Wave Propagation on a 2D Surface
PHY407

Genevieve Beauregard (1004556045)

1 Introduction

We numerically approximated acoustic propagation on a thin 2D area using a simple second-order Finite Difference Finite Time routine (FDTD). The numerical schemes of the project is mostly taken from class work, while the acoustic setup comes from Pengliang Yang's primer, "A numerical tour of wave propagation" [11]. We also implemented a basic acoustic wave with reflective boundary conditions and then attempt to implement a basic Sponge Absorbing Boundary Condition (ABC) at the edges of the walls.

Many of these techniques can be applied to general wave functions, but my sources mostly came from music modelling and geophysics. These techniques are often used in seismic models for modelling the acoustic pressure shockwaves for geophysical models.

The Scripts for this project are the following

- `PHY407_FinalProject_Wave.py` for the FDTD acoustic wave with no damp
- `PHY407_FinalProject_Boundary.py`
- `PHY407_Functions.py`

2 The Classic Wave Equation Problem

We want to solve for the motion of pressure wave propagation through the ground. We simplify this, by only considering the "ground" as a 2D surface (real geophysical models would use three dimensions).

The 2D acoustic wave equation (without damping) with respect to pressure $P(x, y, t)$ is defined as follows:

$$\frac{1}{v(x,y)^2}\frac{\mathrm{d}^2P}{\mathrm{d}t^2} = \left(\frac{\mathrm{d}^2P}{\mathrm{d}x^2} + \frac{\mathrm{d}^2P}{\mathrm{d}y^2}\right) + F(x,y,t) \tag{1}$$

where $v(x, y)$ is the compressional p-wave velocity (as opposed to s-waves which are traverse waves). We treat v like a constant in the isotrophic case. $F(x, y, t)$ is a source term; a way to add disturbances in to the model. We will model a simple case where $F(x, y, t) = 0$, reducing our problem to a simple homogenous wave equation.

This assumes no energy loss to damping of our medium. We will model damping in the next section.

2.1 Parameters

We will solve the wave equation on a rectangular surface length L_x and height L_y. We will use the Neumann boundary condition on the walls:

$$\frac{\mathrm{d}P}{\mathrm{d}x} = \frac{\mathrm{d}P}{\mathrm{d}y} = 0x = 0, L_x, y = 0, L - y \tag{2}$$

And the initial conditions

$$P(x, y, 0) = A \exp\left\{\frac{1}{2}\left[\left(\frac{x - x_0}{\sigma_x}\right)^2 + \left(\frac{y - y_0}{\sigma_y}\right)^2\right]\right\} \tag{3}$$

$$\frac{\mathrm{d}P(x, y, 0)}{\mathrm{d}t} = 0 \tag{4}$$

We set the parameters to the following values;

1. $L_x = L_y = 300\text{m}$
2. $A = 10\text{m}$
3. $x_0 = L_x/2 = 150\text{m}$
4. $y_0 = L_y/2 = 150\text{m}$
5. $\sigma_x = L_x/100 = 3\text{m}$
6. $\sigma_y = L_y/100 = 3\text{m}$
7. $v(x, y) = 340\text{m}$ (about the speed of sound in air)

Our initial condition is basically a very tall and narrow gaussian peak at the centre of the box.

2.2 Method

2.2.1 Discretization of Spatial and Temporal Domain

We first create a discrete grid of N_x x N_y points in the $x - y$ plane to represent an square 2D surface at an initial neutral static equilibrium position.

Spatial data at a given time t is stored N_x x N_y arrays where the i-coordinate represents the x position and j-coordinate represents the y position.

We use a space step of $\Delta x = \Delta y = 1\text{m}$ and step of $\Delta t = 0.005\text{m}$. We will discuss these choices in our analysis section.

