PART II. Incompressible flow

We would now like to turn our attention to finite-difference methods for incompressible flow. The governing equations are conservation of mass, which can be written for incompressible flow as:

$$\nabla \cdot \vec{u} = 0 \tag{1}$$

so that the velocity field is divergence-free, and conservation of momentum:

$$\frac{\partial \vec{u}}{\partial t} + (\vec{u} \cdot \nabla)\,\vec{u} = -\frac{1}{\rho}\nabla p + \nu \Delta \vec{u} \tag{2}$$

where $\vec{u}$ is the velocity; ρ is density; p is pressure; and ν is the kinematic viscosity. The del operator in two-dimensional Cartesian coordinates is:

$$\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right) \tag{3}$$

and the Laplacian is:

$$\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \tag{4}$$

The approach to the full momentum equation may be simplified by considering separately the following three subcategories:

(1) **Nonlinear advection.** This is a problem which by now we understand pretty well. The governing equation is:

$$\frac{\partial s}{\partial t} + (\vec{u} \cdot \nabla)\, s = 0 \tag{5}$$

For the model problem, we can think of s as some scalar. This step is the design point of the full approach, since the errors introduced by the hyperbolic methodology are the most troubling. The time-step restriction enters through a limitation on the CFL number, σ:

$$\sigma = \frac{|\vec{u}|\,\Delta t}{\Delta x} \le C_a \tag{6}$$

where C_a is some constant value less than 1.

(2) **Diffusion**. The model equation for this piece of the puzzle is:

$$\frac{\partial s}{\partial t} = \varepsilon \Delta s \tag{7}$$

where ε represents some diffusion coefficient. If we were to use an explicit method to solve this equation, the time step restriction is:

$$\frac{\varepsilon\Delta t}{\Delta x^2} \le C_d \tag{8}$$

(3) **Poisson equation.** The continuity equation for incompressible flow can be thought of as a constraint on the velocity field, specifically that the velocity field is divergence-free. This constraint allows us to develop another model equation to examine the influence of the field induced by the pressure gradient. If we apply the divergence operator to the momentum equation (2) above, switch the order of differentiation, and utilize the continuity equation (1), we obtain:

$$\nabla \cdot [(\vec{u} \cdot \nabla)\vec{u}] = -\frac{1}{\rho}\Delta p \tag{9}$$

or, rewriting:

$$\Delta p = -\rho\nabla \cdot [(\vec{u} \cdot \nabla)\vec{u}] \tag{10}$$

which suggests the use of the model equation:

$$\Delta\phi = \rho \tag{11}$$

A discussion of step (1) above was given in part I. To examine steps (2) and (3) above, we will be looking at fundamentally different solution techniques. Rather than explicitly solving for the temporal evolution of u, we will be discussing *implicit* methods. For example, to solve for p in equation (10) above, we could discretize the Laplacian operator, and solve the resulting system of linear equations to obtain p. Since the inversion will likely become a rather cumbersome process involving large matrices, we would be wise to also explore iterative methods and multigrid. We first look at the Poisson equation in section 1, and in section 2 explore the model advection equation.

1. Poisson's equation.

We want to focus our attention on the Poisson equation:

$$\Delta\phi = \rho \tag{12}$$

We can think of the quantity ϕ in more general terms as representing a potential field, and ρ as the associated charge density which induces that field. Or we can think of a linear distribution of temperature at steady-state caused by diffusion and a heat source term. Although we will discuss only the two-dimensional case, the extension to three dimensions is just a matter of bookkeeping. We want to define ρ and ϕ at the cell centers on a square 2D grid with equal mesh spacing, h, in both dimensions, as shown in figure +1 below.

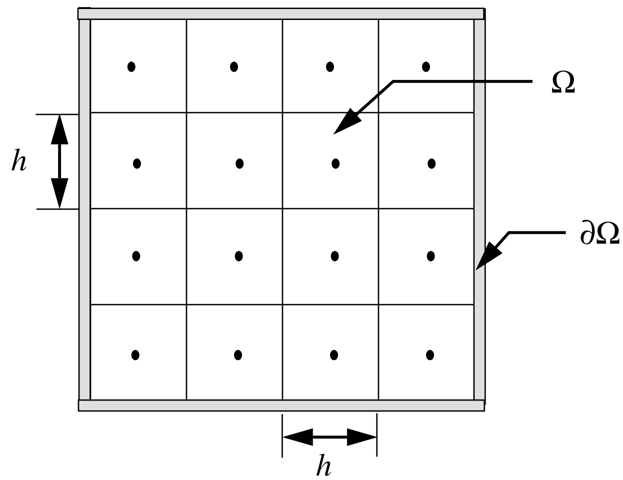


FIGURE 1. General setup of domain.

The dimensions in physical space will be denoted by $\underset{\sim}{x} = (x, y)$ and range from *0* to *1,* while the dimensions in logical space will be $\underset{\sim}{i} = (i, j)$ and extend from *1* to *N* in both directions. We will consider two different sets of boundary conditions: doubly periodic ($\phi(0, y) = \phi(1, y)$; $\phi(x, 0) = \phi(x, 1)$), and initially homogeneous Dirichlet conditions ($\phi = 0$ on $\partial\phi$). These boundary conditions will make Fourier analysis and numerical implementation simpler. At the moment, we will punt on the question of how to make the world conform to our choice of boundary conditions (see section xxx.xx for a discussion of boundary conditions).

To discretize the Laplacian:

$$\Delta\phi = \frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2}$$

$$\approx \frac{\phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j}}{h^2} + \frac{\phi_{i,j+1} - 2\phi_{i,j} + \phi_{i,j-1}}{h^2}$$

or, introducing new notation for the discretized form of the standard five-point Laplacian:

$$(\Delta^h\phi)_{ij} = \frac{\phi_{i+1,j} + \phi_{i-1,j} - 4\phi_{i,j} + \phi_{i,j+1} + \phi_{i,j-1}}{h^2}$$

On the interior of the domain, the discretized form of Poisson's equation is:

$$(\Delta^h\phi)_{ij} = \rho_{ij} \tag{13}$$

The boundary conditions will dictate how we deal with the points near the boundary. For periodic b.c., we will extend the solution in a periodic way, e.g., $\phi_{0,j} = \phi_{N,j}$. For Dirichlet b.c., we will extend the domain and linearly extrapolate so that the value of ϕ at the boundary is nil, i.e., $\phi_{o,j} = -\phi_{1,j}$, as shown in figure +1 below:

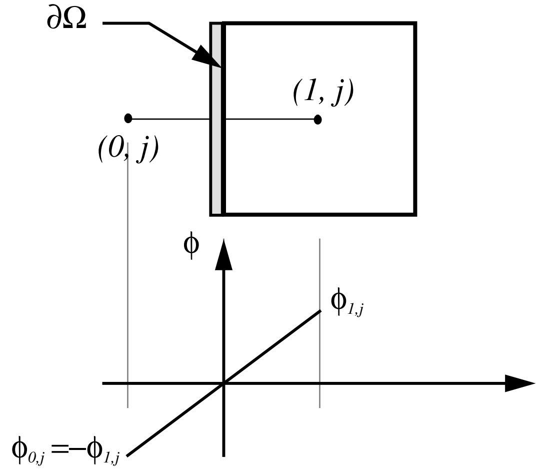


FIGURE 2. Treatment of Dirichlet boundary conditions

Each cell in the interior must satisfy the Poisson equation, while each cell on the edge of the domain must satisfy boundary conditions. The result is that we end up with a big matrix equation of the form:

$$[A]\underset{\sim}{\phi} = \underset{\sim}{\rho} \tag{14}$$

We can solve for $\underset{\sim}{\phi}$ by calculating $\underset{\sim}{\phi} = [A]^{-1}\rho$. Therefore, we want to invert the matrix *[A]* with as little grief as possible. We next examine the merits of direct solvers, iterative and multigrid methods.

