Numerical Methods for Elliptic Equations-III

Grétar Tryggvason
Spring 2011

ADI Methods for Elliptic Equations

ADI for elliptic equation is analogous to ADI in parabolic equation

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

In discrete form

\[
f^{n+1}_{ij} - f^n_{ij} = \alpha \Delta_x [\delta_y f + \delta_x f] + S
\]

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

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

Convergence of ADI

- Iteration parameter \( \rho \), \( \frac{\alpha \Delta t}{2 b^2} \), usually varies with iteration
- For example (Wachspress)

\[
\rho_k = \frac{\alpha \Delta t}{2 b^2} = \left( \frac{a}{b} \right)^{k-1/2}, \quad k = 1, \ldots, n
\]

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

Krylov methods

Most of these methods are formulated as minimization techniques, where the following function is minimized

\[ F = x^T A x - x^T b \]

The search direction and the step are selected in a "best" way.

Preconditioning

If the inverse of the matrix A was known, then the solution could be written down in a straight forward way

\[ A x = b \]
\[ A^{-1} A = I \]
\[ A^{-1} A x = A^{-1} b \]
\[ I x = x = A^{-1} b \]

Iterative methods generally converge faster if we have a matrix A that is "close" to the identity matrix I.

Many methods work only for symmetric systems. While the pressure equation is usually symmetric, implicit methods usually do not lead to symmetric matrices.

Usually the system is PRECONDITIONED to make it better behaved.

If we have a matrix M that is in some way close to the inverse of A, then the system

\[ M A x = M b \]

Has the same solution as the original system

\[ A x = b \]

And should be easier to solve.
Finding the "best" preconditioner is an active research topic, but the simplest choice is:
\[ m_{ij} = \begin{cases} 
\alpha_{ij} & \text{if } i = j \\
0 & \text{otherwise} 
\end{cases} \]

Jacobi preconditioner

Symmetric Successive Overrelaxation (SSOR) Preconditioner
\[
A = D + L + L^T \\
M = (D + L)D^{-1}(D + L)^T \\
M(\beta) = \frac{1}{2-\beta}D + L\left(\frac{1}{\beta}D + L\right)^T
\]


**Steepest Decent** (for symmetric positive A)

Find the residual and correct the guess by going in the direction of the residual

\[ r = b - Ax \]
\[ \alpha = (r, r) / (\Delta r, r) \]
\[ s = s + \alpha \Delta r \]

Notice that we do not need A separately, only the product of A and a vector

\[ \chi^{(1)} \] 1\text{st iterate} \\
\[ r^{(1)} \] 1\text{st residual at 1\text{st step}} \\
\[ \alpha_1 \] multiple of the search direction vector

Computational Fluid Dynamics

Other choices include the so-called Incomplete Cholesky factorization or even multigrid methods

In general it appears that finding an effective preconditioner is more important than exactly what Krylov method we use

The conjugate gradient method

Generally iterative methods generate a sequence of approximations that are used to construct a new approximation. Ideally, we only need to keep a few approximations, but one of the more popular technique, GMRES, requires all the previous iterates. This leads to the restarted GMRES

\[ \chi^{(0)} = b \cdot A x^{(0)} \] for some initial guess \( x^{(0)} \)

\[ \begin{align*}
\text{for } i = 1, \ldots \\
& \text{solve } M^{(1)} x = r^{(1)} \\
& \beta_{1,1} = \frac{r_{1,1}}{\Delta r_{1,1}} \\
& \text{if } i = 1 \\
& \quad p^{(0)} = \Delta r \\
& \text{else} \\
& \quad \beta_{i,1} = \frac{q_{1,i}}{\Delta r_{1,i}} \\
& \quad p^{(i)} = p^{(i-1)} + \beta_{i,1} \Delta r^{(i)} \\
& \text{end if} \\
& \quad q = A p^{(i)} \\
& \gamma_{i,1} = \alpha_{i,1} / p_{i,1} \\
& \quad \alpha_i = \alpha_{i,1} + \frac{q_{i,1}^2}{p_{i,1}^2} \\
& \quad x^{(i)} = x^{(i-1)} + \alpha_i \Delta r^{(i)} \\
& \quad r^{(i)} = r^{(i-1)} - \alpha_i A \Delta r^{(i)} \\
& \text{Check convergence; continue if needed}
\end{align*} \]

