Numerical Methods for Elliptic Equations-I

Grétar Tryggvason
Spring 2010

Elliptic equations often arise due to the application of conservation principles to quantities whose fluxes are proportional to their gradient

$\frac{\partial}{\partial x}F = -S$ where the flux is given by $F = -\alpha\frac{\partial f}{\partial x}$

gives $\frac{\partial}{\partial x}\alpha\frac{\partial f}{\partial x} = S$

In 2 or 3 dimension:

$$\left.\begin{array}{l}\nabla\cdot\mathbf{F} = -S\\ \mathbf{F} = -\alpha\nabla\mathbf{f}\end{array}\right\}\Rightarrow \nabla\cdot\alpha\nabla\mathbf{f} = S$$

If the transport coefficient is constant:

$$\alpha\nabla^2\mathbf{f} = S$$

One-Dimensional Boundary Value Problems

$$\frac{\partial}{\partial x}a\frac{\partial f}{\partial x} + b\frac{\partial f}{\partial x} + cf = s$$

$\frac{\partial f}{\partial x}$ or f given — $\frac{\partial f}{\partial x}$ or f given

periodic

Notice that if f is not given on the boundary, f is not uniquely determined

Two-Dimensional

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = 0; \quad \nabla^2 f = 0$$

Laplace's Equation

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = S; \quad \nabla^2 f = S$$

Poisson's Equation

$$\frac{\partial}{\partial x}a\frac{\partial f}{\partial x} + \frac{\partial}{\partial y}b\frac{\partial f}{\partial y} = S; \quad \nabla\cdot\phi\nabla f = S$$

On the boundaries (BC)

$f = f_0(x,y)$ Dirichlet

$\frac{\partial f}{\partial n} = g_0(x,y)$ Neumann

Three-Dimensional

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2} = 0$$

Laplace's Equation

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2} = s$$

Poisson's Equation

$$\frac{\partial}{\partial x}a\frac{\partial f}{\partial x} + \frac{\partial}{\partial y}b\frac{\partial f}{\partial y} + \frac{\partial}{\partial z}c\frac{\partial f}{\partial z} = s$$

Examples of Elliptic Equations

$$\nabla^2 T = -\frac{\dot{q}}{k}$$ Steady conduction equation

$$\nabla^2 \psi = -\omega$$ 2-D stream function equation

$$\nabla_h^2 P_{i,j} = \frac{1}{\Delta t}\nabla_h \cdot \mathbf{u}_{i,j}^t$$ Projection method (Step 2)

$$\mathbf{u}\cdot\nabla f - \nabla^2 f = 0$$ Steady state advection/diffusion

$$\nabla^4 f = 0$$ Biharmonic equation

One-Dimensional Boundary Value Problems —Direct Methods

$$U\frac{\partial f}{\partial x} - D\frac{\partial^2 f}{\partial x^2} = 0$$

$$U\frac{f_{i+1} - f_{i-1}}{2h} - D\frac{f_{i+1} - 2f_i + f_{i-1}}{h^2} = 0$$

$$\frac{R}{2}(f_{i+1} - f_{i-1}) - (f_{i+1} - 2f_i + f_{i-1}) = 0$$

$$\left(1 + \frac{1}{2}R\right)f_{i-1} - 2f_i + \left(1 - \frac{1}{2}R\right)f_{i+1} = 0$$

$$a_{j-1}f_{j-1} - d_j f_j + c_{j+1}f_{j+1} = b_j$$

$$R = \frac{Uh}{D}$$

$$a_j f_{j-1} - d_j f_j + c_j f_{j+1} = b_j$$

Write out

$$\begin{aligned}
d_1 f_1 + c_1 f_2 \quad &= b_1 \\
a_2 f_1 + d_2 f_2 + c_2 f_3 \quad &= b_2 \\
&\ddots \\
a_{N-1} f_{N-2} + d_{N-1} f_{N-1} + c_{N-1} f_N &= b_{N-1} \\
a_N f_{N-1} + d_N f_N &= b_N
\end{aligned}$$

If the endpoints are given

$$b_1 = -a_1 f_0$$

$$b_N = -c_N f_{N+1}$$


```
% solving a tridiagonal system
nx=50;r=0.1;
a=zeros(nx,1)+(1+r);d=zeros(nx,1)-2;c=zeros(nx,1)+(1-r);
b=zeros(nx,1); x=zeros(nx,1); b(nx)=-(1-r);

% forward elimination
for j=2:nx
  d(j)=d(j)-(a(j)/d(j-1))*c(j-1);
  b(j)=b(j)-(a(j)/d(j-1))*b(j-1);
  x(j)=b(j)/d(j);
end

% backward substitution
for j=nx-1:-1:1
  x(j)=(b(j)-c(j)*x(j+1))/d(j);
end

plot(x)
```


Matlab functions

For simple problems MATLAB has a number of functions to deal with matrices.

