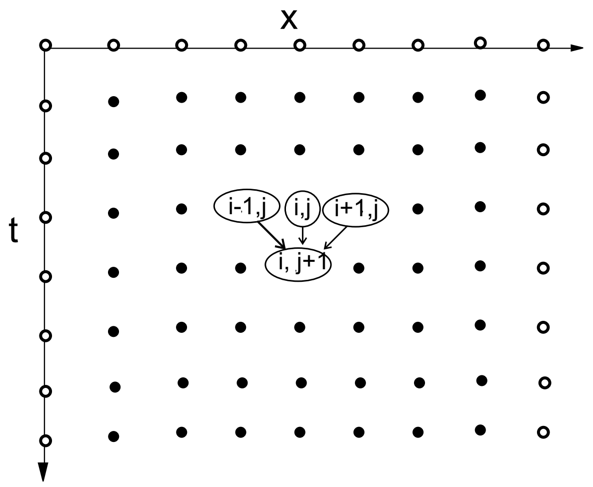

Fig. 24.2 The algorithm for the heat equation in which the temperature at the location $x = i\Delta x$ and time $t = (j+1)\Delta t$ is computed from the temperature values at three points of an earlier time. The nodes with white centers correspond to known initial and boundary conditions. (The boundaries are placed artificially close for illustrative purposes.)

Laplace equation, we use the more-accurate central-difference approximation for the (second) space derivative:

$$\frac{\partial^2 T(x,t)}{\partial x^2} \simeq \frac{T(x+\Delta x,t)+T(x-\Delta x,t)-2T(x,t)}{(\Delta x)^2} \tag{24.15}$$

When we substitute these approximations for the derivatives into the heat equation (24.4), we obtain the heat difference equation:

$$\frac{T(x,t+\Delta t)-T(x,t)}{\Delta t} = \frac{K}{C\rho}\frac{T(x+\Delta x,t)+T(x-\Delta x,t)-2T(x,t)}{\Delta x^2} \tag{24.16}$$

We reorder this equation to a form in which the solution can be stepped forward in time:

$$T_{i,j+1} = T_{i,j} + \eta\left[T_{i+1,j}+T_{i-1,j}-2T_{i,j}\right] \qquad \eta = \frac{K\Delta t}{C\rho\Delta x^2} \tag{24.17}$$

where $x = i\Delta x$ and $t = j\Delta t$. This algorithm is called *explicit* because it provides a solution in terms of known values of the temperature. If we tried to solve for the temperature at all lattice sites simultaneously, then we would have an *implicit* algorithm that requires us to solve equations involving unknown values of the temperature (Fig. 24.2). We see that the temperature at space–time point $(i, j+1)$ is computed from the three temperature values at an earlier time j, and at adjacent space values $i \pm 1, i$. We start the solution at the top row, moving it forward in time for as long as we want, keeping the temperature along the ends fixed at 0°C. Fig. 24.3 shows the solution so obtained.

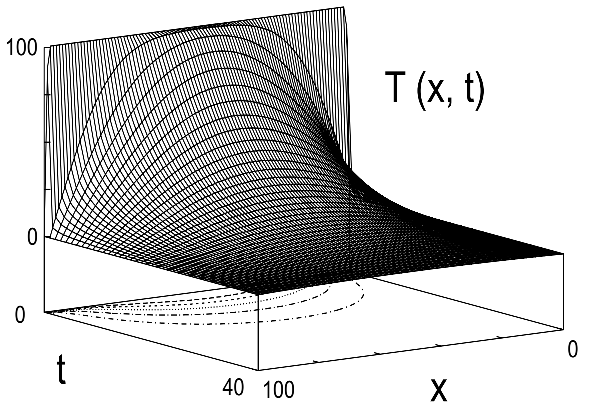

Fig. 24.3 A numeric calculation of the temperature versus position and versus time, with isotherm contours projected onto horizontal plane.

