```
  public void handleMouseAction(InteractivePanel panel,
      MouseEvent evt) {
    switch(panel.getMouseAction()) {
    case InteractivePanel.MOUSE_DRAGGED:
    case InteractivePanel.MOUSE_PRESSED:
       double x = panel.getMouseX(); // mouse x in world units
       double y = panel.getMouseY();
       int i = frame.xToIndex(x);    // closest array index
       int j = frame.yToIndex(y);
       frame.setMessage("V="+decimalFormat.format(potential[i][j]));
       break;
    case InteractivePanel.MOUSE_RELEASED:
       panel.setMessage(null);
       break;
    }
  }

  public static void main(String[] args) {
     SimulationControl.createApp(new LaplaceApp());
  }
}
```

As the algorithm loops through the grid sites, it first checks if each grid site is a conductor. If it is, the site is skipped. If not, a new potential is calculated and assigned to the proper element in the `potential` array. A local variable named `maximumError` keeps track of the maximum difference between the potential at a site and the average potential of the four neighbors. This variable is used to determine the end of the simulation.

In Problems 10.10–10.12 you are asked to modify `LaplaceApp` to compute the potential for various geometries.

Problem 10.10 Numerical solution of the potential within a rectangular region

(a) Modify `LaplaceApp` to determine the potential $V(x, y)$ in a square region with linear dimension $L = 10$. The boundary of the square is at a potential $V = 10$. Choose the grid size $\Delta x = \Delta y = 1$. Before you run the program, guess the exact form of $V(x, y)$ and set the initial values of the interior potential 10% lower than the exact answer. How many iterations are necessary to achieve 1% accuracy? Decrease the grid size by a factor of two and determine the number of iterations that are now necessary to achieve 1% accuracy.

(b) Consider the same geometry as in part (a), but set the initial potential at the interior sites equal to zero except for the center site whose potential is set equal to four. Does the potential distribution evolve to the same values as in part (a)? What is the effect of a poor initial guess? Are the final results independent of your initial guess?

(c) Modify `LaplaceApp` so that the value of the potential at the four sides is 5, 10, 5, and 10, respectively (see Figure 10.1). Sketch the equipotential surfaces. What happens if the potential is 10 on three sides and 0 on the fourth? Start with a reasonable guess for the initial values of the potential at the interior sites and iterate until 1% accuracy is obtained.

*(d) Consider the same initial choice of the potential as in part (b) and focus your attention on the potential at the sites near the center of the square. If the central site has an initial potential of four, what is the potential at the nearest neighbor sites after the first iteration? Follow the distribution of the potential as a function of the number of iterations and verify that the nature of the relaxation of the potential to its correct distribution is closely related to *diffusion* (see Chapter 7). It may be helpful to increase the number of sites in the grid and the initial value of the potential at the central site to see the nature of the relaxation more clearly. ■

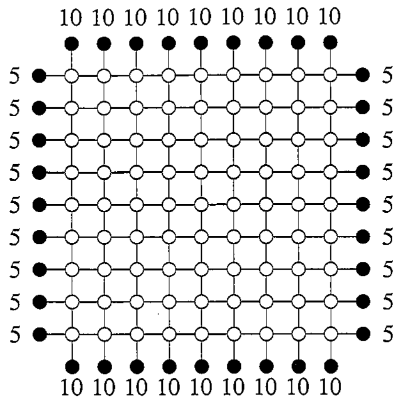

Figure 10.1 Potential distribution considered in Problem 10.10c. The number of interior sites in each direction is nine.

In Problem 10.10, we implemented a simple version of the relaxation method known as the Jacobi method. In particular, the new potential at each site is based on the values of the potentials at the neighboring sites at the previous iteration. After the entire lattice is visited, the potential at each site is updated *simultaneously*. The difficulty with this relaxation method is that it converges very slowly. The use of more general relaxation methods is discussed in many texts (cf. Sadiku or Press et al.). In Problem 10.11 we consider a method known as *Gauss–Seidel* relaxation.

Problem 10.11 Gauss–Seidel relaxation

(a) Modify the program that you used in Problem 10.10 so that the potential at each site is updated sequentially. That is, after the average potential of the nearest neighbor sites of site i is computed, update the potential at i immediately. In this way the new potential of the next site is computed using the most recently computed values of its nearest neighbor potentials. Are your results better, worse, or about the same as for the simple relaxation method?

(b) Imagine coloring the alternate sites of a grid red and black so that the grid resembles a checkerboard. Modify the program so that all the red sites are updated first, and then all the black sites are updated. This ordering is repeated for each iteration. Do your results converge any more quickly than in part (a)?

*(c) The slow convergence of the relaxation methods we have explored is due to the fact that it takes a long time for a change in the potential at one site to effect changes further away. We can improve the Gauss–Seidel method by using an *overrelaxation* method that updates the new potential as follows:

$$V_{\text{new}}(x, y) = w V_{\text{ave}}(x, y) + (1 - w) V(x, y), \tag{10.15}$$