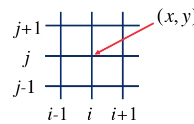
Help matfun: general

Help sparfun:sparse matrices

Elementary Iterative Methods

Solving the Poisson equation

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = S$$



Applying central differencing

$$\frac{f_{i+1,j} - 2f_{i,j} + f_{i-1,j}}{\Delta x^2} + \frac{f_{i,j+1} - 2f_{i,j} + f_{i,j-1}}{\Delta y^2} = S_{i,j}$$

Discretized Poisson Equation

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

Rearranging for $f_{i,j}$

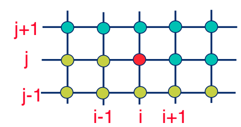
$$f_{i,j} = \frac{1}{4}\left[f_{i+1,j} + f_{i-1,j} + f_{i,j+1} + f_{i,j-1} - h^2 S_{i,j}\right]$$

$$f_{i,j}^{n+1} = \frac{1}{4}\left[f_{i+1,j}^{n} + f_{i-1,j}^{n} + f_{i,j+1}^{n} + f_{i,j-1}^{n} - h^2 S_{i,j}\right]$$ Jacobi

$$f_{i,j}^{n+1} = \frac{1}{4}\left[f_{i+1,j}^{n} + f_{i-1,j}^{n+1} + f_{i,j+1}^{n} + f_{i,j-1}^{n+1} - h^2 S_{i,j}\right]$$ Gauss-Seidel

$$f_{i,j}^{n+1} = \frac{\beta}{4}\left[f_{i+1,j}^{n} + f_{i-1,j}^{n+1} + f_{i,j+1}^{n} + f_{i,j-1}^{n+1} - h^2 S_{i,j}\right] + (1-\beta) f_{i,j}^{n}$$ SOR

The Jacobi iteration can be improved somewhat by using new values as soon as they become available.




```
for j=1:m
for i=1:n
    iterate
end
end
```

$$f_{i,j}^{n+1} = \frac{1}{4}\left(f_{i+1,j}^{n} + f_{i-1,j}^{n+1} + f_{i,j+1}^{n} + f_{i,j-1}^{n+1} - h^2 S_{i,j}\right)$$

From a programming point of view, Gauss-Seidler iteration is even simpler than Jacobi iteration since only one vector with f values is needed.

The Gauss-Seidler iteration can be accelerated even further by various acceleration techniques. The simplest one is the Successive Over-Relaxation (SOR) iteration

$$f_{i,j}^{n+1} = \frac{\beta}{4}\left(f_{i+1,j}^{n} + f_{i-1,j}^{n+1} + f_{i,j+1}^{n} + f_{i,j-1}^{n+1} - h^2 S_{i,j}^{n}\right) + (1-\beta) f_{i,j}^{n}$$

The SOR iteration is very simple to program, just as the Gauss-Seidler iteration. The user must select the coefficient. It must be bounded by 1<β<2. β=1.5 is usually a good starting value.

The iteration must be carried out until the solution is sufficiently accurate. To measure the error, define the residual:

$$R_{i,j} = \frac{f_{i+1,j} + f_{i-1,j} + f_{i,j-1} + f_{i,j+1} - 4f_{i,j}}{h^2} - S_{i,j}$$

At steady-state the residual should be zero. The point-wise residual or the average absolute residual can be used, depending on the problem. Often, simpler criteria, such as the change from one iteration to the next is used

Iteration versus time integration

Jacobi as a time integration

The solution of:

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = 0$$

can be thought of as the steady-state solution of

$$\frac{\partial f}{\partial t} = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Using the discretization derived earlier:

$$\frac{f_{i,j}^{n+1} - f_{i,j}^{n}}{\Delta t} = \frac{f_{i+1,j}^{n} + f_{i-1,j}^{n} + f_{i,j-1}^{n} + f_{i,j+1}^{n} - 4 f_{i,j}^{n}}{h^2}$$

or

$$f_{i,j}^{n+1} = f_{i,j}^{n} + \left(\frac{\Delta t}{h^2}\right)\left(f_{i+1,j}^{n} + f_{i-1,j}^{n} + f_{i,j-1}^{n} + f_{i,j+1}^{n} - 4 f_{i,j}^{n}\right)$$

Rearrange:

$$f_{i,j}^{n+1} = \left(1 - 4\frac{\Delta t}{h^2}\right) f_{i,j}^{n} + \left(\frac{\Delta t}{h^2}\right)\left(f_{i+1,j}^{n} + f_{i-1,j}^{n} + f_{i,j-1}^{n} + f_{i,j+1}^{n}\right)$$

