

of the inner and outer products

$$\begin{aligned}
 (\mathbf{A} + \vec{u} \otimes \vec{v})^{-1} &= (\mathbf{A} \cdot (\mathbf{1} + \mathbf{A}^{-1} \cdot \vec{u} \otimes \vec{v}))^{-1} \\
 &= (\mathbf{1} + \mathbf{A}^{-1} \cdot \vec{u} \otimes \vec{v})^{-1} \cdot \mathbf{A}^{-1} \\
 &= [\mathbf{1} - (\mathbf{A}^{-1} \cdot \vec{u} \otimes \vec{v}) + \\
 &\quad (\mathbf{A}^{-1} \cdot \vec{u} \otimes \vec{v}) \cdot (\mathbf{A}^{-1} \cdot \vec{u} \otimes \vec{v}) - \dots] \cdot \mathbf{A}^{-1} \\
 &= [\mathbf{1} - (\mathbf{A}^{-1} \cdot \vec{u} \otimes \vec{v}) + \\
 &\quad \mathbf{A}^{-1} \cdot \vec{u} \otimes \underbrace{(\vec{v} \cdot \mathbf{A}^{-1} \cdot \vec{u})}_{\beta} \otimes \vec{v} - \dots] \cdot \mathbf{A}^{-1} \\
 &= [\mathbf{1} - (\mathbf{A}^{-1} \cdot \vec{u} \otimes \vec{v})(1 - \beta + \beta^2 - \dots)] \cdot \mathbf{A}^{-1} \\
 &= \mathbf{A}^{-1} - \frac{(\mathbf{A}^{-1} \cdot \vec{u}) \otimes (\vec{v} \cdot \mathbf{A}^{-1})}{1 + \beta} \quad (7.50)
 \end{aligned}$$

Matrix (7.48) is banded-diagonal, having diagonal rows of 1s N elements on either side of the diagonal, and in higher dimensions there will be more bands. Although these are sparse matrices and it can be possible to solve them directly (see [Press *et al.*, 1992] for techniques), the effort required in two or more dimensions can quickly become prohibitive.

One alternative that is applicable for constant-coefficient linear problems is *Fourier Transform* methods. In 2D, the discrete Fourier transform of the field is

$$\hat{u}_{m,n} = \sum_{j=0}^{M-1} \sum_{k=0}^{N-1} u_{j,k} e^{2\pi i m j / M} e^{2\pi i n k / N} \quad (7.51)$$

and the inverse transform is

$$u_{j,k} = \frac{1}{N^2} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \hat{u}_{m,n} e^{-2\pi i j m / M} e^{-2\pi i k n / N} \quad (7.52)$$

Plugging the transforms of u and ρ into equation (7.46), and recognizing that the transform of a function can vanish everywhere only if the function itself is equal to zero, gives

$$\begin{aligned}
 \hat{u}_{m,n} (e^{2\pi i m / M} + e^{-2\pi i m / N} + e^{2\pi i n / M} + e^{-2\pi i n / N} - 4) \\
 = \hat{\rho}_{m,n} (\Delta x)^2 \quad (7.53)
 \end{aligned}$$

Rearranging terms and simplifying the complex exponentials,

$$\hat{u}_{m,n} = \frac{\hat{\rho}_{m,n} (\Delta x)^2}{2 \cos \frac{2\pi m}{M} + 2 \cos \frac{2\pi n}{N} - 4} \quad (7.54)$$

Therefore, the forward transform of the source term can be calculated, this can be used to find the \hat{u} , and then the inverse transform can be taken to find u . This solution imposes periodic boundary conditions; some other boundary conditions can be imposed by choosing the form of the expansion (for example, using only sines if the solution vanishes on the boundary).

