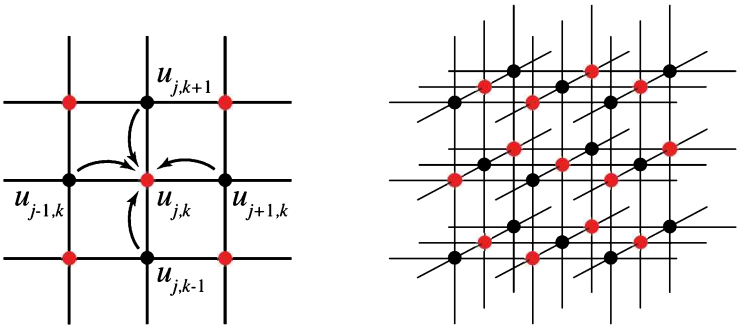

Figure 36.1. *Examples of a checkerboard red/black ordering for two dimensions (left) and three dimensions (right). The grid points with the same color can be updated in parallel without conflict.*

significantly improve the convergence rate of the method, so it is worthwhile to carefully choose ω either through various estimation techniques or by experimentation when developing a high-speed solver.

It should also be noted that for the SOR method, the order in which the points are updated can be changed. The sequential algorithm described above is difficult to parallelize, but a reorganization using a checkerboard pattern can be useful for the case of solving Equation (36.8). Figure 36.1 illustrates the pattern used for parallelizing the SOR method in two and three dimensions using red/black ordering. In this framework, the grid points of the same color can be updated in parallel without conflicting with each other because of interdependencies. Thus, the SOR method updates all the red grid points in one step, and then updates all the black grid points in the second step.

36.1 ▪ Problems to Solve

36.1.1 ▪ Diffusion with Sources/Sinks

Consider the elliptic equation

$$\nabla^2 u = \frac{10\lambda}{\sqrt{\pi}} e^{-\lambda^2((x-1)^2+y^2)} - \frac{10\lambda}{\sqrt{\pi}} e^{-\lambda^2((x+1)^2+y^2)}, \tag{36.10}$$

$$\frac{\partial u}{\partial x}(\pm 2, y) = 0,$$

$$u(x, \pm 1) = 0,$$

in the domain $-2 \leq x \leq 2$, $-1 \leq y \leq 1$ for the parameter value $\lambda = 100$. Here, the two terms represent approximations of a Dirac delta point source of strength 10 at $(-1, 0)$ and a sink at $(1, 0)$. This is a model of a plate with two sides held at a fixed temperature, and the other two sides insulated, and a source and sink of heat. The value of λ controls how narrow or wide the approximate Dirac delta is spread, with larger values being more narrow, and approaching the true Dirac delta as $\lambda \to \infty$.

Assignment. Solve Equation (36.10) using the SOR method with an error tolerance of 10^{-9}. Your program should take four arguments so that your program named `diffusion` can be executed like this:

```
$ diffusion 128 1.8 1e-9 10000
```

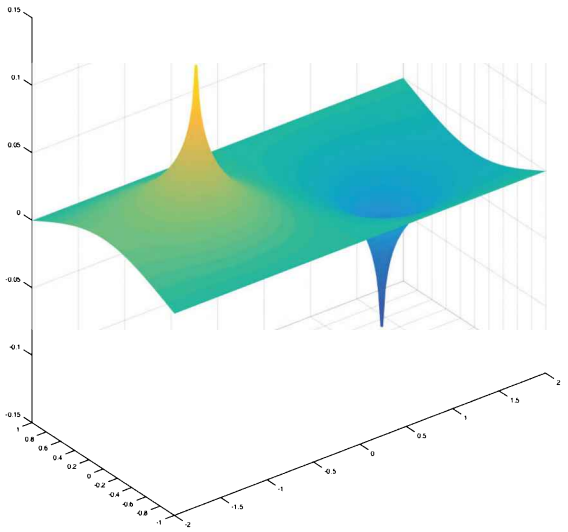

Figure 36.2. *Solution for Equation (36.10) when* $\lambda = 100$. *Note that the vertical scale is exaggerated compared to the horizontal scale.*

where the number of grid points in the y-direction is $N = 128$, the double precision parameter is $\omega = 1.8$, the convergence tolerance is $\tau = 10^{-9}$, and a fail-safe stopping criterion for the maximum number of iterations is $K = 10000$. The domain is rectangular, so for the input value of N in the y-direction, this will lead to $\Delta y = \frac{2}{N-1}$. To make $\Delta x = \Delta y$, take the number of grid points in the x-direction to be $M = 2N - 1$. The fail-safe stopping criterion should be set up so that if the number of iterations exceeds K, then the calculation terminates with a warning message that it failed to converge and what the current maximum error is.

Write the final result for `u` to the file named "`Sources.out`", which must be the $2N - 1 \times N$ solution.

Experiment with various values of the relaxation factor ω to determine the value that requires the fewest iterations to meet the error tolerance.

Your program should also print to the screen the total time required to complete the calculation divided by the number of iterations used. Vary the values of N and generate a plot that compares the time to complete one iteration versus N.

The solution for the case of $\lambda = 100$ is shown in Figure 36.2.

36.1.2 ▪ Stokes Flow

Stokes flow is an approximation for incompressible fluid flow when the flow is highly viscous or slow moving. It is a linearization of the more general Navier–Stokes equations for modeling fluid flow. The equations for Stokes flow are

$$\mu \nabla^2 \mathbf{u} = \nabla p - \mathbf{f}, \tag{36.11}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{36.12}$$