MODULE 10.2

# Diffusion: Overcoming Differences

## Downloads

For several computational tools, the text's website has a *Diffusion* file containing the simulation this module develops and a *10_2QRQ.pdf* file containing system-dependent Quick Review Questions and answers available for download.

## Introduction

Heat energy is transferred by **thermal conduction** within or between objects where a temperature gradient exists. Particles or groups of particles with a higher temperature (more kinetic energy) transfer some of their energy to those at a lower temperature (less kinetic energy) upon collision. Thus, we have a **diffusion** of energy.

This diffusion of thermal energy presented a real problem for astronauts returning from a mission. As they brought their craft into the earth's atmosphere, the vehicle was traveling at about 40,000 km/h, generating tremendous friction. The temperatures on the exterior heat shield equaled 2760 °C, which is just over half the temperature of the sun's surface. Fortunately, the shield was effective enough to allow the cabin temperature to remain at 21°C (NASA Spinoff 1988, 2011).

During the late 1960s through the early 1970s, as part of the Apollo Mission, the United States sent manned spacecraft to the Moon. The heat shields for these vessels effectively fended off the diffusion of all that heat energy, generated upon atmospheric reentry, into the spacecraft. Each heat shield was coated with an ablative material—a substance that was allowed to char, dissipating energy and forming a protective coating, which did not allow the heat into the spacecraft itself (NASA Spinoff 1988, 2011).

A private company designed the heat shield for NASA, and the two entities collaborated in subsequent years to develop a number of fire-retardant paints and foams for military and civilian use. One of these, called Chartek, and derivative products are widely used by the oil and gas industries. Further product development led to Interchar, a fire-retardant commonly used to coat steel for construction. With a very thin layer (1–8 mm), Interchar does not hinder architectural design. Steel does not burn, but very high temperatures can weaken the metal. So, by delaying the transfer of heat energy to the steel, firefighters may be able to put out a fire before irreparable damage is done; and importantly, the coating delays loss of structural integrity for the evacuation of personnel (NASA Spinoff 1988, 2011).

## Problem

In this module, we want to model the heat diffusion through a thin metal bar that has a constant application of heat and cold at designated locations on the bar (Cunningham 2007). We also want to develop an animated scientific visualization to depict the diffusion process.

## Initializing the System

To simplify the situation, we apply heat and cold through the thickness of the bar and assume that each internal point on a line perpendicular to the top surface of the bar has the same temperature. If a point on the top surface has temperature 25 °C, then every point directly below that location is at 25 °C. Moreover, we assume that the bar is in a still room and that the immediate surroundings are at the same temperatures as the bar. Temperature diffuses within the bar, but external conditions do not affect the temperatures. Thus, we model the bar in two dimensions, length and width.

In many simulations, we model such a dynamic area with an $m \times n$ grid, or lattice, or a 2D rectangular array, or matrix, of numbers (Figure 10.2.1). Each cell in the lattice contains a value representing a characteristic of a corresponding location. For example, in a cellular automaton simulation of the diffusion of heat through a metal bar, a cell can contain that small square's average temperature in degrees Celsius. In a simulation involving a landscape, a cell might contain a moisture, nutrient, or veg-

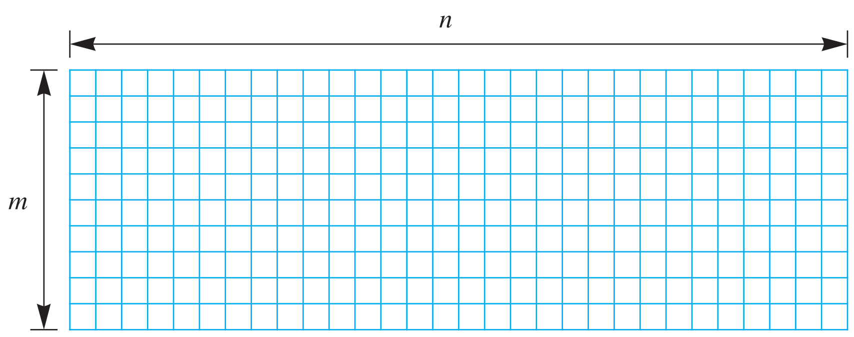

**Figure 10.2.1** Cells to model area

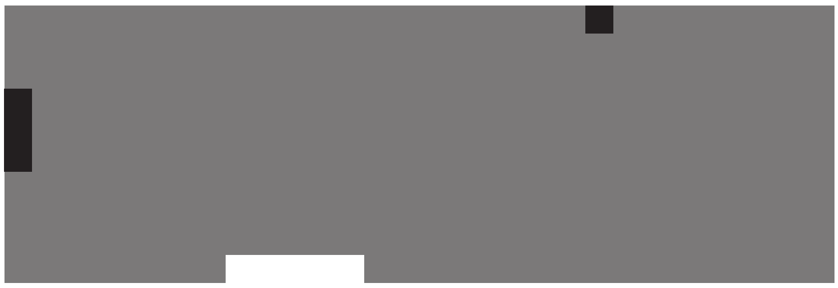

**Figure 10.2.2** Initialized metal bar with black representing hot, white cold, and gray an intermediate temperature

etation level from 0.0 to 1.0. We can use a similar gradient to indicate the amount of pollution spreading through a lake.