The Fourier t
lems with bound
mation process is
found by remem
problem (whethe

This means that
with the asympt
The simplest

$$u_{j,k}^{n+1} = u_{j,k}^n +$$

Here the time s
possible step tha
1/4, leading to

$$u_j^i$$

This has a very
each lattice site
process is repea
related algorith
available

(assuming that

These both v
seen by rewritin

and then separa

For the Jacobi
hand side:

The convergen
($\mathbf{L} + \mathbf{U}$). The
the largest eige
called the *spec*
radius is asymp

The Fourier transform method is so simple only for linear constant-coefficient problems with boundary conditions along the coordinate axes; some kind of iterative approximation process is needed for more general problems. An important class of techniques is found by remembering that Poisson's equation is the steady-state solution of a diffusion problem (whether or not it originally arises from diffusion)

$$\frac{\partial u}{\partial t} = \nabla^2 u - \rho \quad (7.55)$$

This means that all of the techniques for solving diffusion problems can be applied here, with the asymptotic answer giving the solution to Poisson's equation.

The simplest is *Jacobi's method*, which just takes forward time differences

$$u_{j,k}^{n+1} = u_{j,k}^n + \frac{\Delta t}{(\Delta x)^2} (u_{j+1,k}^n + u_{j-1,k}^n + u_{j,k+1}^n + u_{j,k-1}^n - 4u_{j,k}^n) - \Delta t \rho_{j,k} \quad (7.56)$$

Here the time step does not have any physical significance; we just want the largest possible step that converges to the solution. In 2D the Courant condition is $\Delta t / (\Delta x)^2 \leq 1/4$, leading to

$$u_{j,k}^{n+1} = \frac{1}{4} (u_{j+1,k}^n + u_{j-1,k}^n + u_{j,k+1}^n + u_{j,k-1}^n) - \frac{(\Delta x)^2}{4} \rho_{j,k} \quad (7.57)$$

This has a very natural interpretation: starting from a random guess, at each time step each lattice site is set to the average of its neighbors (and a source term is added). This process is repeated until the solution stops changing, a technique called *relaxation*. A related algorithm, the *Gauss-Seidel* method, uses updated values as soon as they become available

$$u_{j,k}^{n+1} = \frac{1}{4} (u_{j+1,k}^n + u_{j-1,k}^{n+1} + u_{j,k+1}^n + u_{j,k-1}^{n+1}) - \frac{(\Delta x)^2}{4} \rho_{j,k} \quad (7.58)$$

(assuming that the updating proceeds down rows).

These both work, but the convergence is too slow for them to be useful. This can be seen by rewriting them in terms of the matrix problem

$$\mathbf{A} \cdot \vec{u} = \vec{\rho} \quad (7.59)$$

and then separating \mathbf{A} into lower-triangular, diagonal, and upper-triangular parts

$$(\mathbf{L} + \mathbf{D} + \mathbf{U}) \cdot \vec{u} = \vec{\rho} \quad (7.60)$$

For the Jacobi method, the lower- and upper-triangular parts are moved to the right hand side:

$$\mathbf{D} \cdot \vec{u}_{n+1} = -(\mathbf{L} + \mathbf{U}) \cdot \vec{u}_n + \vec{\rho} \quad (7.61)$$

The convergence rate will be determined by the eigenvalues of the iteration matrix $-\mathbf{D}^{-1}(\mathbf{L} + \mathbf{U})$. The magnitude of all of the eigenvalues must be less than 1 for stability, and the largest eigenvalue determines the overall convergence rate (the largest eigenvalue is called the *spectral radius* ρ_s). For a large $N \times N$ square lattice problem, the spectral radius is asymptotically equal to [Ames, 1992]

$$\rho_{\text{Jacobi}} \simeq 1 - \frac{\pi^2}{2N^2} \quad (7.62)$$

Therefore, reducing the error by a factor of 10 requires $-\ln 10 / \ln \rho_s \simeq N^2$ steps. In the Gauss-Seidel method, the lower-triangular part is moved over to the left side:

$$(\mathbf{L} + \mathbf{D}) \cdot \vec{u}_{n+1} = -\mathbf{U} \cdot \vec{u}_n + \vec{\rho} \quad (7.63)$$

For the square 2D lattice, this has a spectral radius of

$$\rho_{\text{Gauss-Seidel}} \simeq 1 - \frac{\pi^2}{N^2} \quad (7.64)$$

and so the number of steps needed to reduce the error by a factor of 10 is half that required by the Jacobi method. For a 100×100 lattice, both of these methods require $\sim 10^4$ steps for an improvement of a factor of 10 in the answer, which is usually prohibitive. The Gauss-Seidel method is preferable to the Jacobi method because it converges faster and does not require auxiliary storage, but something better than both is needed.

