Adaptive gridding for numerical modelling of wave propagation

James Mathews
Supervisor: Dr John T. Etgen, BP

January 9, 2013

Contents

1 Introduction and modelling assumptions

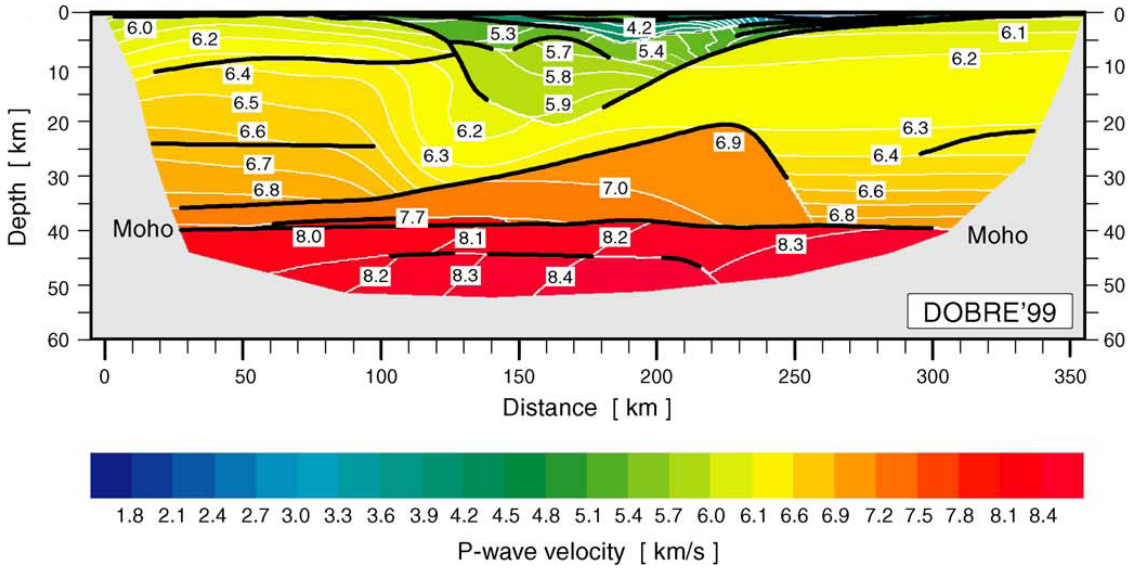
We are looking to model waves propagating through the earth, to do this we solve the scalar acoustic wave equation, since we assume the earth is a non moving fluid. The velocity is dependent on position, so we have:

$$\frac{\partial^2 \phi}{\partial t^2} = c^2(x, y, z) \left(\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} + \frac{\partial^2 \phi}{\partial z^2} \right). \tag{1.1}$$

A typical two dimensional velocity model for the earth is shown in Figure 1. The velocity (also referred to as sound speed) generally increases with depth, but different types of rock and the

compostion of the earth cause the velocity to be non uniform in depth. The boundaries between the different layers seen in Figure 1 are known as reflector boundaries, since when the wave meets the boundary some of it is reflected. To model solutions of the wave equation numerically, we

Figure 1: Velocity model of the crust and mantle in East Ukraine [3].



need to consider both the accuracy and computational time of the model, and achieve a trade-off between these two. To minimise computational time we choose as few points as possible to perform calculations at, so we have fewer equations to solve. But if we choose too few then the grid cells between the points become too large in comparison with the wavelength of the source signal, so waves disperse with increasing travel time - this phenomena is known as grid dispersion [2]. How do we decide the maximum sampling rate for the grid?

It turns out that for a finite difference scheme, a general rule of thumb for choosing the sampling rate is that we should choose [2]

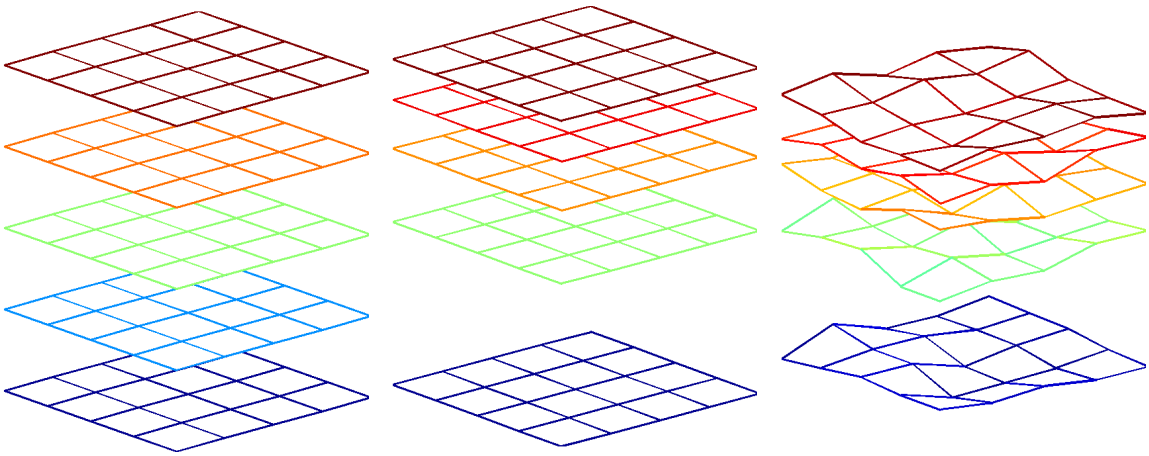
$$\Delta x = \frac{c_{\min}}{f_N \cdot S}. \tag{1.2}$$

In this equation Δx is the sampling rate (how far between grid points), c_{min} is the minimum velocity or sound speed and f_N is the Nyquist frequency which we can just take as double the dominant frequency of the input wave. S is a technical value which is experimentally calculated and dependent on the finite difference scheme, in the scheme we choose later we have $S = 3$. As we travel deeper into the earth, the sound speed increases and thus from (1.2) it is clear that we can choose grid points further apart. Since the sound speed increases by a factor of three or four times as waves propagate into the depths of the earth, we can choose cells significantly larger when deeper into the earth. Thus to ensure accuracy but minimise computational time we need a non uniform grid. We look to implement a numerical method where the distance Δz between the z grid points depends on the sound speed, although not uniformly. In Figure 2 we show some examples of non uniform grids.

