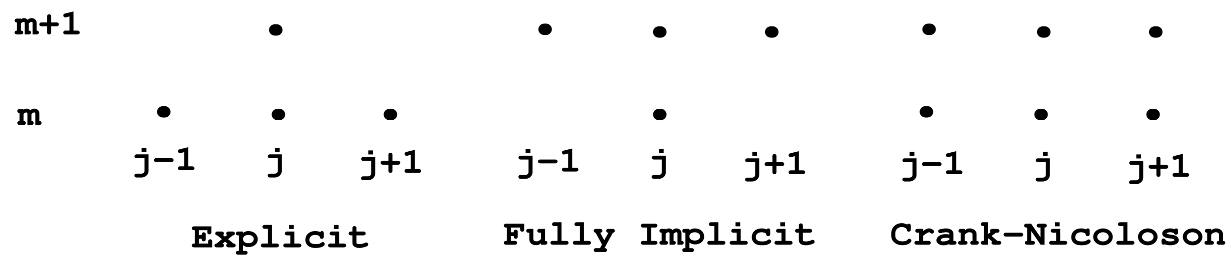

Figure 3.16: Difference Methods for Parabolic Equations.

This, does not mean, however, that the method gives accurate answers for any values of Δt and h. It is still first-order accurate in time and second-order accurate in space, so that we must use small enough step sizes to control the discretization error.

A potentially better implicit method is the Crank-Nicolson method, which is the average between the explicit and implicit methods.

$$u_j^{m+1} - u_j^m = \frac{\Delta t}{2h^2}(u_{j+1}^{m+1} - 2u_j^{m+1} + u_{j-1}^{m+1} + u_{j+1}^m - 2u_j^m + u_{j-1}^m) \tag{3.99}$$

or in vector form

$$(I + \frac{\mu}{2}A)\mathbf{u}^{m+1} = (I - \frac{\mu}{2}A)\mathbf{u}^m + \mathbf{b} \quad m = 0, 1, \ldots \tag{3.100}$$

where A is the $(2, -1)$ tridiagonal matrix (i.e. 2's down the diagonal and -1's on the off-diagonals). Here one is solving a tridiagonal system of equations at each time step. The advantage of the Crank-Nicolson formulation is that is is unconditionally stable and it is second-order accurate in time and space.

An easy way to remember the difference between the methods is to visually remember how their difference approximations involve their neighboring mesh points as shown in figure 3.4.

Now that we have our feet wet, let's consider two- and three-dimensional parabolic field equations. For now, let's consider our unit square again (the same as for elliptic equations), such that we have the Dirichlet boundary condition

$$u(x, y, t) = g(x, y) \tag{3.101}$$

on the sides of the square and the initial condition

$$u(x, y, 0) = f(x, y) \tag{3.102}$$

Our 2D explicit method becomes

$$u_{ij}^{m+1} = u_{ij}^m + \mu(u_{i,j+1}^m + u_{i,j-1}^m + u_{i+1,j}^m + u_{i-1,j}^m - 4u_{i,j}^m) \tag{3.103}$$

where we still have the stability condition

$$\Delta t \leq \frac{h^2}{4} \tag{3.104}$$

Our 2D implicit method becomes

$$u_{ij}^{m+1} = u_{ij}^m + \mu(u_{i,j+1}^{m+1} + u_{i,j-1}^{m+1} + u_{i+1,j}^{m+1} + u_{i-1,j}^{m+1} - 4u_{i,j}^{m+1}) \tag{3.105}$$

which is again, unconditionally stable. However, to utilize this method requires the solution of the following system of equations at each time step.

$$(4 + \frac{h^2}{\Delta t})u_{ij}^{m+1} - u_{i,j+1}^{m+1} - u_{i,j-1}^{m+1} - u_{i+1,j}^{m+1} - u_{i-1,j}^{m+1} = \frac{h^2}{\Delta t}u_{ij}^m \tag{3.106}$$

Therefore, at each time step, we must solve the equivalent of a 2D Poisson problem. Obviously, the computational costs could be tremendous. We could implement a Crank-Nicolson formulation, but it too, suffers the same computational malady.

To overcome this difficulty, let us consider a slightly different kind of method known as *time-splitting methods*. In time-splitting methods, the time interval, $(t, t + \Delta t)$ is further subdivided such that, in essence, one only has to solve a one-dimensional problem at each step.

