A Project Report

On

Reflection & Refraction of wave in a non-ideal medium

BY

Shubhan Mital 2019B4A30900H

Under the supervision of **Dr. Sumit Kumar Vishwakarma**

SUBMITTED IN FULFILLMENT OF THE REQUIREMENTS OF

MATH F366 : LAB PROJECT

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI

HYDERABAD CAMPUS

(May 2023)

ACKNOWLEDGMENTS

I would like to thank Dr. Sumit Kumar Vishwakarma without whose guidance this project would not be successful. Specifically, I would like to thank Sumit sir again for his patience as well as his teaching in the courses such as Optimization and Numerical Methods which helped me very much in understanding the project thoroughly, I would also like to thank the Mathematics department of BPHC for providing us chance to do design projects.

Birla Institute of Technology and Science-Pilani, Hyderabad Campus

Certificate

This is to certify that the project report entitled "**Reflection & Refraction of wave in a non-ideal medium"** submitted by Mr. Mohit Agrawal (ID No.2019B4AA0918H), Mr. Raghav Gupta (ID No. 2019B4A30927H) and Mr. Shubhan Mital (ID No.2019B4A30900H) in fulfillment of the requirements of the course MATH F366, Lab Project Course, embodies the work done by them under my supervision and guidance.

Date: 3rd May 2023

(Dr. Sumit Kumar Vishwakarma**)**

BITS- Pilani, Hyderabad Campus

ABSTRACT

This work proposes a very effective implicit finite difference approach for the propagation of acoustic waves. In comparison to traditional central difference derivative operators, the proposed method uses fewer points to estimate derivatives. The coefficients that appear on the left and right sides of the implicit scheme are calculated to the requisite precision based on the Taylor series expansion of the function at each point. The comparison between many characteristics of the implicit scheme and numerous CD schemes of various orders in the study demonstrates how effective, precise, and little in size the suggested technique is. While minimising errors, it can significantly reduce the amount of computational resources. This scheme's use for seismic wave propagation in 2D and 3D acoustic medium is illustrated in the study. In conclusion, this article makes a substantial addition to geophysics and offers helpful insights into the creation of numerical schemes for wave propagation.

TABLE OF CONTENTS

REVIEW OF THE PAPER

The paper "Efficient method for acoustic wave simulation using implicit FD operator" describes a very effective numerical method for acoustic wave propagation. Malkoti and colleagues are the authors of the article, which was released in the Journal of Applied Geophysics in 2019.

The significance of numerical seismic simulations in seismic processes, including complete waveform inversion and reverse time migration methods, is covered in the first section of the study. The precision of the numerical derivative operator is closely related to the accuracy of a numerical scheme. The classical central derivative (CD) operator and the CD operator over the staggered grid are two examples of the several types of numerical derivative operators that have been used for wave propagation. These techniques do, however, have accuracy and computing resource limits.

In contrast to typical central difference derivative operators, the approach presented in this study is an implicit finite difference scheme that uses fewer points to estimate derivatives. The Taylor series expansion of the function at each point serves as the foundation for the implicit scheme. The coefficients that appear on the left (,) and right (a, b, c) sides are chosen with the necessary precision.

We first use the Taylor series to extend either side of the equation, then we match the coefficients of the corresponding derivative to get the coefficients in the equation. The comparison shows us how the coefficients are related.

Below is the derivation of it.

Eq (4) from Section 2.2

$$f_i'' + \alpha (f_{i+1}'' + f_{i-1}'') + \beta (f_{i+2}'' + f_{i-2}'') = a \frac{f_{i+1} - 2f_i + f_{i-1}}{h^2} + b \frac{f_{i+2} - 2f_i + f_{i-2}}{(2h)^2} + c \frac{f_{i+3} - 2f_i + f_{i-3}}{(3h)^2}$$

applying fourier on both sides

LHS

$$f_{i+1}'' = f_i'' + f_i''' h + f_i^4 \frac{h^2}{2!} + f_i^5 \frac{h^3}{3!} + \dots$$

$$f_{i-1}'' = f_i'' - f_i''' h + f_i^4 \frac{h^2}{2!} - f_i^5 \frac{h^3}{3!} + \dots$$

$$f_{i+2}'' = f_i'' + 2f_i''' h + 4f_i^4 \frac{h^2}{2!} + 8f_i^5 \frac{h^3}{3!} + \dots$$

$$f_{i-2}'' = f_i'' - 2f_i''' h + 4f_i^4 \frac{h^2}{2!} - 8f_i^4 \frac{h^3}{3!} + \dots$$

Putting the above equations in LHS of eqn (4), we get

$$f_i'' + \alpha \left(2f_i'' + 2f_i^4 \frac{h^2}{2!} + 2f_i^6 \frac{h^4}{4!}\right) + \beta \left(2f_i'' + 8f_i^4 \frac{h^2}{2!} + 32 f_i^6 \frac{h^4}{4!}\right) \quad \text{— (a)}$$

RHS

$$f_{i+1} = f_i + f_i' h + f_i'' \frac{h^2}{2!} + f_i^3 \frac{h^3}{3!} + f_i^4 \frac{h^4}{4!} + \dots$$

$$f_{i-1} = f_i - f_i' h + f_i'' \frac{h^2}{2!} - f_i^3 \frac{h^3}{3!} + f_i^4 \frac{h^4}{4!} + \dots$$

$$f_{i+2} = f_i + 2f_i' h + 2f_i'' h^2 + 4f_i^3 \frac{h^3}{3} + 8f_i^4 \frac{h^4}{3} + \dots$$

$$f_{i-2} = f_i - 2f_i' h + 2f_i'' h^2 - 4f_i^3 \frac{h^3}{3} + 8f_i^4 \frac{h^4}{2} + \dots$$

$$f_{i+3} = f_i + 3f_i'h + 9f_i''\frac{h^2}{2} + 27f_i'''\frac{h^3}{6} + 81f_i^4\frac{h^4}{24} + \dots$$

$$f_{i-3} = f_i - 3f_i'h + 9f_i''\frac{h^2}{2} - 27f_i'''\frac{h^3}{6} + \dots$$

Putting these equations in RHS of eqn (4), we get

$$\left(\frac{a}{h^2}\right)\left(2f_i''\frac{h^2}{2!} + 2f_i^4\frac{h^4}{24} + \dots\right) + \left(\frac{b}{(2h)^2}\right)\left(4f_i''h^2 + 4f_i^4\frac{h^4}{3!} + \dots\right)$$

$$+\left(\frac{c}{(3h)^2}\right)\left(18f_i''\frac{h^2}{2} + 162f_i^4\frac{h^4}{24} + \dots\right) \quad -(b)$$