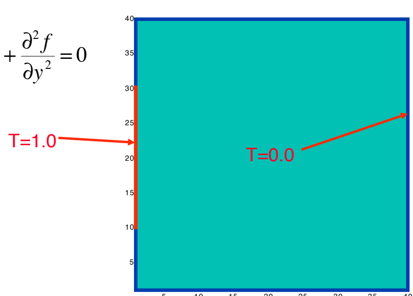
Select the maximum time step: $\frac{\Delta t}{h^2} = \frac{1}{4}$

$$f_{i,j}^{n+1} = \frac{1}{4}\left(f_{i+1,j}^{n} + f_{i-1,j}^{n} + f_{i,j-1}^{n} + f_{i,j+1}^{n}\right)$$

Which is exactly the Jacobi iteration

Example

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = 0$$




```
% two-dimensional steady-state problem by SOR
n=40;m=40;nstep=5000;alpha=0.05;length=2.0;h=length/(n-1);
T=zeros(n,m);bb=1.7;
T(10:n-10,1)=1.0;
for l=1:nstep,
   for i=2:n-1, for j=2:m-1
      T(i,j)=bb*0.25*(T(i+1,j)+...
             T(i,j+1)+T(i-1,j)+T(i,j-1))+(1.0-bb)*T(i,j);
   end,end
   % find residual
   res=0;
   for i=2:n-1, for j=2:m-1
      res=res+abs(T(i+1,j)+...
             T(i,j+1)+T(i-1,j)+T(i,j-1)-4*T(i,j))/h^2;
   end,end
   l,res/((m-2)*(n-2)) % Print iteration and residual
   if (res/((m-2)*(n-2)) < 0.001), break,end
end;
contour(T);
```

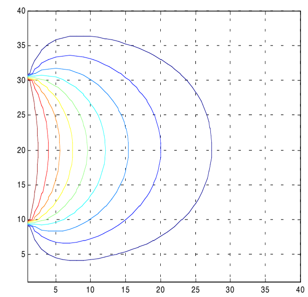
Computational Fluid Dynamics I

Average absolute error: 0.001

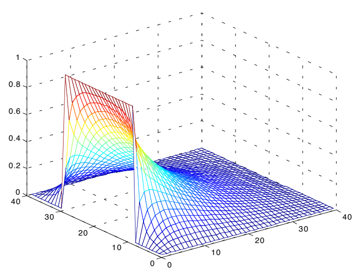
Number of iterations

Jacobi:	1989
Gauss-Seidler:	986
SOR (1.5):	320
SOR (1.7):	162
SOR (1.9):	91
SOR (1.95):	202

Computational Fluid Dynamics I



Computational Fluid Dynamics I



Computational Fluid Dynamics I

Note on Boundary Conditions

Computational Fluid Dynamics I
Boundary Conditions for Iterative Method

Dirichlet conditions are easily implemented.

For Neumann condition, the simplest approach is

$$\frac{\partial f}{\partial n} = 0 \;\Rightarrow\; f_{i,0} - f_{i,1} = 0 \quad \text{(1st order)}$$

Update interior points $f_{i,1}, f_{i,2}, f_{i,3}, \cdots$ and then set $f_{i,0} = f_{i,1}$

This generally does not converge.

Instead, incorporate BC directly into the equations

$$f_{i,1} = \frac{1}{4}\left[f_{i-1,1} + f_{i+1,1} + f_{i,2} + \overset{= f_{i,1}}{f_{i,0}} - h^2 S_{i,j}\right]$$

$$f_{i,1} = \frac{1}{3}\left[f_{i-1,1} + f_{i+1,1} + f_{i,2} - h^2 S_{i,j}\right]$$

Computational Fluid Dynamics I

With only a few exceptions, Iterative Methods are used to solve systems of equations resulting from the discretization of elliptic equations or implicit methods in CFD

Computational Fluid Dynamics I
http://users.wpi.edu/~gretar/me612.html

Numerical Methods for Elliptic Equations-II

Grétar Tryggvason
Spring 2010

Computational Fluid Dynamics I
Outline

Computational Fluid Dynamics I

Convergence of Iterative Methods

Computational Fluid Dynamics I
Convergence

A One Dimensional Example

An equation in the form

$$x = F(x)$$

can be solved by iterative procedure:

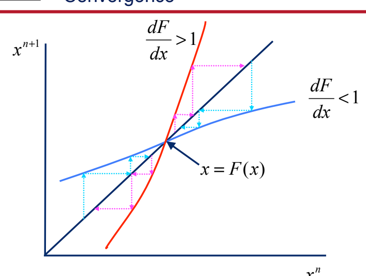
$$x^{n+1} = F(x^n)$$

for which convergence is achieved when

$$x^{n+1} \approx x^n \quad \text{or} \quad \left| \frac{x^{n+1}}{x^n} - 1 \right| < \varepsilon$$

When does the iteration converge?

