Ultrasound in Medicine - MPHYG900
Coursework 2

Student Number - 14062340

November 2017

2

The Taylor series expansions for $f(x+\Delta x)$ and $f(x-\Delta x)$ are shown below (up to and including second order terms):

$$f(x+\Delta x) = f(x) + \Delta x \frac{\partial f(x)}{\partial x} + \frac{\Delta x^2}{2}\frac{\partial^2 f(x)}{\partial x^2} + ... \tag{1}$$

$$f(x-\Delta x) = f(x) - \Delta x \frac{\partial f(x)}{\partial x} + \frac{\Delta x^2}{2}\frac{\partial^2 f(x)}{\partial x^2} - ... \tag{2}$$

Adding equations 1 and 2 together and rearranging the result gives:

$$\frac{\partial^2 f(x)}{\partial x^2} = \frac{f(x-\Delta x) - 2f(x) + f(x+\Delta x)}{\Delta x^2} + O(\Delta x^2) \tag{3}$$

Where $O(\Delta x^2)$ is the truncation error associated with a second order accurate expression in x.

Next, the solution to the wave equation in 3D is:

$$\frac{\partial^2 p}{\partial t^2} = {c_0}^2 \nabla^2 p \tag{4}$$

Ignoring the z component of the Laplace operator on the right hand side for a 2D solution:

$$\frac{\partial^2 p}{\partial t^2} = {c_0}^2 \left(\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2}\right) \tag{5}$$

Using the notation $p^n_{i,j}$ (which is the pressure at time index n and spatial indices i and j), allows the time and spatial derivatives in equation 5 to be replaced with the finite difference relationship in equation 3 (omitting truncation error):

$$\frac{p^{n+1}_{i,j} - 2p^n_{i,j} + p^{n-1}_{i,j}}{\Delta t^2} \approx {c_0}^2 \left(\frac{p^n_{i+1,j} - 2p^n_{i,j} + p^n_{i-1,j}}{\Delta x^2} + \frac{p^n_{i,j+1} - 2p^n_{i,j} + p^n_{i,j-1}}{\Delta y^2}\right) \tag{6}$$

Rearranging equation 6 gives:

$$p_{i,j}^{n+1} \approx 2p_{i,j}^{n} - p_{i,j}^{n-1} + \Delta t^2 {c_0}^2 \left(\frac{p_{i+1,j}^{n} - 2p_{i,j}^{n} + p_{i-1,j}^{n}}{\Delta x^2} + \frac{p_{i,j+1}^{n} - 2p_{i,j}^{n} + p_{i,j-1}^{n}}{\Delta y^2} \right) \tag{7}$$

Equation 7 is therefore the updated expression for acoustic pressure in 2D using a second-order accurate central difference scheme.

3

Equation 7 was implemented in a MATLAB script in a vectorised fashion (included in an appendix). The time step was set to the upper bound of the stability limit in 2D for the central difference scheme used in this simulation:

$$\Delta t = \frac{\Delta x}{c_0 \sqrt{2}} \tag{8}$$

Where $\Delta x = \Delta y$. The following actions were implemented to allow the decay in intensity of the pressure wave to be visualised:

- The maximum and minimum pressure values were obtained after the simulation to allow use of appropriate z-axis limits.
- A 3D surface plot was produced.
- The plot had an accompanying colourbar to show how the pressure field in the plot maps to the colourbar.

100 grid points were used in both x and y directions, with a grid spacing of 1mm, and sound speed of 1,500 m/s.

Snapshots of the pressure field after 10, 50 and 100 time steps are shown below.

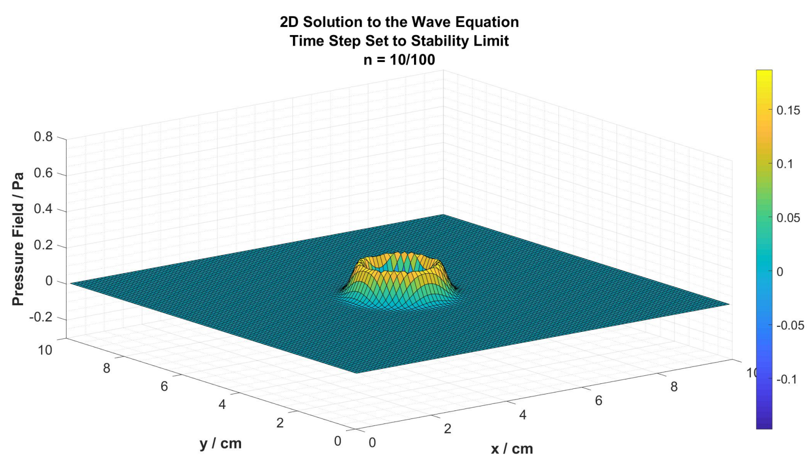

Figure 1: Snapshot of pressure field after 10 time steps.