1.1 Direct solvers

One possibility is using brute force in the form of Gaussian elimination with partial pivoting and *LU* decomposition. (Further discussion of this technique is available in any number of classical books on numerical methods.) But this turns out to be a poor option since there are N^2 equations and N^2 unknowns. The number of operations for a general linear system would be of $O((N^2)^3) = O(N^6)$. This is unacceptably large, especially if we contrast this with that of a hyperbolic problem, which is more typically of $O(N^2)$ operations. We note that the structure of the matrix depends on the order in which we write the equations. If we set up the system with a modicum of wisdom, we can obtain a matrix which is dense in nonzero values near the diagonals and full of zeroes in the upper left and lower right corners. It is easier to invert such a matrix than a fully populated one. The bandwidth, i.e., the number of lines of nonzero values off the diagonal, is of order N for Dirichlet boundary conditions, as shown in figure +1, below.

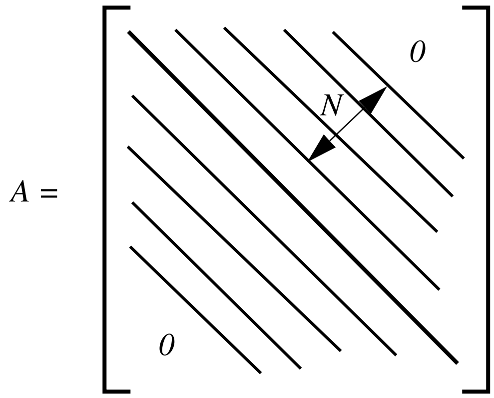


FIGURE 3. Bandwidth structure of the matrix *A*

The number of operations needed for the LU decomposition of this matrix is roughly the number of equations multiplied by the square of the bandwidth, or approximately $O(N^4)$. We also have to consider that the result of the LU decomposition will need to be stored in $O(N^3)$ floating-point memory locations. By comparison, the explicit hyperbolic solver to which this scheme will be coupled requires $O(N^2)$ storage locations, and the same number of floating point operations. For this reason, we seek out methods for which the computational effort and storage required is closer to that of for the hyperbolic solver. We are left with the task of examining iterative solutions, the topic of the next section.

1.2 Iterative solutions.

Ultimately, we want to solve the steady-state Poisson equation:

$$\Delta\phi = \rho \tag{15}$$

For iterative solutions, we form the associated time evolution equation for $\tilde{\phi}$ and carry the solution to "steady state". Instead of solving immediately for $\phi(\vec{x})$, we will first consider $\tilde{\phi}(\vec{x}, t)$ which satisfies:

$$\frac{\partial\tilde{\phi}}{\partial t} = \Delta\tilde{\phi} - \rho \tag{16}$$

We expect that as $t \to \infty$, that $\tilde{\phi}(\vec{x}, t) \to \phi(\vec{x})$. How can be convinced of this statement? Let's define the remainder as:

$$\delta = \tilde{\phi} - \phi \tag{17}$$

What equation does δ satisfy? Since $\tilde{\phi} \neq \phi(t)$ and the system is linear, it is easy to show that

$$\frac{\partial \delta}{\partial t} = \Delta \delta \tag{18}$$

For doubly periodic boundary conditions, we can use a Fourier transform to diagonalize the system and show that all Fourier modes decay as $t \to \infty$. We ought to point out at this point that "time" has a different meaning in this discussion as compared with explicit solutions of unsteady hyperbolic problems. We're not really marching in time for this case, but rather we're iterating somewhat abstractly to obtain the steady-state solution to Poisson's equation. With this clarification in mind, if we let the iterative index be l and the time step be λ, we can write the discretized iterative equation for $\tilde{\phi}$ as:

$$\tilde{\phi}^{l+1} = \tilde{\phi}^{l} + \lambda (\Delta^h \tilde{\phi}^{l} - \rho) \tag{19}$$

Since we will be solving the above equation iteratively as the simulation time $t \to \infty$, we would like the time step λ to be as large as possible.

1.2.1 Convergence and stability

If we define the remainder δ^l:

$$\delta^l = \tilde{\phi}^{l} - \phi \tag{20}$$

and use the definition $\Delta^h \phi = \rho$, we can show that δ must satisfy:

$$\delta^{l+1} = \delta^l + \lambda \Delta^h \delta^l \tag{21}$$

The question then becomes one of how quickly does the solution converge so that $\delta^l \to 0$. More formally, for some operator L where:

$$L\delta = (I + \lambda \Delta^h)\,\delta \tag{22}$$

Does $L\delta$ reduce the norm of δ? Is it true that:

$$\| L\delta \| \leq \| \delta \| \tag{23}$$

The natural norm to choose is the L_2 norm, defined by:

$$\| \Phi \|_2^2 = h^2 \sum |\Phi_{i,j}|^2 \tag{24}$$

where the summation is over the entire domain. For doubly periodic boundary conditions, Fourier analysis indicates:

$$\delta_{j_x, j_y} = \sum_{k_x = -\frac{N}{2}+1}^{\frac{N}{2}+1} \sum_{k_y = -\frac{N}{2}+1}^{\frac{N}{2}+1} a_{k_x} a_{k_y} e^{2\pi i (k_x j_x + k_y j_y) h} \tag{25}$$

where index space is temporarily denoted by *(j_x, j_y)* rather than *(i, j)* to avoid confusion with the $i = \sqrt{-1}$ in the exponential expression of equation (-1) above. Then

$$(L\delta)_{j_x, j_y} = \sum_{k_x = -\frac{N}{2}+1}^{\frac{N}{2}+1} \sum_{k_y = -\frac{N}{2}+1}^{\frac{N}{2}+1} a_{k_x} a_{k_y} \Lambda(\beta_x, \beta_y) e^{2\pi i (k_x j_x + k_y j_y) h} \tag{26}$$

where the wavenumber in the *x* direction is defined as:

$$\beta_x = 2\pi i k_x h \tag{27}$$

and similarly in the *y* direction:

$$\beta_y = 2\pi i k_y h \tag{28}$$

We also define Λ as the result of extending the shift operation to two dimensions, i.e., Λ is the symbol of the scheme (recall the discussion of symbols in section xxx.xx.x). From a simpleminded comparison of equations (-3) and (-4) above, we see that our condition for stability is met (i.e., as we slap *L* on δ, the norm decreases), if $\Lambda < 1$. If the operator *L* represents the iterative equation (22) above and using the standard five-point cell-centered Laplacian, then we can write the operator as:

$$L = I + \frac{\lambda}{h^2} [S_x + S_x^{-1} + S_y + S_y^{-1} - 4I] \tag{29}$$

where *S*'s represent shifts and *I* is the identity. The symbol $\Lambda(\beta_x, \beta_y)$ of the scheme is:

$$\begin{aligned} \Lambda(\beta_x, \beta_y) &= 1 + \frac{\lambda}{h^2} [e^{i\beta_x} + e^{-i\beta_x} + e^{i\beta_y} + e^{-i\beta_y} - 4] \\ &= 1 + \frac{\lambda}{h^2} [2\cos\beta_x + 2\cos\beta_y - 4] \end{aligned} \tag{30}$$

