6. Solving the Poisson equation

So far we have dealt only with either the advection or momentum equations or the vorticity transport equation. We also need methods to solve the elliptic Poisson equation for the stream function ψ

$$(1) \qquad \nabla^2\psi = \frac{\partial^2\psi}{\partial x^2} + \frac{\partial^2\psi}{\partial y^2} = \zeta$$

Using standard second-order finite differences, (1) can be approximated by

$$\frac{\psi_{i+1,j} + \psi_{i-1,j} + \psi_{i,j+1} + \psi_{i,j-1} - 4\psi_{i,j}}{\Delta x^2} = \zeta_{i,j}$$

We know the $\zeta_{i,j}$, we want the $\psi_{i,j}$. In practice, it is a very large set of equations that we are trying to solve

Example:
the set of equations is for a 5 by 6 domain

$$\begin{pmatrix} B & I & 0 & 0 \\ I & B & I & 0 \\ 0 & I & B & I \\ 0 & 0 & I & B \end{pmatrix} \begin{pmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \end{pmatrix} = \begin{pmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{pmatrix}$$

→ function of $\zeta_{i,j}$ + boundary conditions $\begin{pmatrix} \psi_{1,j} & \psi_{i,1} \\ \psi_{5,j} & \psi_{i,6} \end{pmatrix}$

where $B = \begin{pmatrix} -4 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & -4 \end{pmatrix}$

$$\phi_i = \begin{pmatrix} \psi_{2,i} \\ \psi_{3,i} \\ \psi_{4,i} \end{pmatrix}$$

Block tridiagonal set of equations. Can be used to our advantage in many schemes to solve the set

6.1. Iterative methods

These methods require the solution of a linear system of equations. The order of this system may be very large. However the properties of these systems allows us to construct effective iterative methods for their solution.

We want to solve the matrix equation
$A x = b$ where x and b are n-dimensional vectors and A is a matrix of order n. We are trying then to find the zeros of the vector function $f(x) = Ax - b$. We can then convert this to a fixed point problem by defining the function $g(x) = x - f(x)$ $= (I - A)x + b$. We are then now looking for vectors x such that $x = g(x)$. The easiest iterative method is

{ choose an initial guess x_0.
{ define $x^{m+1} = g(x^m)$

$$x^{m+1} = (I - A)x^m + b = M x^m + b$$

We denote the error ε^m by $\varepsilon^m = x - x^m$. Then

$$\varepsilon^{m+1} = M \varepsilon^m = M^m \varepsilon(0)$$

The convergence will therefore depends on the conditions under which M^m will approach zero. If M is convergent or $S_R(M) < 1$, then the error will eventually reach zero.

↳ 28 Feb 1991

In practice, we express any matrix A as the sum $A = D - E - F$

where D is diagonal, E and F are strictly lower and upper triangular n^{th} order matrix.

$$Ax = b \Rightarrow Dx = (E+F)x + b$$

(a) Method #1 or
- Ordinary iteration
- Gauss-Jacobi iteration
- Point Jacobi
- Point total step iteration method
- Unaccelerated Richardson method
- Methods of simultaneous displacements

We define the iteration scheme as

(2) $$\boxed{x^{m+1} = D^{-1}(E+F)x^m + D^{-1}b}$$

or in expanded form

$$x_i^{m+1} = -\sum_{\substack{j=1 \\ j\neq i}}^{n} (a_{ij}/a_{ii})\, x_j^m + b_i/a_{ii}$$

The method will converge if
$\delta_R(\Omega) < 1$ with $\Omega = D^{-1}(E+F)$

(b) Method #2 or
- Gauss-Seidel iteration
- Point Gauss-Seidel
- Point single-step iteration
- Liebmann method
- Methods of successive displacements

The iteration scheme is defined as

(3) $$\boxed{(D-E)x^{m+1} = Fx^m + b}$$

In expanded form

$$a_{ii}\, x_i^{m+1} = -\sum_{j<i} a_{ij}\, x_j^{m+1} - \sum_{j>i} a_{ij} x_j^m + b_i$$

The iteration matrix Ω is $\Omega = (D-E)^{-1}F$
and converge if $\delta_R(\Omega) < 1$

This methods require the storage of only one vector x since we replace the components of x^m by components of x^{m+1} as soon as they are computed