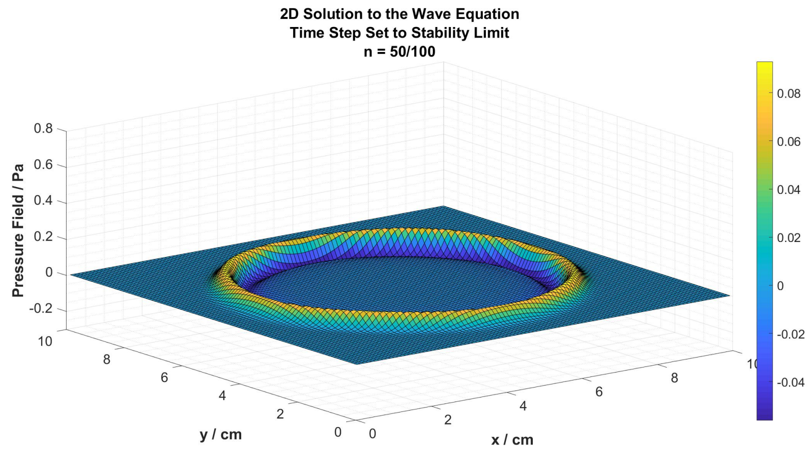

Figure 2: Snapshot of pressure field after 50 time steps.

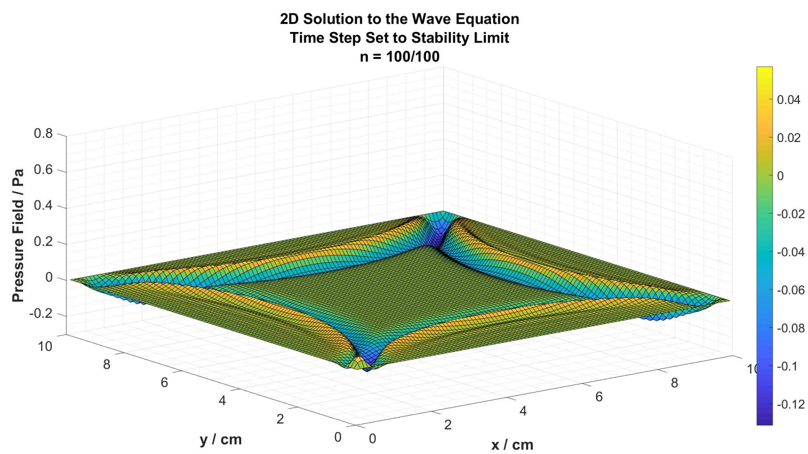

Figure 3: Snapshot of pressure field after 100 time steps.

4

The wave is both completely reflected and inverted when it reaches the boundary. To explain this the following assumption within the implementation of the finite difference scheme needs to be characterised.

The new pressure value at each grid point depends on the previous two values in time at that same location, and the previous four values either side of that location in space (in two dimensions). Therefore the finite difference stencil cannot update edge values, and these values at the very edge of the pressure field are left equal to zero by default. These values are *used*, but they are never *updated*.

This results in what is known as a pressure release boundary condition. The boundary effectively has an extremely low impedance (Z_2), compared to the interior impedance (Z_1) within the pressure field. This effectively gives a reflection coefficient of $R = -1$, and a transmission coefficient of $T = 0$. All of this results in the wave being both completely reflected and inverted when it reaches the boundary (this is equivalent to a pressure wave travelling from water into air).