In the case of heat diffusing through a thin metal bar, we might initialize each cell to be some ambient temperature, say *AMBIENT* = 25 °C, except for hot and cold spots, which might have the values *HOT* = 50 °C and *COLD* = 0 °C, respectively. The following algorithm ***initBar*** initializes the grid for such a bar with two hot spots—a larger one in the middle of the first column and a smaller one three-fourths the way on the first row—and one fairly large cold spot one-third of the way along the last row (Cunningham 2007). Because the hot and cold spots are always present, we define a function, ***applyHotCold***, which we can call elsewhere, to assign the values *HOT* and *COLD* to appropriate cells in a bar. Using black to represent *HOT*, white for *COLD*, and a proportional shade of gray for temperatures between these values, Figure 10.2.2 illustrates the top 2D surface of such an initialized bar with a $10 \times 30$ grid.

***initBar*(*m*, *n*, *hotSites*, *coldSites*)**

Function to return an $m \times n$ grid of temperatures: Cells with coordinates in *hotSites* have the value *HOT*; cells with coordinates in *coldSites* have the value *COLD*; and all other cells have the value *AMBIENT*

***Pre*:** *m* and *n* are positive integers.
*hotSites* and *coldSites* are lists of coordinates for hot and cold sites, respectively.
*AMBIENT*, *HOT*, and *COLD* are global constants, and $COLD \le AMBIENT \le HOT$.
***Post:*** An $m \times n$ grid of values as described before has been returned.
***Algorithm:***

```
ambientBar ← m by n matrix of AMBIENT values
return applyHotCold(ambientBar, hotSites, coldSites)
```

***applyHotCold*(*bar*, *hotSites*, *coldSites*)**

Function to accept a grid of temperatures and to return a grid with heat and cold applied at *hotSites* and *coldSites*, respectively

```
Pre:  bar is a grid of values.
      hotSites and coldSites are lists of coordinates inside the grid for hot
      and cold sites, respectively.
      AMBIENT, HOT, and COLD are global constants, and $COLD \le AMBIENT \le HOT$.
Post: A grid of values as described above has been returned.
Algorithm:
      newBar ← bar
      assign HOT to every newBar cell with coordinates in hotSites
      assign COLD to every newBar cell with coordinates in coldSites
      return newBar
```

### Quick Review Question 1

From the text's website, download your computational tool's *10_2QRQ.pdf* file for this system-dependent question on initializing the grid.

## Heat Diffusion

At each simulation iteration, we apply a function, ***diffusion***, to each cell site to determine its temperature at the next time step. The cell's value at the next instant depends on the cell's current value (*site*) and the values of its four or eight nearest **neighbors**, as in Figure 10.2.3. The four neighbors along with the site itself in Figure 10.2.3a comprise the **von Neumann neighborhood** of a site, while the nine nodes in Figure 10.2.3b form the **Moore neighborhood** of a site. For diffusion of heat through a metal bar, we employ Moore neighborhoods.

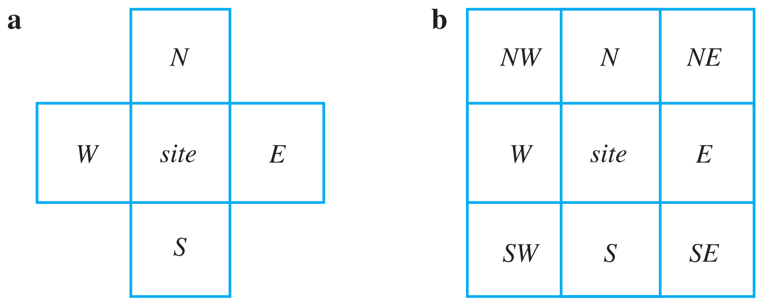

**Figure 10.2.3** Cells that determine a site's next value

**Definitions** In a two-dimensional grid, the **von Neumann neighborhood** of a site is the set of cells directly to the north, east, south, and west of the site and the site itself. As well as these five cells, the **Moore neighborhood** of a site includes the corner cells to the northeast, southeast, southwest, and northwest of the site. The four or eight neighborhood cells not including the site are the site's **neighbors**.

We base our model of diffusion on **Newton's law of heating and cooling**, which states that the rate of change of the temperature with respect to time of an object is proportional to the difference between the temperature of the object and the temperature of its surroundings. Similarly, we can say that the change in a cell's temperature, $\Delta site$, from time $t$ to time $t + \Delta t$ is a **diffusion rate parameter** ($r$) times the sum of each difference in the temperature of a neighbor ($neighbor_i$) and the cell's temperature ($site$), as follows:

$$\Delta site = r\sum_{i=1}^{8} (neighbor_i - site), \text{ where } 0 < r < 1/8 = 0.125$$

Thus, the site's temperature at time $t + \Delta t$ is the following:

$$site + \Delta site = site + r\sum_{i=1}^{8} (neighbor_i - site)$$

where $0 < r < 0.125$ and the sum is over the eight neighbors. With subtraction of $r \cdot site$ occurring 8 times, the formula simplifies to the following weighted sum of temperatures of the cell and its neighbors:

$$site + \Delta site = (1 - 8r)site + r\sum_{i=1}^{8} neighbor_i, \text{ where } 0 < r < 0.125$$

Similar diffusion formulas, which we explore in the projects, can have smaller coefficients for the corners than for the north, east, south, and west neighbors. However, the sum of the coefficients, which are fractions or percentages, for each of the nine cells in the neighborhood should be 1.0, or 100%.