One of the oldest non-stationary iterative method is the conjugate gradient method

\[ \chi^{(0)} \] 0\text{th iterate} \\
\[ r^{(0)} \] 0\text{th residual at 0\text{th step}} \\
\[ \alpha_0 \] multiple of the search direction vector

For \( M=I \) we have the unpreconditioned version

\[
\begin{bmatrix}
1 & 2 \\
2 & 1
\end{bmatrix}
\begin{bmatrix}
x \\
y
\end{bmatrix}
= 
\begin{bmatrix}
5 \\
4
\end{bmatrix}
\]

\[ x^{(0)} = [0 0]^T \]

\[ x^{(1)} = [1.0033 1.9970]^T \]

\[ x^{(2)} = [0.9520 1.9906]^T \]

\[ x^{(3)} = [1.0687 2.1373]^T \]

\[ x^{(4)} = [1.6942 1.3554]^T \]

\[ x^{(5)} = [0.0490 0.0511]^T \]

\[ x^{(6)} = [0.9953 1.9906]^T \]

\[ x^{(7)} = [0.0436 0.0199]^T \]

\[ x^{(8)} = [1.0033 1.9970]^T \]
At the present time there does not seem to be a “best” Krylov method. In addition to the relatively simple early methods like the conjugate gradient method, GMRES is fairly popular (particularly the restarted version) and BiCGSTAB has been used by a number of people.

Preconditioned Bi-CGSTAB

From: Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods
http://www.netlib.org/templates/Templates.html

Books:
Barrett, Berry, Chan, Deline, Donato, Dongarra, Eijkhout, Pozo, Romine, and Van der Vorst
Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods
http://www.netlib.org/templates/Templates.html

Y. Saad. Iterative methods for sparse linear systems (2nd edition)
http://www-users.cs.umn.edu/~saad/books.html

Although iterative methods are the dominant technique for solutions of elliptic equations in CFD, Fast Direct Methods exists for special cases. The methods require simple domains (rectangles), simple equations (separable), and simple boundary conditions (periodic, or the derivative or the function equal to zero at each boundary)

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

separable

\[
\frac{\partial}{\partial x} a(x,y) \frac{\partial f}{\partial x} + \frac{\partial}{\partial y} b(x,y) \frac{\partial f}{\partial y} = S
\]

non-separable

Fast Direct Methods

The fast Fourier transform

\[
f_j = \sum_{k=1}^{N} \hat{f}_k e^{i \frac{2 \pi k j}{N}}
\]

inverse

\[
\hat{f}_k = \sum_{j=1}^{N} f_j e^{-i \frac{2 \pi k j}{N}}
\]

Can be evaluated in \(2N \log_2 N\) operations

The Cooley-Tukey algorithm
Δc \quad (d/dx)(du/dx) + (d/dy)(du/dy) + \lambda u = f(x,y).

c

Coordinates:

difference approximation to the helmholtz equation in cartesian

* * * * * * * * *  purpose    * * * * * * * * * * * * * * * * * *

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

*              the national science foundation                                     *

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

*                   which is sponsored by                                                  *

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

*                boulder, colorado  (80307)  u.s.a.                                *

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

*         the national center for atmospheric research                *

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

*                             of                                                                               *

*        john adams, paul swarztrauber and roland sweet          *

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

*                  (version 3.1 , october 1980)                                        *