5

The time step was increased to 2% above the stability limit, by multiplying the right hand side of equation 8 by a factor of 1.02:

$$\Delta t = 1.02 \times \left(\frac{\Delta x}{c_0 \sqrt{2}} \right) \tag{9}$$

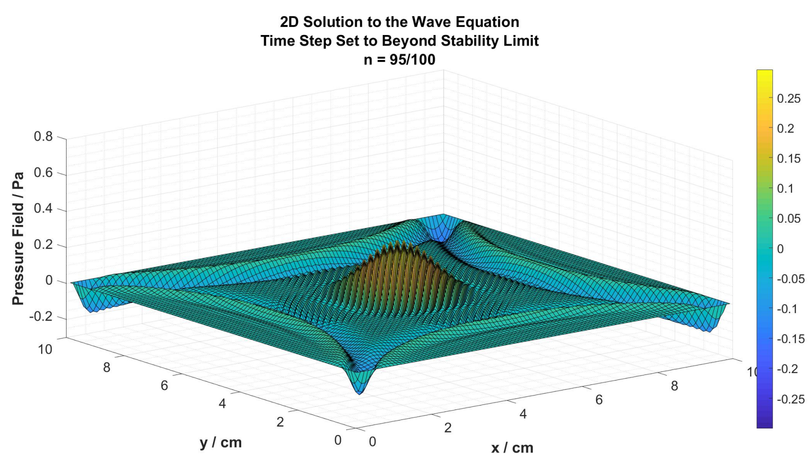

Figure 4: Snapshot of pressure field as instability is beginning.

As can be seen in the above plot, this caused instability to develop in the centre of the propagating pressure field (shown by the central jagged region) as the simulation progressed. This occurred because the time step was set to above the 2D stability limit, and the numerical model became unstable. This means that the errors in the numerical solution grow without bound as the simulation progresses (whereas in a stable numerical model the errors in the numerical solution remain bounded). Eventually the unstable numerical model blows up and produces nonsense values (infinity (Inf) and not a number (NaN) values), with the error continuing to grow and the solution being completely corrupted.

6

Using a time step set to the stability limit (as described in equation 8), the value of the pressure field at grid position (40,40) was recorded for each time step throughout the simulation. The resulting pressure vector is plotted against time below.

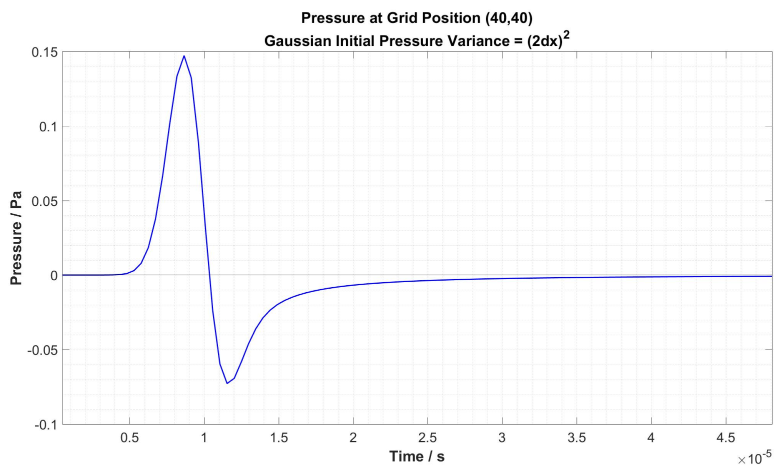

Figure 5: Plot of pressure at grid position (40,40) versus time.

The pressure signal shown above is fairly smooth throughout the whole simulation, with no signs of distortion.

7

The variance of the Gaussian initial pressure was then decreased from $(2dx)^2$ to $(dx)^2$, and, as for question 6, the pressure at grid position (40,40) was recorded for each time step throughout the simulation. The resulting pressure vector is plotted against time overleaf.