Faster

(c) Method #3 or — SOR Successive Over Relaxation

$A = D - L - U$

and $D^{-1}A = I - \mathcal{L} - U$

($\mathcal{L}$ strictly lower and U upper matrices)

- Extrapolated Liebman
- Relaxation methods of successive displacements
- Gauss-Seidel relaxation
- Gauss-Southwell "

* We define R, a residual vector which is the error vector at any stage of the calculation.

$$R^m = D^{-1}b - x^m + \mathcal{L}x^{m+1} + Ux^m$$

If we consider the iteration scheme

$$(I - \mathcal{L})x^{m+1} = Ux^m + D^{-1}b$$

* We now define the iterative scheme to be used

(4)
$$\boxed{x^{m+1} = x^m + \alpha R^m}$$

where α is called a relaxation parameter

In general, we can show that $0 < \alpha < 2$ for the method to work.

This is an error correction method. If $\alpha > 1$ we are overcorrecting the scheme. If $\alpha < 1$ we are undercorrecting the scheme. If $\alpha = 1$ similar to method #2. The iterative scheme can be rewritten

$$x^{m+1} = x^m - \alpha x^m + \alpha\mathcal{L}x^{m+1} + \alpha Ux^m + \alpha D^{-1}b$$

Solving for x^{m+1} gives

$$(I - \alpha\mathcal{L})x^{m+1} = (I - \alpha I + \alpha U)x^m + \alpha D^{-1}b$$

$$x^{m+1} = (I - \alpha\mathcal{L})^{-1}(\mathcal{L} - \alpha I + \alpha U)x^m + (I - \alpha\mathcal{L})^{-1}\alpha D^{-1}b$$

or

$$\boxed{x^{m+1} = Mx^m + (I - \alpha\mathcal{L})^{-1}\alpha D^{-1}b}$$

with $M = (I - \alpha L)^{-1} (I - \alpha I + \alpha U)$

Method # 3 will then converge if $S_R(M) < 1$, we then want to know the optimal α for which $\| M_3 \|$ is a minimum.

* If we define $M(\alpha) = (I - \alpha L)^{-1} (\alpha U + (1-\alpha) I)$
then $S_R(M) \geq |\alpha - 1|$

(5) $\phi(\lambda) = \det(\lambda I - M) = \det(\lambda I - (I - \alpha L)^{-1}(\alpha U + (1-\alpha)I))$
$= \det[\lambda(I - \alpha L) - (\alpha U + (1-\alpha) I)]$
since $\begin{cases} \det[AB] = \det A \det B \\ \det(I - \alpha L) = 1 \end{cases}$

If $\lambda_i(\alpha)$ are the roots of $\phi(\lambda)$ (and eigenvalues of M)
then from polynomials theory

$$(-1)^n \prod^n \lambda_i(\alpha) = \phi(0), \text{ but}$$

also $\phi(0) = (\alpha - 1)^n$ from (5).

This then $\Rightarrow$ that $\max |\lambda_i| \geq |\alpha - 1|$

$\Longrightarrow$ This method may converge only if $0 < \alpha < 2$.
but not for other values.

d) The steady solution

As discussed above, the solution of a steady elliptic equation by iteration is analogous to solving a time-dependent problem to an asymptotic steady state (error $= 0$ or $\frac{\partial \psi}{\partial t} = 0$). Suppose we consider the time dependent diffusion equation for ψ with a source term, ξ, and a diffusion term

(6) $$\boxed{\frac{\partial \psi}{\partial t} = \nabla^2 \psi - \xi}$$

We are not interested in the significance of the transients, but as the solution approaches a steady state, it also approaches the desired solution for the Poisson equation.

In order to solve the equation over a region, we need to know on the boundary curve C enclosing the region either

(1) the values of ψ (Dirichlet condition)
(2) its' normal derivatives (Neuman condition)
or (3) a combination of the two.