*      separable elliptic partial differential equations                 *

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

*              elmbda,f,idimf,pertrb,ierror,w)

subroutine hwscrt (a,b,m,mbdcnd,bda,bdb,c,d,n,nbdcnd,bdc,bdd,

\Delta

The algorithm is

1 Find \hat{b}_{l,m} by FFT

2 Find \hat{f}_{l,m} as shown before

3 find \hat{f}_{l,j} by FFT

Have

\Delta f_{l,j} = -2 \sum_{l=1}^{N} \sum_{m=1}^{N} \hat{f}_{l,m} e^{i(l+m)\pi/N} \left( -2 \cos \frac{2\pi}{N} l + \cos \frac{2\pi}{N} m \right)

also

b_{l,j} = \sum_{l=1}^{N} \sum_{m=1}^{N} \hat{b}_{l,m} e^{i(l+m)\pi/N}

but

\Delta f_{l,j} = \hat{b}_{l,j}

Solve:

\hat{f}_{l,m} = \frac{-\hat{b}_{l,m}}{2 \left( -2 \cos \frac{2\pi}{N} l + \cos \frac{2\pi}{N} m \right)}

As outlined the method is applicable to periodic boundaries only. Other simple boundary

conditions can be handled by simple changes

(using cosine or sine series).

Other fast direct methods, such as Cyclic

Reduction, are based on similar ideas

See FISHPACK

\Delta

Computational Fluid Dynamics

Computational Fluid Dynamics

Resources
A large number of pre-written software packages for the solution of elliptic equations is available.

**In MATLAB:**

Linear Equations (iterative methods).
- pcg - Preconditioned Conjugate Gradients Method.
- bicg - BiConjugate Gradients Method.
- bicgstab - BiConjugate Gradients Stabilized Method.
- cgs - Conjugate Gradients Squared Method.
- gmres - Generalized Minimum Residual Method.
- qmr - Quasi-Minimal Residual Method.

**In MATLAB:**

- MUDPACK
  - [http://www.scd.ucar.edu/css/software/mudpack/](http://www.scd.ucar.edu/css/software/mudpack/)
- FISHPACK
  - [http://www.scd.ucar.edu/css/software/fishpack/](http://www.scd.ucar.edu/css/software/fishpack/)
- Multigrid website
  - [http://www.mgnet.org/](http://www.mgnet.org/)
- Netlib is a collection of mathematical software, papers and databases.
  - [http://www.netlib.org/](http://www.netlib.org/)

**Examples of elliptic equations**

- Direct Methods for 1D problems
- Elementary Iterative Methods
- Note on Boundary Conditions
- SOR on vector computers
- Iteration as Time Integration
- Convergence of Iterative Methods—elementary considerations
- Multigrid methods
- Convergence of Iterative Methods—Formal Discussion
- ADI for elliptic equations
- Krylov Methods
- Fast Direct Method
- Resources

**The Advection-Diffusion equation**

The Cell Reynolds number

1D Advection/diffusion equation

\[
\frac{\partial f}{\partial t} + U \frac{\partial f}{\partial x} = D \frac{\partial^2 f}{\partial x^2}
\]

Forward in time/centered in space (FTCS)

\[
\frac{f^{n+1}_i - f^n_i}{\Delta t} + U \frac{f^n_{i+1} - f^n_{i-1}}{2h} = D \frac{f^n_{i+1} - 2f^n_i + f^n_{i-1}}{h^2}
\]

Stability limits

\[
\frac{U\Delta t}{2D} \leq 1 \quad \Rightarrow \quad \Delta t \leq \frac{2D}{U}
\]

\[
\frac{D\Delta t}{h^2} \leq \frac{1}{2} \quad \Rightarrow \quad \Delta t \leq \frac{h^2}{2D}
\]

\[
\Delta t \to 0 \quad \text{For high and low D}
\]