2.2.2 Finite Difference Time Domain

We then use a first order finite difference scheme to calculate the spatial derivatives, denoting the indices i, j as the x-coordinates and y coordinates respective and the superscript n as the time step.

$$\left.\frac{\mathrm{d}^2 P}{\mathrm{d}x^2}\right|_{i,j}^{n} \approx \frac{P^n_{i+1,j} - 2P^n_{i,j} + P^n_{i-1,j}}{(\Delta x)^2} \tag{5}$$

$$\left.\frac{\mathrm{d}^2 P}{\mathrm{d}y^2}\right|_{i,j}^{n} \approx \frac{P^n_{i,j+1} - 2P^n_{i,j} + P^n_{i,j-1}}{(\Delta y)^2} \tag{6}$$

For simplicity, we will set $\Delta x = \Delta y$. Taking the sum of the two derivatives we get:

$$\nabla^2 P_{i,j} = \left.\frac{\mathrm{d}^2 P}{\mathrm{d}x^2} + \frac{\mathrm{d}^2 P}{\mathrm{d}y^2}\right|_{i,j}^{n} = \frac{P^n_{i+1,j} - 2P^n_{i,j} + P^n_{i-1,j}}{(\Delta x)^2} + \frac{P^n_{i,j+1} - 2P^n_{i,j} + P^n_{i,j-1}}{(\Delta y)^2} \tag{7}$$

$$= \frac{P^n_{i+1,j} + P^n_{i-1,j} + P^n_{i,j+1} + P^n_{i,j-1} - 4P^n_{i,j}}{(\Delta x)^2} \tag{8}$$

We will define a function `FDLaplaceEstimate` to handle this. As we are doing neuman boundary conditions at the edges, we can just ignore the edge terms.

As this is a second order ODE with respect to the temporal domain, we implement a simple verlet time-step, with $\Delta t = h$:

$$\left.\frac{\mathrm{d}^2 P}{\mathrm{d}t^2}\right|_{i,j} = \frac{P^{n+1}_{i,j} - 2P^n_{i,j} + P^{n-1}_{i,j}}{(h)^2} \tag{9}$$

Rearranging everything:

$$\frac{P^{n+1}_{i,j} - 2P^n_{i,j} + P^{n-1}_{i,j}}{(h)^2} = v^2(\nabla^2 P_{i,j}) + F^n_{i,j} \tag{10}$$

We require the forward time step at $n+1$ at point $\{i, j\}$, so we rearrange everything in terms of of the previous time-steps:

$$P^{n+1}_{i,j} = v^2 h^2 \nabla^2 P_{i,j} + h^2 F_{i,j} + 2P^n_{i,j} - P^{n-1}_{i,j}. \tag{11}$$

We account for the $n = -1$ term by 'inventing' a point with our initial condition. Using our central finite difference scheme our initial velocity at $n = 0$ would be equivalent to:

$$\partial_t P^0_{i,j} = \frac{P^1_{i,j} - P^{-1}_{i,j}}{2\Delta x} \tag{12}$$

When we substitute it back into 2.2.2 we have;

$$P^{n+1}_{i,j} = P^n_{i,j} - \Delta t(\partial_t P^0_{i,j}) + \frac{1}{2}(v\Delta t)^2 \nabla P^n_{i,j} \tag{13}$$

2.3 Results

Our numerically obtained waves can be viewed in Fig 1. An animation of the results is attached under the file Wave.mp4.