## Quick Review Question 2

Suppose the diffusion rate parameter is 0.1 and the temperatures in the cells are as in Figure 10.2.4. Calculate the temperature in the center cell at the next time step.

With diffusion rate (*diffusionRate*) and temperatures of a cell (*site*) and its eight neighbors (*N*, *NE*, *E*, *SE*, *S*, *SW*, *W*, *NW*) as parameters, the function *diffusion* computes and returns the new temperature for the cell.

| | | |
|---|---|---|
| 2 | 3 | 4 |
| 0 | 5 | 6 |
| 1 | 3 | 7 |

**Figure 10.2.4** Temperatures in a section of the grid for Quick Review Question 1

***diffusion(diffusionRate, site, N, NE, E, SE, S, SW, W, NW)***

Function to return the new temperature of a cell

```
Algorithm:
    return (1 - 8diffusionRate)site
        + diffusionRate·(N + NE + E + SE + S + SW + W + NW)
```

## Boundary Conditions

We must be able to apply the function *diffusion* to every grid point, such as in Figure 10.2.1, including those on the boundaries of the first and last rows and the first and last columns. However, the *diffusion* function has parameters for the grid point (*site*) and its neighbors (*N*, *NE*, *E*, *SE*, *S*, *SW*, *W*, *NW*). Thus, to apply *diffusion* we extend the boundaries by one cell in each direction, creating what we call **ghost cells**. Several choices exist for values in those cells:

- Give every extended boundary cell a constant value, such as 25. Thus, the boundary insulates. Figure 10.2.5 outlines an original square grid, which has white cells, with thick black lines, while the constant extension is in color. We call the situation where the boundary has a constant value an **absorbing boundary condition**. In the case of the diffusion of heat through a metal bar, the boundary is similar to the bar being placed in a well ventilated room at 25 °C.
- Give every extended boundary cell the value of its immediate neighbor. Thus, the values on the original first row occur again on the new first row of ghost cells. Similar situations occur on the last row and the first and last columns (Figure 10.2.6). Such immediate repetitions are called **reflecting,** or **reflective, boundary conditions**. In the case of the spread of temperature, the boundary tends to propagate the current local situation: The air in the room is still, and the air temperature around the bar tends to mimic the temperature of the bar.
- Wrap around the north-south values and the east-west values in a fashion similar to a donut, or torus. Extend the north boundary with a ghost row that is a copy of the original south boundary row, and extend the south boundary with a copy of the original north boundary row. Similarly, expand the column boundaries on the east and west sides. Thus, for a cell on the north boundary, its neighbor to the north is the corresponding cell to the south (Figure 10.2.7). Such conditions are called **periodic boundary conditions**. In the case of a simulation of heat diffusion, the area is a closed, continuous environment with the situation at one boundary effecting its opposite boundary cells.

In the application of heat diffusion, because we assume that the immediate surroundings are at the same temperatures as on the surface of the bar, we choose to employ reflecting boundary conditions to minimize the impact of the surroundings. In the beginning, we attach new first and last rows, as in Figure 10.2.8, by **concatenating**, or attaching, the original grid's first row, the original grid, and the last row to create a new lattice, *latNS*.

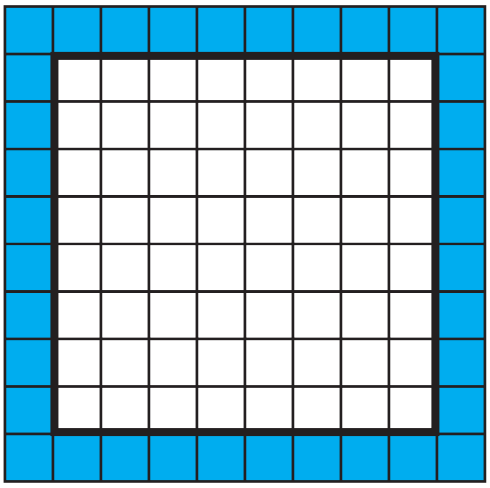

**Figure 10.2.5** Absorbing boundary conditions: Grid with extended boundaries and each ghost having a constant value

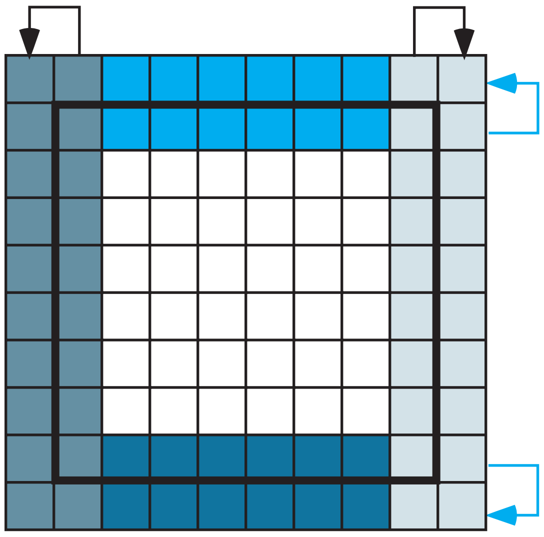

**Figure 10.2.6** Reflecting boundary conditions: Grid with extended boundaries and each ghost cell having the value of its immediate neighbor in the original grid

## Quick Review Question 3

Answer the following questions about Figure 10.2.4 as an extremely small entire thermal grid.

