Chapter 3

Computational Scheme

3.1: Introduction on Numerical Methods

For numerical computations, the mathematical equations that need to be computed must be first cast into finite difference forms. This process is known as discretization. In other words, to do discretization, continuous differential equations or integral equations of the mathematical model are replaced by finite algebraic equations in order to allow numerical solution using computers.

There are many ways to discretize a mathematical equation. In this process, there are a few factors that need to be considered. The first factor involves the accuracy of a numerical solution. Due to mathematical equations that are in finite algebraic form, there exist two sources of errors. The first source of error, the truncation error is caused by approximation as some functions are written in the first few terms in a Taylor expansion. The second source of error, round-off errors, is due to the accuracy with which a particular variable is stored in the memory of the computer. The numbers stored in the computer are in terms of an exponent and a decimal fraction, and they are not exact due to truncation. If a calculation is correct to the i decimal places, it is less accurate compared to $i+1$ decimal places.

The second factor requires a system of algebraic equations generated by discretization process to be consistent with the original partial differential equation. In the limit that the grid spacing tends to zero ($\Delta x, \Delta z \rightarrow 0$) the system of algebraic equations is equivalent to the partial differential equation at each grid point. This is necessary if the approximate solution is to converge to the solution of the partial differential equation under consideration. However, it is not a sufficient condition as it does not follow that the solution of a system of algebraic equations approaches the solution of partial differential equation.

A solution of algebraic equation is said to be convergent if the approximation solution approaches the exact solution of the partial differential equation for each value of the independent variable as the grid spacing tends to zero. Thus it requires $T_i^k \to \overline{T}(x_i, z_k)$ at $\Delta x, \Delta z \to 0$. The difference between the exact solution of the partial differential equation and the exact solution of the system of algebraic equations is called the solution error, e_i^k that is

$$e_i^k = \overline{T}(x_i, z_k) - T_i^k .$$

Convergence implies that the solution error should reduce to zero as the grid spacing is shrunk to zero.

The third factor that needs to be considered is the stability of the numerical scheme. The concept of stability concerns the growth, or decay, of errors introduced at any stage of the computation. In most calculations, there will be iterations. Therefore, the errors – round-off error and truncation error – at each calculation will be accumulated. A particular method is stable if all errors produced in the application of the algorithm are bounded. In other words, after n iterations ($n \to \infty$), the approximate solution will reduce to the exact solution.

Between these factors, we will have to find a compromise that gives quality of representation and efficiency. A simulation that meets all requirements – stability, accuracy – may not be efficient to implement due to the consumption of extended computational resources. Therefore, higher order schemes may be used to increase the efficiency of such program.

The finite difference scheme defined, by Strikwerda[19], as a grid of points in the (x, z) plane. Let Δx and Δz be positive numbers, then the grid will be the points $(x_n, z_m) = (n\Delta x, m\Delta z)$ for arbitrary integers n and m. For a function v defined on the grid, one can write v_m^n for the value of v at the grid point (x_n, z_m). The values of Δx and Δz for a grid have to be very small i.e. $\Delta x, \Delta z \to 0$ for an approximation to be the differential equations. The diagram below shows the points on a fixed grid.

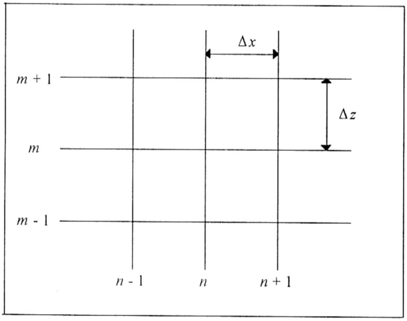

Figure 3.1

Say for a given equation

$$u_x + au_z = 0$$

it can be discretized in many ways such as in the following equations

$$\frac{u_m^{n+1} - u_m^n}{\Delta x} + a\frac{u_{m+1}^n - u_m^n}{\Delta z} = 0 \qquad \text{(a)}$$