Computational Fluid Dynamics I
Convergence



The above figure suggests that

$$\left| \frac{dF}{dx} \right| \leq 1 \quad \text{for convergence}$$

Computational Fluid Dynamics I
Convergence

A One Dimensional Example

For the linear equation

$$x^{n+1} = a x^n$$

We must have:

$$|a| \leq 1$$

for convergence

Computational Fluid Dynamics I
Convergence

For multidimensional problems we have:

$$\mathbf{x}^{\alpha+1} = \mathbf{M}\mathbf{x}^{\alpha}$$

For symmetric M it can be shown that its eigenvectors form a complete and orthogonal set and span the space of x. It is therefore possible to write:

$$\mathbf{x} = y_1\mathbf{v}_1 + y_2\mathbf{v}_2 + \cdots = \sum_j y_j \mathbf{v}_j$$

where

$$\mathbf{M}\mathbf{v}_j = \lambda_j \mathbf{v}_j \qquad j = 1 \ldots M \times N$$

Computational Fluid Dynamics I
Convergence

Hence it is possible to write

$$\mathbf{x}^{\alpha+1} = \mathbf{M}\mathbf{x}^{\alpha}$$

as

$$\begin{aligned} y_1^{\alpha+1}\mathbf{v}_1 + y_2^{\alpha+1}\mathbf{v}_2 + \cdots &= \mathbf{M}\left(y_1^{\alpha}\mathbf{v}_1 + y_2^{\alpha}\mathbf{v}_2 + \cdots\right) \\ &= y_1^{\alpha}\mathbf{M}\mathbf{v}_1 + y_2^{\alpha}\mathbf{M}\mathbf{v}_2 + \cdots \\ &= y_1^{\alpha}\lambda_1\mathbf{v}_1 + y_2^{\alpha}\lambda_2\mathbf{v}_2 + \cdots \end{aligned}$$

or

$$\begin{aligned} y_1^{\alpha+1} &= \lambda_1 y_1^{\alpha} \\ y_2^{\alpha+1} &= \lambda_2 y_2^{\alpha} \\ \vdots \quad & \quad \vdots \end{aligned}$$

Which are the same as for the 1-D example. Therefore:

$$|\lambda_{\max}| \le 1$$

for convergence

Computational Fluid Dynamics I

More Formal Discussion of Iterative Methods

Computational Fluid Dynamics I

If $\Delta x = \Delta y = h \Rightarrow f_{i+1,j} + f_{i-1,j} + f_{i,j+1} + f_{i,j-1} - 4f_{i,j} = h^2 S_{i,j}$

$$\begin{bmatrix} -4 & 1 & 0 & 0 & \cdots & 1 & 0 & \cdots & \cdots & \cdots \\ 1 & -4 & 1 & 0 & \cdots & 0 & 1 & \cdots & \cdots & \cdots \\ 0 & 1 & -4 & 1 & \cdots & \cdots & 0 & 1 & \cdots & \cdots \\ \vdots & & & & & & & & & \\ \vdots & & & & & & & & & \\ 1 & 0 & & & & & & & & \\ 0 & 1 & & & & & & & & \\ 0 & 0 & 1 & & & & & & & \\ & & & & & & & & & \\ & & & & & & & 0 & 1 & -4 \end{bmatrix} \begin{bmatrix} f_{1,1} \\ f_{1,2} \\ \vdots \\ f_{1,J} \\ f_{2,1} \\ f_{2,2} \\ \vdots \\ f_{I,J-1} \\ f_{I,J} \end{bmatrix} = h^2 \begin{bmatrix} S_{1,1} \\ S_{1,2} \\ \vdots \\ S_{1,J} \\ S_{2,1} \\ S_{2,2} \\ \vdots \\ S_{I,J-1} \\ S_{I,J} \end{bmatrix}$$

Sparse matrix: only 5 non-zero entries in each row

Computational Fluid Dynamics I

A few definitions:

$$[D] = \begin{bmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & d \end{bmatrix} \qquad [U] = \begin{bmatrix} 0 & a & b & c \\ 0 & 0 & d & e \\ 0 & 0 & 0 & f \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad [L] = \begin{bmatrix} 0 & 0 & 0 & 0 \\ a & 0 & 0 & 0 \\ b & c & 0 & 0 \\ d & e & f & 0 \end{bmatrix}$$