To obtain $|\Lambda| \le 1$ we need

$$-2 \le \frac{\lambda}{h^2} [2\cos\beta_x + 2\cos\beta_y - 4] \le 0 \tag{31}$$

It is clear that the expression in brackets above is never positive. Therefore, we need to show that

$$\frac{\lambda}{h^2}\left[2\cos\beta_x + 2\cos\beta_y - 4\right] \geq -2 \tag{32}$$

Since the minimum value on cosine is -1, we can show that the condition in equation (32 above is met when

$$-\frac{8\lambda}{h^2} > -2$$

or

$$\frac{\lambda}{h^2} < \frac{1}{4} \tag{33}$$

When the time step is less than $h^2/4$, all of the Fourier modes of δ eventually decay. How many iterations will this take? How much does each Fourier amplitude decrease as a function of β_x and β_y?

One thing which we can do is examine the smooth parts of the solution. This means looking at the long-wavelength limit of the solution as $\beta_x, \beta_y \longrightarrow 0$. Using a Taylor expansion about $\beta_x = \beta_y = 0$:

$$\begin{aligned}
\Lambda(\beta_x, \beta_y)\big|_{\beta_x = \beta_y = 0} &= 1 + \frac{\lambda}{h^2}\left\{2\left[1 - \frac{\beta_x^2}{2} + \ldots\right] + 2\left[1 - \frac{\beta_y^2}{2} + \ldots\right] - 4 \quad \right\} \\
&\approx 1 - \frac{\lambda\left[\beta_x^2 + \beta_y^2\right]}{h^2} \\
&= 1 - \frac{\lambda\left[2\pi\right]^2\left[(k_x h)^2 + (k_y h)^2\right]}{h^2}
\end{aligned}$$

or

$$\Lambda(\beta_x, \beta_y)\big|_{\beta_x = \beta_y = 0} \approx 1 - 4\lambda\pi^2\left[k_x^2 + k_y^2\right] \tag{34}$$

From stability considerations, we have shown in equation (-2) above that $\lambda \leq \frac{h^2}{4}$. We can then deduce that the dependence of the symbol on the mesh spacing is:

$$\Lambda(0, 0) \approx 1 - Ch^2 \tag{35}$$

where C is some constant. This is a slow rate of convergence.

We can also consider the behavior of high-wavenumber modes. We note that the highest wavenumber is π. This means that if $\beta_x = \beta_y = \pi$, the symbol becomes:

$$\Lambda(\pi,\pi) = 1 + \frac{\lambda}{h^2}[2(-1) + 2(-1) - 4]$$

$$= 1 - \frac{8\lambda}{h^2}$$

However, if $\lambda = \frac{h^2}{8}$, we arrive at the result:

$$\Lambda(\pi,\pi)\big|_{\lambda = h^2/8} = 0 \tag{36}$$

This finding indicates that this scheme is remarkably efficient at damping out the high-wavenumber modes with the right value of λ. In practice, we find that modes in the range $\frac{\pi}{2} \le \beta_x, \beta_y \le \pi$ decay rapidly (within a few iterations) if we choose $\lambda = h^2/8$. Therefore, it is really the long wavelengths which we must worry about.

Note that "long" and "short" wavelengths are measured relative to the mesh spacing *h*. We would expect short-wavelength disturbances to decay within a few iterations. The remainder δ could at that point be represented accurately on a coarser mesh, say double the original mesh spacing. If we then ran for a few iterations on this coarser mesh, we'd get rid of somewhat longer wavelengths. The idea, then, is to relax on a series of continually coarser grids in order to destroy successively longer-wavelength components of the remainder. (We have assumed (so far) that the total number of mesh points is a power of 2.)

Place somewhere relevant: For further discussion of the multigrid technique, see Briggs (19xx). "A Multigrid Tutorial." SIAM Publications.

We can also talk about the residual problem associated with Poisson's equation. Assume we have an estimate of $\tilde{\phi}$ at some time. The residual matrix *R* is given by:

$$R = -\Delta^h\tilde{\phi} + \rho \tag{37}$$

Using the residual as a source term, we can solve for:

$$\Delta^h\delta = R \tag{38}$$

and then use the result to update the solution:

$$\phi = \tilde{\phi} + \delta \tag{39}$$

These steps are equivalent to solving $\Delta^h\phi = \rho$. How can we be sure of this?

Proof: What is the solution to $\Delta^h\delta = R$?

$$\Delta^h\delta = R$$

$$= \rho - \Delta^h\tilde{\phi}$$

$$\delta = \Delta^{-1}\rho - \tilde{\phi}$$

Using the definition of δ:

$$-\tilde{\phi} + \Delta^{-1}\rho = \Delta^{-1}\rho - \tilde{\phi}$$

We are being consistent.

1.2.2 Procedure for implementation of multigrid

We would like to solve the discrete Poisson equation $\Delta^h\phi = \rho$ using multigrid on a square mesh of dimension $N \times N$, where $N = 2^m$. We will discuss the mechanics of implementing this scheme by considering the subroutine

MG_relax(ρ, h, l, $\tilde{\phi}$)

The inputs are: ρ, which is a generic forcing term on the r.h.s.; h is the mesh spacing, l is the current level (which represents the coarseness of the mesh); and $\tilde{\phi}$ is the first guess at the solution ϕ. The subroutine returns a new approximation to ϕ. Within the subroutine, the following steps are taken:

(1) **Relaxation scheme:** Update the guess for $\tilde{\phi}$ over all points (i, j) by the local relaxation scheme. So far, the only one we have considered is of the form:

$$\tilde{\phi}^{l+1} = \tilde{\phi}^{l} + \lambda\left(\Delta^h\phi^l - \rho\right) \tag{40}$$

Perform this operation p times. The result is that we get rid of the high-frequency components of the remainder δ.

(2) **Calculate the residual:** As defined previously,

$$R = \rho - \Delta^h\phi \tag{41}$$

We know that the current guess satisfies $\Delta^h\delta = R$. In addition, all high-wavenumber components of R are damped if the high-wavenumber components of δ are damped, since this is an elliptic problem. Thus, R and δ can be represented accurately on a coarser grid.