Now comparing the eqn (a) & (b) we get

h^0, f'' : $(1 + 2\alpha + 2\beta) = 2\frac{1}{2!}(a+b+c)$

h^2, f^4 : $(2)\frac{1}{2!}(\alpha + 2^2\beta) = 2\frac{1}{4!}(a + 2^2b + 3^2c)$

$\vdots$

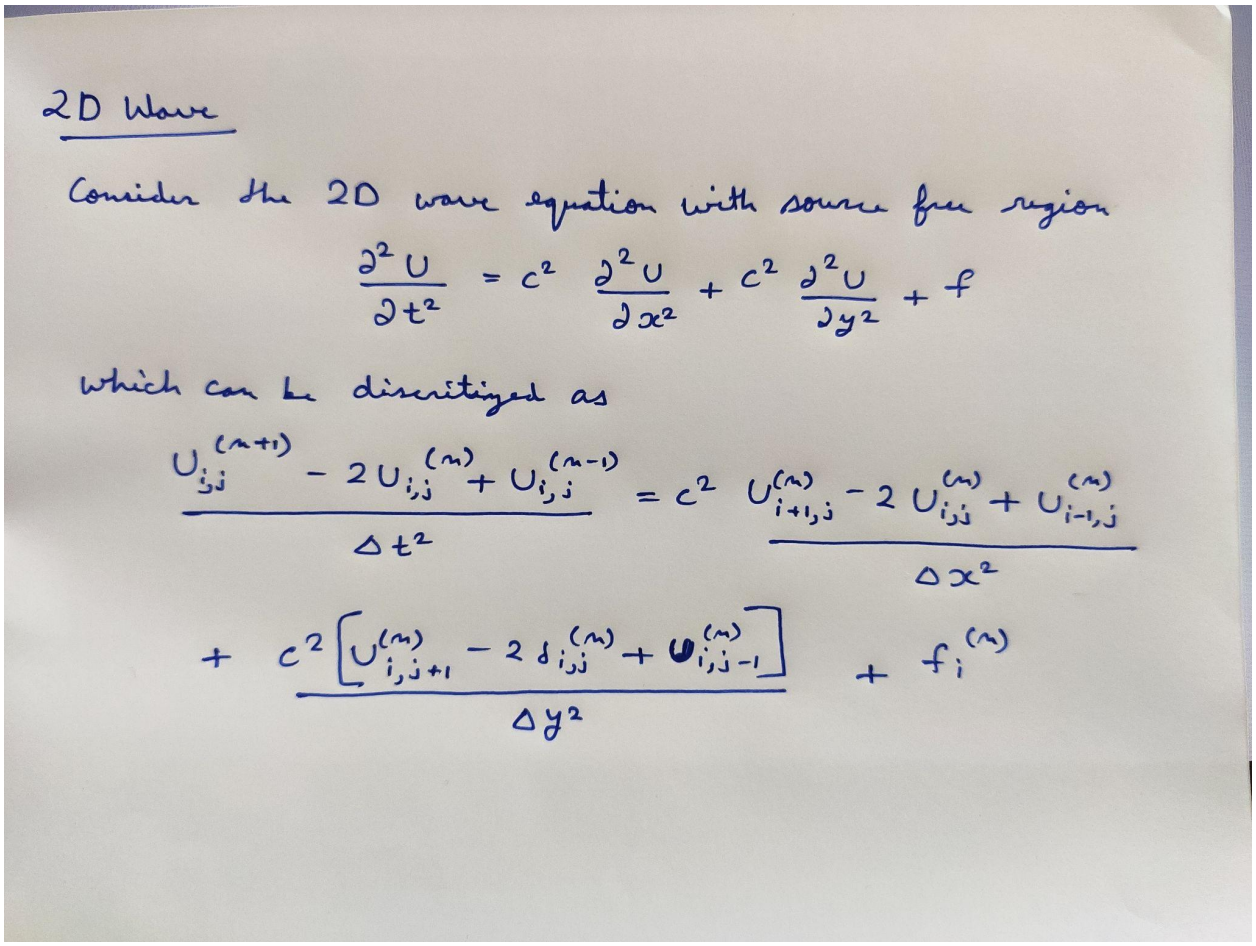
↳ eqn (5)

The suggested approach provides a number of benefits over traditional central difference systems. Compared to lengthy explicit finite difference operators, it has a smaller stencil and is more local in nature. As a result, it becomes more effective and uses less processing power while reducing mistakes.

The implicit scheme's spectral properties are comparable to those of traditional CD schemes of 10th order. For all schemes, including the implicit scheme, the error characteristics (i.e., minima and maxima locations) are equivalent.

This scheme's use for seismic wave propagation in 2D and 3D acoustic medium is illustrated in the study. The findings demonstrate how effective, precise, and small the suggested approach is. The computational expense of the plan may be cut practically in half.

Finite Difference Scheme of 2D Boundaries Wave Equation Based on Absorbing and Reflecting Time Dependent Wave Propagation Modelling



The boundary has the ability to reflect or absorb the wave.

We consider the end points as Dirichlet boundary conditions given by :

$$u\left(0,y,t\right)=0,u\left(Lx,y,t\right)=0 \text{ and } u\left(x,0,t\right)=0,u\left(x,Ly,t\right)=0$$

For absorption across the boundary, the absorbing boundary conditions are:

$$\left.\frac{\partial u}{\partial x}\right|_{x=0}=c\left.\frac{\partial u}{\partial t}\right|_{x=0},\ \left.\frac{\partial u}{\partial x}\right|_{x=Lx}=-c\left.\frac{\partial u}{\partial t}\right|_{x=Lx}$$

$$\left.\frac{\partial u}{\partial y}\right|_{y=0}=c\left.\frac{\partial u}{\partial t}\right|_{y=0},\ \left.\frac{\partial u}{\partial y}\right|_{y=Ly}=-c\left.\frac{\partial u}{\partial t}\right|_{y=Ly}$$

which can be discretized by

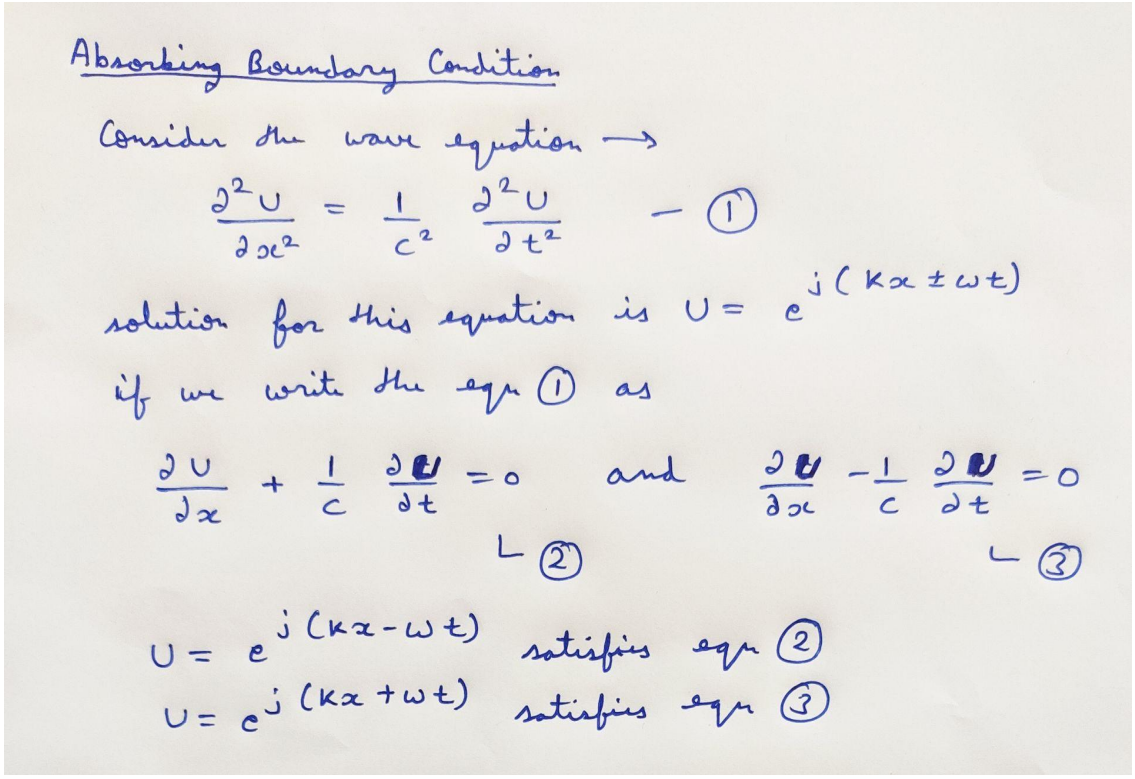
$$u_{1,j}^{n+1}=u_{2,j}^{n}+\frac{CFL-1}{CFL+1}\left[u_{2,j}^{n+1}-u_{1,j}^{n}\right]$$

$$u_{n_x,j}^{n+1}=u_{n_x-1,j}^{n}+\frac{CFL-1}{CFL+1}\left[u_{n_x-1,j}^{n+1}-u_{n_x,j}^{n}\right]$$

$$u_{i,1}^{n+1}=u_{i,2}^{n}+\frac{CFL-1}{CFL+1}\left[u_{i,2}^{n+1}-u_{i,1}^{n}\right]$$

$$u_{i,n_y}^{n+1}=u_{i,n_y-1}^{n}+\frac{CFL-1}{CFL+1}\left[u_{i,n_y-1}^{n+1}-u_{i,n_y}^{n}\right]$$