Applying FTCS, (6) can be rewritten as

$$\psi_{i,j}^{n+1} = \psi_{i,j}^{n} + \frac{\Delta t}{\Delta x^2}\left(\psi_{i+1,j}^{n} + \psi_{i-1,j}^{n} + \psi_{i,j-1}^{n} + \psi_{i,j+1}^{n} - 4\psi_{i,j}^{n}\right) - \Delta t\, \zeta_{i,j} \tag{7}$$

assuming $\Delta x = \Delta y$ for the time being

The errors for the iterative values $\psi_{i,j}^{n}$ are then

$$\varepsilon_{i,j}^{n} = \underbrace{\psi_{i,j}}_{\text{exact value or } "\infty" \text{ iterations}} - \psi_{i,j}^{n} \tag{8}$$

(7) can be rewritten as

$$\varepsilon_{i,j}^{n+1} = \varepsilon_{i,j}^{n} + \frac{\Delta t}{\Delta x^2}\left[\varepsilon_{i+1,j} + \varepsilon_{i-1,j} + \varepsilon_{i,j+1} + \varepsilon_{i,j-1} - 4\varepsilon_{i,j}\right]$$

which is independent of ζ and therefore the stability properties of (6) are not affected by ζ

The Courant criterion for the difference equation in two dimension is $\alpha \frac{\Delta t}{\Delta x^2} \leq 1/4$ (instead of $1/2$ for one dimension).
For $\alpha = 1$, the criteria is $\Delta t \leq \Delta x^2/4$
Since we wish to approach the asymptotic solution as fast as possible, we consider the largest $\Delta t = \frac{\Delta x^2}{4}$ which gives for (7)

(9) $$\psi_{i,j}^{n+1} = 1/4 \left[\psi_{i+1,j}^{n} + \psi_{i-1,j}^{n} + \psi_{i,j+1}^{n} + \psi_{i,j-1}^{n} - \Delta x^2 \xi_{i,j} \right]$$

This is the solution by Method #1 for $\Delta x = \Delta y$
Each ψ^{n+1} is calculated independent of the sequence in (i,j) and therefore in a sense simultaneously.

If we define a mesh ratio $\delta = \frac{\Delta x}{\Delta y}$, the same method gives

(10) $$\psi_{i,j}^{n+1} = \frac{1}{2(1+\delta^2)} \left[\psi_{i+1,j}^{n} + \psi_{i-1,j}^{n} + \delta^2 \psi_{i,j+1}^{n} + \delta^2 \psi_{i,j-1}^{n} - \Delta x^2 \xi_{i,j} \right]$$

The analysis of the convergence rate can proceed from the analysis of the error equation.

$$E^{k+1} = G E^{k}$$

Tuesday, March 5

Fraenkel (1950):
The highest and lowest wavelength error components damp most slowly. Thus regardless of the initial error distribution, these components will dominate for k large.

Now we can improve method #1.
Equation (10) is a two-time level equation

which requires storage of $\psi_{i,j}^{n+1}$ and $\psi_{i,j}^{n}$
If we sweep the $\psi_{i,j}^{n}$ by new values whenever it is possible in equation (10), then we obtain

$$\psi_{i,j}^{n+1} = \frac{1}{2(1+\delta^2)} \left[\psi_{i+1,j}^{n} + \psi_{i-1,j}^{n+1} + \delta^2 \psi_{i,j-1}^{n+1} + \delta^2 \psi_{i,j+1}^{n} - \Delta x^2 \zeta_{i,j} \right] \tag{11}$$

This is the solution by Method #2 and only one storage level is needed. Frankel (1950) showed that asymptotically, k Method #2 iterations are worth 2k Method #1 iterations and only require half the storage.

Now, it was also found that optimum convergence could be achieved by "over-relaxing" or "under-relaxing" depending on whether neighboring residuals were of the same or opposite sign.

Frankel (1950) developed a method of applying over relaxation to Method #2. It is called Successive Over-Relaxation or SOR. or Method #3

(11) can be rewritten as the following with the bracketed term multiplied by a relaxation coefficient α

$$\psi_{i,j}^{n+1} = \psi_{i,j}^{n} + \frac{\alpha}{2(1+\delta^2)} \left[\psi_{i+1,j}^{n} + \psi_{i-1,j}^{n+1} + \delta^2 \psi_{i,j+1}^{n} + \delta^2 \psi_{i,j-1}^{n+1} - \Delta x^2 \zeta_{i,j} - 2(1+\delta^2) \psi_{i,j}^{n} \right] \tag{12}$$

We saw in section c) that $1 \leq \alpha < 2$ for

convergence (over-relaxation). The optimum value α_o depends on the mesh, the shape of the domain and the type of boundary conditions.