(3) **Average the residual onto a coarser grid:** Let R^c denote the averaged R on the coarser grid. As a matter of fact, let's erupt in a flurry of new notation: let the new coarse level be $l^c = l - 1$; $h^c = 2h$; and the new first guess on the coarse grid to be $\tilde{\delta}^c = 0$. Dimensionally, the source term R looks like a charge; we would like to conserve charge in forming R^c. We have specified that the mesh spacing h is equivalent in both

the x and y directions. The interpolated value for the residual on the coarse mesh can then be seen as the average value of the component cells of the fine mesh, as shown in figure +1.

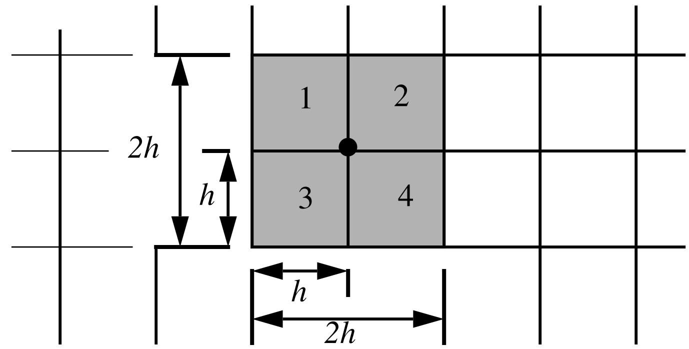


FIGURE 4. Calculation of residual on the coarse grid

We can then calculate the residual on the coarse grid with mesh spacing *2h x 2h* from the fine-grid mesh cells 1 through 4 by:

$$R^c = \frac{R^1 + R^2 + R^3 + R^4}{4} \tag{42}$$

Note that if the mesh spacing is not uniform, a logical approach would be to use weighting factors which account for the differing areas on the fine-grid cells.

(4) **Solve for the coarse-grid correction recursively.** As long as the level $l > 0$, we call the MG_relax routine recursively:

δ^c = MG_relax(R^c, h^c, l^c, $\tilde{\delta}^c$)

The effect of this step is to solve $\Delta^{2h}\delta^c = R^c$, thereby approximating δ on a coarser grid.

(5) **Interpolate the coarse-grid correction onto the finer grid, one level at a time.** We would like the interpolation operator I to be the adjoint of the averaging operator. For the averaging operator used above, the interpolation operator is simply $\delta^c = \delta_1 = \delta_2 = \delta_3 = \delta_4$. The next step is to update $\tilde{\phi} = \tilde{\phi} + \delta$.

(6) **Relax.** In this context, "relax" means that we perform

$$\tilde{\phi}^{l+1} = \tilde{\phi}^l + \lambda\,(\Delta^h \phi^l - \rho)$$

using the updated value of $\tilde{\phi}$ of step (5).

We know from stability considerations that λ must be less than $h^2/4$, and that if $\lambda = h^2/8$, short-wavelength oscillations are readily suppressed. Can we do better? What is the optimal point relaxation scheme? What happens if Δ^h is not invertible? These topics are addressed in the following sections.

1.2.2.1 Point Jacobi iteration

To update for new values of $\tilde{\phi}_{ij}$, using point Jacobi iteration:

$$\tilde{\phi}^{l+1}_{i,j} = \tilde{\phi}^{l}_{i,j} + \lambda\left[(\Delta^h \tilde{\phi}^l)_{i,j} - \rho_{i,j} \right] \tag{43}$$

If we choose the maximum allowable time step, $\lambda = h^2/4$. Using a five-point cell-centered Laplacian, equation (-1) becomes:

$$\begin{aligned} \phi^{l+1}_{i,j} &= \phi^{l}_{i,j} + \frac{h^2}{4}\left\{ \frac{[\tilde{\phi}^l_{i,j+1} + \tilde{\phi}^l_{i,j-1} + \tilde{\phi}^l_{i+1,j} + \tilde{\phi}^l_{i-1,j} - 4\tilde{\phi}^l_{i,j}]}{h^2} - \rho_{i,j} \right\} \\ &= \frac{1}{4}[\tilde{\phi}^l_{i,j+1} + \tilde{\phi}^l_{i,j-1} + \tilde{\phi}^l_{i+1,j} + \tilde{\phi}^l_{i-1,j}] - \rho_{i,j}\frac{h^2}{4} \end{aligned} \tag{44}$$

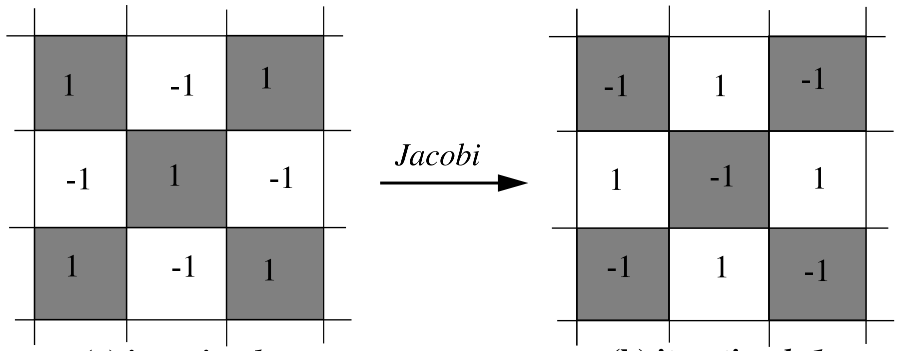
If we define

$$\delta^l = \tilde{\phi}^l - \phi \tag{45}$$

where ϕ is the exact solution to $\Delta^h \phi = \rho$, then

$$\delta^{l+1}_{i,j} = \frac{1}{4}[\delta^l_{i,j+1} + \delta^l_{i,j-1} + \delta^l_{i+1,j} + \delta^l_{i-1,j}] \tag{46}$$

We note that we have lost one entire subtraction by using this form relative to equation (-3). The chief advantage of looking at δ rather than ϕ is that we can set the boundary values of δ to zero for the case of any Dirichlet boundary conditions, which can simplify the numerics. In either case, we would expect high frequencies to be readily damped in a simulation using this iterative procedure, thus providing lots of smoothing. This is desirable; however, there is an important drawback to this formulation. Through this choice of time step, we've removed the dependence on the current value of $\tilde{\phi}$ at point *(i, j)*; we've also opened the door to a major headache because we've removed realistic communication between adjacent cells. Consider a checkerboard pattern of red and black. With this formulation, the red cells do not “talk” to the black cells and vice versa. If, for example, $\delta\phi$ attains the following values in the cells in figure +1:



(a) iteration *l* **(b) iteration *l+1***

FIGURE 5. Checkerboard pattern which spells trouble for point Jacobi

If this pattern is extended, the conscientious reader can verify that the cells flip helplessly from one value to the other with each iteration.