Below is the derivation of these boundary conditions for 1D case:



Here K = wave number (it is the number of waves present per meter)

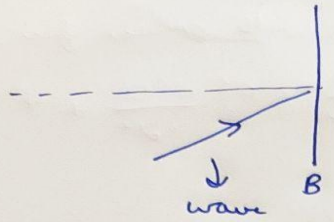
K = ω/C

Now, if we consider a Right hand side boundary and a wave travelling towards it and if we impose the condition given in eqn-2 then the right travelling wave will not see any reflection.

So eqn-2 condition is the perfect boundary condition for 1-D.

What happens in 2D if we impose this condition?

For 2D Wave



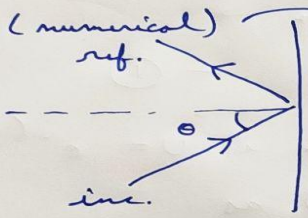
Eqn of the wave $\longrightarrow$ $e^{j(\omega t - k_x x - k_y y)}$

Now at the Boundary (B) we will impose

$$\frac{\partial U}{\partial x} + \frac{1}{c}\frac{\partial U}{\partial t} = 0$$

will give $-j k_x + j\frac{\omega}{c} \neq 0$ as $k_x^2 + k_y^2 = (\omega/c)^2$

so the wave will get numerically reflected.



$k_x = k\cos\theta$

$k_y = k\sin\theta$

inc : $e^{j(\omega t - k\cos\theta x - k\sin\theta y)}$

ref : $R e^{j(\omega t + k\cos\theta x - k\sin\theta y)}$ (R → ref. Coefficient)

Total field = inc + ref

Now at the Boundary, we will impose

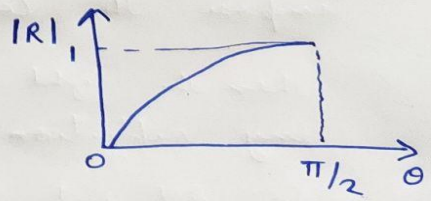
$$\left(\frac{\partial}{\partial x} + \frac{1}{c}\frac{\partial}{\partial t}\right)(\text{total}) = 0$$