Choosing a non uniform grid such as (b) or (c) in Figure 2 leads to several issues:

(1) How do we choose the grid points in the z direction? How many do we choose since the formula above no longer applies?

(2) How can we discretise the wave equation using finite differences when points are non-uniform? How accurate should the finite difference scheme be?



(a) Uniform grid in x, y, z. (b) Non uniform grid, but Δz only varies with depth and not x, y position. (c) Non uniform grid as Δz varies with both depth and x, y position.

Figure 2: Example of uniform and non uniform grids.

We address the first question in Section 2 by considering the grid as a simple case of a moving mesh, and we touch upon the theory behind these meshes. We also briefly mention x and y adaptivity and simple ways to implement them in this case. We answer the second question in Section 3, where we present a tenth order finite difference scheme on a non uniform grid and also carefully consider the boundary terms. In Section 4 we give some suggestions on how to further improve the model before summarising what we have found in Section 5.

2 The moving mesh

When implementing a numerical solution to a partial differential equation on a non uniform grid we need to come up with a method to choose the grid points. To do this we introduce a mesh density function $\boldsymbol{\rho}$ which we want to distribute evenly over the range of z. This means that when $\boldsymbol{\rho}$ is large we choose grid points close together and when $\boldsymbol{\rho}$ is small we choose grid points further away. We review the basic principles behind the concept here and we will also consider suitable mesh density functions. Most of the material in this Section is based upon [1] and [4].

2.1 Equidistribution principle

Given a mesh density function $\boldsymbol{\rho}(z,t)$ with $z \in [a,b]$, to equidistribute $\boldsymbol{\rho}$ (as a function of z) over the K points of the mesh $a = z_1 < z_2 < \ldots < z_K = b$ we require that

$$\int_{z_1(t)}^{z_2(t)} \boldsymbol{\rho}(z,t)\mathrm{d}z = \int_{z_2(t)}^{z_3(t)} \boldsymbol{\rho}(z,t)\mathrm{d}z = \ldots = \int_{z_{K-1}(t)}^{z_K(t)} \boldsymbol{\rho}(z,t)\mathrm{d}z.$$

holds for all t. We can rewrite this as

$$\int_a^{z_j(t)} \boldsymbol{\rho}(z,t)\mathrm{d}z = \frac{j-1}{K-1}\sigma(t) \text{ for each } j = 1, \ldots, K, \tag{2.1}$$

where $\sigma(t) = \int_a^b \boldsymbol{\rho}(z,t)dz$. From (2.1) it is clear that if $\boldsymbol{\rho}$ is positive then we can indeed find a mesh which equidistribute the mesh density function.

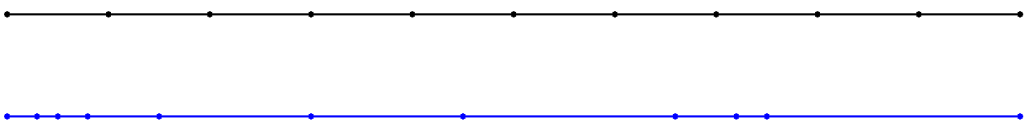


Figure 3: Example of a coordinate transformation z when $\Omega_c = \Omega$.

To calculate the mesh we introduce a transformation between the computational domain $\Omega_c = [0, 1]$ in which we associate points with ξ and the physical domain $\Omega = [a, b]$ with transformation $z : \Omega_c \times [0, T] \to \Omega$. Given grid points $\xi_j = \frac{j-1}{K-1}$, a uniform mesh on the computational domain, we get a non uniform mesh on the physical domain with points $z_j(t) = z(\xi_j, t)$ as shown in Figure 3. We want to be able to calculate this transformation, and indeed we can if we just know the mesh density function $\boldsymbol{\rho}$. To see this, from the continuous version of (2.1) we have

$$\int_a^{z(\xi,t)} \boldsymbol{\rho}(z, t)dz = \xi\sigma(t),$$

and upon differentiating this expression with respect to ξ we get the following differential equation which we usually have to solve numerically:

$$\boldsymbol{\rho}(z, t)\frac{\partial z}{\partial \xi} = \sigma(t). \tag{2.2}$$

It can be shown that the equidistributed mesh both optimises the mesh and solution dependent factor in error estimates of the numerical solution of the PDE, which is studied in [4, §2.1].