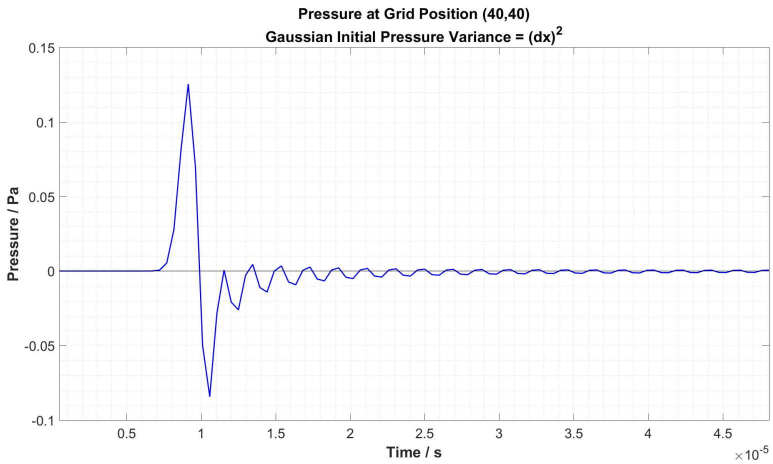

Figure 6: Plot of pressure at grid position (40,40) versus time for reduced variance of the Gaussian initial pressure.

As the simulation progresses, the signal has become distorted due to numerical dispersion. To understand this, the following explanation needs to be characterised. A Gaussian distribution does not contain a single frequency, but rather it is a sum of a range of waves which have a range of frequencies.

In the exact solution to the *finite difference approximation* that this simulation is solving (equation 7), an unwanted numerical error is introduced, and as such, wave velocity depends on frequency in this solution. (This is in contrast to the exact solution to the *2D wave equation*, in which all waves travel at the same speed, regardless of frequency.) Therefore, as a Gaussian wave is composed of a range of frequencies, different components of the wave will travel at different speeds in the solution to this numerical model. The further the wave travels, the more distorted it will become and it will begin to break up. This is known as numerical dispersion.

By decreasing the variance of the initial Gaussian pressure distribution and making it steeper, it will contain more higher frequency components. This will increase the range of velocities which the different components of the wave travel at in this solution. Accordingly, the wave will become more susceptible to numerical dispersion, and this results in a distorted signal (as is seen in Figure 6), especially as the simulation progresses. The effect of this could be reduced by using a higher order finite difference formula to solve the wave equation, or by reducing the size of the grid spacing and time step used in the simulation.

Another factor to consider in interpreting this change is that the recorded signal in Figure 6 has diverged away from its convergent solution. To make sense of this the following needs to be characterised. By decreasing the variance of the initial Gaussian pressure distribution and making it steeper, its wavelength has been decreased, but the grid spacing used for sampling (Δx) has remained constant. This has the result of decreasing the amount of grid points per wavelength, which has the effect of increasing the size of Δx (relative to the wavelength of the pressure distribution).

This decreased spatial sampling resolution causes the numerical solution to diverge away from the result toward which it would converge if the spatial sampling resolution was increased (i.e. if Δx was made smaller).

Making Δx smaller and smaller in an iterative fashion is known as a convergence analysis, as the numerical result will eventually converge on an exact solution and stop changing, even if Δx is made smaller. At this point the right-hand side of equation 7 is a good approximation to the left-hand side of the same equation. This occurs because equation 7 is *consistent* (i.e. it is correct in the limit that Δx, Δy and Δt reduce to zero) and the numerical model used is *stable*. Lax's equivalence theorem states that a numerical scheme will be *convergent* if it is also consistent and stable.