The Gauss-Seidel method can be rewritten in a suggestive form as follows:

$$\begin{aligned} (\mathbf{L} + \mathbf{D}) \cdot \vec{u}_{n+1} &= -\mathbf{U} \cdot \vec{u}_n + \vec{\rho} \\ \vec{u}_{n+1} &= (\mathbf{L} + \mathbf{D})^{-1} [-\mathbf{U} \cdot \vec{u}_n + \vec{\rho}] \\ &= \vec{u}_n - (\mathbf{L} + \mathbf{D})^{-1} \cdot [\mathbf{U} \cdot \vec{u}_n - \vec{\rho}] - \vec{u}_n \\ &= \vec{u}_n - (\mathbf{L} + \mathbf{D})^{-1} \cdot [(\mathbf{L} + \mathbf{D} + \mathbf{U}) \cdot \vec{u}_n - \vec{\rho}] \\ &= \vec{u}_n - (\mathbf{L} + \mathbf{D})^{-1} \cdot [\mathbf{A} \cdot \vec{u}_n - \vec{\rho}] \\ &= \vec{u}_n - (\mathbf{L} + \mathbf{D})^{-1} \cdot \vec{E}_n \quad (7.65) \end{aligned}$$

where \vec{E}_n is the error at the n th time step. In each update, the error gets multiplied by $(\mathbf{L} + \mathbf{D})^{-1}$ and subtracted from the state. The idea of *Successive Over-Relaxation (SOR)* is to extrapolate this correction and subtract a larger change

$$\vec{u}_{n+1} = \vec{u}_n - \alpha(\mathbf{L} + \mathbf{D})^{-1} \cdot \vec{E}_n \quad (7.66)$$

It can be shown that this converges for $0 < \alpha < 2$ [Ames, 1992]. When $\alpha = 1$ this is just the Gauss-Seidel method, $\alpha < 1$ is underrelaxation (which slows the convergence), and $1 < \alpha < 2$ is overrelaxation. The convergence rate depends on the value of α ; choosing a value that is too large is as bad as choosing one that is too small because the solution will overshoot the final value. The optimal relaxation rate is

$$\alpha = \frac{2}{1 + \sqrt{1 - \rho_{\text{Jacobi}}^2}} \quad (7.67)$$

which leads to an asymptotic spectral radius of

$$\rho_{\text{SOR}} \simeq 1 - \frac{2\pi}{N} \quad (7.68)$$

This reduces the number of steps needed to reduce the error by a factor of 10 to $\mathcal{O}(N)$, which is now proportional to the grid size rather than the square of the grid size. Written out in components, for the 2D problem SOR is

$$u_{j,k}^{n+1} = (1 - \alpha)u_{j,k}^n + \frac{\alpha}{4}(u_{j+1,k}^n + u_{j-1,k}^n + u_{j,k+1}^n + u_{j,k-1}^n) - \frac{\alpha(\Delta x)^2}{4}\rho_{j,k} \quad (7.69)$$

SOR is very easy (although this can be late). An alternative at roughly the same techniques have been the final solution of coarse-graining then interpolating to the Richardson they are more com

[Ames, 1992] An Equat Ames emph

(7.1) Consider

- (a) Write
- (b) What
- (c) Use t
- (d) Use t (hint:
- (e) Do d
- (f) Num with
- (g) If th

assu

and on t (h) Rep diti

$\geq N^2$ steps. In left side:

(7.63)

(7.64)

if that required time $\sim 10^4$ steps prohibitive. The erges faster and ded.