2.2 Moving Mesh PDE

To transform (2.2) into a Moving Mesh PDE (MMPDE) which is easier to solve we differentiate again with respect to ξ to eliminate σ and we get the equation

$$\frac{\partial}{\partial \xi}\left(\boldsymbol{\rho}(z, t)\frac{\partial z}{\partial \xi}\right) = 0,$$

and thus is it clear that the solution of the below PDE, the so-called MMPDE5 [4], will be the equidistributed mesh:

$$\frac{\partial z}{\partial t} = \frac{1}{\tau}\frac{\partial}{\partial \xi}\left(\boldsymbol{\rho}(z, t)\frac{\partial z}{\partial \xi}\right). \tag{2.3}$$

In this equation τ is a user controlled parameter which can be adjusted to improve the speed of convergence. There are many other different MMPDEs listed in [1, 4] of which two more are the modified MMPDE5

$$\frac{\partial z}{\partial t} = \frac{1}{\tau\boldsymbol{\rho}(z, t)}\frac{\partial}{\partial \xi}\left(\boldsymbol{\rho}(z, t)\frac{\partial z}{\partial \xi}\right),$$

and the widely used smoothed moving mesh equation

$$\left(1 - \gamma\frac{\partial^2}{\partial \xi^2}\right)\frac{\partial z}{\partial t} = \frac{1}{\tau}\frac{\partial}{\partial \xi}\left(\boldsymbol{\rho}(z, t)\frac{\partial z}{\partial \xi}\right), \tag{2.4}$$

with γ controlling the smoothness of the mesh. The derivation of these in one dimension are in [1, 4] and rely on varying a functional, and we not repeat the derivation here but instead give a

heuristic justification. It is clear that if our mesh is equidistributed then the right hand side of all these MMPDEs are zero and so the mesh points stay still, thus the moving mesh PDEs move the mesh over time to the equidistributed state. The smoothed moving mesh equation (2.4) is widely used for several reasons since it produces extremely stable meshes and has natural generalisations to higher dimensions, while it also ensures that no mesh tangling occurs as we evolve the mesh [1]. However, since we will be using a simple monitor function mesh tangling is not a concern so we will use the simplest moving mesh PDE, MMPDE5 (2.3).

2.3 Discretisation of the Moving Mesh PDE

In this section we will semi-discretise and fully discretise the MMPDE we have chosen to used, which will give us an ODE or system of equations to solve to find the equidistributed mesh. We do not need to use a particular high order method for the discretisation since we are interested in an accurate solution of the PDE, not the moving mesh PDE. The semi-discretisation of the MMPDE (2.3) using a central difference scheme on the computation domain Ω_c is given by

$$\frac{dz_j}{dt} = \frac{1}{\tau(\Delta\xi)^2}\left[\frac{\boldsymbol{\rho}_{j+1}+\boldsymbol{\rho}_j}{2}(z_{j+1}-z_j) - \frac{\boldsymbol{\rho}_j+\boldsymbol{\rho}_{j-1}}{2}(z_j-z_{j-1})\right], \tag{2.5}$$

where $\Delta\xi = \frac{1}{K-1}$, $z_j \approx z(\xi_j, t)$, $\boldsymbol{\rho}_j = \boldsymbol{\rho}(z_j, t)$ and we make the obvious changes to the discretisation at $j = 1$ and $j = K$. This can then be discretised using backwards Euler with the mesh density function $\boldsymbol{\rho}$ at the previous time step as in [4] or using forward Euler as in [1] which is as follows:

$$\frac{z_j^{n+1}-z_j^n}{\Delta t} = \frac{1}{\tau(\Delta\xi)^2}\left[\frac{\boldsymbol{\rho}_{j+1}^n+\boldsymbol{\rho}_j^n}{2}(z_{j+1}^n-z_j^n) - \frac{\boldsymbol{\rho}_j^n+\boldsymbol{\rho}_{j-1}^n}{2}(z_j^n-z_{j-1}^n)\right], \tag{2.6}$$

where $\Delta\xi = \frac{1}{k-1}$, Δt is the timestep, $t_n = n\Delta t$, $z_j^n \approx z(\xi_j, t_n)$ and $\boldsymbol{\rho}_j^n = \boldsymbol{\rho}(z_j, t_n)$. To evolve the mesh we firstly take $\boldsymbol{\rho}_0 := \boldsymbol{\rho}(z, 0)$ and apply either discretised MMPDE, (2.5) or (2.6), in artificial time to find the equidistributed mesh at time $t = 0$. We then use the discretised MMPDE to evolve the equidistributed mesh with time as the PDE evolves (which causes $\boldsymbol{\rho}$ to evolve), solving the discretised systems of the MMPDE and PDE alternatively. However, if our mesh density function is independent of time then we can simply solve the discretisation in artificial time to find the equidistributed mesh which we then fix and solve the PDE on the fixed mesh.

2.4 Choice of mesh density function

Clearly, the equidistributed mesh we solve the PDE on depends entirely on the choice of the mesh density function $\boldsymbol{\rho}$, so it is critical that we choose an appropriate function. Typically, the mesh density function is based on one of several error estimates, such as interpolation error estimates, *a priori* error estimates or *a posteriori* error estimates. These are all derived in [4, §2.4,§2.8,§2.9] and we will not repeat the derivation here. For example, the optimal mesh density function for linear interpolation which minimises the error in the L^2 norm is given by

$$\boldsymbol{\rho} = \left(1 + \frac{1}{\boldsymbol{\alpha}}\left|\frac{\partial^2 u}{\partial z^2}\right|^2\right)^{\frac{1}{5}}, \text{ where } \boldsymbol{\alpha} = \left[\frac{1}{b-a}\int_a^b \left|\frac{\partial^2 u}{\partial z^2}\right|^{2/5} \mathrm{d}z\right]^5$$

and $u(z, t)$ is the solution of the partial differential equation. Other popular choices [4, §2.4] are the arc-length mesh density function $\boldsymbol{\rho} = (1 + \left|\frac{\partial u}{\partial z}\right|^2)^{\frac{1}{2}}$ and the curvature mesh density function $\boldsymbol{\rho} = (1 + \left|\frac{\partial^2 u}{\partial z^2}\right|^2)^{\frac{1}{4}}$.