$k = \omega/c$

we get $(-k\cos\theta + Rk\cos\theta) + \frac{1}{c}(\omega + R\omega) = 0$

$$\Rightarrow R = \frac{\cos\theta - 1}{\cos\theta + 1}$$

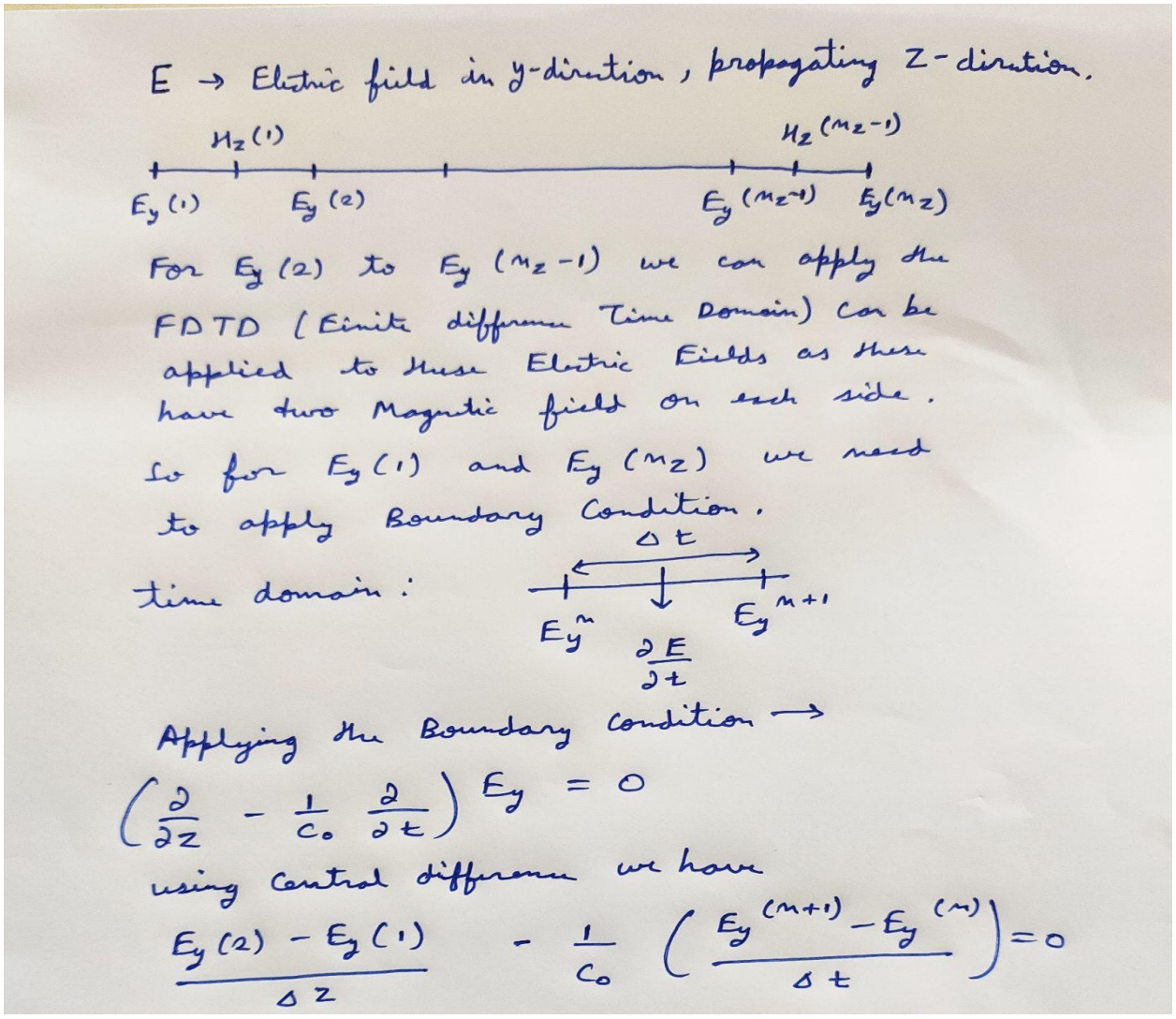
Reflection Coefficient



Now for $\theta = 0 \Rightarrow R = 0$

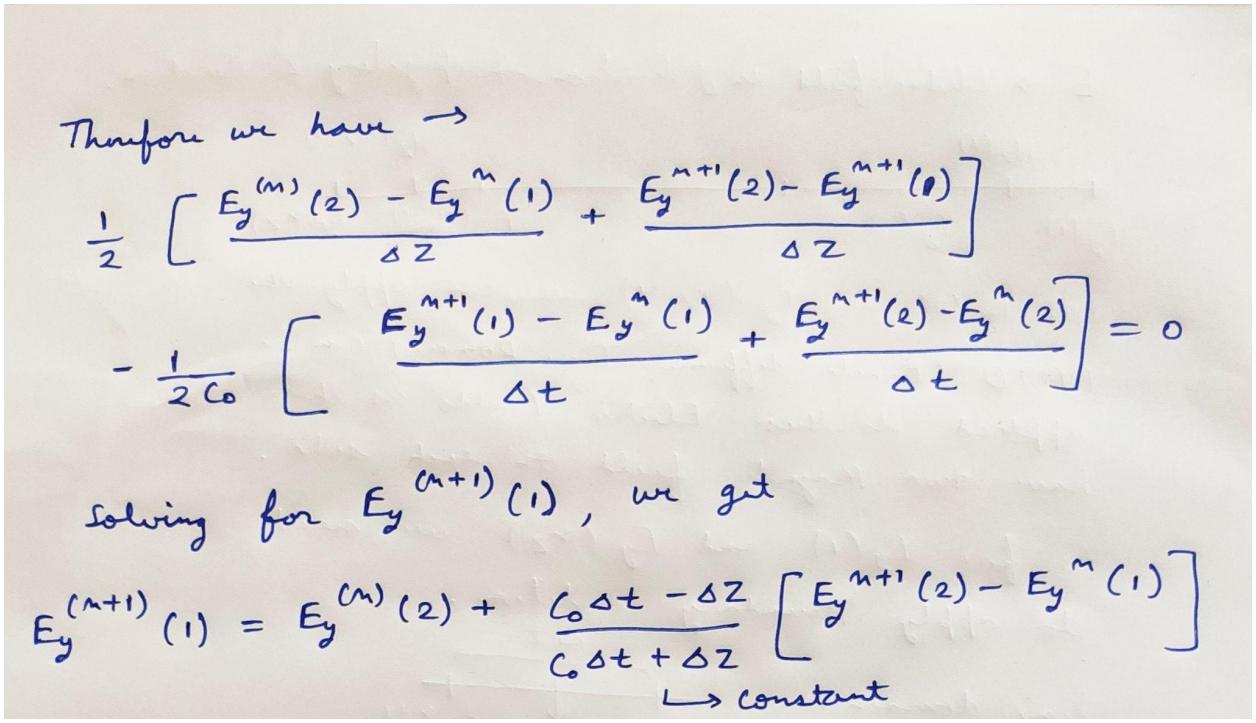
↳ perfectly absorbed.

MUR Boundary Condition of 1D



But this equation we got after applying the boundary condition is not correct. The problem is that the spatial derivative is defined either at Ey(n) or Ey(n+1) but not at Ey(n+½). So we need to take the average of the spatial derivative at these two points. Same thing we have to do for the time derivative also.

Below is the updated equation taking into account the average:



This eqn we got is similar to the discretized absorbing boundary condition we have in terms of CFL.

MATLAB & PYTHON CODES ALONG WITH RESULTS

1) 2D ABSORB

2D ABSORB MATLAB CODE

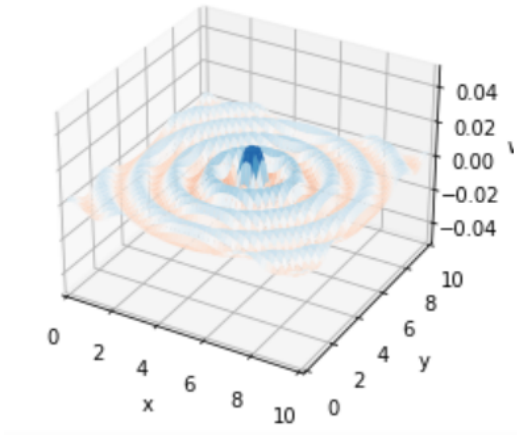
```
%%Solving a PDE
clear;
%Equation
% wtt= c^2 wxx + c^2 wyy + f

%%Domain
%space
Lx=10;
Ly=10;
dx=0.1;
dy=dx;
nx=fix(Lx/dx);
ny=fix(Ly/dy);
x=linspace(0,Lx,nx);
y=linspace(0,Ly,ny);

%Time
T=10;

%%Field variables
%Variables
wn=zeros(nx,ny);
wnm1=wn; % w at time n-1
wnp1=wn; % w at time n+1

%Parameters
CFL=0.5;
c=1;
dt=CFL*dx/c;

%%Initial Conditions
%% Time Stepping loop
t=0;

while(t<T)
  %Reflecting boundary conditions
  %wn(:,[1 end])=0;
  %wn([1 end],:)=0;

  %Absorbing boundary conditions
  wnp1(1,:)=wn(2,:) + ((CFL-1)/(CFL+1))* (wnp1(2,:)-wn (1,:));
  wnp1(end,:)=wn(end-1,:)+ ((CFL-1)/(CFL+1))*(wnp1(end-1,:)-wn(end,:));
  wnp1(:,1)=wn(:,2)+ ((CFL-1)/(CFL+1))*(wnp1(:,2)-wn (:,1));
  wnp1(:, end)=wn(:, end-1) + ((CFL-1)/(CFL+1))*(wnp1(:, end-1)-wn(:, end));
```

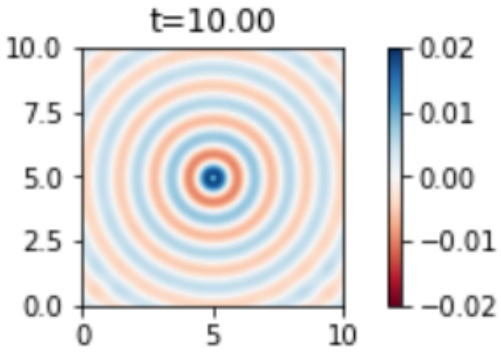
```
  %Solution
  t=t+dt;
  wnm1=wn; wn=wnp1; %Saving current and previous arrays

  %Source
  wn(50,50)=dt^2*20*sin(30*pi*t/20);

  for i=2:nx-1, for j=2:ny-1
          wnp1(i,j)= 2*wn(i,j)-wnm1(i,j)...
              +CFL^2*(wn(i+1,j)+wn(i,j+1)-4*wn(i,j)+wn(i-1,j)+wn(i,j-1));
      end,end

  %Check convergence
  %Visualize at selected steps
  clf;
  subplot(2,1,1);
  imagesc(x,y,wn'); colorbar; caxis([-0.02,0.02])
  title(sprintf('t=%.2f' , t));
  subplot(2,1,2);
  mesh(x,y,wn'); colorbar; caxis([-0.02 0.02])
  axis([0 Lx 0 Ly -0.05 0.05]);
  shg; pause(0.01);
end
```