**a.** Give the size of the grid extended to accommodate boundary conditions.
**b.** Give the values in the first row of the extended matrix, assuming fixed boundary conditions with fixed value 0.
**c.** Give the values in the first row of the extended matrix, assuming reflecting boundary conditions, where we copy rows first.

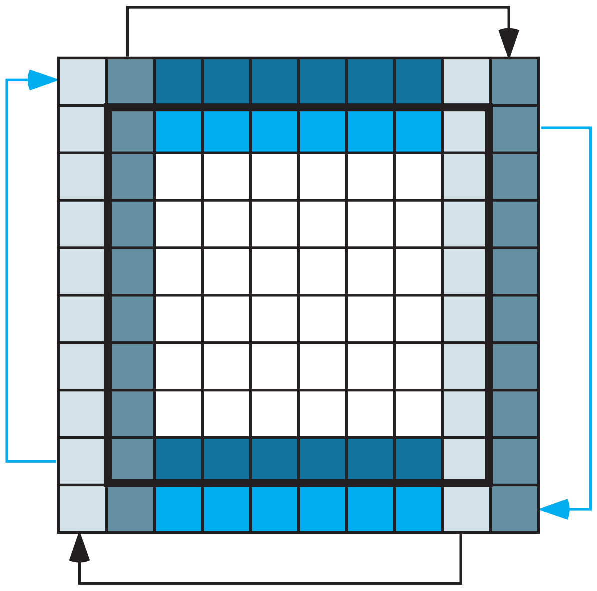

**Figure 10.2.7** Extended grid with periodic boundary conditions

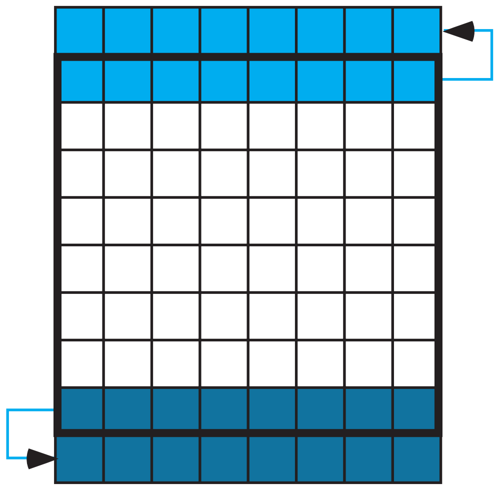

**Figure 10.2.8** Grid extended by having a new first row that is a copy of the first row on the original grid and having a new last row that is a copy of the last row on the original grid

**d.** Give the values in the first row of the extended matrix, assuming periodic boundary conditions, where we copy rows first.

## Quick Review Question 4

From the text's website, download your computational tool's *10_2QRQ.pdf* file for this system-dependent question that extends a grid as in Figure 10.2.8 by attaching a copy of the first row to the beginning and a copy of the last row to the end of the original grid to form a new grid, *latNS*.

To extend the grid with reflecting boundary conditions in the east and west directions, we concatenate the first column of *latNS* from Quick Review Question 4, *latNS*, and the last column of *latNS*. For some computational tools, it is easier to first transpose the lattice *latNS*, perform the same manipulation with the rows as in Quick Review Question 4, and then transpose the resulting lattice.

To consolidate these tasks, we define a function, ***reflectingLat***, using reflecting boundary conditions to extend by one cell in each direction the lattice. Pseudocode for the function follows.

***reflectingLat*(*lat*)**

Function to accept a grid and to return a grid extended one cell in each direction with reflecting boundary conditions

```
Pre:  lat is a grid.
Post: A grid extended one cell in each direction with reflecting boundary
      conditions was returned.
Algorithm:
      latNS ← concatenation of first row of lat, lat, and last row of lat
      return concatenation of first column of latNS, latNS, and last column
      of latNS
```

### Quick Review Question 5

From the text's website, download your computational tool's *10_2QRQ.pdf* file for this system-dependent question that extends a lattice, as in Figure 10.2.9.