For a Dirichlet problem in a rectangular domain of the size $(I-1)\Delta x$ by $(J-1)\Delta y$, Frankel (1950) showed that

$$\alpha_o = 2\left(\frac{1-\sqrt{1-\beta}}{\beta}\right)$$

with $$\beta = \left(\frac{\cos\left(\frac{\pi}{I-1}\right) + \delta^2 \cos\left(\frac{\pi}{J-1}\right)}{1+\delta^2}\right)^2$$

Because of its simplicity and effectiveness, the SOR method has been the most popular of the iterative methods for solving the Poisson equation in computational fluid dynamics problems. However, this method takes considerable computer time and has been now replaced by faster, accurate direct methods in most problems.

The SOR method is very flexible and can be used under a wide range of conditions, including irregular boundaries, interior points .....

6.2. Direct methods

There is another class of solvers, the direct methods, that treat the finite difference equations as a large linear system and employ some tools from linear algebra

to operate on the matrix of the finite difference coefficients. These methods take advantage of the sparseness and regular structure of the coefficient matrices to minimize storage requirements and operation counts. Although they require more storage than the iterative methods, they require fewer operations.

Let's now consider the Poisson equation on a rectangle

(13) $$\nabla^2 \psi = \zeta$$ ($n \times m$ domain)

As described at the beginning of this chapter, the FD form can be expressed as

(14) $$M\phi = g$$ in matrix form

where M is of the form

$$\begin{pmatrix} B & I & 0 & \cdots & & \\ I & B & I & 0 & \cdots & \\ 0 & I & B & I & 0 & \cdots \\ \vdots & & \ddots & \ddots & \ddots & \\ & & & I & B & I \\ & & & & I & B \end{pmatrix}$$

($m-2$ Dirichlet + n cyclic) index k

The vectors ϕ and g are vectors of subvectors ϕ_k of solution values along the k^{th} row of the mesh.

(correspond to the S_i and L_i of the beginning of the chapter)

and B of the form

$$\begin{pmatrix} -4 & 1 & 0 & \cdots & & \\ 1 & -4 & 1 & 0 & \cdots & \\ 0 & 1 & -4 & 1 & 0 & \\ \vdots & \vdots & & \ddots & & 1 \\ & & & & 1 & -4 \end{pmatrix}$$

index j ($n-2$ Dirichlet + n cyclic)

The matrix A is very sparse and shows a very regular block structure. All the direct methods take advantage of this structure. Irregularly shaped boundaries destroy this structure and thus prohibit the use of direct method.

a) <u>Generalized form of Hockney's method or Fourier Transform method</u>

We define the Matrix Q such that

$$Q^{-1} B Q = \Lambda$$

where $\Lambda = \begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_M \end{pmatrix}$. Thus Q is the matrix whose colums are the eigenvectors of B and the λ_i, the corresponding eigenvalues

We can also define the transform

$$\bar{\phi}_k = Q^{-1} \phi_k \quad ; \quad G_k = Q^{-1} g_k \tag{15}$$

For the k^{th} row of (14), we have

$$I \phi_{k-1} + B \phi_k + I \phi_{k+1} = g_k$$

Multiplying by Q^{-1} and using (15)

$$I \bar{\phi}_{k-1} + \underbrace{Q^{-1} B Q}_{\text{eigenvalues}} \bar{\phi}_k + I \bar{\phi}_{k+1} = G_k$$

Then for each value of λ_ν, eigenvalue, we get a tridiagonal system

$$\bar{\phi}_{\nu,k-1} + \lambda_\nu \bar{\phi}_{\nu,k} + \bar{\phi}_{\nu,k+1} = G_{\nu,k} \tag{16}$$

Each of these tridiagonal systems can be easily solved by Gaussian elimination. The solution is then given by the inverse transform

$$\phi_k = Q \Phi_k \tag{17}$$

* In practice, if the Fourier transform of (15) are defined as

$$\begin{cases} \phi_{j,k} = \sum_\nu \Phi_{\nu,k}\, e^{i\frac{2\pi\nu j}{n}} \\ g_{i,k} = \sum_\nu G_{\nu,k}\, e^{i\frac{2\pi\nu i}{n}} \end{cases}$$