1.2.2.2 Gauss-Seidel relaxation with red/black ordering (GSRB)

A half-step incremental approach is used in Gauss-Seidel relaxation for which the values are updated in separate sweeps. Point-Jacobi iteration is used on half of the cells on the first pass, and on the other group on the second pass. In the first step:

$$\tilde{\phi}_{i,j}^{l+1/2} = \begin{cases} \tilde{\phi}_{i,j}^{l} + \lambda\left[(\Delta^h \tilde{\phi}^l)_{i,j} - \rho_{i,j} \right] \ldots if \ldots i+j = even \\ \tilde{\phi}_{i,j}^{l} \ldots if \ldots i+j = odd \end{cases} \tag{47}$$

Next, in step two:

$$\tilde{\phi}_{i,j}^{l+1} = \begin{cases} \tilde{\phi}_{i,j}^{l+1/2} \ldots if \ldots i+j = even \\ \tilde{\phi}_{i,j}^{l+1/2} + \lambda\left[(\Delta^h \tilde{\phi}^{l+1/2})_{i,j} - \rho_{i,j} \right] \ldots if \ldots i+j = odd \end{cases} \tag{48}$$

Or, for $\lambda = h^2/4$ and the standard five-point Laplacian:

$$\delta_{i,j}^{l+1/2} = \begin{cases} \frac{1}{4}\left[\delta_{i,j+1}^{l} + \delta_{i,j-1}^{l} + \delta_{i+1,j}^{l} + \delta_{i-1,j}^{l}\right] \ldots if \ldots i+j = even \\ \delta_{i,j}^{l} \ldots if \ldots i+j = odd \end{cases} \tag{49}$$

and

$$\delta_{i,j}^{l+1} = \begin{cases} \delta_{i,j}^{l+1/2} \ldots if \ldots i+j = even \\ \frac{1}{4}\left[\delta_{i,j+1}^{l+1/2} + \delta_{i,j-1}^{l+1/2} + \delta_{i+1,j}^{l+1/2} + \delta_{i-1,j}^{l+1/2}\right] \ldots if \ldots i+j = odd \end{cases} \tag{50}$$

The goal of using GSRB is to transfer the error from high to low wavenumbers; we will then let the coarser grids upon application of multigrid worry about the low-wavenumber errors. Ultimately, the red and black cells are linked via the boundary conditions. The checkerboard picture which goes along with Gauss-Seidel with red/black ordering is shown in figure +1 below:

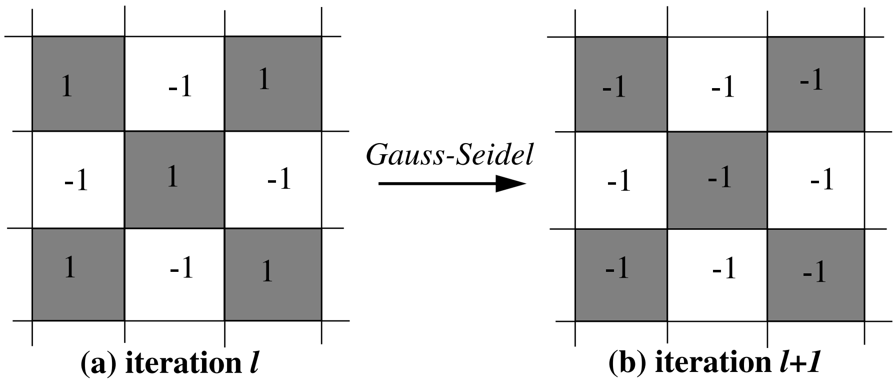


FIGURE 6. Updating the checkerboard using Gauss-Seidel relaxation

Note that this method extremely efficient at *eliminating*, not just damping, modes in the range of $\frac{\pi}{2} \leq \dot{\beta}_x, \beta_y \leq \pi$. We must also note that GSRB damps all other modes to some extent. However, it does not increase the error, i.e.,

$$\left\| \delta^{l+1} \right\| < \left\| \delta^{l} \right\| \tag{51}$$

We conclude that we can live with this since we are more worried about damping high wavenumber error and, after all, GSRB is stable, which is most desirable.

1.2.2.3 Solvability conditions -- the Fredholm alternative

We want to examine the consequences if Δ^h is not invertible. Given any rectangular matrix *[A]* (not even necessarily square any more), we would like to solve:

$$A\underset{\sim}{v} = \underset{\sim}{f} \tag{52}$$

This has a solution if and only if $\underset{\sim}{f}$ is orthogonal to the null space of A^T.[1] Stated another way, a solution exists if and only if, for all w for which $A^T w = 0$, the inner product $\langle \underset{\sim}{f}, \underset{\sim}{w} \rangle = 0$.

The necessity of this condition is easy to show. Given $A\underset{\sim}{v} = \underset{\sim}{f}$

$$A^T \underset{\sim}{w} = 0$$

The inner product of $\langle \underset{\sim}{f}, \underset{\sim}{w} \rangle$ is:

$$\begin{aligned} \langle \underset{\sim}{f}, \underset{\sim}{w} \rangle &= \langle A\underset{\sim}{v}, \underset{\sim}{w} \rangle \\ &= \langle \underset{\sim}{v}, A^T \underset{\sim}{w} \rangle \\ &= \langle \underset{\sim}{v}, 0 \rangle \\ &= 0 \end{aligned}$$

It is a little trickier to show that this is sufficient for a general matrix A. However, for a symmetric matrix the proof is straightforward. Assume A is p x p. We find a basis which diagonalizes A, using the right-eigenvector matrix, R so that:

$$R^{-1}AR = \begin{bmatrix} \lambda_1 & & 0 \\ & \dots & \\ 0 & & \lambda_p \end{bmatrix} \tag{53}$$

with

$$R = (r_1, \dots, r_p) \tag{54}$$

If, say, $\lambda_p = 0$, then we run into trouble when we try to solve $v_p = f_p / \lambda_p$.

We must also be aware that solutions are not unique. Assuming that $A\underset{\sim}{w} = 0$ (a different $\underset{\sim}{w}$ this time), we know that $A\underset{\sim}{v} = \underset{\sim}{f} = A(\underset{\sim}{v} + \underset{\sim}{w})$ for linear systems. The value of $\underset{\sim}{v}$ can be determined up to a constant. However, the nonuniqueness is not a problem in the physical application we are looking at. Recall that the motivation behind all of this is to solve Poisson's equation:

$$\Delta\phi = \rho \tag{55}$$

However, we may not really be interested in ϕ; what we may want to find out about is the field induced by ϕ, i.e., $\nabla\phi$. For example, in the case of incompressible fluids, this is the pressure gradient. We know that the addition of a constant to ϕ does not change the value of $\nabla\phi$.

1. $\underset{\sim}{x}$ is in the null space of the matrix A if $Ax_i = 0$. E.g., for Poisson's equation with periodic b.c., $\phi = Const$ is in the nullspace of Δ^h

There are two important implications of this analysis relevant to our discussion of point relaxation. First, we effectively toss out the nullspace component of the solution by looking at its gradient. Secondly, even if there is some component of null space in the initial guess of ϕ, then we expect that that component will not grow. We want to iteratively update ϕ by:

$$\phi^{l+1} = \phi^{l} + \lambda (A\phi^{l} - f) \tag{56}$$

Let's examine the r.h.s. of equation (-1) step by step. A applied to null space is zilch; f is orthogonal to null space. Therefore, there are no terms on the r.h.s which could increase the null space amplitude (except for roundoff error). Checking for the increase of ϕ in null space can be used as a handy debugging tool.