In our case we want to choose a mesh density function $\boldsymbol{\rho}$ proportional to the inverse of the velocity, so at high velocities we will have small $\boldsymbol{\rho}$ and thus since we equidistribute $\boldsymbol{\rho}$ we will choose mesh points further apart. For given x,y coordinates (x_n, y_m) we define

$$c_{n,m}(z) := c(x_n, y_m, z) \text{ and } \boldsymbol{\rho}_{n,m}(z) := \frac{1}{c_{n,m}(z)},$$

and then

$$c_{n,m,k} = c_{n,m}(z_k) \text{ and } \boldsymbol{\rho}_{n,m,k} := \boldsymbol{\rho}_{n,m}(z_k).$$

If we now denote each mesh point by $(x_n, y_m, z_{n,m,k})$, where $n = 1, \ldots, N$, $m = 1, \ldots, M$, $k = 1, \ldots, K$, so $z_{n,m,k}$ is the z-coordinate associated to (x_n, y_m), then (2.6) becomes

$$\frac{z^{l+1}_{n,m,k} - z^l_{n,m,k}}{\Delta t} = \frac{1}{\tau(\Delta\xi)^2}\left[\frac{\boldsymbol{\rho}^l_{n,m,k+1} + \boldsymbol{\rho}^l_{n,m,k}}{2}(z^l_{n,m,k+1} - z^l_{n,m,k}) - \frac{\boldsymbol{\rho}^l_{n,m,k} + \boldsymbol{\rho}^l_{n,m,k-1}}{2}(z^l_{n,m,k} - z^l_{n,m,k-1})\right]. \quad (2.7)$$

This gives a system of $NMKL$ (where $\Delta t = \frac{1}{L-1}$) linear equations to solve for the mesh points. Notice at this point we could reduce the number of equations that we have to solve by considering a mesh density function that is uniform in x (or in y):

$$c_x(y, z) := \frac{1}{x_N - x_1}\int_{x_1}^{x_N} c(x, y, z)\mathrm{d}x \text{ and } \boldsymbol{\rho}_{m,k} := \frac{1}{c_x(y_m, z_k)}.$$

And then the reduced system below has only MKL equations,

$$\frac{z^{l+1}_{m,k} - z^l_{m,k}}{\Delta t} = \frac{1}{\tau(\Delta\xi)^2}\left[\frac{\boldsymbol{\rho}^l_{m,k+1} + \boldsymbol{\rho}^l_{m,k}}{2}(z^l_{m,k+1} - z^l_{m,k}) - \frac{\boldsymbol{\rho}^l_{m,k} + \boldsymbol{\rho}^l_{m,k-1}}{2}(z^l_{m,k} - z^l_{m,k-1})\right]. \quad (2.8)$$

We could even define a mesh density function that is uniform in both x and y as follows:

$$c_{x,y}(z) := \frac{1}{(x_N - x_1)(y_M - y_1)}\int_{y_1}^{y_M}\int_{x_1}^{x_N} \boldsymbol{\rho}(x, y, z)\mathrm{d}x\mathrm{d}y \text{ and } \boldsymbol{\rho}_k := \frac{1}{c_{x,y}(z_k)},$$

which would give a system of KL equations. In pratice we do not bother with reducing the number of equations, since the time taken to generate the mesh is fairly insignificant compared to the total time to generate the mesh and perform simulations on it. This is because once we have a mesh we will perform many (ranging from 1000's to 100,000's) simulations of waves propagating on it, each modelling a different scenario.

We finish by stating how many points we need to choose in the z direction, K, and we will consider the values of N and M in the next section. Recall from equation (1.2) how we choose Δz in a uniform grid. If we have a non uniform grid then we choose Δz based on the average speed in the z direction, such as

$$\Delta z = \frac{1}{f_N \cdot S}\min_{n,m}\int_{z_1}^{z_K} c_{n,m}(z)\mathrm{d}z \text{ and } K = \left\lceil \frac{z_K - z_1}{\Delta z} \right\rceil. \quad (2.9)$$

This ensure that in each grid cell $\Delta z \approx \frac{c_{\min}}{f_N \cdot S}$ where $c_{\min}$ is the minimum sound speed in that cell. Note that this calculation relies on knowing the (x_n, y_m) coordinates (i.e knowing N and M), we could just minimise over all x and y but this requires significantly more computation.

2.5 Adaptivity in x and y coordinates

Since the sound speed increases by three to four times as we penetrate deeper into the earth, we can increase the sampling rate in x and y deep into the earth. Due to how we have chosen the z mesh points, which are dependent on the (x, y) coordinate associated with z, we can not move the x and y points around similarly to the z points. We briefly investigate in §4 allowing all coordinate points to move at the same time. Instead, we need to do it in a uniform way such that we can still discretise the PDE later. One way of doing this is to simply remove grid points as we go deeper into the earth. So for example we find $k^{[2]}$ as the first solution of $\min_{n,m}(c_{n,m,k}) > 2\min_{n,m,k}(c_{n,m,k}) = 2\min_{n,m}(c_{n,m,1})$ (since the sound speed is an increasing function in z). Then, at all levels below $k^{[2]}$ we just use every other grid point in the finite difference scheme, so our stencil changes from the first picture in Figure 4 to the second one. We can also find $k^{[3]}$ such that $\min_{n,m}(c_{n,m,k}) > 3\min_{n,m}(c_{n,m,1})$, then at all levels below $k^{[3]}$ we just use one out of three grid points in the finite difference scheme, so our stencil changes from the second picture in Figure 4 to the third one. We can also define $k^{[4]}$ in a similar way, and we do not worry about $k^{[n]}$ for $n > 4$.