2D ABSORB PYTHON CODE:

```python
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

#Solving a PDE
# Equation
# wtt = c^2 wxx + c^2 wyy + f

# Domain
# Space
Lx = 10
Ly = 10
dx = 0.1
dy = dx
nx = int(Lx/dx)
ny = int(Ly/dy)
x = np.linspace(0, Lx, nx)
y = np.linspace(0, Ly, ny)

# Time
T = 10

# Field variables
# Variables
wn = np.zeros((nx, ny))
wnm1 = wn  # w at time n-1
wnp1 = wn  # w at time n+1

# Parameters
CFL = 0.5
c = 1
dt = CFL*dx/c

# Initial Conditions

# Time Stepping loop
t = 0

while(t < T):
    # Reflecting boundary conditions
#     wn[:, [0, -1]] = 0
#     wn[[0, -1], :] = 0

    # Absorbing boundary conditions
    wnp1[0, :] = wn[1, :] + ((CFL-1)/(CFL+1))* (wnp1[1, :]-wn[0, :])
    wnp1[-1, :] = wn[-2, :] + ((CFL-1)/(CFL+1))*(wnp1[-2, :]-wn[-1, :])
    wnp1[:, 0] = wn[:, 1] + ((CFL-1)/(CFL+1))*(wnp1[:, 1]-wn[:, 0])
    wnp1[:, -1] = wn[:, -2] + ((CFL-1)/(CFL+1))*(wnp1[:, -2]-wn[:, -1])

    # Solution
    t = t+dt
    wnm1 = wn.copy()
    wn = wnp1.copy()  # Saving current and previous arrays

    # Source
    wn[50, 50] = dt**2*20*np.sin(30*np.pi*t/20)

    for i in range(1, nx-1):
        for j in range(1, ny-1):
            wnp1[i, j] = 2*wn[i, j]-wnm1[i, j] + CFL**2 * \
                (wn[i+1, j]+wn[i, j+1]-4*wn[i, j]+wn[i-1, j]+wn[i, j-1])

# Check convergence
# Visualize at selected steps
    plt.subplot(2, 1, 1)
    im = plt.imshow(wn.T, extent=[0, Lx, 0, Ly], cmap='RdBu', vmin=-0.02, vmax=0.02)
    plt.title('t={:.2f}'.format(t))
    plt.clim([-0.02, 0.02])
    plt.colorbar(im)


    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    X, Y = np.meshgrid(x, y)
    ax.plot_surface(X, Y, wn.T, cmap='RdBu', vmin=-0.02, vmax=0.02)
    ax.set_xlabel('x')
    ax.set_ylabel('y')
    ax.set_zlabel('w')
    ax.set_xlim3d(0, Lx)
    ax.set_ylim3d(0, Ly)
    ax.set_zlim3d(-0.05, 0.05)
    plt.show()
    plt.pause(0.05)
```

RESULTS OF 2D ABSORB





2) 2D REFLECT

2D REFLECT MATLAB CODE

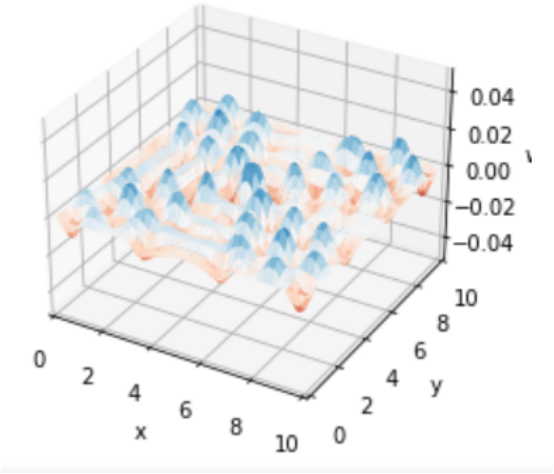
```
%%Solving a PDE
clear;
%Equation
% wtt= c^2 wxx + c^2 wyy + f

%%Domain
%space
Lx=10;
Ly=10;
dx=0.1;
dy=dx;
nx=fix(Lx/dx);
ny=fix(Ly/dy);
x=linspace(0,Lx,nx);
y=linspace(0,Ly,ny);

%Time
T=10;

%%Field variables
%Variables
wn=zeros(nx,ny);
wnm1=wn; % w at time n-1
wnp1=wn; % w at time n+1

%Parameters
CFL=0.5;
c=1;
dt=CFL*dx/c;

%%Initial Conditions
%% Time Stepping loop
t=0;

while(t<T)
  %Reflecting boundary conditions
  wn(:,[1 end])=0;
  wn([1 end],:)=0;

  %Absorbing boundary conditions
  %wnp1(1,:)=wn(2,:) + ((CFL-1)/(CFL+1))* (wnp1(2,:)-wn (1,:));
  %wnp1(end,:)=wn(end-1,:)+ ((CFL-1)/(CFL+1))*(wnp1(end-1,:)-wn(end,:));
  %wnp1(:,1)=wn(:,2)+ ((CFL-1)/(CFL+1))*(wnp1(:,2)-wn (:,1));
  %wnp1(:, end)=wn(:, end-1) + ((CFL-1)/(CFL+1))*(wnp1(:, end-1)-wn(:, end));
```

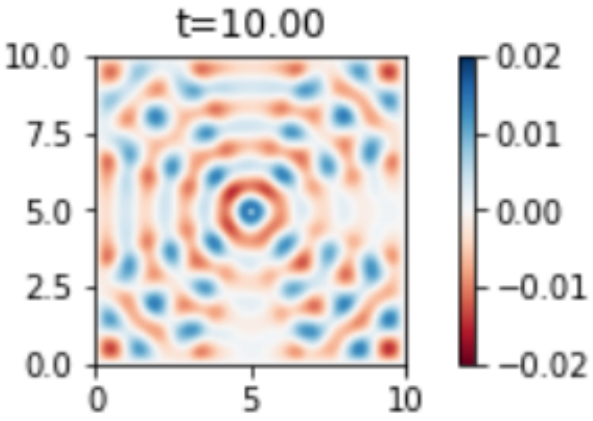
```
  %Solution
  t=t+dt;
  wnm1=wn; wn=wnp1; %Saving current and previous arrays

  %Source
  wn(50,50)=dt^2*20*sin(30*pi*t/20);

  for i=2:nx-1, for j=2:ny-1
          wnp1(i,j)= 2*wn(i,j)-wnm1(i,j)...
              +CFL^2*(wn(i+1,j)+wn(i,j+1)-4*wn(i,j)+wn(i-1,j)+wn(i,j-1));
      end,end

  %Check convergence
  %Visualize at selected steps
  clf;
  subplot(2,1,1);
  imagesc(x,y,wn'); colorbar; caxis([-0.02,0.02])
  title(sprintf('t=%.2f' , t));
  subplot(2,1,2);
  mesh(x,y,wn'); colorbar; caxis([-0.02 0.02])
  axis([0 Lx 0 Ly -0.05 0.05]);
  shg; pause(0.01);
end
```

2D REFLECT PYTHON CODE

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

#Solving a PDE
# Equation
# wtt = c^2 wxx + c^2 wyy + f

# Domain
# Space
Lx = 10
Ly = 10
dx = 0.1
dy = dx
nx = int(Lx/dx)
ny = int(Ly/dy)
x = np.linspace(0, Lx, nx)
y = np.linspace(0, Ly, ny)

# Time
T = 10

# Field variables
# Variables
wn = np.zeros((nx, ny))
wnm1 = wn  # w at time n-1
wnp1 = wn  # w at time n+1

# Parameters
CFL = 0.5
c = 1
dt = CFL*dx/c

# Initial Conditions

# Time Stepping loop
t = 0

while(t < T):
    # Reflecting boundary conditions
    wn[:, [0, -1]] = 0
    wn[[0, -1], :] = 0

    # Absorbing boundary conditions
    # wnp1[0, :] = wn[1, :] + ((CFL-1)/(CFL+1))* (wnp1[1, :]-wn[0, :])
    # wnp1[-1, :] = wn[-2, :] + ((CFL-1)/(CFL+1))*(wnp1[-2, :]-wn[-1, :])
    # wnp1[:, 0] = wn[:, 1] + ((CFL-1)/(CFL+1))*(wnp1[:, 1]-wn[:, 0])
    # wnp1[:, -1] = wn[:, -2] + ((CFL-1)/(CFL+1))*(wnp1[:, -2]-wn[:, -1])