Let's apply this knowledge to the solution of Poisson's equation by multigrid. In that case, the matrix A represents the discretization of the Laplacian, Δ^h. The homogenous part of the solution corresponding to null space is $\phi^h = C$, where C is some constant. The solvability condition is that the inner product

$$\langle \rho, 1 \rangle = \sum \rho_{ij} = 0$$

Physically, this means that there is no net change in the charge. Examining the multigrid procedure step by step:

- **Relax.** There is no term on the r.h.s which increases the null space of ϕ when relaxing by: $\phi^{l+1} = \phi^{l} + \lambda (A\phi^{l} - f)$, as discussed above
- **Compute residual.** By calculating $R = \rho - \Delta^h \phi$, we know that R satisfies the solvability condition.
- **Average residual onto coarse grid.** Solvability is not violated when $R^c = [R_1 + R_2 + R_3 + R_4] / 4$
- **Relax for coarse correction**. δ^c has no null space component.
- **Interpolate.** The fine-grid correction $\delta = I\delta^c$ has no null space components, so that the new $\phi^{l+1} = \phi^{l} + \delta^c$ will not increase the null space.

1.2.2.4 Boundary conditions

So far, we've considered only homogeneous boundary conditions. What if we have inhomogeneous Dirichlet or Neumann boundary conditions? It is simple to convert the inhomogeneous boundary conditions to homogeneous b.c.'s with a modified right-hand side. Let's examine the case of inhomogeneous Dirichlet boundary conditions. We want to solve $\Delta\phi = \rho$ on the domain D subject to the boundary conditions $\phi = g$ on ∂D. Using cell-centered discretization as usual, we write the discretized form of the differential equation as:

$$\Delta^h \phi^h = \rho^h \tag{57}$$

Consider the left edge of the domain at $i_0 - 1/2$, as shown below in figure +1:

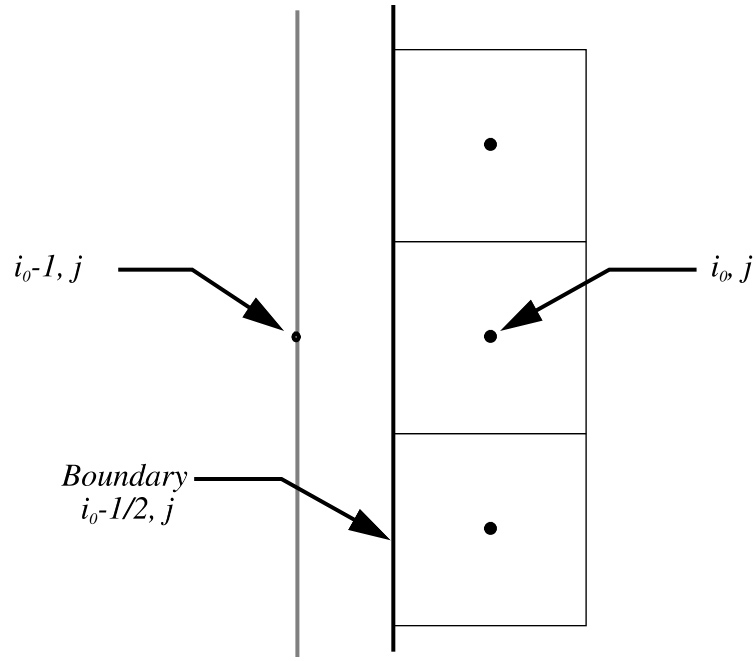


FIGURE 7. Left boundary...

We know from the boundary conditions that in the discrete domain at the boundary, ϕ is given exactly by:

$$\phi^h_{i_0 - 1/2, j} = g^h_{i_0 - 1/2, j} \tag{58}$$

where

$$g^h_{i_0 - 1/2, j} = g\left(\left(i_0 - 1/2 \right) h, jh \right) \tag{59}$$

We can also make a first-order finite-difference approximation considering the phantom node located one point inside the boundary:

$$g^h_{i_0 - 1/2, j} = \frac{\phi^h_{i_0 - 1, j} + \phi^h_{i_0, j}}{2} \tag{60}$$

so that

$$\phi^h_{i_0 - 1, j} = 2 g^h_{i_0 - 1/2, j} - \phi^h_{i_0, j} \tag{61}$$

What does all of this notation buy us (even if it doesn't cost anything)? We claim that we can find a boundary charge ρ^B_{ij} from this mess which we can simply add to the original right-hand side to obtain an equivalent problem for ϕ. For the column of nodes one point inside the domain, at $i = i_{0:}$

$$\Delta^h \phi^h = \rho^h_{ij} + \rho^B_{ij}$$

$$= \frac{1}{h^2}(\phi^h_{i_0-1,j} - 2\phi^h_{i_0,j} + \phi^h_{i_0+1,j}) + \frac{1}{h^2}(\phi^h_{i_0,j-1} - 2\phi^h_{i_0,j} + \phi^h_{i_0,j+1})$$

$$= \frac{1}{h^2}(\,(2g^h_{i_0-1/2,j} - \phi^h_{i_0,j}) - 2\phi^h_{i_0,j} + \phi^h_{i_0+1,j}) + \frac{1}{h^2}(\phi^h_{i_0,j-1} - 2\phi^h_{i_0,j} + \phi^h_{i_0,j+1})$$

For Dirichlet boundary conditions, $\phi^h_{i_0-1,j} = -\phi^h_{i_0,j}$. We then can conclude that the left edge, the boundary charge is given by:

$$\rho^B_{i_0,j} = -\frac{2g^h_{i_0-1/2,j}}{h^2} \tag{62}$$

Similarly, on the bottom edge:

$$\rho^B_{i,j_0} = -\frac{2g^h_{i,j_0-1/2}}{h^2} \tag{63}$$

At the bottom left corner, the charge is:

$$\rho^B_{i_0,j_0} = -\frac{2}{h^2}(g^h_{i_0-1/2,j} + g^h_{i,j_0-1/2}) \tag{64}$$

We can perform a similar analysis for the right and top edges.