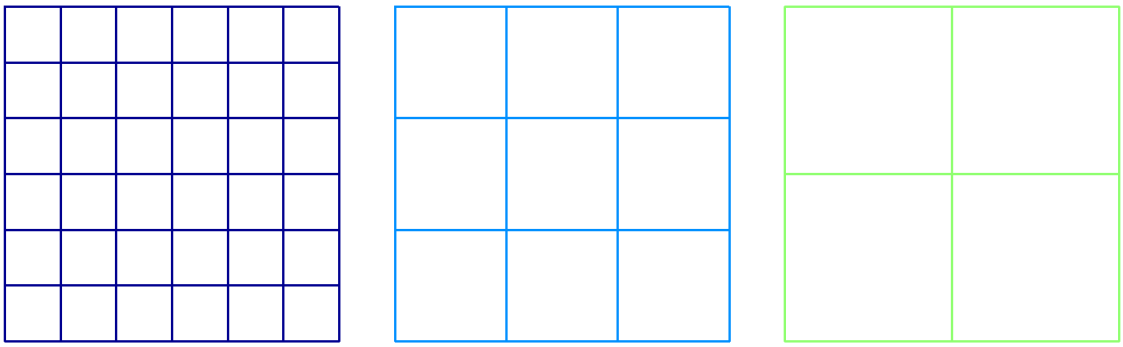


Figure 4: Removing grid points from the stencil for discretising the wave equation.

Clearly we are only going to be left with a grid to perform the discretisation on if the original grid size is a factor of two, three and four, so a multiple of twelve. Let $\Delta x = \Delta y = \min_{x,y} c(x, y, z_1)$ and then choose

$$N = 12\left\lceil \frac{x_N - x_1}{12\Delta x} \right\rceil \text{ and } M = 12\left\lceil \frac{y_M - x_1}{12\Delta y} \right\rceil. \tag{2.10}$$

We note the calculation of the points $k^{[n]}$ does not require much computational effort since we already have to calculate $c_{n,m,k}$ for discretising the wave equation and also to calculate $\boldsymbol{\rho}_{n,m,k}$, and calculating $\min_{n,m}(c_{n,m,1})$ can be done in $O(NM)$ operations.

3 Finite difference scheme for the wave equation

We now turn our attention to discretising the wave equation on the mesh we have calculated in §2. We firstly show how to derive a very accurate finite difference method of $O(\Delta t^4, \Delta x^{10})$ on a uniform mesh based upon [2], which gives the necessary accuracy needed in seismic models. We then show how to adapt this method for a non uniform mesh, before dealing with the boundary.

3.1 Discretised form of the wave equation on a uniform mesh

Assuming we have a uniform mesh, let $\phi^l := \phi(l\Delta t)$ be the discretisation in time and $\phi_{n,m,k} := \phi(n\Delta x, m\Delta y, k\Delta z)$ be the discretisation in space. Using Taylor series to expand in time we get

$$\phi^{l+1} = \phi^l + \frac{\partial\phi^l}{\partial t}\Delta t + \frac{\partial^2\phi^l}{\partial t^2}\frac{(\Delta t)^2}{2!} + \frac{\partial^3\phi^l}{\partial t^3}\frac{(\Delta t)^3}{3!} + \frac{\partial^4\phi^l}{\partial t^4}\frac{(\Delta t)^4}{4!} + \frac{\partial^5\phi^l}{\partial t^5}\frac{(\Delta t)^5}{5!} + O((\Delta t)^6),$$

and

$$\phi^{l-1} = \phi^l - \frac{\partial\phi^l}{\partial t}\Delta t + \frac{\partial^2\phi^l}{\partial t^2}\frac{(\Delta t)^2}{2!} - \frac{\partial^3\phi^l}{\partial t^3}\frac{(\Delta t)^3}{3!} + \frac{\partial^4\phi^l}{\partial t^4}\frac{(\Delta t)^4}{4!} - \frac{\partial^5\phi^l}{\partial t^5}\frac{(\Delta t)^5}{5!} + O((\Delta t)^6),$$

and adding the two equations gives

$$\frac{\partial^2\phi^l}{\partial t^2} = \frac{1}{(\Delta t)^2}\left(\phi^{l+1} + \phi^{l-1} - 2\phi^l - \frac{\partial^4\phi^l}{\partial t^4}\frac{(\Delta t)^4}{12}\right) + O((\Delta t)^4). \tag{3.1}$$

Thus to estimate $\frac{\partial^2\phi^l}{\partial t^2}$ to order $O((\Delta t)^4)$ we need to calculate $\frac{\partial^4\phi^l}{\partial t^4}$, which we do by using the wave equation. We have

$$\begin{aligned}
\frac{\partial^4\phi^l}{\partial t^4} &= \frac{\partial^2}{\partial t^2}\left(\frac{\partial^2\phi^l}{\partial t^2}\right) = c^2\frac{\partial^2}{\partial t^2}\left(\frac{\partial^2\phi^l}{\partial x^2} + \frac{\partial^2\phi^l}{\partial y^2} + \frac{\partial^2\phi^l}{\partial z^2}\right)\\
&= c^2\left[\frac{\partial^2}{\partial x^2}\left(\frac{\partial^2\phi^l}{\partial t^2}\right) + \frac{\partial^2}{\partial y^2}\left(\frac{\partial^2\phi^l}{\partial t^2}\right) + \frac{\partial^2}{\partial z^2}\left(\frac{\partial^2\phi^l}{\partial t^2}\right)\right]\\
&= c^4\left[\frac{\partial^2}{\partial x^2}\left(\frac{\partial^2\phi^l}{\partial x^2} + \frac{\partial^2\phi^l}{\partial y^2} + \frac{\partial^2\phi^l}{\partial z^2}\right) + \frac{\partial^2}{\partial y^2}\left(\frac{\partial^2\phi^l}{\partial x^2} + \frac{\partial^2\phi^l}{\partial y^2} + \frac{\partial^2\phi^l}{\partial z^2}\right) + \frac{\partial^2}{\partial z^2}\Big(\dots\Big)\right]\\
&= c^4\left[\frac{\partial^4\phi^l}{\partial x^4} + \frac{\partial^4\phi^l}{\partial y^4} + \frac{\partial^4\phi^l}{\partial z^2} + 2\left(\frac{\partial^4\phi^l}{\partial x^2\partial y^2} + \frac{\partial^4\phi^l}{\partial x^2\partial z^2} + \frac{\partial^4\phi^l}{\partial y^2\partial z^2}\right)\right].
\end{aligned} \tag{3.2}$$