FFT transform

and are substituted in the FD form of the Poisson's equation, then for the part j,k we get

$$\begin{aligned} &\sum_\nu \Phi_{\nu,k-1}\, e^{i\frac{2\pi\nu j}{n}} + \sum_\nu \Phi_{\nu,k}\, e^{i\frac{2\pi\nu(j-1)}{n}} \\ &+ \text{———} \; e^{i\frac{2\pi\nu j+i}{n}} + \sum_\nu \Phi_{\nu,k+1}\, e^{i\frac{2\pi\nu j}{n}} \\ &- 4 \sum_\nu \Phi_{\nu,k}\, e^{i\frac{2\pi\nu j}{n}} = \sum_\nu G_{\nu,k}\, e^{i\frac{2\pi\nu j}{n}} \end{aligned} \tag{18}$$

Due to the orthogonality of the $e^{i\frac{2\pi\nu j}{n}}$ terms, we have n independent tridiagonal systems for the $\Phi_{\nu,k}$

$$\Phi_{\nu,k-1} + \left(-4 + e^{i2\pi\nu/n} + e^{-i\frac{2\pi\nu}{n}}\right)\Phi_{\nu,k} + \Phi_{\nu,k+1} = G_{\nu,k} \tag{19}$$

so that $\lambda_\nu = -4 + 2\cos(2\pi\nu/n)$

Once these tridiagonal systems are solved the solution is recovered using a back-transform or inverse ~~FFT~~.

Tuesday March 7 1995

b) Cyclic reduction

Hockney recommended for efficiency to reduce the order of the diagonal systems by applying one or more passes of cyclic reduction (or odd-even reduction) before performing the Fourier Transform

For the k^{th} row, (k even) we have

$$\left.\begin{aligned} I\phi_{k-2} + B\phi_{k-1} + I\phi_k &= g_{k-1} \\ I\phi_{k-1} + B\phi_k + I\phi_{k+1} &= g_k \\ I\phi_k + B\phi_{k+1} + I\phi_{k+2} &= g_{k+1} \end{aligned}\right\} \quad (20)$$

Multiplying the second by $-B$ and adding the three gives

$$I\phi_{k-2} + (2I - B^2)\phi_k + I\phi_{k+2} = g_{k-1} + g_{k+1} - Bg_k \quad (21)$$

The system is now reduced to only even-numbered rows. At this point, we can apply the Fourier Transform method on the even rows and then use (20) to get the odd rows. This is referred to as Hockney's method

One does not have to perform the Fourier Transform. If m is a power of 2, $m = 2^{l+1}$, the reduction process can be performed until we are left with only one row of unknowns to solve for.

We can define the recursion

$$\begin{cases} B^{(p+1)} = 2I - \left(B^{(p)}\right)^2 \\ g_k^{(p+1)} = g_{k-2^p}^{(p)} + g_{k+2^p}^{(p)} - B^{(p)} g_k \end{cases}$$

$$\text{for } \begin{cases} k = 2^p, \; m - 2^p, \text{ every } 2^p \\ p = 1, \ldots, l \end{cases}$$

After l reductions, we are left with

$$I \phi_0 + B^{(l)} \phi_{2^l} + I \phi_m = g_{2^l}^{(l)} \tag{22}$$

ϕ_0 and ϕ_m are known from the boundary conditions $\Rightarrow$ ϕ_{2^l} can be found easily

Advantages: * Every $g_k^{(p+1)}$ can overwrite the previous $g_k^{(p)}$ $\Rightarrow$ little storage needed

* Each $B^{(p)}$ is a polynomial of B of order 2^p and can be expressed as a sequence of tridiagonal matrices

$$B^p = A_1 A_2 A_3 \cdots A_{2^p}$$

Disadvantage: The computation of $g^{(p)}$ is subject to severe round-off errors (Buzbee et al., 1970) $\Rightarrow$ unstable for many values of l

There are various ways to stabilize the method ("Buneman variable or cyclic reduction" or "the Buneman algorithm"). These algorithms are mathematically identical to the previous description, but are not prone to round-off errors.

c) Block method

Used mostly for the general form of an elliptic equation which cannot be solved by the two previous methods and is too time consuming if done by iterative methods