Diagonal — Upper triangular — Lower triangular

Computational Fluid Dynamics I

Ultimately, the difference form of the Poisson equation boils down to solving for

$$[A]\mathbf{f} = \mathbf{b}$$

Hence,

$$\mathbf{f} = [A]^{-1}\mathbf{b}$$

Direct method:
- Solving inverse matrix directly (Cramer's rule)
- Inverting matrix more cleverly (Gaussian elimination)
- Other (L-U decomposition, Thomas algorithm)

Computational Fluid Dynamics I

Gaussian Elimination

- Pivoting: rearranging equations to put the largest coefficient on the main diagonal.
- Eliminate the column below main diagonal.
- Repeat until the last equation is reached.
- Back-substitution

$$\begin{array}{l} a_{11}f_1 + a_{12}f_2 + \cdots\cdots = c_1 \\ a_{21}f_1 + a_{22}f_2 + \cdots\cdots = c_2 \\ \vdots \qquad\qquad\qquad\quad \vdots \\ a_{n1}f_1 + a_{n2}f_2 + \cdots\cdots = c_n \end{array} \;\Rightarrow\; \begin{array}{r} a_{11}f_1 + a_{12}f_2 + \cdots\cdots = c_1 \\ a'_{22}f_2 + \cdots\cdots = c_2 \\ \vdots \\ a'_{nn}f_n = c_n \end{array}$$

- Special case: tri-diagonal matrix - Thomas algorithm

Computational Fluid Dynamics I
Convergence – Multi Dimensional -1

General iterative procedure

$$[A]\mathbf{f} = \mathbf{b}$$

Let $[A] = [A_1] - [A_2]$

$$[A_1]\mathbf{f} = [A_2]\mathbf{f} + \mathbf{b}$$

An iterative scheme is constructed as

$$[A_1]\mathbf{f}^{n+1} = [A_2]\mathbf{f}^n + \mathbf{b}$$

For example,

$$[A_1] = [D] = -\frac{1}{4}[I], \quad [A_2] = [B] = [A_1] - [D] \qquad \text{Jacobi}$$

$$[A_1] = [D] - [L], \quad [A_2] = [U] \qquad \text{Gauss-Seidel}$$

Computational Fluid Dynamics I
Convergence – Multi Dimensional -2

$$[A_1]\mathbf{f}^{n+1} = [A_2]\mathbf{f}^n + \mathbf{b}$$

Requirements:

1. $[A_1]$ should be invertible.
2. Iteration should converge, i.e.

$$\lim_{n\to\infty} \mathbf{f}^n = \mathbf{f}$$

Define error at n-th iteration: $\mathbf{e}^n = \mathbf{f} - \mathbf{f}^n$

$$[A_1]\mathbf{e}^{n+1} = [A_2]\mathbf{e}^n$$

$$\longrightarrow \quad \mathbf{e}^{n+1} = [A_1]^{-1}[A_2]\mathbf{e}^n$$

$$\longrightarrow \quad \mathbf{e}^n = \left([A_1]^{-1}[A_2]\right)^n \mathbf{e}^0$$

Computational Fluid Dynamics I
Convergence – Multi Dimensional -3

Condition for convergence

$$\lim_{n\to\infty} \mathbf{e}^n = 0$$

which requires $\lim_{n\to\infty}\left([A_1]^{-1}[A_2]\right)^n = 0$

This is achieved if the modulus of the eigenvalues are less than unity.

Therefore, the convergence condition becomes:

$$\rho = |\lambda_i|_{\max} \leq 1$$

ρ Spectral radius of convergence

λ_i Eigenvalues of matrix $[A_1]^{-1}[A_2]$

Computational Fluid Dynamics I
Convergence – Jacobi Method -1

Jacobi Iteration Method for Poisson Equation
(2nd order central difference with uniform mesh)

$$f_{i,j}^{n+1} = \frac{1}{4}\left[f_{i+1,j}^n + f_{i-1,j}^n + f_{i,j+1}^n + f_{i,j-1}^n - h^2 S_{i,j}\right]$$

$$[A_1]^{-1} = [I]$$

and using a discrete analog of separation of variables, it can be shown that the eigenvalues of $[A_2]$ are

$$\lambda_{mn} = \frac{1}{2}\left[\cos\frac{m\pi}{M} + \cos\frac{n\pi}{N}\right], \; m = 1, \cdots, M-1, \;\; n = 1, \cdots, N-1$$

Therefore, $|\lambda_{mn}| \leq 1$ and the Jacobi method converges.