Denoting x by x_1, y by x_2 and z by x_3, and combining (3.1) and (3.2) we get

$$\phi^{l+1} = -\phi^{l-1} + 2\phi^l + (\Delta t)^2\left(\sum_{i=1}^{3}\frac{\partial^2\phi^l}{\partial x_i^2}\right) + \frac{c^4(\Delta t)^4}{12}\left(\sum_{i=1}^{3}\frac{\partial^4\phi^l}{\partial x_i^4} + 2\sum_{i\neq j}\frac{\partial^4\phi^l}{\partial x_i^2 x_j^2}\right) + O((\Delta t)^6), \tag{3.3}$$

which is accurate in time to $O((\Delta t)^4)$. We now calculate the spatial discretisation to $O((\Delta x)^{10})$, which involves calculating terms of the form $\frac{\partial^4\phi^l}{\partial x_i^4}$ and $\frac{\partial^4\phi^l}{\partial x_i^2 x_j^2}$. To do this we first calculate the second derivative to $O((\Delta x)^{10})$, which gives us for example

$$\frac{\partial^2\phi_{n,m,k}}{\partial x^2} = \frac{1}{(\Delta x)^2}\left[w_0\phi_{n,m,k} + \sum_{j=1}^{5} w_j(\phi_{n+j,m,k} + \phi_{n-j,m,k})\right], \tag{3.4}$$

where the coefficients w_j are as in [2], that is

$$w_0 = -\frac{5269}{900}, w_1 = \frac{10}{3}, w_2 = -\frac{10}{21}, w_3 = \frac{5}{63}, w_4 = -\frac{5}{504}, w_5 = \frac{1}{1525}. \tag{3.5}$$

We then can calculate the fourth order derivatives by using a central difference scheme on the second order derivatives, for example we have

$$\frac{\partial^4\phi_{n,m,k}}{\partial x^2\partial y^2} = \frac{1}{(\Delta y)^2}\left[\frac{\partial^2\phi_{n,m+1,k}}{\partial x^2} + \frac{\partial^2\phi_{n,m-1,k}}{\partial x^2} - 2\frac{\partial^2\phi_{n,m,k}}{\partial x^2}\right]. \tag{3.6}$$

This gives us a complete expression for the discretised form of the wave equation on a uniform mesh, which we get by combining (3.3), (3.4) and (3.6). We do not write out the full system here because it is not particularly insightful, but we show the stencil for the scheme in §3.3.

3.2 Discretised form of the wave equation on a non uniform mesh

We now turn our attention to discretising the wave equation on a mesh where the z coordinate is no longer uniform. We note that (3.3) still holds, but we can no longer discretise $\frac{\partial^2\phi}{\partial z^2}$ as in (3.4) or the mixed fourth derivatives involving z as in (3.6). Instead we use the coordinate transformation in §2.1 and discretise on the uniform grid ξ. We have

$$\frac{\partial\phi}{\partial z} = \frac{\partial\phi}{\partial\xi}\frac{\partial\xi}{\partial z} \text{ or } \frac{\partial\phi}{\partial\xi} = \frac{\partial\phi}{\partial z}\frac{\partial z}{\partial\xi},$$

and thus

$$\frac{\partial^2\phi}{\partial z^2} = \frac{\partial^2\phi}{\partial\xi^2}\left(\frac{\partial\xi}{\partial z}\right)^2 + \frac{\partial\phi}{\partial\xi}\left(\frac{\partial^2\xi}{\partial z^2}\right) \text{ or } \frac{\partial^2\phi}{\partial\xi^2} = \frac{\partial^2\phi}{\partial z^2}\left(\frac{\partial z}{\partial\xi}\right)^2 + \frac{\partial\phi}{\partial z}\left(\frac{\partial^2 z}{\partial\xi^2}\right). \tag{3.7}$$

Taking the second equation in (3.7) and rearranging we get

$$\begin{aligned}\frac{\partial^2\phi_{n,m,k}}{\partial z^2} &= \left(\frac{\partial z}{\partial\xi}\right)^{-2}\left(\frac{\partial^2\phi_{n,m,k}}{\partial\xi^2} - \frac{\partial\phi_{n,m,k}}{\partial z}\frac{\partial^2 z}{\partial\xi^2}\right)\\ &= \left(\frac{\partial z}{\partial\xi}\right)^{-2}\left(\frac{\partial^2\phi_{n,m,k}}{\partial\xi^2} - \frac{\partial^2 z}{\partial\xi^2}\frac{\partial\phi_{n,m,k}}{\partial\xi}\left(\frac{\partial z}{\partial\xi}\right)^{-1}\right).\end{aligned} \tag{3.8}$$