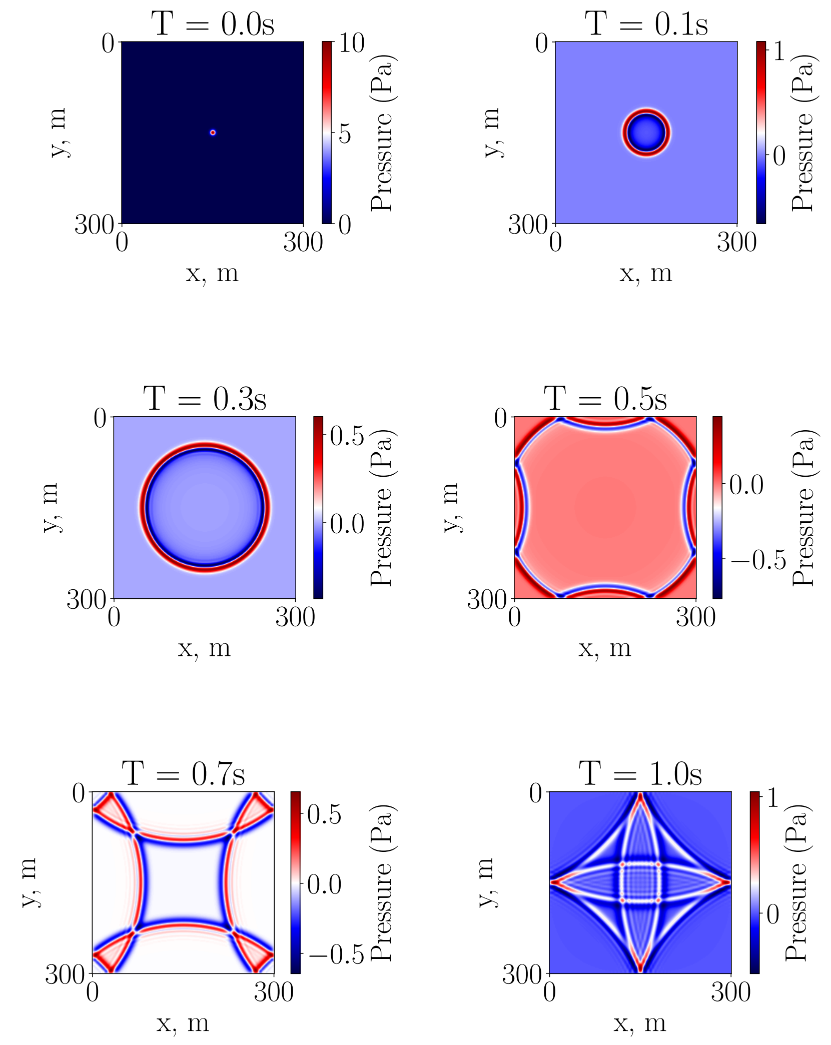


Figure 1: Numerically estimated FDTD wave equation with Newmann conditons at various time points.

2.4 Analysis

The stability of the finite difference scheme is somewhat dependent on the Courant number being less than 1 each [1] [1];

$$C = 2\Delta x \frac{v}{\Delta t} \lesssim 1 \tag{14}$$

and similarly for the Δy value. Our conditions fulfilled this criteria.

FDTD methods generally exhibit anistropic error, due to the dispersion of the wave number [10]. While we managed to avoid seeing explicit phenomema of this, this beheviour may be apparent if we extended the box boundaries.

The error in the FDTD method comes primarily from the central difference schemes which have an error of $O(\Delta t^2)$ and $O(\Delta x^2)$ (or $O(\Delta y^2)$ for the y terms) from the respective grids. This can be mitigated by taking higher order finite differences or varying the stencil shapes. Another potential way to minimise the error is to utilise fourier derivatives for the laplacian.

FDTD methods are generally computationally intensive compared to its spectral domain method counterparts. However, it has owing to improvements in GPU in recent years, it has seen a resurgence due to its parallelization capability as well as the ability to handle irregular boundary conditions. [7].

3 Approximating Open Boundary: Sponge Boundary Conditions

We have derived the Neumann boundary conditions (Dirichlet is similar). If we want to remove the effect of boundaries on a wave function, we need to implement an open boundary condition, sometimes known as the radiation condition:

$$P_t + v^2 \nabla P = 0$$

This is particularly useful if local phenomena away from boundaries needs to be observed. While it is possible to develop a finite difference scheme of the above boundary in 1D, it has been known to be incredibly tricky as it scales up with dimensions [4]. We implement an ABC instead.

We will attempt implement a Sponge Absorbing Boundary condition on the edges of our grid to investigate this effect. We will extend our grid to nb number of layers on the top and bottom of our grid; what was an $N_x \times N_y$ grid points changes to $(N_x + 2nb) \times (N_y + 2nb)$ grid, with additional nb columns and rows at the edges. A good schematic of the setup can be seen in Fig 2. We adapt the methods from Yao et al. [12][2]

At each time step, we want to decay our numerically obtained P^{n+1} at the absorbing layers to minimise the effects of the reflection before it hits the "ghost"

[1]its noted this may be quite difficult [1]

[2]Their scheme was a bit more complicated

boundary layer. It can be shown the solution for a wave with damping is of the following form:

$$A = \exp\{-Dt\}e^{j(\omega t - \vec{k}\vec{r})} \tag{15}$$

where

$$D(l) = \log\left(\frac{1}{\alpha}\right)\frac{3v(x,y)}{2L_{nb}}\left(\frac{l}{L_{nb}}\right)^2$$

where l is the distance between a cell in the absorbing layer and the boundary, α the theoretical reflection coefficient and L_nb is the length of the absorbing layer [12]. The implementation would involve multiplying this exponential factor to the time-stepping solution. [3]

