Elliptic PDEs

Numerical solution of Elliptic PDEs

- We will begin our study of numerical methods for PDEs with elliptic PDEs as they are generally among the easiest to solve (if not always the fastest) and many techniques are applicable to them.
- We will begin with finite differences.
- We will use the Poisson equation as our standard example.

Finite-Differencing the Poisson Equation

- We use centered differences

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u(i+i,j) - 2u(i,j) + u(i-1,j)}{\Delta x^2}$$

- And similarly for u_{yy}. The result is

$$\frac{u(i+1,j) - 2u(i,j) + u(i-1,j)}{\Delta x^2} + \frac{u(i,j+1) - 2u(i,j) + u(i,j-1)}{\Delta y^2} = S(i,j)$$

FD Poisson Equation

- In the case that $\Delta x=\Delta y=h$ we obtain the simpler form

$$u(i-1,j)+u(i,j-1)+u(i+1,j)+u(i,j+1)-4u(i,j)=h^2S(i,j)$$

- This is usually written in the notation

$$u_{i-1,j}+u_{i,j-1}+u_{i+1,j}+u_{i,j+1}-4u_{ij}=h^2S_{ij}$$

- It should be apparent that this is a system of linear equations that we must solve.

Stencils

- This discretization is said to be a *five-point stencil* since five points are required to determine the approximation for each i,j. Pictorially, we can represent it as

$$\nabla^2 u = \frac{1}{h^2} \begin{Bmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{Bmatrix} u_{ij}$$

Boundary Conditions

- Dirichlet boundary conditions are substituted directly into the equation to form the first and last rows of the coefficient matrix.
- Neumann boundary conditions are represented by a difference equation. To use differences we must introduce "fictitious" points outside our grid. For example, if we have a Neumann BC at i,N we introduce N+1. Central differences for the first derivative would give
- u(i,N+1)-u(i,n-1)=2hG or u(i,N+1)=u(i,N -1)+2hG

Example

- Illustration of a rectangular region with Dirichlet BC at two edges and Neumann at the other two.

	ua	ub	uc	
20	$u11$	$u12$	$u13$	20
20	$u21$	$u22$	$u23$	20
20	$u31$	$u32$	$u33$	20
20	$u41$	$u42$	$u43$	20
20	$u51$	$u52$	$u53$	20
	ud	ue	uf	

Iterative Solutions

- The methods we have discussed so far are called direct methods since we solve the system exactly (to numerical accuracy). Another approach is *iterative* or *relaxation* methods. These methods begin with a guess and correct the guess iteratively until the solution stops changing.

Jacobi Iteration

- The simplest example of an iterative method is Jacobi iteration.
- Rarely used in practice because it is so slow to converge
- Initialize U=G

$$u_{ij}^{n+1} = 0.25 * (u_{i,j+1}^{n} + u_{i,j-1}^{n} + u_{i+1,j}^{n} + u_{i-1,j}^{n})$$

Solving The FD Poisson Equation

- We almost never use anything but an iterative solver for the finite-difference equations of the Poisson equation. For example, we might use methods called SOR (successive overrelaxation) or ICCG (incomplete Cholesky conjugate gradient). An example of an SOR solver follows. (Declarations and other bookkeeping issues omitted.)

SOR

- An older but still popular method is successive overrelaxation or SOR.
- Relaxation method: modify Jacobi by adding and subtracting `u_ij`

$$u_{ij}^{n+1} = u_{ij}^{k} + 0.25(u_{i+1\,j}^{k} + u_{i-1,j}^{k} + u_{i,j+1}^{k} + u_{i,j-1}^{k} - 4u_{ij}^{k})$$

- SOR: multiply by a factor (about 1.5 to 2 is usually good)

$$u_{ij}^{n+1} = u_{ij}^{k} + 0.25\omega(u_{i+1\,j}^{k} + u_{i-1,j}^{k} + u_{i,j+1}^{k} + u_{i,j-1}^{k} - 4u_{ij}^{k})$$

SOR Solver

- !2D SOR for Cartesian coordinates. Second-order differencing.

```
!Chebyshev acceleration.

!Boundaries are at 0 and n and are assumed to be set before this
!subroutine is called.

hx2=hx*hx
hy2=hy*hy
h4=hx2*hy2

coef=0.5_r_kind/(hx2+hy2)
coefi=1._r_kind/coef

rjac2=rjac**2

! Update all interior points

do iter=1,2
   do j=1,m-1
      do i=1,n-1
         if ( mod(i+j,2) .eq. mod(iter,2) ) then
!        residual = coef*(hy2*(u(i-1,j)+u(i+1,j))+hx2*(u(i,j-1)+u(i,j+1))   &
!                                 -coefi*u(i,j)-h4*g(i,j))
         residual = (u(i-1,j)+u(i+1,j)+u(i,j-1)+u(i,j+1)   &
                                      -4*u(i,j)-hx2*g(i,j))

         u(i,j) = u(i,j) + weight*0.25_r_kind*residual
         endif
      enddo
   enddo
   if ( n .eq. 1 ) weight=1._r_kind/(1._r_kind-0.5_r_kind*rjac2)
   if ( n .eq. 2 ) weight=1._r_kind/(1._r_kind-0.25_r_kind*rjac2*weight)
enddo
```

Multigrid Methods

- The fastest known solvers for many elliptic equations are multigrid methods
- Suppose you decompose the solution into Fourier components. Relaxation methods generally converge at different rates for different wavelengths.
- Basic idea:
  - Step 1. Reduce high-frequency errors with a simple relaxation (e.g. Gauss-Seidel) on a fine grid
  - Step 2. Downscale the *residual error* to a coarser grid.
    - This may be done recursively down to some coarsest grid
  - Step 3. Apply correction and interpolate up the grid structure to the finest grid