## Applying a Function to Each Grid Point

After extending the grid by one cell in each direction using reflecting boundary conditions, we apply the function *diffusion* to each internal cell and then discard the boundary cells. We define a function, ***applyDiffusionExtended***, that takes an extended lattice, *latExt*, and returns the internal lattice with *diffusion* applied to each site. Figure 10.2.10 depicts an extended grid with the internal grid, which is a copy of the original lattice, in color. The number of rows of *latExt* is $m + 2$, while the number of columns is $n + 2$. As Figure 10.2.10 depicts, the number of rows ($m$) and columns ($n$) of the returned lattice is two less than the number of rows and columns of *latExt*, respectively. We apply the function *diffusion*, which has parameters *diffusionRate*, *site*, *N*, *NE*, *E*, *SE*, *S*, *SW*, *W*, and *NW*, to each internal cell in lattice *latExt*. If array indices in a computational tool begin with 0, these internal cells are in rows 1 through $m$ and columns 1 through $n$. For array indices that start with 1, the internal

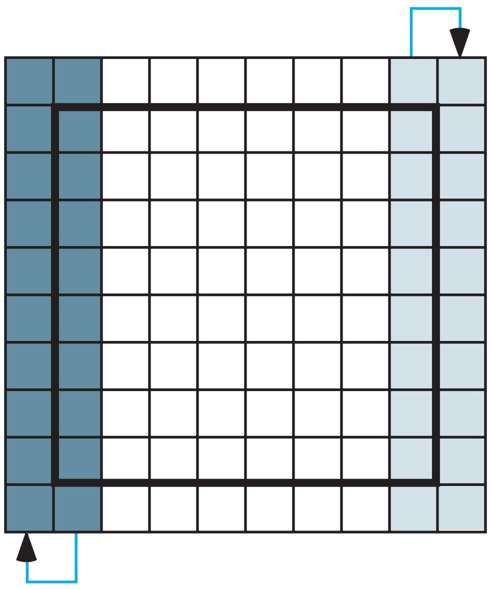

**Figure 10.2.9** Grid from Figure 10.2.8 expanded by having a new first column that is a copy of the first column and a new last column that is a copy of the last column

cells are in rows 2 through $m + 1$ and columns 2 through $n + 1$. We added the boundary rows and columns to eliminate different cases for cells with or without one or more neighbors. Thus, for $i$ going through the indices for the internal rows of the extended array and for $j$ going through the internal column indices, *applyDiffusion-Extended* obtains a value for each cell in a new $m \times n$ lattice by applying *diffusion* to each site with coordinates $i$ and $j$. The site's neighbors with corresponding coordinates are as in Figure 10.2.11.

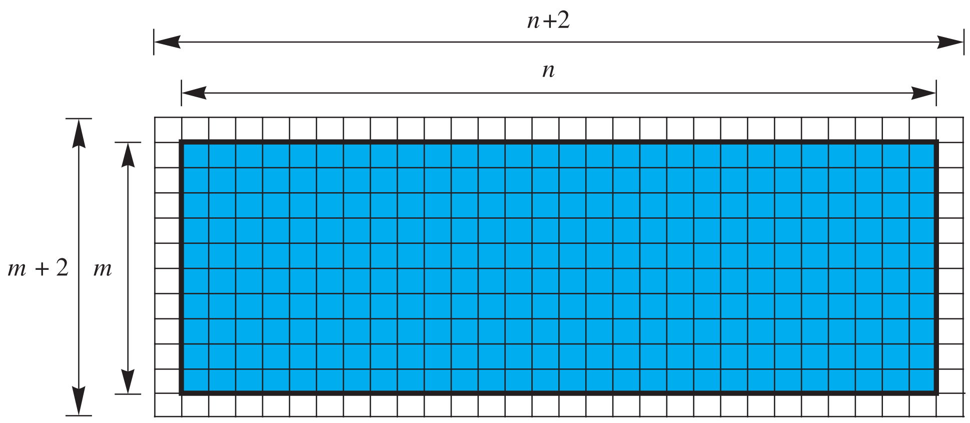

**Figure 10.2.10** Internal grid in color that is a copy of the original grid (Figure 10.2.1) embedded in an extended grid

| | | |
|---|---|---|
| *NW* $(i - 1, j - 1)$ | *N* $(i - 1, j)$ | *NE* $(i - 1, j + 1)$ |
| *W* $(i, j - 1)$ | *site* $(i, j)$ | *E* $(i, j + 1)$ |
| *SW* $(i + 1, j - 1)$ | *S* $(i + 1, j)$ | *SE* $(i + 1, j + 1)$ |

**Figure 10.2.11** Indices for a lattice site and its neighbors

### Quick Review Question 6

Suppose *extMat* is an extended matrix of size 97 × 62.

**a.** Give the size of the matrix *applyDiffusionExtended* returns.
**b.** When $i = 33$ and $j = 25$, give the indices of the site's neighbor to the north.
**c.** For this site, give the indices of its neighbor to the southwest.

### Quick Review Question 7

From the text's website, download your computational tool's *10_2QRQ.pdf* file for this system-dependent question that develops the function *applyDiffusionExtended*.

## Simulation Program

To perform the simulation of diffusion of heat through a metal bar, we define a function, ***diffusionSim***, with parameters *m* and *n*, the number of grid rows and columns, respectively; *diffusionRate*, the rate of diffusion; and *t*, the number of time steps. The function *diffusionSim* returns a list of the initial lattice and the next *t* lattices in the simulation. Pseudocode for *diffusionSim* is presented on the following page.