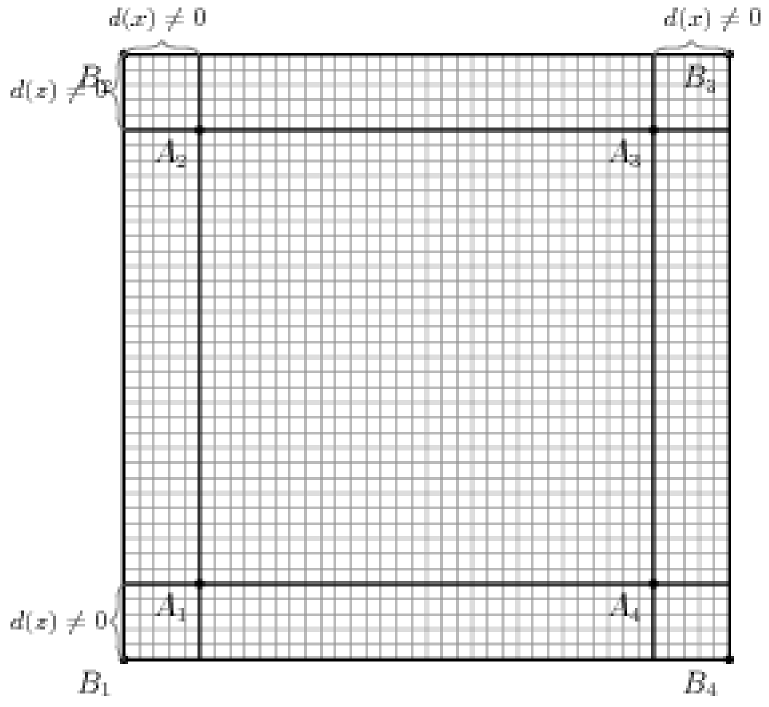


Figure 2: A solution grid with absorbing boundary layers on all four edges. A marks to corners of the solution space we want to consider. For our scheme, we will only implement the top and bottom layers. Image credited to Pengliang Yang [6].

We extended our grid by nb layers on each edge, in a routine similar to 2. We compute a matrix for $d(x)$ and a matrix for $d(x)$. We did some trial and error with the reflection coefficient for awhile with $nb = 20, 50, 100$ and found that $\alpha = 0.99$ works fairly well. We will only show the relevant graphs to save

[3]the authors of the paper used a pseudospectral method to solve the wave equation

space. We implemented the same Neumann boundary on the extended grid (the layers) and utilised the same initial conditions.

3.1 Results

We obtained the plots the results at $T = 1.0$ for selected $\alpha = 0.1, 0.5, 0.9$ and selected $nb = 20, 50, 100$.

Plots for $nb = 20$ and different α can be obtained in Figure 3. Plots for $\alpha = 0.99$ and $nb = 50$ can be obtained in Figure 4. The plots are in the Appendix. An animation is attached to this submission simulating $nb = 50$ and $\alpha = 0.99$ under the filename `DampedWave.mp4` to see the different reflective motions that occur.

We see immediately there is very little effect of the reflection coefficient; beyond the size of the maximum amplitude pockets (the red areas), overall next result of the wave looks very similar to the final result of the free reflective case.

The motion for the reflection appears to be a staggered reflection, this may be due partial reflection occurring at various points in the sponge layer.

However, the number of layers has a large effect on the amplitude reduction as seen in Figure 4. This is to be expected: more absorbing layers gives more space for the wave to loses amplitude by exponential decay before it hits to the true reflective Neumann layer.

3.2 Analysis

This is definitely not a perfect scheme at mimic open boundary layers; it merely dissapates the wave and removes energy from the system at the boundaries. It may however be a decent approximation of a wave hitting a medium with significantly higher damping. It may be possible that I misunderstood the source given the lack of absorption occurring. Overall, the scheme we implemented was not very successful at mimic open boundary conditions. However, if we ignore wave amplitudes below a certain point, it may be used to approximate the movement of the wave peaks with open boundaries.