The same game can be played with Neumann boundary conditions, i.e., if we are given $\frac{\partial\phi}{\partial\hat{n}} = g$, where $\hat{n}$ is the outward normal. The expression for $\phi^h_{i_0-1,j}$ is in this case:

$$\phi^h_{i_0-1,j} = 2hg^h_{i_0-1/2,j} + \phi^h_{i_0,j} \tag{65}$$

1.2.3 Performance of multigrid

Now that we have the mechanics of multigrid firmly in place (presumably), we need to examine the performance of this technique. We initially start out with some guess $\tilde{\phi}$ for the solution ϕ and calculate the norm of the residual, as defined by:

$$RNORM = \left\| \Delta^h \phi - \rho \right\| \tag{66}$$

Then we must set some tolerance criterion, ε, which is a function of the machine's round-off error. Then, as long as *RNORM* is larger than the tolerance, keep using the multigrid algorithm recursively. A section of pseudo-code of the outer loop might look like:

$\tilde{\phi}$ = *[first guess]*

$RNORM^l = \|\Delta^h\phi - \rho\|$

while (RNORM $> \varepsilon\|\rho\|$ *)*

> $l = l + 1$
>
> $\phi = MG\_relax(\rho, h, l, \tilde{\phi})$
>
> $RNORM^l = \|\Delta^h\phi - \rho\|$

end while

Next we want to examine issues such as convergence properties, cost and accuracy.

(1) **Convergence properties of outer loop**

A remarkable fact which can even be proven analytically for some cases, as well as being actually observed in numerical simulations is that:

$$RNORM^{l+1} \le C \cdot RNORM^l \qquad (67)$$

where C is a constant less than one and is a function of the mesh spacing h. For the five-point discretization of the Laplacian that we have been using, $C \cong 0.2$. Other common discretizations yield values in the ballpark of 0.2 - 0.4.

(2) **Cost of computation for a fixed tolerance**

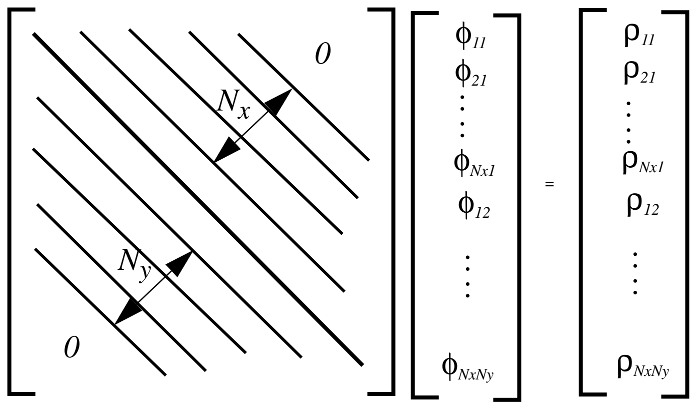
On each level, the number of floating point operations per iteration is on the order of the number of mesh points at that level. If K is denoted as the number of floating point operations and the number of mesh points in each direction is $N = 2^p$, then

$$K\sum_{q=0}^{p} 2^{2p} \le K'N\,(log_2 N) \qquad (68)$$

(3) **Number of mesh points N ≠ 2^p.**

When the number of grid points is not a power of two, represent the number of mesh points in each direction by the largest common power of 2 multiplied by an odd integer, $N_x{}^B$ and $N_y{}^B$ (i.e., $N_x = 2^p N_x{}^B$ and $N_y = 2^p N_y{}^B$). Let's assume that $N_x > N_y$. The next step is to call MG_relax(ρ, h, l, N_x, N_y, $\tilde{\phi}$). As long as $N_x > N_x{}^B$ and $N_y > N_y{}^B$, it makes sense to coarsen the grid and apply the usual multigrid algorithm. When we finally arrive at the coarsest level of $N_x{}^B$ and $N_y{}^B$, then it's time to do a direct solve, using, e.g., Gaussian elimination. The matrix equation can be ordered thusly:

(69)



The matrix *A* is banded. Upon *LU* decomposition, the lower diagonal matrix *L* and the corresponding upper diagonal matrix *U* will both have a bandwidth of about N_x. The storage for these matrices is the product of the number of unknowns multiplied by the bandwidth, so that the storage should be of $O(N_x^2 N_y)$. In terms of floating-point work, the most expensive part will be in performing the *LU* decomposition, requiring $O(N_x^3 N_y)$ floating-point operations. However, this only needs to be done once. The backsolve requires about $O(N_x^2 N_y)$ floating-point operations.

(4) **Accuracy**.

Let's develop some more notation. If the exact continuous solution and source terms are denoted by the subscript *E*, then they are related by the differential equation:

$$\Delta\phi_E = \rho_E \tag{70}$$

We will call the exact continuous solution evaluated at discrete grid points $\phi^D_{E_{i,j}}$. We would like to find the truncation error, i.e., the difference between the discrete operator acting on the exact solution and the discrete r.h.s. of equation (-1). For central differencing

$$(\Delta^h \phi^D_E)_{i\Delta x, j\Delta y} = (\Delta\phi_E)_{i\Delta x, j\Delta y} + O(h^2) \tag{71}$$

But

$$(\Delta\phi_E)_{i\Delta x, j\Delta y} = (\rho_E)_{i\Delta x, j\Delta y}$$
$$= \rho$$
$$= \Delta^h\phi$$

so that

$$\Delta^h \phi^D_E = \Delta^h \phi + O(h^2)$$

Then, by linearity

$$\Delta^h (\phi^D_E - \phi) = O(h^2) \tag{72}$$

The Laplacian operator is bounded and independent of h for Dirichlet boundary conditions. For this case, if we can show that

$$\|\Delta^h\phi\| \geq C\|\phi\| \tag{73}$$

where C is positive and independent of the mesh spacing, we can conclude that

$$\phi_E^D - \phi = O(h^2) \tag{74}$$

1.2.4 Time step considerations

Suppose that the temperature $T(\vec{x}, t)$ is governed by the unsteady heat equation:

$$\frac{\partial T}{\partial t} = \kappa\Delta T + f(\vec{x}) \tag{75}$$

where κ is the heat diffusion coefficient. To approach this numerically, we must replace the Laplacian by its discrete representation, Δ^h, and decide at what time to evaluate it, as well as figure out how to properly discretize the partial derivative w.r.t. t. A number of choices present themselves.

1.2.4.1 Forward Euler

The discretized equation takes the form:

$$\frac{T^{n+1} - T^n}{\Delta t} = \kappa\Delta^h T^n + f \tag{76}$$

or equivalently

$$T^{n+1} = (I + \kappa(\Delta t)\Delta^h)T^n + (\Delta t)f \tag{77}$$

where I is the identity matrix. If the steady-state solution is defined by:

$$\Delta^h T_E^n = -f \tag{78}$$

and the truncation error by:

$$\delta T^n = T^n - T_E^n \tag{79}$$

then, for the solution to converge, i.e., $\|\delta T^{n+1}\| \leq \|\delta T^n\|$, it is a sufficient and necessary condition that

$$\frac{\kappa\Delta t}{h^2} \leq \frac{1}{2n} \tag{80}$$

where Δt is the time step, h is the mesh spacing, and n is the number of spatial dimensions in the problem. Catastrophic failure results from violating this condition, but we are reluctant to accept this overly restrictive time step. Recall that the grand plan is to couple this part of the Navier-Stokes equation to the hyperbolic part. We saw that Lax-Wendroff, Fromm and upwinding performed well with $\Delta t = O(\Delta x)$. In addition, we can see trouble from the outset since the errors are unbalanced. Consideration of global error for the hyperbolic case shows that we worked hard to obtain error of $O(h^2, \Delta t^2)$, indicating that h is proportional to Δt, whereas for the parabolic case just considered, the error is of $O(h^2, \Delta t)$.