### Quick Review Question 8

From the text's website, download your computational tool's *10_2QRQ.pdf* file for this system-dependent question that implements the loop in the *diffusionSim* function.

***diffusionSim*(*m*, *n*, *diffusionRate*, *t*)**

Function to return a list of grids in a simulation of the diffusion of heat through a metal bar

***Pre:*** *m* and *n* are positive integers for the number of grid rows and columns, respectively.
*diffusionRate* is the rate of diffusion.
*t* is the number of time steps.
*diffusion* is a function to return a new temperature for a grid point.

***Post:*** A list of the initial grid and the grid at each time step of the simulation was returned.

***Algorithm:***

```
bar ← initBar(m, n, hotSites, coldSites)
grids ← list containing bar
do the following t times:
    barExtended ← reflectingLat(bar)
    bar ← applyDiffusionExtended(diffusionRate, barExtended)
    bar ← applyHotCold(bar, hotSites, coldSites)
    grids ← the list with bar appended onto the end of grids
return grids
```

## Display Simulation

Visualization helps us understand the meaning of the grids. For each lattice in the list returned by *diffusionSim*, we generate a graphic using grayscale or color. We define a function, ***animDiffusionGray***, with parameter *grids*, which is a list of lattices from the simulation, to produce a grayscale animation of the changing temperatures in the metal bar, with black representing the hottest locations and white the coldest. Starting with the initial bar from Figure 10.2.2 and a diffusion rate of 0.1 and displaying several frames of such an animation, Figure 10.2.12 shows that the bar quickly approaches equilibrium.

### Quick Review Question 9

From the text's website, download your computational tool's *10_2QRQ.pdf* file for this system-dependent question that develops the function *animDiffusionGray*, which produces a grayscale graphic corresponding to each simulation lattice in a list (*grids*).

For a color display, we should employ a coloration that is evocative of the situation, such as red for hot and blue for cold. For display on a monitor, we usually employ the **red-green-blue (RGB) color model**. In the RGB color model, we specify the amounts between 0.0 and 1.0 of red, green, and blue light at each **pixel**, or picture element, or point in the graphics. For our heated bar, we employ only red and blue

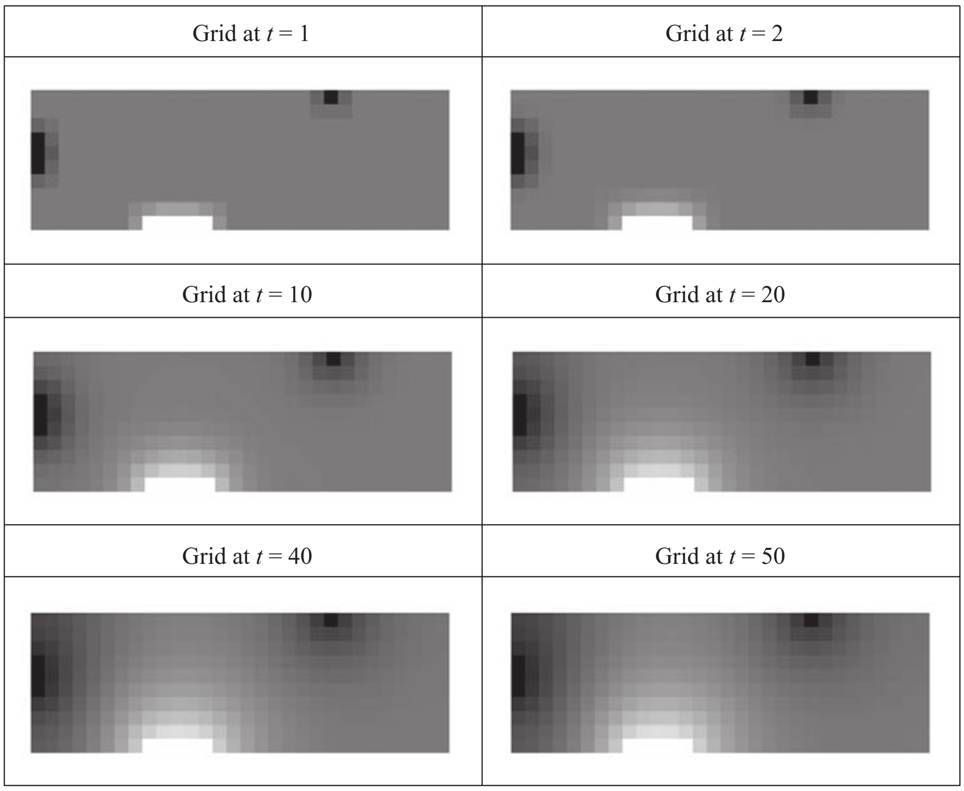

**Figure 10.2.12** Several frames in an animation in grayscale of the spreading of heat through a metal bar

light, so the level of green light is 0.0. In going from the coldest to the hottest values, red increases from 0.0 to 1.0, while blue decreases from 1.0 to 0.0. To obtain a zero-to-one scale with the minimum temperature being *COLD* = 0, we divide a cell's temperature, *temp*, by the maximum temperature, *HOT*, so that the amount of red light is *temp*/*HOT*, expressed as a floating-point number. If *HOT* is 50.0 and *temp* is 0.0, so is *temp*/*HOT* = 0.0/50.0 = 0.0; while if *temp* is 50.0, then *temp*/*HOT* is 1.0. To have the amount of blue light decrease as the temperature decreases, we subtract the fraction from 1.0. If *temp* is 0.0, then 1.0 – *temp*/*HOT* is 1.0; and if *temp* is 50.0, then 1.0 – *temp*/*HOT* is 0.0. Using a temperature scale from 0 °C to 50 °C, Table 10.2.1 gives several RGB for this scaling.