24.4 von Neumann Stability Assessment

When we solve a PDE by converting it to a difference equation, we hope that the solution to the difference equation is a good approximation to the solution of the PDE. If the difference-equation solution diverges, then we know we have a bad approximation. If the difference solution converges, then it generally provides a good approximation to the PDE. The *von Neumann stability analysis* is based on the assumption that eigenmodes of the difference equation can be written as

$$T_{m,j} = \xi(k)^j \, e^{i\,k\,m\,\Delta x} \tag{24.18}$$

where $x = m\Delta x$, $t = j\Delta t$, but $i = \sqrt{-1}$ is the imaginary number. The constant k is an unknown wave vector $(2\pi/\lambda)$, and $\xi(k)$ is an unknown complex function of k. View (24.18) as a basis function that oscillates in space (the exponential), with an amplitude or *amplification factor* $\xi(k)^j$ that changes by an additional power of ξ for each time step. If the general solution to the difference equation can be expanded in terms of these eigenmodes, then the general solution will be stable if the eigenmodes are stable. Clearly, for an eigenmode to be stable, the amplitude ξ cannot grow in time j, which means $|\xi(k)| < 1$ for all values of the parameter k [9,73].

Application of the stability analysis is more straightforward that it might seem. We substitute the expansion (24.18) into the difference equation (24.17):

$$\xi^{j+1}e^{ikm\Delta x} = \xi^{j+}e^{ikm\Delta x} + \eta\left[\xi^{j}e^{ik(m+1)\Delta x} + \xi^{j+}e^{ik(m-1)\Delta x} - 2\xi^{j+}e^{ikm\Delta x}\right]$$

After cancelling some common factors, it is easy to solve for ξ:

$$\xi(k) = 1 + 2\eta[\cos(k\Delta x) - 1] \tag{24.19}$$

In order for $|\xi(k)| < 1$ for all possible k values, we must have

$$\eta = \frac{K\Delta t}{C\rho\Delta x^2} < \frac{1}{4} \tag{24.20}$$

Equation (24.20) tells us that if we make the time step Δt smaller, we always improve stability. But if we decrease the space step Δx, without a simultaneous quadratic *increase* in the time step, we worsen stability. The lack of space–time symmetry arises from our use of stepping in time, but not space.

In general, you should perform a stability analysis for every PDE you have to solve, although it can get complicated [9]. Yet even if you do not, the lesson here is that you may have to try out different *combinations* of Δx and Δt variations until a stable and reasonable solution is obtained. You may expect, nonetheless, that there are choices for Δx and Δt for which the numeric solution fails, and that simply decreasing an individual Δx or Δt, in the hope that this will increase precision, may not improve the solution.

24.4.1 Heat Equation Implementation

Recall, we want to solve for the temperature distribution within an aluminum bar of length $L = 1$ m subject to the boundary and initial conditions

$$T(x=0,t) = T(x=L,t) = 0°\text{C} \qquad T(x,t=0) = 100°\text{C} \tag{24.21}$$

The International system constants for iron's specific heat, thermal conductivity, and density are:

$$C = 0.113\,\text{cal}/°\text{C g} \quad K = 0.12\,\text{cal°C g s} \quad \rho = 7.8\,\text{g/cm}^3 \tag{24.22}$$

1. Write or modify `EqHeat.java` in Listing 24.1 to solve the heat equation.
2. Define a 2D array `T[101][2]` for the temperature as a function of space and time. The first index is for the 100 space divisions of the bar, and the second index for present and past times (because thousands of time steps may be made, we save memory by saving only two times).
3. For time $t = 0$, (*j=1*), initialize `T` so that all points on the bar except the end points are at 100°C. Set the temperatures of the ends to 0°C.
4. Apply (24.14) to obtain the temperature at the next time step.
5. Assign the present-time values of the temperature to the past values: **`T[i][1] = T[i][2], i=1, ..., 101.`**