The computational time is extended by a significant amount due to the increased grid size. Given that the number of layers appears to be the main factor into reducing reflection amplitudes, this scheme would become highly inefficient extremely fast (our code takes awhile to run). There are other more efficient ABC methods such as Perfectly matched layers that are far more efficient at absorbing incident waves.

References

[1] Aslam Abdullah. Simplified von neumann stability analysis of wave equation numerical solution. *Journal of Engineering and Applied Sciences*, 14:4164–4168, 12 2019.

[2] S.V. Bosakov. Eigenfrequencies and modified eigenmodes of a rectangular plate with free edges. *Journal of Applied Mathematics and Mechanics*, 73(6):688 – 691, 2009.

[3] Charles Cerjan, Dan Kosloff, Ronnie Kosloff, and Moshe Reshef. A nonreflecting boundary-condition for discrete acoustic and elastic wave-equations. *Geophysics*, 50:705–708, 04 1985.

[4] Heiner Igel. Computers, waves, simulations: A practical introduction to numerical methods using python. `https://www.coursera.org/learn/computers-waves-simulations/home/welcome`.

[5] Sara Javidpoor, Nassim Ale Ali, and Amer Kabi. Time integration of rectangular membrane free vibration using spline-based differential quadrature. *Journal of Applied and Computational Mechanics*, 2(2):74–79, 2016.

[6] Hans Petter Langtangen. `http://hplgit.github.io/INF5620/doc/notes/main_wave.html#wave:2D3D`, 2012.

[7] DongGun Lee, TaeHyung Kim, and Q-Han Park. Performance analysis of parallelized pstd-fdtd method for large-scale electromagnetic simulation. *Computer Physics Communications*, 259:107631, 2021.

[8] Hisao Nakanishi Nicholas J. Giordano. *Computational Physics (2nd Edition)*. Benjamin Cummings, 2nd edition, 2005.

[9] George Rawitscher, Victo dos Santos Filho, and Thiago Carvalho Peixoto. *Chebyshev Polynomials as Basis Functions*, pages 43–62. Springer International Publishing, Cham, 2018.

[10] Adrian Sescu. Numerical anisotropy in finite differencing. *Advances in Difference Equations*, 2015(1):9, 2015.

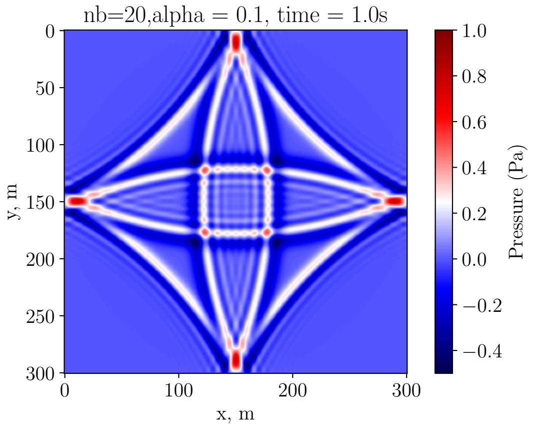
[11] Pengliang Yang. `http://www.ahay.org/RSF/book/xjtu/primer/paper_html/node2.html`, May 2014.

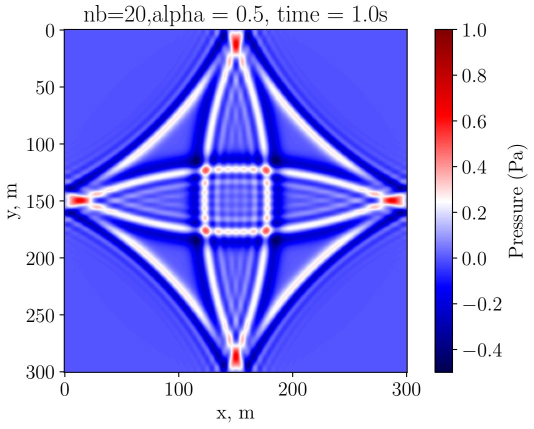
[12] Gang Yao, Nuno V da Silva, and Di Wu. An effective absorbing layer for the boundary condition in acoustic seismic wave simulation. *Journal of Geophysics and Engineering*, 15(2):495–511, 02 2018.