$$\frac{u_m^{n+1} - u_m^n}{\Delta x} + a\frac{u_m^n - u_{m-1}^n}{\Delta z} = 0 \qquad \text{(b)}$$

$$\frac{u_m^{n+1} - u_m^n}{\Delta x} + a\frac{u_{m+1}^n - u_{m-1}^n}{2\Delta z} = 0 \qquad \text{(c)}$$

$$\frac{u_m^{n+1} - \frac{1}{2}\left(u_{m+1}^n + u_{m-1}^n\right)}{\Delta x} + a\frac{u_{m+1}^n - u_{m-1}^n}{2\Delta z} = 0 \qquad \text{(d)}$$

$$\frac{u_m^{n+1} - u_m^{n-1}}{2\Delta x} + a\frac{u_{m+1}^n - u_{m-1}^n}{2\Delta z} = 0 \qquad \text{(e).}$$

Equation (a) is referred to as the forward-time forward-space scheme because of the forward difference approximation, (b) is the forward-time backward-space scheme, (c) is the forward-time central-space scheme, (d) is the Lax-Friedrichs scheme and (e) is the leapfrog scheme. All the schemes mentioned above are classified as explicit schemes, characterized by the fact that all dependent variables are defined from the previous mesh. Finite difference method is notable for the great variety of schemes that can be used to approximate a given partial differential equation.

Equations (a) and (b) are first order in both x and z. While equation (c) is said to have first order accuracy in the z but second order in x. According to Potter[18], equation (c) is not stable for hyperbolic equations. Equation (d) is first order accurate in z but second order accurate in x. However, it is conditionally stable. The last equation,

equation (e) is second order accurate for both x and z since it is a three level scheme. It is also conditionally stable for hyperbolic equation.

The current Lax-Wendroff scheme faces a problem of numerical diffusion. It is difficult to extend the scheme to higher order terms to replace this term

$$\frac{1/2(u^n_{m+1}+u^n_{m-1})}{\Delta x}.$$

It requires a reduction in grid spacing, which then increases the computation time. The next likely scheme to replace the Lax-Wendroff scheme is the leapfrog scheme. However, it is not possible to discretize the governing equations that are used in this project without uncoupling the two interlocking meshes that define separate variables.

To cast the governing equation (2.14)

$$\frac{\partial L_x}{\partial z}=\frac{1}{2L_z}\frac{\partial n_G^2}{\partial x}-\frac{L_x}{L_z}\frac{\partial L_x}{\partial x}$$

into the leapfrog scheme, the variables are separated with L_x and L_z defined in a mesh while ϕ and n_G are defined in the half mesh as required by leapfrog scheme.

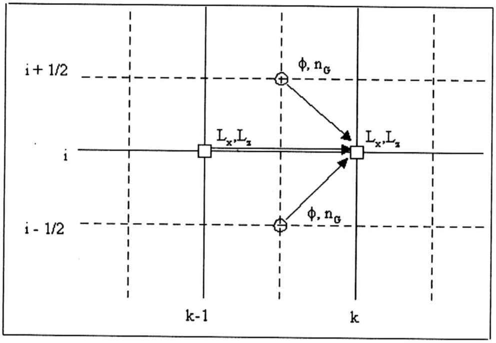

Figure 3.2

However, this equation requires the definition of L_x and L_z not only at grid point (i, k) but also at grid point $(i, k - ½)$ where the variables L_x and L_z are not computed. Definition of L_x and L_z variables at half mesh will cause the two meshes to drift out of phase. Overcoming this problem with interpolation to obtain the values at point $(i, k - ½)$ will change the scheme to be about the same as the Lax-Wendroff scheme.

In order to avoid this problem, the forward-time centred-space (FTCS) scheme is proposed for this project. The instability that arises due to this scheme can be overcome by using an implicit scheme on the equation that gives rise to the instability.

3.2: Numerical Stability of Difference Schemes

A numerically unstable scheme[18] happens when the scheme produces a solution, which is not bounded. Errors that are produced along any calculation and are continually amplified from time step to time step will lead to instability. The amplified errors will quickly swamp the solution and the results obtained will be of no value. Stability of a numerical scheme may then be stated as a small error at any stage of the calculation, which will produce a smaller cumulative error further along the iteration.