Listing 24.1: `EqHeat.java` solves the heat equation for a 1D space and time, by leapfrogging (time stepping) the initial conditions forward in time. You will need to adjust the parameters to obtain a solution like those in the figures.

```java
// EqHeat.java: Solve heat equation via finite differences

import java.io.*;                                          // Import IO library

public class EqHeat {                            // Class constants in MKS units

  public static final int Nx = 11, Nt = 300;                    // Grid sizes
  public static final double Dx = 0.01, Dt = 0.1;               // Step sizes
  public static final double KAPPA = 210.;          // Thermal conductivity
  public static final double SPH = 900.;                     // Specific heat
  public static final double RHO = 2700.;                       // Al density

  public static void main(String[] argv)
                        throws IOException, FileNotFoundException {

    int ix, t;
    double T[][] = new double[Nx][2];
    double cons;

    PrintWriter q = new PrintWriter                      // File for gnuplot
                        (new FileOutputStream("EqHeat.dat"), true);
    for ( ix=1;  ix  <  Nx-1;  ix++ ) T[ix][0] = 100.;   // Initialize
    T[0][0] = 0.;
    T[0][1] = 0.;                                             // Except ends
    T[Nx-1][0] = 0.;
    T[Nx-1][1] = 0.;
    cons = KAPPA/(SPH*RHO)*Dt/(Dx*Dx);                // Integration factor
    System.out.println("constant = " + cons);
                                                                   // t loop
    for ( t=1;  t <= Nt;  t++ )  {
                                                                   // x loop
      for ( ix=1;  ix  <  Nx-1;  ix++ ) T[ix][1] = T[ix][0]
                 + cons*(T[ix + 1][0] + T[ix-1][0]-2.*T[ix][0]);
      if ( t%10==0 || t==1 )  {                        // Save every N steps
        for ( ix = 0; ix<Nx;  ix++ ) q.println(T[ix][1]);
        q.println();                                    // Blank line ends row
      }
      for ( ix = 1; ix<Nx-1; ix++ ) T[ix][0]=T[ix][1];  // New to old
    }                                                       // End t loop
    System.out.println("data stored in EqHeat.dat");
  }                                                          // End main
}                                                           // End class
```

6. Start with 50 time steps. Once you are confident the program is running properly, use thousands of steps to see the bar cool smoothly with time. For every ~500 time steps, print the time and temperature along the bar.

24.5 Assessment and Visualization

1. Check that your program gives a temperature distribution that varies smoothly along the bar, and which agrees with the boundary conditions.
2. Check that your program gives a temperature distribution that varies smoothly with time and attains equilibrium. You may have to vary the time and space steps to obtain well-behaved solutions.
3. Compare the analytic and numeric solutions (and the wall times needed to compute them). If the solutions differ, suspect the one which does not appear smooth and continuous.
4. Make surface plots of the temperature vs. position for several times.
5. Better yet, make a surface plot of the temperature vs. position vs. time.
6. Plot the *isotherms* (contours of constant temperature).
7. **Stability test:** Check (24.20) that the temperature diverges in t if $\eta > 1/4$.
8. **Material dependence:** Repeat the calculation for iron. Take note that the stability condition requires you to change the size of the time step.
9. **Initial sinusoidal distribution** $\sin(\pi x/L)$**:** Compare to analytic solution,

$$T(x,t) = \sin(\pi x/L)e^{-\pi^2 Kt/(L^2 C\rho)} \tag{24.23}$$

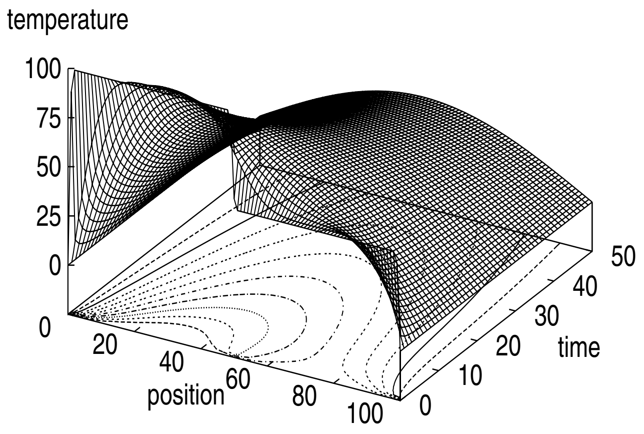

Fig. 24.4 Temperature versus position and time when two bars of differing temperature are placed in contact at $t = 0$. The projected contours show the isotherms.

10. **Two bars in contact:** Two identical bars 0.25 m long are placed in contact along one of their ends with their other ends kept at 0°C (Fig. 24.4). One is kept in a heat bath at 100°C, and the other at 50°C. Determine how the temperature varies with time and location.

11. **Radiating bar (Newton's cooling):** Imagine now, that instead of being insulated along its length, a bar is in contact with an environment at a temperature T_e. Newton's law of cooling (radiation) says that the rate of temperature change due to radiation is

$$\frac{\partial T}{\partial t} = -h(T - T_e) \tag{24.24}$$

where h is a positive constant. This leads a the modified heat equation

$$\frac{\partial T(x,t)}{\partial t} = \frac{K}{C\rho}\frac{\partial^2 T}{\partial^2 x} - hT(x,t) \tag{24.25}$$

Modify the algorithm to include Newton's cooling, and compare the cooling of this bar with that of the insulated bar.