We can get around the stability problem by making the temporal differencing implicit, e.g., using backward Euler.

1.2.4.2 Backward Euler

This scheme is also first order in time, but it is implicit, which leads to unconditional stability. The discretized form of the governing equation is:

$$\frac{T^{n+1} - T^n}{\Delta t} = \kappa \Delta^h T^{n+1} + f \tag{81}$$

or

$$(I - \kappa (\Delta t) \Delta^h) T^{n+1} = T^n + (\Delta t) f \tag{82}$$

where I is the identity matrix.

If we perform a Fourier analysis (see similar treatment for the hyperbolic case in section xxx.xxx.xx) with the five-point Laplacian discretization, we can show that the amplitude of the k_xth/k_yth Fourier mode behaves in the following fashion:

$$a^{n+1}_{k_x, k_y} \{ 1 - \Delta t \kappa [-4 + 2\cos\beta_x + 2\cos\beta_y] \} \frac{1}{h^2} = a^n_{k_x, k_y}$$

or

$$a^{n+1}_{k_x, k_y} = \frac{a^n_{k_x, k_y}}{1 + \Delta t \kappa [4 - 2\cos\beta_x - 2\cos\beta_y]} \tag{83}$$

Since the denominator is less than unity, the conclusion is that the amplitudes of Fourier modes decay.

The implicit nature of algorithm is a mixed blessing. Unlike forward Euler, stability does not restrict the size of the time step (although we should note that accuracy will effectively limit its size.) On the other hand, it means that for almost all practical problems, we now have a linear system to solve. As before, we can expect that careful organization of the system will allow us to have a banded matrix. Therefore, we will probably want to resort to multigrid for an efficient algorithm if we use this discretization. One important consideration remains: we still have a mismatch in the temporal and spatial errors. As is the case for forward Euler, $\Delta t = O(h^2)$. Since we are stuck with an implicit method anyways, with just a little extra work, we can fix the error mismatch by using Crank-Nicholson, as shown in the next section.

1.2.4.3 Crank-Nicholson

For this case, the discretization is:

$$T^{n+1} = T^n + \frac{\Delta t}{2}\kappa\Delta^h\left(T^{n+1} + T^n\right) + (\Delta t) f \tag{84}$$

or

$$\left(I - \frac{\Delta t}{2}\kappa\Delta^h\right) T^{n+1} = \left(I + \frac{\Delta t}{2}\kappa\Delta^h\right) T^n + (\Delta t) f \tag{85}$$

This method costs no more to implement than backward Euler and is similarly unconditionally stable. We have also gained the bonus of a scheme which has second-order accuracy in time. Performing a Fourier analysis gives:

$$a^{n+1}_{k_x, k_y}\left\{1 - \frac{\Delta t}{2}\kappa\left[-4 + 2\cos\beta_x + 2\cos\beta_y\right]\frac{1}{h^2}\right\} =$$

$$= a^n_{k_x, k_y}\left\{1 + \frac{\Delta t}{2}\kappa\left[-4 + 2\cos\beta_x + 2\cos\beta_y\right]\frac{1}{h^2}\right\}$$

To simplify, if we define

$$\mu = \frac{\Delta t}{2h^2}\kappa\left[-4 + 2\cos\beta_x + 2\cos\beta_y\right] \tag{86}$$

then we can write

$$a^{n+1}_{k_x, k_y} = a^n_{k_x, k_y}\left[\frac{1-\mu}{1+\mu}\right] \tag{87}$$

We can see that the upper and lower bounds on the growth of the Fourier amplitudes are:

$$-1 \le \frac{1-\mu}{1+\mu} \le 1$$

This scheme is unconditionally stable, although we still must worry about solving a linear system. What's the best approach for solving this? The cost of a direct solver is prohibitive, just as it was in the case of the Poisson equation. We would like to be able to use the multigrid methodology developed in the previous section. Let's denote the temperature at time level $n+1$ as $\tilde{T}$. We note that the form of the equation (85) above can be written as a linear operator on $\tilde{T}$ being equal to a right-hand side:

$$L\tilde{T} = \rho \tag{88}$$

if the linear operator is:

$$L = I - \frac{\Delta t}{2}\kappa\Delta^h \tag{89}$$

For example, using the standard five-point Laplacian, we can write $L\tilde{T}$ at any point *(i, j)* as:

$$(L\tilde{T})_{i,j} = \left(1 + \frac{4\Delta t\kappa}{2h^2}\right)\tilde{T}_{i,j} - \frac{\Delta t\kappa}{2h^2}\left[\tilde{T}_{i+1,j} + \tilde{T}_{i-1,j} + \tilde{T}_{i,j+1} + \tilde{T}_{i,j-1}\right] \tag{90}$$

The r.h.s of equation (-3) is made up of values of temperature at time level n as well as the original source term f:

$$\rho = \left(I + \frac{\Delta t}{2}\kappa\Delta^h\right)T^n + (\Delta t)f \tag{91}$$

Is this kosher? What did we need for multigrid to work? A number of considerations present themselves:

(1) **Linearity.** Although not strictly essential, the linearity of the system makes our task simpler. Recall that, in the previous section, we were able to develop a scheme based on the variation between the guess of the solution and the updated value of the solution. For $L\phi = \rho$. we defined the residual to be $R = \rho - L\tilde{\phi}$ where $\tilde{\phi}$ was the initial guess of the solution ϕ; we defined the remainder to be $\delta = \phi - \tilde{\phi}$; and then performed multigrid recursively on the system $L\delta = R$ with homogeneous Dirichlet boundary conditions; and updated the solution by $\phi = \tilde{\phi} + \delta$. The size of the residual gave a direct measure of how close we were in coming to the solution.

(2) **Damping of short wavelengths by the local relaxation scheme.** This meant that, after a relaxation step, we could accurately represent the solution on a coarser grid. We can transfer the high wavenumber error to low wavenumber if we choose λ to be:

$$\lambda = -\left[1 + \frac{4\Delta t\kappa}{2h^2}\right]^{-1} \tag{92}$$

Although $\lambda = \lambda(h, \Delta t)$ now, rather than being directly related to the time step alone, this choice of λ does allow us to squelch the high wavenumber error.

(3) **Implicit assumption of ellipticity.** For elliptic problems, if we know the degree to which ρ is smooth, then we also know how smooth ϕ is; and vice versa. Why is this important? We know that if we eliminate the short wavelengths in the error, then we also get rid of the short wavelengths in the r.h.s. as well. (Technically, the smoothness of $L\phi$ at a point $\vec{x}$ is determined by the smoothness of ρ in the neighborhood of $\vec{x}$.) For

transonic flows, there are both hyperbolic regions (in the supersonic portion of the flow), as well as elliptic regions (subsonic flow). Although multigrid is used in transonic flow problems, caution must be exercised in the regions of supersonic flow.