**Table 10.2.1**
Several RGB Color Model Values of the Amounts of Red and Blue for Temperatures from 0 °C to 50 °C

| **Temperature (°C)** | *0* | *10* | *25* | *40* | *50* |
|---|---|---|---|---|---|
| **red fraction (temperature/50)** | 0.0 | 0.2 | 0.5 | 0.8 | 1.0 |
| **blue fraction (1.0 – temperature/50)** | 1.0 | 0.8 | 0.5 | 0.2 | 0.0 |

### Quick Review Question 10

From the text's website, download your computational tool's *10_2QRQ.pdf* file for this system-dependent question that develops the function, ***animDiffusionColor***, which produces a color graphic corresponding to each simulation lattice in a list (*grids*).

## Exercises

*On the text's website,* Diffusion *files for several computational tools contain the code for the simulation of the module. Complete the following exercises using your computational tool.*

1. Write a function to extend a grid using absorbing boundary conditions with the constant value on the boundary being 25.
2. Write a function to extend a grid using periodic boundary conditions.

## Projects

*On the text's website,* Diffusion *files for several computational tools contain the code for the simulation of the module. Complete the following projects using your computational tool.*

*For an additional project, see Project 4 from Module 13.4, "Probable Cause—Modeling with Markov Chains."*

1. **a.** Determine how long it takes, *t*, for the bar modeled in this module to reach equilibrium, where from time *t* to time *t* + 1 the values in each cell vary by no more than plus or minus some small value, such as ±0.001.
   **b.** Repeat Part a, applying heat and cold for 10 time steps and then removing such heating and cooling.
2. Develop simulations and animations for the bar modeled in this module using several boundary conditions: three simulations of absorbing boundary conditions with constant values 0, 25, and 50 and periodic boundary conditions. Along with the reflecting boundary conditions, describe the results. Discuss the advantages and disadvantages of each approach and the situations, such as heat or pollution diffusion, for which each is most appropriate.
3. Instead of using the formula for diffusion in the section "Heat Diffusion," employ the filter in Figure 10.2.13. Thus, to obtain the value at a site for time *t* + 1, we add 25% of the site's temperature at time *t*, 12.5% of the north, east, south, and west cells at time *t*, and 6.25% of the corner cells to the northeast, southeast, southwest, and northwest. This sum is called a **weighted sum** with each nutrition value carrying a particular weight as indicated by the table. Revise the model using this configuration and compare the results with that of the module.
4. **a.** Model a bar at 100 °C that has a constant application of a 25 °C external source on its boundary. Generate plots of the temperatures at a corner

| | | |
|---|---|---|
| 0.0625 | 0.125 | 0.0625 |
| 0.125 | 0.25 | 0.125 |
| 0.0625 | 0.125 | 0.0625 |

**Figure 10.2.13** Filter for Project 3

and in the middle of the bar versus time. Describe the shapes of the graphs.

**b.** Repeat Part a with the bar being at –50 °C.

**c.** Discuss the results.

**5.** Consider a small, shallow body of water that initially has a constant amount of nutrient. A cypress toward one edge of the water consumes nutrients at a constant rate, so that at each time step the amount of nutrients in the corresponding cell decreases by a fixed amount. Suppose shore is on three sides and a larger body of water is on the fourth side. Nutrients from the larger body of water diffuse into the smaller area. Model and visualize the situation for the small body of water. Find a rate of diffusion and a rate of nutrient consumption so that the tree always has nourishment. Use the formula for diffusion in the section "Heat Diffusion" or the filter variation in Project 3.

**6.** Suppose an industry constantly spills pollutants into a containment pond, which initially has water. Using a diffusion rate of 0.1, how long will it take for the concentration of pollutants in the middle of the pond to reach 25%? Give your assumptions and discuss the results.

**7.** Model and visualize a situation in which diffusion tends to occur more in one direction than another, say more from the east than from the west. Thus, design a filter similar to that in Project 3 that favors directional diffusion. Such a configuration could be used in modeling diffusion on the surface of flowing water. Give your assumptions and discuss the results.

**8.** Suppose a dye is dissolved in water, which is poured on top of a gel. Model and visualize a cross section of the diffusion of the dye into the gel. Compare your results with the time-lapsed video at (Wikipedia Contributors, "Diffusion"). For your parameters, determine $t$ to match the diffusion time in the video.