We are now ready to discretise $\frac{\partial^2\phi}{\partial z^2}$. Firstly we can calculate

$$\frac{\partial^2\phi_{n,m,k}}{\partial\xi^2} = \frac{1}{(\Delta\xi)^2}\left[w_0\phi_{n,m,k} + \sum_{j=1}^{5} w_j(\phi_{n,m,k+j} + \phi_{n,m,k-j})\right], \tag{3.9}$$

where the coefficients w_j are as in (3.5) and also that

$$\frac{\partial\phi_{n,m,k}}{\partial\xi} = \frac{1}{(\Delta\xi)}\left[\sum_{j=1}^{5} v_j(\phi_{n,m,k+j} - \phi_{n,m,k-j})\right], \tag{3.10}$$

where the coefficients v_j are as in [5], that is

$$v_1 = \frac{5}{6}, v_2 = -\frac{5}{21}, v_3 = \frac{5}{84}, v_4 = -\frac{5}{504}, v_5 = \frac{1}{1525}.$$

Next, we calculate the derivatives $\frac{\partial z}{\partial\xi}$ and $\frac{\partial^2 z}{\partial\xi^2}$ by exactly the same methods, so we get

$$\frac{\partial z}{\partial\xi} = \frac{1}{(\Delta\xi)}\left[\sum_{j=1}^{5} v_j(z_{n,m,k+j} - z_{n,m,k-j})\right], \tag{3.11}$$

and

$$\frac{\partial^2 z}{\partial\xi^2} = \frac{1}{(\Delta\xi)^2}\left[w_0 z_{n,m,k} + \sum_{j=1}^{5} w_j(z_{n,m,k+j} + z_{n,m,k-j})\right]. \tag{3.12}$$

Thus combining equations (3.8), (3.9), (3.10), (3.11) and (3.12) we get a spatial discretisation of $\frac{\partial^2\phi_{n,m,k}}{\partial z^2}$ w to $O((\Delta\xi)^{10})$. To calculate the fourth order mixed derivatives needed we calculate for example

$$\frac{\partial^4\phi_{n,m,k}}{\partial x^2 z^2} = \left(\frac{\partial z}{\partial\xi}\right)^{-2}\left(\frac{\partial^2}{\partial\xi^2}\left(\frac{\partial^2\phi_{n,m,k}}{\partial x^2}\right) - \frac{\partial^2 z}{\partial\xi^2}\frac{\partial}{\partial\xi}\left(\frac{\partial^2\phi_{n,m,k}}{\partial x^2}\right)\left(\frac{\partial z}{\partial\xi}\right)^{-1}\right),$$

and then employ a central difference scheme, giving us

$$\frac{\partial^4\phi_{n,m,k}}{\partial x^2 z^2} = \left(\frac{\partial z}{\partial \xi}\right)^{-2}\left(\frac{1}{(\Delta\xi)^2}\left[\frac{\partial^2\phi_{n,m,k+1}}{\partial x^2}+\frac{\partial^2\phi_{n,m,k-1}}{\partial x^2}-2\frac{\partial^2\phi_{n,m,k}}{\partial x^2}\right]\right.$$
$$\left.-\left(\frac{\partial z}{\partial \xi}\right)^{-1}\frac{\partial^2 z}{\partial \xi^2}\frac{1}{2\Delta\xi}\left[\frac{\partial^2\phi_{n,m,k+1}}{\partial x^2}+\frac{\partial^2\phi_{n,m,k-1}}{\partial x^2}\right]\right), \qquad (3.13)$$

which we combine with (3.4), (3.11) and (3.12) to get the discretisation.

3.3 Boundary terms and implementing x and y adaptivity

Having dealt with the case of non uniform z we now deal with the x and y adaptivity we discussed in §2.5. Before doing this, we consider the discretisation at the boundary. When we consider the discretisation in both the uniform and the non uniform mesh, at the next time level $\phi^l_{n,m,k}$ we are using one point at the same spatial coordinates and two time levels before, and the spatial stencil one time level before is shown in Figure 5. It is clear we need to change the stencil at boundary points, and to do this we simply change the points taken in the tenth order discretisation. At central points we take five points to the right and to the left, so at boundary terms we change the number of points we take to the left and the right. At the first point we take ten terms to the right, the second point we take one to the left and nine to the right, and so on until we reach the sixth point where we take five points either side. This carries on until the $(N-5)$th term (for the x boundary) where we take five either side, then at the $(N-4)$th term we take six points to the left and four to the right, until we reach the Nth point where we take ten points to the left. Since we are no longer taking central differences the coefficients change, but they can be calculated and have some symmetry. These boundary coefficients for the tenth order approximation are derived for first order derivatives in [5] and we could calculate similarly the coefficients for second order derivatives.

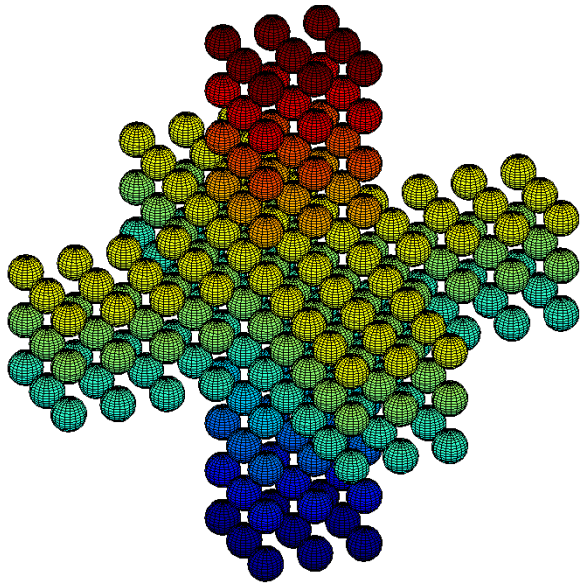


Figure 5: Stencil for a $O(\Delta t^4, \Delta x^{10})$ second derivative finite difference method.