**Resource:**

- FTCS \( O(\Delta t, h^2) \)
  - \( \frac{U\Delta t}{2D} \leq 1 \quad \& \quad \frac{D\Delta t}{h^2} \leq \frac{1}{2} \)
- Upwind \( O(\Delta t, h) \)
  - \( \frac{U\Delta t}{h} + \frac{D\Delta t}{h} \leq 1 \)
- L-W \( O(\Delta t^2, h^2) \)
  - \( \left( \frac{U\Delta t}{h} \right)^2 \leq 2 \frac{D\Delta t}{h^2} \leq 1 \)
- C-N \( O(\Delta t^2, h^2) \)
  - Unconditionally stable
**Computational Fluid Dynamics**

Steady state solution to the advection/diffusion equation

\[ U \frac{\partial f}{\partial x} = D \frac{\partial^2 f}{\partial x^2} \]

Exact solution

\[ f = \frac{\exp(R_j x / L) - 1}{\exp(R_j) - 1} \quad R_j = \frac{U_L}{D} \]

Numerical solution of:

\[ U \frac{f_{j+1} - f_j}{2h} = D \frac{f_{j+1} - 2f_j + f_{j-1}}{h^2} \]

Centered difference approximation

\[ f_j = \frac{q_j + 1}{2} \]

Solving for \( q \) gives two solutions:

\[ q_1 = 1 \quad \text{and} \quad q_2 = \frac{2 + R}{2 - R} \]

The general solution is:

\[ f_j = C_1 q_j^R + C_2 q_j^L \]

or

\[ f_j = C_1 \left( \frac{2 + R}{2 - R} \right)^R \]
Apply the boundary conditions

\[ f_0 = C_1 + C_2 \left( \frac{2 + R}{2 - R} \right) = C_1 + C_2 = 0 \]

\[ f_N = C_1 + C_2 \left( \frac{2 + R}{2 - R} \right) = 1 \]

The final solution is:

\[ f_j = \left( \frac{2 + R}{2 - R} \right)^j - 1 \]

\[ f_j = \frac{\left( \frac{2 + R}{2 - R} \right)^j - 1}{\left( \frac{2 + R}{2 - R} \right) - 1} \]

Upwind

\[ \frac{U}{h} \frac{f_j - f_{j-1}}{h} = D \frac{f_{j+1} - 2 f_j + f_{j-1}}{h^2} \]

or

\[ (R + 2) f_j - (R + 1) f_{j+1} - f_{j-1} = 0 \]

Try solutions

\[ f_j = q^j \]

giving

\[ q^j - (R + 2) q^j + (R + 1) = 0 \]

Solution

\[ f_j = \frac{1 - (1 + R)^j}{1 - (1 + R)} \]
When centered differencing is used for the advection/diffusion equation, oscillations may appear when the Cell Reynolds number is higher than 2. For upwinding, no oscillations appear. In most cases the oscillations are small and the cell Reynolds number is frequently allowed to be higher than 2 with relatively minor effects on the result.

\[ R = \frac{U h}{D} < 2 \]

2D example

\[ \frac{\partial f}{\partial t} + U \frac{\partial f}{\partial x} + V \frac{\partial f}{\partial y} = D \left( \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \right) \]

\[ f = 0 \]
\[ f = 1 \] (Computations using centered differences on a 32 by 32 grid)
\[ f = 0 \]

\[ \text{Re}_{\text{cell}} = 3.2258 \]
\[ D = 0.02 \]
\[ t = 1.5088 \]

\[ \text{Re}_{\text{cell}} = 6.4516 \]
\[ D = 0.01 \]
\[ t = 1.5088 \]

\[ \text{Re}_{\text{cell}} = 12.9032 \]
\[ D = 0.005 \]
\[ t = 1.5088 \]

Fine grid

\[ \text{Re}_{\text{cell}} = 3.2958 \]
\[ D = 0.02 \]
\[ t = 1.50 \]

Coarser grid

\[ \text{Re}_{\text{cell}} = 6.6716 \]