flows:

]

(7.65)

gets multiplied over-Relaxation

(7.66)

$\alpha = 1$ this is just onvergence), and ie of α ; choosing ause the solution

(7.67)

(7.68)

or of 10 to $\mathcal{O}(N)$, grid size. Written

$\frac{x)^2}{\rho_{j,k}}$ (7.69)

SOR is very easy to program, but does require determining the relaxation parameter α (although this can be estimated empirically, since if α is too large the solution will oscillate). An alternative is to use ADI (which permits larger time steps), but that converges at roughly the same rate. For large problems that require repeated fast solution both techniques have been superseded by *multigrid* methods [Press *et al.*, 1992], which find the final solution on N grid points in $\mathcal{O}(N)$ steps. These methods are based on iteratively coarse-graining the problem to produce a simpler one that can be solved quickly, and then interpolating to find the approximate solution at higher resolution. This is analogous to the Richardson extrapolation methods for ODEs, but because of the extra dimensions they are more complicated to implement.

7.4 SELECTED REFERENCES

[Ames, 1992] Ames, William F. (1992). *Numerical Methods for Partial Differential Equations*. 3rd edn. Boston, MA: Academic Press.

Ames is a classic reference for numerical methods for PDEs, with a bit more emphasis on mathematical rigor than on practical advice.

7.5 PROBLEMS

(7.1) Consider the 1D wave equation

$$\frac{\partial^2 u}{\partial t^2} = v^2 \frac{\partial^2 u}{\partial x^2} \quad (7.70)$$

- Write down the straightforward finite-difference approximation.
- What order approximation is this in time and in space?
- Use the von Neumann stability criterion to find the mode amplitudes.
- Use this to find a condition on the velocity, time step, and space step for stability (hint: consider the product of the two amplitude solutions).
- Do different modes decay at different rates for the stable case?
- Numerically solve the wave equation for the evolution from an initial condition with $u = 0$ except for one nonzero node, and verify the stability criterion.
- If the equation is replaced by

$$\frac{\partial^2 u}{\partial t^2} = v^2 \frac{\partial^2 u}{\partial x^2} + \gamma \frac{\partial}{\partial t} \frac{\partial^2 u}{\partial x^2} \quad (7.71)$$

assume that

$$u(x, t) = Ae^{i(kx - \omega t)} \quad (7.72)$$

and find a relationship between k and ω , and simplify it for small γ . Comment on the relationship to the preceding question.

- Repeat the numerical solution of the wave equation with the same initial conditions, but include the damping term.

- (7.2) Write a program to solve a 1D diffusion problem on a lattice of 500 sites, with an initial condition of zero at all the sites, except the central site which starts at the value 1.0. Take $D = \Delta x = 1$, and use fixed boundary conditions set equal to zero.
- Use the explicit finite difference scheme, and look at the behavior for $\Delta t = 1$, 0.5, and 0.1. What step size is required by the Courant condition?
 - Now repeat this using implicit finite differences and compare the stability.
- (7.3) Use ADI to solve a 2D diffusion problem on a lattice, starting with randomly seeded values.
- (7.4) Use SOR to solve Laplace's equation in 2D, with boundary conditions $u_{j,1} = u_{1,k} = 0$, $u_{N,k} = -1$, $u_{j,N} = 1$, and explore how the convergence rate depends on α , and how the best choice for α depends on the lattice size.

We have seen how to solve problems on a lattice, and how to use approximations. As problems are not directly applicable to inhomogeneous problems, for example, in steady-state problems, it is often more convenient to write it in terms of a Poisson equation, and find the general solution of the solution.

These limit the applicability of these methods. They are used for solutions to problems that are posed as boundary value problems, and for problems with spatial components in which the problem is the solution.

Because the number of unknowns is enormous, many people use direct methods for matrix problems. For example, the input geometry of a commercial package (http://www.ansys.com) for magnetic field analysis, there are also

The method of solving ordinary differential equations. Let $u(\vec{x}, t)$ be the solution on time, to