Computational Fluid Dynamics I
Convergence – Jacobi Method -2

For a large matrix

$$\lambda_{mn,\max} = \frac{1}{2}\left[\cos\frac{\pi}{M} + \cos\frac{\pi}{N}\right] \qquad \text{largest eigenvalues for } m = 1, \; n = 1$$

$$\cong 1 - \frac{1}{4}\left[\frac{\pi^2}{M^2} + \frac{\pi^2}{N^2}\right] + \cdots$$

Thus, for a large matrix, $|\lambda_{mn,\max}|$ is only slightly less than unity.

➔ Very slow convergence

Computational Fluid Dynamics I
Convergence – Gauss-Seidel -1

Gauss-Seidel Iteration Method for Poisson Equation
(2nd order central difference with uniform mesh)

$$f_{i,j}^{n+1} = \frac{1}{4}\left[f_{i+1,j}^{n} + f_{i-1,j}^{n+1} + f_{i,j+1}^{n} + f_{i,j-1}^{n+1} - h^2 S_{i,j}\right]$$

$$[A_1] = [D] - [L], \quad [A_2] = [U]$$

$$\underbrace{\begin{bmatrix} 4 & 0 & 0 & 0 & \cdots & 0 \\ -1 & 4 & 0 & 0 & \cdots & 0 \\ 0 & -1 & 4 & 0 & \cdots & 0 \\ 0 & 0 & -1 & 4 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \cdots & \vdots \\ 0 & \cdots & \cdots & 0 & -1 & 4 \end{bmatrix}}_{A_1} \begin{bmatrix} f_{1,1}^{n+1} \\ f_{1,2}^{n+1} \\ \vdots \\ \vdots \\ \vdots \\ f_{I,J}^{n+1} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ \vdots & \vdots & & & 0 & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix}}_{A_2} \begin{bmatrix} f_{1,1}^{n} \\ f_{1,2}^{n} \\ \vdots \\ \vdots \\ \vdots \\ f_{M,N}^{n} \end{bmatrix} - h^2 \begin{bmatrix} S_{1,1} \\ S_{1,2} \\ \vdots \\ \vdots \\ \vdots \\ S_{M,N} \end{bmatrix}$$

Computational Fluid Dynamics I
Convergence – Gauss-Seidel -2

It can be shown that the eigenvalues of matrix $[A_1]^{-1}[A_2]$ are simply square of the eigenvalues of Jacobi method

$$\lambda_{mn,\max} = \frac{1}{4}\left[\cos\frac{\pi}{M} + \cos\frac{\pi}{N}\right]^2$$

Thus, Gauss-Seidel method is twice as fast as the Jacobi method.

Computational Fluid Dynamics I
Successive Overrelaxation - 1

Successive Overrelaxation

Consider the Gauss-Seidel method

$$\left([D] - [L]\right)\mathbf{f}^{n+1} = [U]\mathbf{f}^{n} + \mathbf{b}$$

If Gauss-Seidel is an attempt to change the solution as

$$\mathbf{f}^{n+1} = \mathbf{f}^{n} + \mathbf{d}$$

Accelerate the change by introducing a parameter

$$\mathbf{f}^{n+1} = \mathbf{f}^{n} + \beta\mathbf{d}, \ \beta > 1$$

Computational Fluid Dynamics I
Successive Overrelaxation - 2

Hence, SOR first uses Gauss-Seidel to compute intermediate solution, $\tilde{\mathbf{f}}$

$$\left([D] - [L]\right)\tilde{\mathbf{f}} = [U]\mathbf{f}^{n} + \mathbf{b} \quad \text{or} \quad [D]\tilde{\mathbf{f}} = [L]\tilde{\mathbf{f}} + [U]\mathbf{f}^{n} + \mathbf{b}$$

Then accelerate the next iteration solution

$$\mathbf{f}^{n+1} = \mathbf{f}^{n} + \beta\left(\tilde{\mathbf{f}} - \mathbf{f}^{n}\right) = \beta\tilde{\mathbf{f}} + (1-\beta)\mathbf{f}^{n}$$

Combining the two steps

$$\mathbf{f}^{n+1} = \frac{\beta}{c}\left([L]\tilde{\mathbf{f}} + [U]\mathbf{f}^{n} + \mathbf{b}\right) + (1-\beta)\mathbf{f}^{n}$$

since $[D] = c[I]$

Computational Fluid Dynamics I
Successive Overrelaxation - 3

Example: Poisson Equation (2nd order CD, uniform mesh)

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

G-S

$$[D]\tilde{\mathbf{f}} = [L]\tilde{\mathbf{f}} + [U]\mathbf{f}^{n} + \mathbf{b}$$

$$f_{i,j}^{n+1} = \frac{1}{4}\left[f_{i-1,j}^{n+1} + f_{i,j-1}^{n+1} + f_{i+1,j}^{n} + f_{i,j+1}^{n} - h^2 S_{i,j}\right]$$