    # Solution
    t = t+dt
    wnm1 = wn.copy()
    wn = wnp1.copy()  # Saving current and previous arrays

    # Source
    wn[50, 50] = dt**2*20*np.sin(30*np.pi*t/20)

    for i in range(1, nx-1):
        for j in range(1, ny-1):
            wnp1[i, j] = 2*wn[i, j]-wnm1[i, j] + CFL**2 * \
                (wn[i+1, j]+wn[i, j+1]-4*wn[i, j]+wn[i-1, j]+wn[i, j-1])

# Check convergence
# Visualize at selected steps
    plt.subplot(2, 1, 1)
    im = plt.imshow(wn.T, extent=[0, Lx, 0, Ly], cmap='RdBu', vmin=-0.02, vmax=0.02)
    plt.title('t={:.2f}'.format(t))
    plt.clim([-0.02, 0.02])
    plt.colorbar(im)


    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    X, Y = np.meshgrid(x, y)
    ax.plot_surface(X, Y, wn.T, cmap='RdBu', vmin=-0.02, vmax=0.02)
    ax.set_xlabel('x')
    ax.set_ylabel('y')
    ax.set_zlabel('w')
    ax.set_xlim3d(0, Lx)
    ax.set_ylim3d(0, Ly)
    ax.set_zlim3d(-0.05, 0.05)
    plt.show()
    plt.pause(0.05)
```

RESULTS OF 2D REFLECT





3) 3D ABSORB

3D ABSORB MATLAB CODE

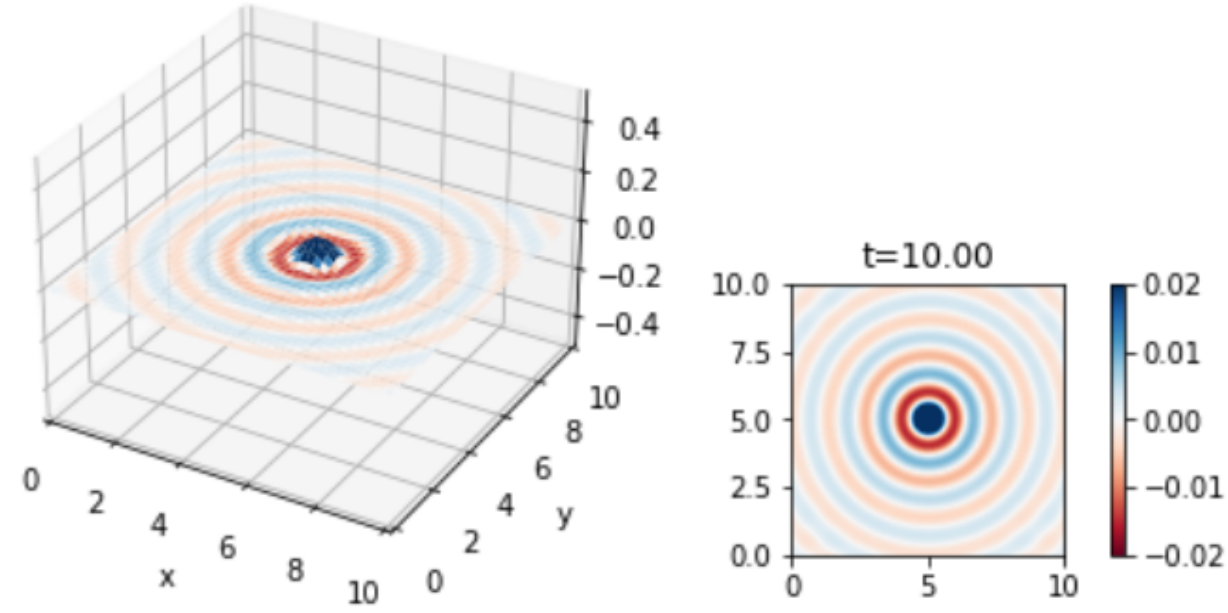
```
%%Solving a PDE
clear;
%Equation
% wtt= c^2 wxx + c^2 wyy + c^2 wzz + f

%%Domain
%space
Lx=10;
Ly=10;
Lz=10;
dx=0.1;
dy=dx;
dz=dx;
nx=fix(Lx/dx);
ny=fix(Ly/dy);
nz=fix(Lz/dz);
x=linspace(0,Lx,nx);
y=linspace(0,Ly,ny);
z=linspace(0,Lz,nz);

%Time
T=10;

%%Field variables
%Variables
wn=zeros(nx,ny,nz);
wnm1=wn; % w at time n-1
wnp1=wn; % w at time n+1

%Parameters
CFL=0.5;
c=1;
dt=CFL*dx/c;

%%Initial Conditions
%% Time Stepping loop
t=0;

while(t<T)
  %Reflecting boundary conditions
  %wn(:,[1 end],:)=0;
  %wn([1 end],:,:)=0;
  %wn(:,:,[1 end])=0;

  %Absorbing boundary conditions
  wnp1(1,:,:)=wn(2,:,:) + ((CFL-1)/(CFL+1))* (wnp1(2,:,:)-wn (1,:,:));
  wnp1(end,:,:)=wn(end-1,:,:)+ ((CFL-1)/(CFL+1))*(wnp1(end-1,:,:)-wn(end,:,:));
  wnp1(:,1,:)=wn(:,2,:)+ ((CFL-1)/(CFL+1))*(wnp1(:,2,:)-wn (:,1,:));
  wnp1(:,end,:)=wn(:, end-1,:) + ((CFL-1)/(CFL+1))*(wnp1(:, end-1,:)-wn(:, end,:));
  wnp1(:,:,1)=wn(:,:,2)+ ((CFL-1)/(CFL+1))*(wnp1(:,:,2)-wn (:,:,1));
  wnp1(:,:,end)=wn(:,:,end-1) + ((CFL-1)/(CFL+1))*(wnp1(:,:,end-1)-wn(:,:,end));
```

```
  %Solution
  t=t+dt;
  wnm1=wn; wn=wnp1; %Saving current and previous arrays

  %Source
  wn(50,50,50)=dt^2*200*sin(30*pi*t/20);

  for i=2:nx-1, for j=2:ny-1, for k=2:nz-1
          wnp1(i,j,k)= 2*wn(i,j,k)-wnm1(i,j,k)...
              +CFL^2*(wn(i+1,j,k)+wn(i,j+1,k)+wn(i,j,k+1)-6*wn(i,j,k)+wn(i-1,j,k)+wn(i,j-1,k)+wn(i,j,k-
1));
       end, end, end

  %Check convergence
  %Visualize at selected steps
  clf;
  subplot(2,1,1);
  imagesc(x,y,wn(:,:,nz/2)'); colorbar; caxis([-0.02,0.02])
  title(sprintf('t=%.2f' , t));
  subplot(2,1,2);
  mesh(x,y,wn(:,:,nz/2)'); colorbar; caxis([-0.02 0.02])
  axis([0 Lx 0 Ly -0.05 0.05]);
  shg; pause(0.01);
end
```

3D ABSORB PYTHON CODE

```
import numpy as np
import matplotlib.pyplot as plt

# Equation
# wtt = c^2 wxx + c^2 wyy + c^2 wzz + f

# Domain
# space
Lx = 10
Ly = 10
Lz = 10
dx = 0.1
dy = dx
dz = dx
nx = int(Lx/dx)
ny = int(Ly/dy)
nz = int(Lz/dz)
x = np.linspace(0, Lx, nx)
y = np.linspace(0, Ly, ny)
z = np.linspace(0, Lz, nz)