**9.** Often because of imperfections, variations in media, or other factors, diffusion does not proceed deterministically but varies slightly with an element of chance. Revise the function *diffusion*, which the section "Heat Diffusion" describes, to be stochastic. Instead of multiplying each $neighbor_i$ by $r$, the rate of diffusion, multiply each neighboring temperature by a different $(1 + rnd_i)r$, where $rnd_i$ is a normally distributed random number with mean 0 and standard deviation 0.5. Adjust the coefficient of *site* so that the sum of all the coefficients is 1. Run the model 100 times for 20 time steps and determine the mean and range of temperatures for a designated cell towards the middle of the bar.

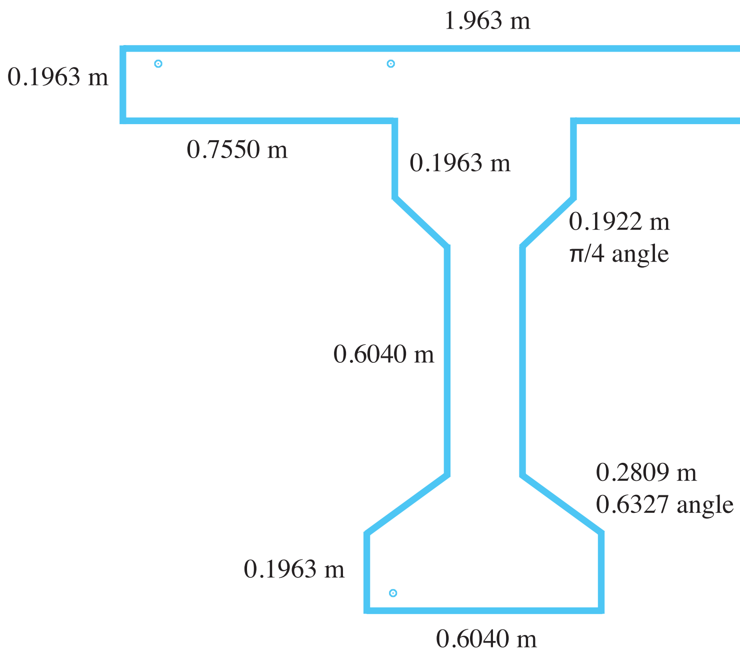

**Figure 10.2.14** Cross section of a bridge support, based on Podroužek (2008)

**10.** Repeat Project 9 using the filter described in Project 3.

**11.** Application of deicing salts in the winter can degrade concrete reinforced structures, such as bridges, because of the ingress of harmful substances such as chloride ions. Engineers incase steel in concrete to protect against corrosion. However, when the concentration of chloride reaches a critical concentration, perhaps 0.4% $Cl^-$ per unit of concrete content, the concrete no longer can protect the steel. Develop a cellular automaton simulation of the diffusion of chloride in a T-shaped cross section of a bridge support, as in Figure 10.2.14. Assume deicing salts can seep into the structure from all surfaces except the top. Referring to Project 9, employ stochastic diffusion with a basic diffusion rate of 0.125 and von Neumann neighborhoods. For 30 years of constant exposure, apply a 2% per unit chloride ion concentration from the salt to all external surfaces except the upper surface. Have the basic time step be 165 days. Small circles indicate where reinforcing bars intersect the T cross section. Averaging the results for many simulations, say 100 or 1000, determine the chloride ion concentration at the locations for these reinforcing bars after 30 years of continuous exposure (Podroužek 2008).

**12.** Model in 3D the diffusion of heat through a bar. Assume that the bar is sitting on a table in a room with good circulation. The part of the table on which the bar rests has approximately the same temperatures as the corresponding locations on the bottom of the bar, but the air around the bar remains almost constantly 25 °C.

## Answers to Quick Review Question

From the text's website, download your computational tool's *10_2QRQ.pdf* file for answers to the system-dependent questions.

**2.** 3.6 = (1 – 8 × 0.1)(5) + 0.1(2 + 3 + 4 + 0 + 6 + 1 + 3 + 7)
**3.** **a.** 5 × 5
**b.** 0, 0, 0, 0, 0
**c.** 2, 2, 3, 4, 4
**d.** 7, 1, 3, 7, 1
**5.** **a.** 95 × 60
**b.** (32, 25)
**c.** (34, 24)

## References

Cunningham, Steve. 2007. *Computer Graphics: Programming in OpenGL for Visual Communication*, Upper Saddle River, NJ: Prentice-Hall.
NASA Spinoff. 1988. "Spinoff from Mooncraft Technology." NASA.
NASA Spinoff. 2011. "Fire-Resistant Reinforcement Makes Steel Structures Sturdier." http://spinoff.nasa.gov/Spinoff2006/ps_3.html (accessed June 15 2012)
Podroužek, Jan, and Břetislav Teplý. 2008. "Modelling of Chloride Transport in Concrete by Cellular Automata." *Engineering Mechanics*, (15)3: 213–222.
Wikipedia Contributors, "Diffusion," *Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/wiki/Diffusion (accessed June 13, 2012)
Wikipedia Contributors, "Heat Transfer," *Wikipedia, The Free Encyclopedia*, http://en.wikipedia.org/wiki/Heat_transfer (accessed June 13, 2012)