Combining

$$\mathbf{f}^{n+1} = \frac{\beta}{c}\left([L]\tilde{\mathbf{f}} + [U]\mathbf{f}^{n} + \mathbf{b}\right) + (1-\beta)\mathbf{f}^{n}$$

$$f_{i,j}^{n+1} = \frac{\beta}{4}\left[f_{i-1,j}^{n+1} + f_{i,j-1}^{n+1} + f_{i+1,j}^{n} + f_{i,j+1}^{n} - h^2 S_{i,j}\right] + (1-\beta)f_{i,j}^{n}$$

Computational Fluid Dynamics I
Successive Overrelaxation - 4

Convergence of SOR

From $[D]\tilde{\mathbf{f}} = [L]\tilde{\mathbf{f}} + [U]\mathbf{f}^{n} + \mathbf{b}$

$$\mathbf{f}^{n+1} = \mathbf{f}^{n} + \beta\left(\tilde{\mathbf{f}} - \mathbf{f}^{n}\right)$$

Eliminating $\tilde{\mathbf{f}}$ and solving for $\mathbf{f}^{n+1}$

$$\mathbf{f}^{n+1} = \overbrace{\left([I] - \beta[D]^{-1}[L]\right)^{-1}\left\{(1-\beta)[I] + \beta[D]^{-1}[U]\right\}}^{[M]_{SOR}}\mathbf{f}^{n} + \left([I] - \beta[D]^{-1}[L]\right)^{-1}\beta[D]^{-1}\mathbf{b}$$

Convergence depends on the eigenvalues of $[M]_{SOR}$

For the discretized Poission operator, it can be shown that that eigenvalues of the SOR matrix are:

$$\mu^{1/2} = \frac{1}{2}\left[\lambda\beta + \sqrt{\lambda^2\beta^2 - 4(\beta - 1)}\right]$$

where λ is an eigenvalue of the Jacobi matrix

$$[M]_J = [D]^{-1}([L]+[U])$$

Note that $\mu = \lambda^2$ if $\beta = 1$ (Gauss-Seidel)

Minimum μ occurs at

$$\beta_{\text{opt}} = \frac{2}{1+\sqrt{1-\lambda_{\max}^2}}$$

Typically

$$\beta_{\text{opt}} \approx 1.7 \sim 1.9$$

$$|\lambda_{\max}| \approx 1$$

For problems with irregular geometry and non-uniform mesh, β_{opt} must be found by trial and error.

Typical Comparison Chart

	$\lambda_{\max}$ $(\mu_{\max})$	Iterations
Jacobi	0.9945	1250
Gauss-Seidel	0.9890	625
SOR	0.7906	29

Ferziger, J. H., *Numerical Method for Engineering Application* (1981)

Numerical Methods for Elliptic Equations-III

Grétar Tryggvason
Spring 2010

Examples of elliptic equations
Direct Methods for 1D problems
Elementary Iterative Methods
Iteration as Time Integration
Example
Boundary Conditions
Convergence of Iterative Methods
- 1D Example
- Formal Discussion

SOR on vector computers

SOR on Vector Computers

In large computer application (vector or parallel platform), SOR faces difficulties in using constantly updated values.

Remedy: Two separate grid system (red & black)

do i =1, nx, 2

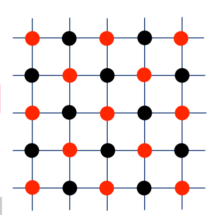
$$f_{i,j} = \frac{\beta}{4}\left[f_{i-1,j} + f_{i,j-1} + f_{i+1,j} + f_{i,j+1} - h^2 S_{i,j}\right] + (1-\beta) f_{i,j}$$

enddo

do i=2, nx, 2

$$f_{i,j} = \frac{\beta}{4}\left[f_{i-1,j} + f_{i,j-1} + f_{i+1,j} + f_{i,j+1} - h^2 S_{i,j}\right] + (1-\beta) f_{i,j}$$

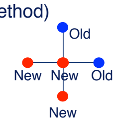
enddo



Computational Fluid Dynamics I
Successive Line Overrelaxation (SLOR) - 1

Line Relaxation Method (Line Gauss-Seidel Method)

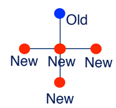
$$f_{i,j}^{n+1} = \frac{1}{4}\left[f_{i-1,j}^{n+1} + f_{i,j-1}^{n+1} + f_{i+1,j}^{n} + f_{i,j+1}^{n} - h^2 S_{i,j}\right]$$