General form of an elliptic equation

$$\left.\begin{aligned} &a(x,y)\,\phi_{xx} + b(x,y)\,\phi_{xy} + c(x,y)\,\phi_{yy} + \\ &d(x,y)\,\phi_x + e(x,y)\,\phi_y + f(x,y)\,\phi = g(x,y) \end{aligned}\right\} \quad (23)$$

In finite-difference form, the matrices are

$$M\phi = \begin{bmatrix} A_1 & C_1 & & & \\ B_2 & A_2 & C_2 & & \\ & B_3 & A_3 & C_3 & \\ & & \ddots & \ddots & \ddots \\ & & & B_m & A_m \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \vdots \\ \phi_m \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ \vdots \\ g_m \end{bmatrix}$$

with

$$A_K = \begin{bmatrix} f_{1,K} - 2\left(\frac{a_{1,K}}{\Delta X^2} + \frac{C_{1,K}}{2\Delta X}\right) & \frac{a_{1,K}}{\Delta X^2} + \frac{d_{1,K}}{2\Delta X} & & \\ \frac{a_{2,K}}{\Delta X^2} - \frac{d_{2,K}}{2\Delta X} & f_{2,K} - 2\left(\frac{a_{2,K}}{\Delta X^2} + \frac{C_{2,K}}{\Delta Y^2}\right) & \ddots & \\ & \ddots & \ddots & \ddots \end{bmatrix}_{n \times n}$$

$$B_K = \begin{bmatrix} \frac{C_{1,K}}{\Delta Y^2} - \frac{e_{1,K}}{2\Delta Y} & \frac{-b_{1,K}}{4\Delta X \Delta Y} & & \\ \frac{b_{2,K}}{4\Delta X \Delta Y} & \ddots & \ddots & \\ & \ddots & & \end{bmatrix}_{n \times n}$$

$$C_K = \begin{bmatrix} \frac{C_{1,K}}{\Delta Y^2} + \frac{e_{1,K}}{2\Delta Y} & \frac{b_{1,K}}{4\Delta X \Delta Y} & & \\ \frac{-b_{2,K}}{4\Delta X \Delta Y} & \ddots & \ddots & \\ & \ddots & & \end{bmatrix}_{n \times n}$$

$$\phi_K = \begin{bmatrix} \phi_{1,K} \\ \phi_{2,K} \\ \vdots \\ \phi_{n,K} \end{bmatrix} \qquad g_K = \begin{bmatrix} g_{1,K} \\ g_{2,K} \\ \vdots \\ g_{n,K} \end{bmatrix}$$

A can be factored. $A = LU$

L = Lower Triangle

U = Upper ———

$$L = \begin{bmatrix} I & 0 & & \\ \bar{B}_2 & I & 0 & \\ & \bar{B}_3 & I & 0 \\ & & \ddots & \ddots \\ & & & \bar{B}_m \; I \end{bmatrix} \qquad U = \begin{bmatrix} \bar{A}_1 & C_1 & 0 & \\ 0 & \bar{A}_2 & C_2 & 0 \\ & 0 & \bar{A}_3 & C_3 \\ & & \ddots & \ddots \\ & & & \bar{A}_m \end{bmatrix}$$

$\bar{A}_k$ and $\bar{B}_k$ are given by

$$\begin{cases} \bar{A}_{k-1} \bar{B}_k = B_k \\ \bar{A}_k = A_k - \bar{B}_k C_{k-1} \end{cases} \qquad \bar{A}_1 = A_1 \tag{24}$$

On a vector computer, these recursions can be performed very efficiently since A_k, B_k, C_k are tridiagonal. To obtain ϕ_k, we first solve $L\phi = g$ (forward sweep) by

(25) $$\phi^*_k = g_k - \bar{B}_k \phi_{k-1} \qquad (\phi_1 = g_1)$$
$k = 2, m$

Then $U\phi = \phi$ is solved (backward sweep)

(26) $$\begin{cases} A_m \phi_m = \phi^*_m \\ A_k \phi_k = \phi^*_k - C_k \phi^*_{k+1} \end{cases} \qquad k = m-1, 1$$

This method can handle very general boundary conditions, but are as before restricted to rectangular domain. Very fast on a vector machine, but is <u>machine specific</u>. Canned subroutines are available.

fg ch Poisson 91