Probably the simplest time-splitting method was put forth by Peaceman and Rachford and is known as the *Alternating Direction Implicit (ADI)* method. Here we have

$$u_{ij}^{m+1/2} = u_{ij}^m + \mu(u_{i+1,j}^{m+1/2} + u_{i-1,j}^{m+1/2} - 2u_{i,j}^{m+1/2} + u_{i,j+1}^m + u_{i,j-1}^m - 2u_{i,j}^m), \tag{3.107}$$

which leads to

$$u_{ij}^{m+1} = u_{ij}^{m+1/2} + \mu(u_{i,j+1}^{m+1} + u_{i,j-1}^{m+1} - 2u_{i,j}^{m+1} + u_{i+1,j}^{m+1/2} + u_{i-1,j}^{m+1/2} - 2u_{i,j}^{m+1/2}), \tag{3.108}$$

where $\mu = \frac{\Delta t}{2h^2}$. This is a two-step method in which intermediate values $u_{ij}^{m+1/2}, j = 1, \ldots, N$ are computed at the first step. These $u_{ij}^{m+1/2}$ values can be interpreted as approximate values of the solution. Note that the $\frac{1}{2}$ that appears in the definition of μ is because the time step is $\frac{1}{2}\Delta t$.

It we define $\alpha = 2h^2/\Delta t$, then for fixed j and $i = 1, \ldots, N$ we have,

$$(2 + \alpha)u_{ij}^{m+1/2} - u_{i+1,j}^{m+1/2} - u_{i-1,j}^{m+1/2} = (\alpha - 2)u_{ij}^m + u_{i,j+1}^m + u_{i,j-1}^m \tag{3.109}$$

This is a tridiagonal system of equations whose solution is $u_{ij}^{m+1/2}$, $i = 1, \ldots, N$. Once the intermediate values of $u_{ij}^{m+1/2}$ have been computed, the final values u_{ij}^{m+1} are computed by solving the N tridiagonal systems

$$(2 + \alpha)u_{ij}^{m+1} - u_{i,j+1}^{m+1} - u_{i,j-1}^{m+1} = (\alpha - 2)u_{ij}^{m+1/2} + u_{i+1,j}^{m+1/2} + u_{i-1,j}^{m+1/2} \tag{3.110}$$

for $j = 1, \ldots, N$, for each $i = 1, \ldots, N$.

I've just given a very brief overview of finite difference methods for time-dependent problems. The methods that are usually used in research codes are more sophisticated versions of the ADI or Crank-Nicolson type methods.

3.5 Computational Considerations

Thus, solving elliptic and parabolic engineering field problems by the finite difference method reduces to solving a linear system of equations. So are we done? What about solving the system of equations? How do we do that? What kind of considerations should we be thinking about? Should we use Gaussian elimination, perhaps LU decomposition, or maybe an iterative method?

To help us answer some of these questions, let's look at the structure of the matrix. It is symmetric, positive definite, and very sparse. Since we used a five point method, there will not be any more than five non-zero elements in any given row (some will have three or four near the top and bottom). Just to give you an idea of how size effects our problem let's consider that N is of moderate size, $N = 100$. Then there are $N^2 = 10^4$ unknowns and the matrix is $10,000 \times 10,000$. In each row there are at most five nonzero elements regardless of the size of N. As N grows, the matrix becomes more sparse. Let's say we decide to use Gaussian elimination to solve the system of equations. Gaussian elimination requires on the order of n^3 operations to solve an $n \times n$ system of equations. Therefore, if we have a $10^4 \times 10^4$ linear system, it requires on the order of 10^{12} operations. At a rate of 10^6 operations per second (1 MFLOP), it would require several hours to solve such a problem. Furthermore, if we were to extend this into a three-dimensional system, we would have a system of size 10^6 requiring 10^{18} operations which is currently beyond the scope of even our fastest computers.

Before leaving finite difference methods, we should note that we have only touched the surface and given an introductory treatment of the plethera of special methods that have been developed over the years. A few of the topics we neglected were: formulations in other coordinate systems (spherical, polar, parabolic, etc.), the finite difference time domain (FDTD) method, which has become popular in Electrical Engineering, and many others. For those wishing to find details about these topics, the reader is invited to consult the references [25, 26].