If u(x,t) is the exact solution of the initial-value problem, and u_i^k is the solution of the finite difference equations at time step n, the error of the approximation is then

$$U_i^k - u(ix, kt) = \varepsilon^k .$$

If this error, ε^k, grows, then, u_i^k will lose its accuracy. Therefore, it is desirable to use a scheme that this error will maintain in its order. For a linear partial differential equation, the behaviour of the error can be analyzed using Fourier analysis. The amplification factor, which is defined as

$$g = \frac{u_i^{k+1}}{u_i^k}$$

will determine the stability of the numerical scheme. The scheme will be stable[18, 20] if the amplification factor satisfies the von Neumann stability condition.

$$|g| \leq 1 + O(h)$$

The amplification factors of the five given explicit schemes that were mentioned above, are as shown.

Difference Scheme	Amplification factor, g
(1) Forward time forward space	$-\frac{a\Delta x}{\Delta z}\left(e^{ik\Delta x}-1\right)+1$
(2) Forward time backward space	$-\frac{a\Delta x}{\Delta z}\left(1-e^{-ik\Delta x}\right)+1$
(3) Forward time centred space	$-\frac{a\Delta x}{\Delta z}i\sin(k\Delta x)+1$
(4) Lax-Friedrichs	$-\frac{a\Delta x}{\Delta z}i\sin(k\Delta x)+\cos(k\Delta x)$
(5) Leapfrog	$g^2+\frac{a\Delta x}{\Delta z}g-1=0$

Table 3.1

In considering the stability of the equations (2.14) to (2.17) based on FTCS numerical scheme, each of the terms in these equations are tested for their contributions to the instability of the scheme. Taking into account the results produced and the stability of the computational program, each term is omitted one by one. The partial L_x term in the x direction from equation (2.14) when replaced with a trial solution that fits the L_x function is able to give a stable scheme that is able to propagate up to 10m. This term is then identified as the one that causes instability in this scheme.

Using this procedure it has been shown that the partial L_x term of equation (2.14) is the main cause of instability for the FTCS scheme. One of the ways to overcome the instability of this FTCS scheme is to use implicit scheme on equation (2.14).

The obvious characteristic[16 - 19] of an implicit scheme is the evaluation of spatial terms at the unknown time level $(k+1)$. Equation (2.14) will give an implicit form that evaluates the partial L_x term along the z-axis at the $(k+1)$ level based on the neighbouring values of L_x at the same level. These neighbouring values are yet to be known, thus giving an algebraic equation at that node. Each node along the x-axis will have such an equation that relies on each other. This then causes coupling of the equations of each node $(i,k+1)$ at the $(k+1)-th$ time level. In order to advance the solution, it is necessary to consider all the nodes i and the corresponding equations, arising a need to solve a system of algebraic equations. As such, a set of matrix equations can be written for the unknown values of the variables at $(i,k+1)$ time level.

The implicit finite difference representation of equation (2.14) is given by

$$\frac{L_{xi}^{k+1}-L_{xi}^{k}}{\Delta z}=\frac{1}{2L_{zi}^{k}}\left[\frac{\left(n_G^2\right)_{i+1}^{k+1}-\left(n_G^2\right)_{i-1}^{k+1}}{2\Delta x}\right]-\frac{L_{xi}^{k}}{L_{zi}^{k}}\left[\frac{L_{xi+1}^{k+1}-L_{xi-1}^{k+1}}{2\Delta x}\right]$$

By doing so, it is enough to obtain a conditionally stable scheme that can be further improved by applying the implicit method to the other equations. However, the solution of a partial differential equation by the implicit method requires the solving of the matrix, which requires twice as much computer time as solving the same number of explicit equations. The implicit method is then applied to the most unstable among the equations used in this project. The stability by this reasoning is adequate for up to 10m from the origin.