# time
T = 10

# field variables
# variables
wn = np.zeros((nx, ny, nz))
wnm1 = wn  # w at time n-1
wnp1 = wn  # w at time n+1

# parameters
CFL = 0.5
c = 1
dt = CFL*dx/c

# initial conditions

# time stepping loop
t = 0

while t < T:
    # Reflecting boundary conditions
#     wn[:,[0,-1],:] = 0
#     wn[[0,-1],:,:] = 0
#     wn[:,:, [0,-1]] = 0

     # Absorbing boundary conditions
    wnp1[0,:,:] = wn[1,:,:] + ((CFL - 1)/(CFL + 1)) * (wnp1[1,:,:] - wn[0,:,:])
    wnp1[-1,:,:] = wn[-2,:,:] + ((CFL - 1)/(CFL + 1)) * (wnp1[-2,:,:] - wn[-1,:,:])
    wnp1[:,0,:] = wn[:,1,:] + ((CFL - 1)/(CFL + 1)) * (wnp1[:,1,:] - wn[:,0,:])
    wnp1[:,-1,:] = wn[:,-2,:] + ((CFL - 1)/(CFL + 1)) * (wnp1[:,-2,:] - wn[:,-1,:])
    wnp1[:,:,0] = wn[:,:,1] + ((CFL - 1)/(CFL + 1)) * (wnp1[:,:,1] - wn[:,:,0])
    wnp1[:,:,-1] = wn[:,:,-2] + ((CFL - 1)/(CFL + 1)) * (wnp1[:,:,-2] - wn[:,:,-1])

    # solution
    t = t + dt
    wnm1 = wn.copy()
    wn = wnp1.copy()  # saving current and previous arrays

    # source
    wn[50,50,50] = dt**2 * 200 * np.sin(30*np.pi*t/20)

    for i in range(1, nx-1):
        for j in range(1, ny-1):
            for k in range(1, nz-1):
                wnp1[i,j,k] = 2*wn[i,j,k] - wnm1[i,j,k] \
                    + CFL**2 * (wn[i+1,j,k] + wn[i,j+1,k] + wn[i,j,k+1] - 6*wn[i,j,k] + wn[i-1,j,k] +
wn[i,j-1,k] + wn[i,j,k-1])

    # check convergence
    # visualize at selected steps
    plt.clf()
    plt.subplot(2,1,1)
    im=plt.imshow(wn[:,:,nz//2].T, origin='lower', extent=[0,Lx,0,Ly],cmap='RdBu', vmin=-0.02,
vmax=0.02)
    plt.colorbar(im)
    plt.clim([-0.02,0.02])
    plt.title('t={:.2f}'.format(t))

    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    X, Y = np.meshgrid(x, y)
    ax.plot_surface(X, Y, wn[:,:,nz//2].T, cmap='RdBu', vmin=-0.02, vmax=0.02)
    ax.set_xlabel('x')
    ax.set_ylabel('y')
    ax.set_zlabel('z')
    ax.set_xlim3d(0, Lx)
    ax.set_ylim3d(0, Ly)
    ax.set_zlim3d(-0.5, 0.5)
    plt.show()
    plt.pause(0.05)
```

RESULTS OF 3D ABSORB



4) 3D REFLECT

3D REFLECT MATLAB CODE

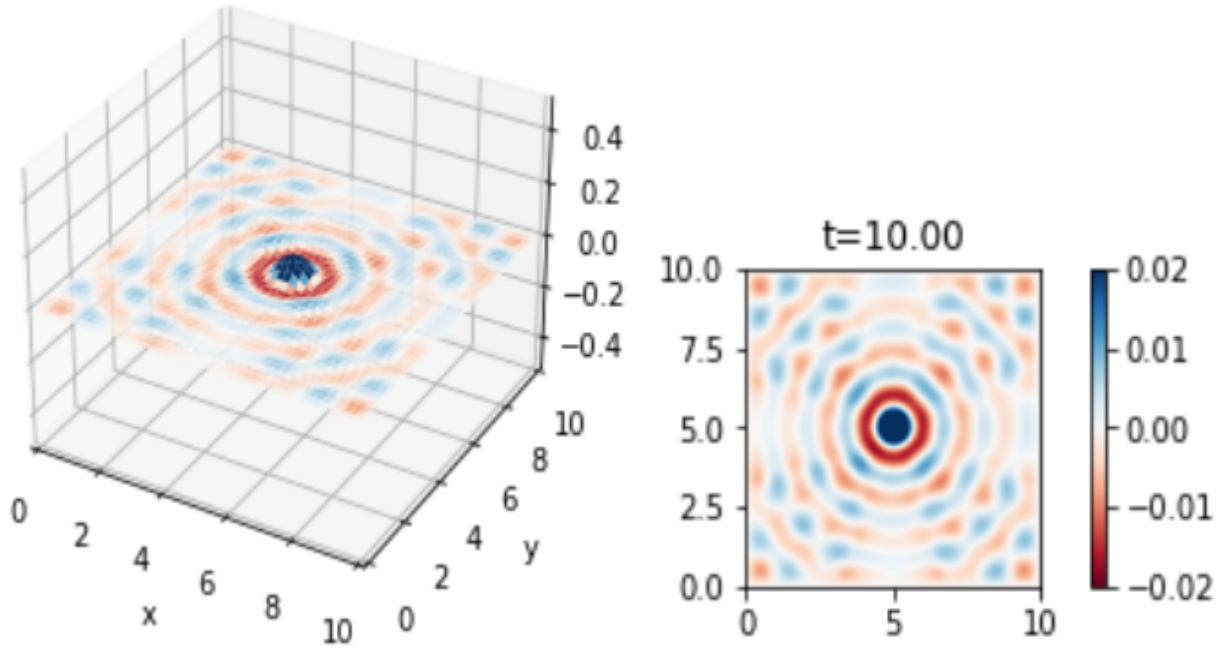
```
%%Solving a PDE
clear;
%Equation
% wtt= c^2 wxx + c^2 wyy + c^2 wzz + f

%%Domain
%space
Lx=10;
Ly=10;
Lz=10;
dx=0.1;
dy=dx;
dz=dx;
nx=fix(Lx/dx);
ny=fix(Ly/dy);
nz=fix(Lz/dz);
x=linspace(0,Lx,nx);
y=linspace(0,Ly,ny);
z=linspace(0,Lz,nz);

%Time
T=10;

%%Field variables
%Variables
wn=zeros(nx,ny,nz);
wnm1=wn; % w at time n-1
wnp1=wn; % w at time n+1

%Parameters
CFL=0.5;
c=1;
dt=CFL*dx/c;

%%Initial Conditions
%% Time Stepping loop
t=0;

while(t<T)
  %Reflecting boundary conditions
  wn(:,[1 end],:)=0;
  wn([1 end],:,:)=0;
  wn(:,:,[1 end])=0;

  %Absorbing boundary conditions
  %wnp1(1,:,:)=wn(2,:,:) + ((CFL-1)/(CFL+1))* (wnp1(2,:,:)-wn (1,:,:));
  %wnp1(end,:,:)=wn(end-1,:,:)+ ((CFL-1)/(CFL+1))*(wnp1(end-1,:,:)-wn(end,:,:));
  %wnp1(:,1,:)=wn(:,2,:)+ ((CFL-1)/(CFL+1))*(wnp1(:,2,:)-wn (:,1,:));
  %wnp1(:,end,:)=wn(:, end-1,:) + ((CFL-1)/(CFL+1))*(wnp1(:, end-1,:)-wn(:, end,:));
  %wnp1(:,:,1)=wn(:,:,2)+ ((CFL-1)/(CFL+1))*(wnp1(:,:,2)-wn (:,:,1));
  %wnp1(:,:,end)=wn(:,:,end-1) + ((CFL-1)/(CFL+1))*(wnp1(:,:,end-1)-wn(:,:,end));
```