8 Appendix

The fully commented MATLAB script used to run all of the above simulations is shown below.

```
% Simple finite difference solution to the 2D wave equation. Gradients in
% time and space are both calculated using a 2nd-order accurate central
% difference scheme. Adapted from 1D solution provided by Dr Bradley
% Treeby (Ultrasound in Medicine Course - MPHYx900 - UCL).

% clear all variables in the workspace, close all active figures and clear
% command window

clc
close all
clear all

% ----------
% Question 3
% ----------
```

```

% set the literals (hard-coded numbers used in the script) and initial
% conditions

% number of grid points in x and y directions
Nx = 100;
Ny = 100;

% grid spacing (m) in x and y directions
dx = 1e-3;
dy = 1e-3;

% sound speed (m/s)
c0 = 1500;

% number of time steps
Nt = 100;

% create the grid axis
x = (1:Nx)*dx;
y = (1:Ny)*dy;

% set the position of the source
x_pos = (Nx/2)*dx;
y_pos = (Ny/2)*dy;

% set the initial pressure to be a Gaussian
variance = (2*dx)^2;
gaussian_x = exp( -(x - x_pos).^2 / (2 * variance) );
gaussian_y = exp( -(y - y_pos).^2 / (2 * variance) );
p_n = gaussian_x' * gaussian_y;

% set the size of the time step to its maximum within the stability limit,
% where in 2D the stability limit is given by dt <= dx/(sqrt(2)*c0)
dt = dx/(sqrt(2)*c0);

% set pressure at (n - 1) to be equal to pressure at n (this implicitly sets the
% initial particle velocity to be zero)
p_nm1 = p_n;

% preallocate the pressure at (n + 1) (this is updated during the time
```

```
% loop)
p_np1 = zeros(size(p_n));

% open a new figure in a maximised window (to facilitate saving images)
figure('units','normalized','outerposition',[0 0 1 1])

%to initialise vectors containing max and min z values for each time step
%(to facilitate setting appropriate z axis correctly later)
min_vector=zeros(1,Nt);
max_vector=zeros(1,Nt);

%to initialise vector containing value of pressure field at grid position
% (40,40), which is used in Question 6
p_out=zeros(1,Nt);

% To set line widths (for 2D plots) and font size (for all plots)
LW=1.5;
fs=20;

% calculate pressure in a loop through the time steps
for n = 1:Nt

    % -----------
    % CALCULATION
    % -----------

    % calculate the new value for the pressure, leaving edge values as
    % zero

    %let T = time derivative terms of 2D solution
    T = 2*p_n(2:end-1,2:end-1) - p_nm1(2:end-1,2:end-1);
    %let X = x derivative terms of 2D solution (noting Matlab has
    %(row,column) --> (Y,X) notation when addressing a matrix location)
    X = p_n(2:end-1,1:end-2) - 2*p_n(2:end-1,2:end-1) + p_n(2:end-1,3:end);
    %let Y = y derivative terms of 2D solution
    Y = p_n(1:end-2,2:end-1) - 2*p_n(2:end-1,2:end-1)
```

```
        + p_n(3:end,2:end-1);

    %calculate pressure at n+1 in 2D using the above three terms
    p_np1(2:end-1,2:end-1) = T + ( c0^2 * dt^2 * (X/dx^2 + Y/dy^2) );

    %copy the value of p at n to p at (n - 1)
    p_nm1 = p_n;

    % copy the pressure at (n + 1) to the pressure at n
    p_n = p_np1;

    % --------
    % PLOTTING
    % --------

    % plot the pressure field (in units of cm and pascals)
    s=surf(x*100,y*100,p_np1);

    % set the limits on the z-axis (derived from max_vector and min_vector
    % at end of loop)
    set(gca, 'ZLim', [-0.3, 0.8]);

    % add a title, subtitle, label axes, and colorbar
    title({'2D Solution to the Wave Equation'; ...
        'Time Step Set to Stability Limit';['n = ' ...
        ,num2str(n),'/',num2str(Nt)]},'FontSize',fs+1,'FontWeight','bold')
    xlabel('x / cm','FontWeight','bold','FontSize',fs-1)
    ylabel('y / cm','FontWeight','bold','FontSize',fs-1)
    zlabel('Pressure Field / Pa','FontWeight','bold','FontSize',fs-1)
    colorbar

    % formatting gridlines and axes label fontsizes
    grid minor
    ax = gca;
    ax.FontSize = fs-3;

    % force the plot to update
```

```
    drawnow;

    % briefly pause before continuing the loop
    pause(0.1)

    % to update max and min pressure values for this time step to overall
    % max_vector and min_vector
    max_vector(n)=max(max(p_np1));
    min_vector(n)=min(min(p_np1));