3.3: Governing Equations

The generalized eikonal equation, equation of continuity and the ray tracing equations are simplified to two dimensions. The two dimensions taken into account are the x-axis and the z-axis. The ray direction of propagation will be set in the z-axis that is the horizontal axis while x is the vertical axis. The governing equations are given below.

From the ray tracing equation (2.14 – 2.15)

$$\frac{\partial L_x}{\partial z} = \frac{1}{2L_z}\frac{\partial n_G^2}{\partial x} - \frac{L_x}{L_z}\frac{\partial L_x}{\partial x}, \tag{3.1}$$

$$\frac{\partial L_z}{\partial z} = \frac{1}{2L_z}\frac{\partial n_G^2}{\partial z} - \frac{L_x}{L_z}\frac{\partial L_z}{\partial x}, \tag{3.2}$$

the continuity equation (2.16)

$$\frac{\partial \phi}{\partial z} = -\frac{1}{2}\frac{\phi}{L_z}\left(\frac{\partial L_x}{\partial x} + \frac{\partial L_z}{\partial z}\right) - \frac{L_x}{L_z}\frac{\partial \phi}{\partial x}, \tag{3.3}$$

and the generalized eikonal equation (2.17)

$$n_G^2 = n_0^2 + \frac{1}{k_0^2\phi}\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial z^2}\right)\phi . \tag{3.4}$$

From these four differential equations, the calculation of ϕ, L_x, L_z, and n can be carried out. There are, in all, four variables in a two-dimension system. To implement the finite difference scheme, all the points will be defined in a fixed grid. Equation (3.3) and (3.4) will be discretized in a FTCS scheme while equation (3.1) will be that of an implicit scheme, after which equation (3.2) will be replaced by using

$$L_z = \sqrt{n_G^2 - L_x^2} .$$

The finite difference approximations to the last two equations are then

$$\frac{\phi_i^{k+1}-\phi_i^k}{\Delta z} = -\frac{\phi_i^k}{2L_{zi}^k}\left(\frac{L_{xi+1}^k - L_{xi-1}^k}{2\Delta x} + \frac{L_{zi}^k - L_{zi}^{k-1}}{\Delta z}\right) - \frac{\Delta z L_{xi}^k}{L_{zi}^k}\left(\frac{\phi_{i+1}^k-\phi_{i-1}^k}{2\Delta x}\right) \text{ and}$$

$$n_G^2 = n_0^2 + \frac{1}{k_0^2\phi}\left(\frac{\phi_{i+1}^{k+1} - 2\phi_i^{k+1}+\phi_{i-1}^{k+1}}{(\Delta x)^2} + \frac{\phi_i^{k+1}-2\phi_i^k+\phi_i^{k-1}}{(\Delta z)^2}\right).$$

While the implicit scheme leaves equation (3.1) as

$$\left(-tL_{xi}^k\right)L_{xi-1}^{k+1} + \left(L_{xi}^k\right)L_{xi}^{k+1} + \left(tL_{xi}^k\right)L_{xi+1}^{k+1} = L_{zi}^k L_{xi}^k + \frac{t}{2}\left[\left(n_G^2\right)_{i+1}^{k+1} - \left(n_G^2\right)_{i-1}^{k+1}\right]$$

where t represents $\frac{\Delta z}{2\Delta x}$. The i subscript represents points along the x-axis while k superscript represents points along the z-axis. The left-hand side of this equation will form a column matrix for L_x at $(k+1)-th$ level while a tridiagonal system of equations will form a $(m \times m)$ matrix as i is evaluated from zero to its upper limit. The size of the square matrix depends on the number of points that exist in between x equals to zero to the upper limit along the x-axis. The right hand side equation, which is a column matrix, is evaluated based on results from iterations before this step. By reducing the square matrix to unity, one is able to obtain the values of L_x at every point on $(k+1)-th$ level. The corresponding values of L_z at these points can then be calculated from n_G and L_x. The numerical iteration of this finite difference scheme is carried out from $z=0$ to $z = z_{final}$.