We deal with the x and y adaptivity by implementing artificial boundaries at the points $k^{[2]}$, $k^{[3]}$ and $k^{[4]}$ if these points exist, and deal with points at the boundary as we have done above. Between 0 and $k^{[2]}$ we have differences Δx and Δy and we have points such as $\phi_{n-i,m,k}$ and $\phi_{n,m+j,k}$ in the finite difference scheme. Then between $k^{[2]}$ and $k^{[3]}$ we use differences $2\Delta x$ and $2\Delta y$ and use instead points such as $\phi_{n-2i,m,k}$ and $\phi_{n,m+2j,k}$, and similarly between $k^{[3]}$ and $k^{[4]}$. By doing this we ensure we have enough equations and unknowns to be able to solve the system, and we can not solve the system if we simply increase the differences at $k^{[i]}$ without making some sort of boundary modification.

3.4 Initial Conditions

We apply a Ricker wave as the initial condition, as in [2], which gives us two pieces of information. Firstly, the dominant frequency of the wave allows us to calculate the Nyquist frequency f_N and from that we can calculate the values of N,M and K from (2.9) and (2.10). It also tells us the solution of the wave equation at time zero, with the majority of points having zero intital conditions, while those around the epicentre of the Ricker wave will have non zero initial conditions.

4 Improvements and alternative meshes

We will now briefly discuss some possible improvements or alternatives to the mesh we derived in §2. We will not go into much detail and will just outline the methods, and the material is treated in some depth in both [1] and [4].

4.1 Different mesh density functions

We firstly consider using an alternative mesh density function $\boldsymbol{\rho}$ to one that is inversely proportional to the sound speed. Although choosing $\boldsymbol{\rho}$ inversely proportional to the sound speed is the obvious choice, we might be able to reduce computational time while maintaining accuracy if we choose a mesh density function based on an error estimate, such as the ones we briefly discussed in §2.4. We would experimentally try different mesh density functions and see how they compare to the method we have described in §2. The downside of implementing another mesh density function $\boldsymbol{\rho}$ is that we then have to solve the discretised system of the MMPDE and PDE alternatively, which can lead to some difficulties since we get a time lag, see [4, §2.6].

4.2 *p*-adaptivity

Another method that we could incorporate is p-adaptivity, which is where the order of the finite difference method varies according to the geometry of the region. When we reduce the order of the finite difference method (in regions where we have a smooth mesh) we reduce the computational time for each simulation, and thus any time savings we get will be magnified since we perform so many simulations on the model. The main drawback of incorporating the p-adaptivity is choosing which method to use in which region, with most industrial use based on concepts rather than solid theory. This component would be added to the mesh generation "engine" once we get a realistic model without varying the order.

4.3 Three dimensional moving mesh

In §2 we considered a moving mesh in just the z direction, but we could extend this to two or three dimensions using a similar method. Instead of a mesh density function $\boldsymbol{\rho}$ we get a mesh density matrix $\boldsymbol{M}$, and we could choose a matrix with components inversely proportional to the sound speed. We also need an extra condition, the alignment condition, which orients all the cells in the mesh and ensures that we can uniquely determine a transformation from a given mesh density matrix. From the equidistribution principle in §2.1 and the alignment condition, we can solve the system (again by discretising) to produce a coordinate transformation from the computational domain $\Omega_c = [0,1]^3$ to the physical domain Ω. The generation of such a mesh is covered in [1] and [4, §4]. Unfortunately, while calculating the moving mesh isn't so difficult, modifying the wave

equation to a form we can discretise on Ω_c becomes very computational heavy, and is much harder than the case in §3.2. For example, [4, §3] only performs the calculations in two dimensions, which is already getting very complicated. Trying to then implement a $O(\Delta t^4, \Delta x^{10})$ finite difference method would probably increase the computational time compared to the method in §2 and §3 since each simulation is more time consuming due to the complex discretisation.

5 Summary

We have investigated and presented a method to discretise the wave equation on a non uniform grid, based on a moving mesh. This is a relatively new area of numerical analysis with much research being carried out by Huang, Russell, Budd and co. A full rundown of the current state of the area is given in the review paper [1] and recent textbook [4]. Further work to extend this project would involve performing simulations on a industry standard velocity model and comparing the method we have presented with other methods. We would compare against methods such pseudospectral and finite difference methods on uniform grid to determine both accuracy and saving in computational time. Based on these simulations we could modify the method with some of the improvements we suggested in §4 and see how they compare in accuracy and time.

References

[1] C.J. Budd, W. Huang, and R.D. Russell. Adaptivity with moving grids. *Acta Numerica*, 18(1):111–241, 2009.

[2] M.A. Dablain. The application of high-order differencing to the scalar wave equation. *Geophysics*, 51(1):54–66, 1986.

[3] M. Grad, D. Gryn, A. Guterch, T. Janik, R. Keller, R. Lang, SB Lyngsie, V. Omelchenko, VI Starostenko, RA Stephenson, et al. Dobrefraction'99 velocity model of the crust and upper mantle beneath the donbas foldbelt (east ukraine). *Tectonophysics*, 371(1):81–110, 2003.

[4] W. Huang and R.D. Russell. *Adaptive moving mesh methods*. Springer, 2011.

[5] M. Sari, G. Gürarslan, and A. Zeytinoğlu. High-order finite difference schemes for solving the advection-diffusion equation. *Mathematical and Computational Applications*, 15(3):449–460, 2010.