```
  %Solution
  t=t+dt;
  wnm1=wn; wn=wnp1; %Saving current and previous arrays

  %Source
  wn(50,50,50)=dt^2*200*sin(30*pi*t/20);

  for i=2:nx-1, for j=2:ny-1, for k=2:nz-1
          wnp1(i,j,k)= 2*wn(i,j,k)-wnm1(i,j,k)...
              +CFL^2*(wn(i+1,j,k)+wn(i,j+1,k)+wn(i,j,k+1)-6*wn(i,j,k)+wn(i-1,j,k)+wn(i,j-1,k)+wn(i,j,k-
1));
       end, end, end

  %Check convergence
  %Visualize at selected steps
  clf;
  subplot(2,1,1);
  imagesc(x,y,wn(:,:,nz/2)'); colorbar; caxis([-0.02,0.02])
  title(sprintf('t=%.2f' , t));
  subplot(2,1,2);
  mesh(x,y,wn(:,:,nz/2)'); colorbar; caxis([-0.02 0.02])
  axis([0 Lx 0 Ly 0 Lz]);
  shg; pause(0.01);
end
```

3D REFLECT PYTHON CODE

```
import numpy as np
import matplotlib.pyplot as plt

# Equation
# wtt = c^2 wxx + c^2 wyy + c^2 wzz + f

# Domain
# space
Lx = 10
Ly = 10
Lz = 10
dx = 0.1
dy = dx
dz = dx
nx = int(Lx/dx)
ny = int(Ly/dy)
nz = int(Lz/dz)
x = np.linspace(0, Lx, nx)
y = np.linspace(0, Ly, ny)
z = np.linspace(0, Lz, nz)

# time
T = 10

# field variables
# variables
wn = np.zeros((nx, ny, nz))
wnm1 = wn  # w at time n-1
wnp1 = wn  # w at time n+1

# parameters
CFL = 0.5
c = 1
dt = CFL*dx/c

# initial conditions

# time stepping loop
t = 0

while t < T:
    # Reflecting boundary conditions
    wn[:,[0,-1],:] = 0
    wn[[0,-1],:,:] = 0
    wn[:,:, [0,-1]] = 0

     # Absorbing boundary conditions
#     wnp1[0,:,:] = wn[1,:,:] + ((CFL - 1)/(CFL + 1)) * (wnp1[1,:,:] - wn[0,:,:])
#     wnp1[-1,:,:] = wn[-2,:,:] + ((CFL - 1)/(CFL + 1)) * (wnp1[-2,:,:] - wn[-1,:,:])
#     wnp1[:,0,:] = wn[:,1,:] + ((CFL - 1)/(CFL + 1)) * (wnp1[:,1,:] - wn[:,0,:])
#     wnp1[:,-1,:] = wn[:,-2,:] + ((CFL - 1)/(CFL + 1)) * (wnp1[:,-2,:] - wn[:,-1,:])
#     wnp1[:,:,0] = wn[:,:,1] + ((CFL - 1)/(CFL + 1)) * (wnp1[:,:,1] - wn[:,:,0])
#     wnp1[:,:,-1] = wn[:,:,-2] + ((CFL - 1)/(CFL + 1)) * (wnp1[:,:,-2] - wn[:,:,-1])

    # solution
    t = t + dt
    wnm1 = wn.copy()
    wn = wnp1.copy()  # saving current and previous arrays

    # source
    wn[50,50,50] = dt**2 * 200 * np.sin(30*np.pi*t/20)

    for i in range(1, nx-1):
        for j in range(1, ny-1):
            for k in range(1, nz-1):
                wnp1[i,j,k] = 2*wn[i,j,k] - wnm1[i,j,k] \
                    + CFL**2 * (wn[i+1,j,k] + wn[i,j+1,k] + wn[i,j,k+1] - 6*wn[i,j,k] + wn[i-1,j,k] +
wn[i,j-1,k] + wn[i,j,k-1])

    # check convergence
    # visualize at selected steps
    plt.clf()
    plt.subplot(2,1,1)
    im=plt.imshow(wn[:,:,nz//2].T, origin='lower', extent=[0,Lx,0,Ly],cmap='RdBu', vmin=-0.02,
vmax=0.02)
    plt.colorbar(im)
    plt.clim([-0.02,0.02])
    plt.title('t={:.2f}'.format(t))

    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    X, Y = np.meshgrid(x, y)
    ax.plot_surface(X, Y, wn[:,:,nz//2].T, cmap='RdBu', vmin=-0.02, vmax=0.02)
    ax.set_xlabel('x')
    ax.set_ylabel('y')
    ax.set_zlabel('z')
    ax.set_xlim3d(0, Lx)
    ax.set_ylim3d(0, Ly)
    ax.set_zlim3d(-0.5, 0.5)
    plt.show()
    plt.pause(0.05)
```

RESULTS OF 3D REFLECT



CONCLUSION

In conclusion, this paper presents a highly efficient implicit finite difference scheme for acoustic wave propagation that offers significant advantages over other numerical schemes. The authors have demonstrated the effectiveness of this method through extensive testing and comparison with other central difference schemes. By reducing computational resources and minimizing errors, this method has the potential to revolutionize numerical simulations in a variety of fields, from oil and gas exploration to medical imaging.

The fact that this approach needs fewer points than central difference schemes does, which lowers the computational cost and adds less phase error, is one of its main advantages. Additionally, the 6th order implicit scheme has computational costs that are almost identical to those of the 10th order explicit scheme and can be easily expanded from 1D to higher dimensions (2D/3D). Because of this, it is a desirable alternative for modelling wave propagation in complicated medium.

The authors have also contrasted the numerical scheme and derivative operator's spectrum properties with a number of other central difference schemes. They have demonstrated that their approach performs better than these other plans in terms of accuracy and effectiveness. Additionally, they have successfully used an implicit FD scheme that is sixth-order accurate to simulate wave propagation in both 2D and 3D acoustic media.

This method has significant potential for practical applications in fields such as oil and gas exploration, medical imaging, and seismic workfl ows. By improving the accuracy and efficiency of numerical simulations in these fields, it can help researchers better understand complex phenomena and make more informed decisions. For example, it could be used to optimize drilling operations or improve medical diagnoses.

Overall, this paper represents an important contribution to the field of numerical simulation and offers exciting possibilities for future research and development. The authors' innovative approach has opened up new avenues for exploring complex wave propagation phenomena using numerical methods. As such, it is likely to inspire further research in this area for years to come.

REFERENCES

1. Ajay Malkoti, Nimisha Vedanti, Ram Krishan Tiwari. "A highly efficient implicit finite difference scheme for acoustic wave propagation", Journal of Applied Geophysics, 2018
2. dspace.univ-ouargla.dz
3. coek.info
4. Repository.library.csuci.edu
5. https://www.youtube.com/watch?v=fD_C90c9yR4&ab_channel=cfurse
6. https://www.youtube.com/watch?v=b0TtxBz4Qag&ab_channel=NPTEL-NOCIITM
7. https://www.youtube.com/watch?v=kKzRsorjHoA&list=PLyqSpQzTE6M-_vDPKKboGoAmhqlXuaKIl&index=53&ab_channel=NPTEL-NOCIITM