4 Appendix

4.0.1 Code Implementation FDTD first order

```
Define Parameters L_x, L_y.
Initialize Constants

Define Grid Resolution of each side N

Create Mesh XX, YY
Set initial P0, V0

Initialize time array

Initialize array stack of dimension (len(time), n_x, n_y)

Obtain first time step and store in stackarray

Populate array stack with initial P0, P1
Initialize spate partial Derivative Matrix dPy2, 2Px2

Define GetNextP:
        Calculate spatial derivates using array methods
        Obtain next time step as per equation
        Enforce Boundary Condition Dirichlet P_now = 0
    return P_next

Def FDLaplaceEstimate:
    Calculate laplacian
    return Laplacian Array

For loop over time from n = 1 to len(time):
    Get Next P
    Store P
Plot
```





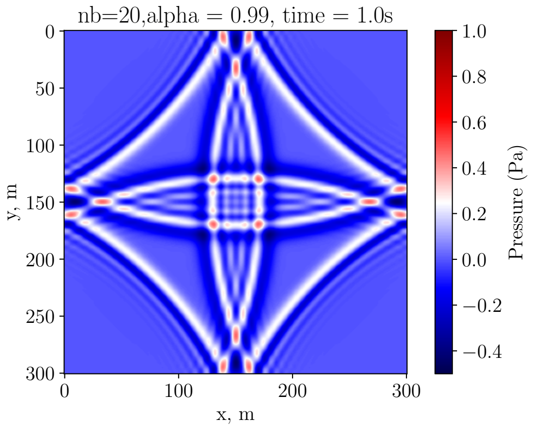


Figure 3: Comparing the reflection coefficient values against the layers shows very little change in the disapation.

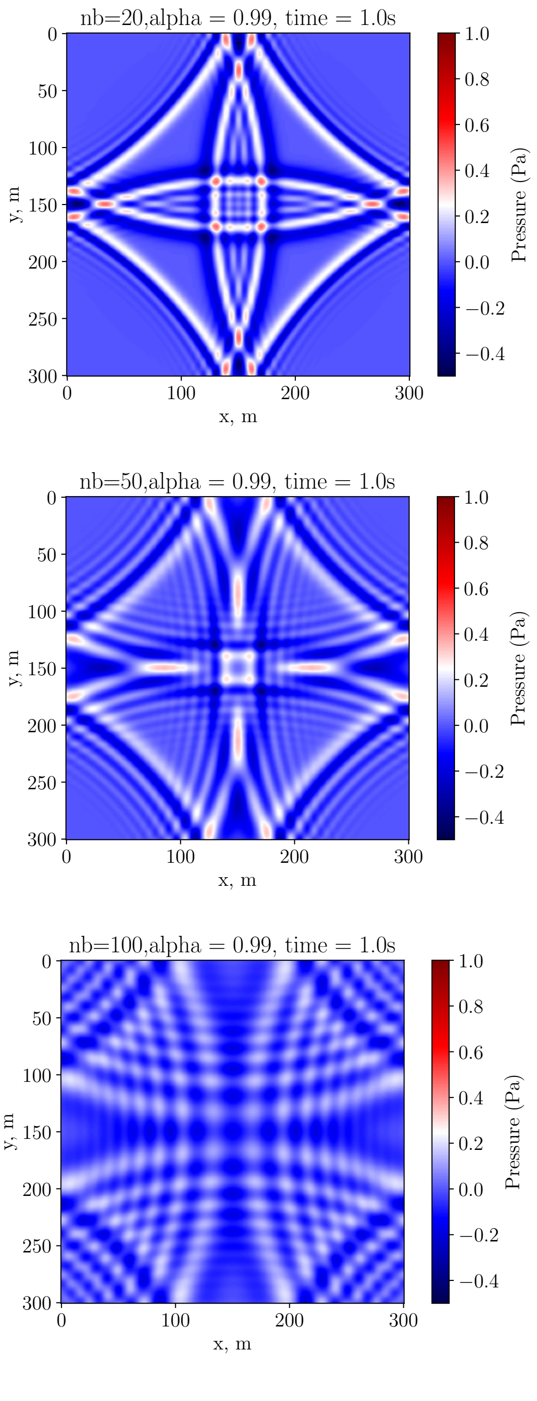


Figure 4: Different nb number of layers, for $\alpha = 0.99$. Notice the reduction in amplitude.