Adding one more coupling

$$f_{i,j}^{n+1} = \frac{1}{4}\left[f_{i-1,j}^{n+1} + f_{i,j-1}^{n+1} + f_{i+1,j}^{n+1} + f_{i,j+1}^{n} - h^2 S_{i,j}\right]$$

$$-\frac{1}{4} f_{i-1,j}^{n+1} + f_{i,j}^{n+1} - \frac{1}{4} f_{i+1,j}^{n+1} = \frac{1}{4}\left[f_{i,j-1}^{n+1} + f_{i,j+1}^{n} - h^2 S_{i,j}\right]$$



➔ Thomas algorithm

Computational Fluid Dynamics I
Successive Line Overrelaxation (SLOR) - 2

SLOR = Line Relaxation + Overrelaxation

Apply line relaxation for intermediate solution

$$-\frac{1}{4}\tilde{f}_{i-1,j} + \tilde{f}_{i,j} - \frac{1}{4}\tilde{f}_{i+1,j} = \frac{1}{4}\left[f_{i,j-1}^{n+1} + f_{i,j+1}^{n} - h^2 S_{i,j}\right]$$

and then overrelax

$$f_{i,j}^{n+1} = \beta \tilde{f}_{i,j} + (1-\beta) f_{i,j}^{n}$$

which is no more complicated than line relaxation.

Computational Fluid Dynamics I
Successive Line Overrelaxation (SLOR) - 3

Notes on SLOR

- Exact eigenvalues are unknown.
- To ensure convergence, $\beta \le 2$
- Converges approximately twice as fast as Gauss-Seidel.
- May be faster than pointwise SOR, but each iteration takes longer with Thomas algorithm.
- Improved convergence is due to the direct effect of the boundary condition in each row.

Computational Fluid Dynamics I
Alternating-Direction Implicit - 1

ADI for elliptic equation is analogous to ADI in parabolic equation

$$\frac{\partial f}{\partial t} = \alpha\left[\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}\right] - S$$

In discrete form

$$f_{i,j}^{n+1} - f_{i,j}^{n} = \alpha \Delta t\left[\delta_{xx} f + \delta_{yy} f\right] + S$$

and take it to the limit to obtain the steady solution.

$$\left(\frac{\partial f}{\partial t} = 0\right)$$

Computational Fluid Dynamics I
Alternating-Direction Implicit - 2

ADI for $\quad f_{i,j}^{n+1} - f_{i,j}^{n} = \alpha \Delta t\left[\delta_{xx} f + \delta_{yy} f\right] - S$

is written as

$$f^{n+1/2} - f^{n} = \frac{\alpha \Delta t}{2h^2}\left[\left(f_{i+1,j}^{n+1/2} - 2 f_{i,j}^{n+1/2} + f_{i-1,j}^{n+1/2}\right) + \left(f_{i,j+1}^{n} - 2 f_{i,j}^{n} + f_{i,j-1}^{n}\right)\right] - S'_{i,j}$$

$$f^{n+1} - f^{n+1/2} = \frac{\alpha \Delta t}{2h^2}\left[\left(f_{i+1,j}^{n+1/2} - 2 f_{i,j}^{n+1/2} + f_{i-1,j}^{n+1/2}\right) + \left(f_{i,j+1}^{n+1} - 2 f_{i,j}^{n+1} + f_{i,j-1}^{n+1}\right)\right] - S''_{i,j}$$

or

$$\left(1 - \rho_n \delta_{xx}\right) f^{n+1/2} = \left(1 + \rho_n \delta_{yy}\right) f^{n} - S'_{i,j}$$

$$\left(1 - \rho_n \delta_{yy}\right) f^{n+1} = \left(1 + \rho_n \delta_{xx}\right) f^{n+1/2} - S''_{i,j}$$

$$\left(\rho_n = \frac{\alpha \Delta t_n}{2}\right)$$

Computational Fluid Dynamics I
Alternating-Direction Implicit - 3

Convergence of ADI

- Iteration parameter $\rho_n = \dfrac{\alpha \Delta t_n}{2}$ usually varies with iteration
- For example (Wachspress)

$$\frac{\rho_k}{h^2} = \frac{\alpha \Delta t_k}{2h^2} = b\left(\frac{a}{b}\right)^{(k-1)/(n-1)}, \quad k = 1, \cdots, n$$

a lower bound eigenvalue

b upper bound eigenvalue

- Comparison with SOR is difficult
- ADI can be efficient if appropriate parameters are found.

Although the iterative methods discussed here are important for understanding iterative methods, they are rarely used for practical applications due to their slow convergence rate.

The exception is the SOR method, which was widely used in the 70's and early 80's. Due to its simplicity, it is an excellent choice during code development or for runs where programming time is of more concern than computer time.