    %to save plots at time steps 10, 50, 100
    if n == 10
        saveas(figure(1),'Plot_Time_Step_10','jpg');
    elseif n==50
        saveas(figure(1),'Plot_Time_Step_50','jpg');
    elseif n ==100
        saveas(figure(1),'Plot_Time_Step_100','jpg');
    end

    %to save the value of the pressure field at (40,40) in vector p_out
    p_out(n)=p_np1(40,40);

end

%to compute the overall max and min z values for all time steps, in order
%to allow for suitable limits to be implemented on z-axis
zmax=max(max_vector); % =0.7650 --> round to 0.8
zmin=min(min_vector); % =-0.2940 --> round to -0.3

% ----------
% Question 5
% ----------

%Increase the time step to 2% above the stability limit
dt = 1.02 * (dx/(sqrt(2)*c0));

%Reset pressure matrix values and open a new figure
p_n = gaussian_x' * gaussian_y;
p_nm1 = p_n;
p_np1 = zeros(size(p_n));
figure('units','normalized','outerposition',[0 0 1 1])
```

```

%Repeat above calculations and plotting from Question 3 with increased time
%step value

for n = 1:Nt

    % -----------
    % CALCULATION
    % -----------

    %forming terms for the expressions in the 2D solution
    T = 2*p_n(2:end-1,2:end-1) - p_nm1(2:end-1,2:end-1);
    X = p_n(2:end-1,1:end-2) - 2*p_n(2:end-1,2:end-1) + p_n(2:end-1,3:end);
    Y = p_n(1:end-2,2:end-1) - 2*p_n(2:end-1,2:end-1) + p_n(3:end,2:end-1);

    %calculate pressure at n+1 in 2D using the above three terms
    p_np1(2:end-1,2:end-1) = T + ( c0^2 * dt^2 * (X/dx^2 + Y/dy^2) );

    %update the pressure matrix values
    p_nm1 = p_n;
    p_n = p_np1;

    % -----------
    % PLOTTING
    % -----------

    % plot the pressure field (in units of cm and pascals)
    surf(x*100,y*100,p_np1);

    % set the limits on the z-axis to the same as Question 3
    set(gca, 'ZLim', [-0.3, 0.8]);

    % add a title, subtitle, label axes, and colorbar
    title({'2D Solution to the Wave Equation'; ...
        'Time Step Set to Beyond Stability Limit';['n = ' ...
        ,num2str(n),'/',num2str(Nt)]},'FontSize',fs+1,
```

```
            'FontWeight','bold')
    xlabel('x / cm','FontWeight','bold','FontSize',fs-1)
    ylabel('y / cm','FontWeight','bold','FontSize',fs-1)
    zlabel('Pressure Field / Pa','FontWeight','bold','FontSize',fs-1)
    colorbar

    % formatting gridlines and axes label fontsizes
    grid minor
    ax = gca;
    ax.FontSize = fs-3;

    % force the plot to update
    drawnow;

    % briefly pause before continuing the loop
    pause(0.1)

    %to save plot when instability begins (at roughly time step 95 by
    %inspection)
    if n == 95
        saveas(figure(2),'Plot_Time_Step_95_Unstable','jpg');
    end

end

% ----------
% Question 6
% ----------

%To plot vector p_out against time (p_out = pressure at grid position
%(40,40))
time = dt*(1:Nt); %first form a time vector in seconds
figure('units','normalized','outerposition',[0 0 1 1])
plot(time,p_out,'color','b','linewidth',LW)
title({'Pressure at Grid Position (40,40)'; ...
        'Gaussian Initial Pressure Variance = (2dx)^{2}'}, ...
        'FontSize',fs+1,'FontWeight','bold')
xlabel('Time / s','FontWeight','bold','FontSize',fs-1)
ylabel('Pressure / Pa','FontWeight','bold','FontSize',
```

```
    fs-1)

%plot formatting
grid minor
ax = gca;
ax.FontSize = fs-3;
xlim([min(time) max(time)]) %tight x-axis fit
line(xlim,[0 0],'color','k');  % plot x-axis

%to save plot
saveas(figure(3),'Pressure_Field_40_40_Original_Variance','jpg');

% ----------
% Question 7
% ----------

%Repeat analysis with reduced variance of Gaussian initial pressure
variance = dx^2;
gaussian_x = exp( -(x - x_pos).^2 / (2 * variance) );
gaussian_y = exp( -(y - y_pos).^2 / (2 * variance) );
%Reset time step value to stability limit
dt = (dx/(sqrt(2)*c0));
%reset pressure matrix values
p_n = gaussian_x' * gaussian_y;
p_nm1 = p_n;
p_np1 = zeros(size(p_n));

%to initialise a vector containing value of pressure field at grid position
% (40,40) for reduced variance
p_out_reduced_var=zeros(1,Nt);

%Rerun the simulation
for n = 1:Nt

    % -----------
    % CALCULATION
    % -----------

    %forming terms for the expressions in the 2D solution
    T = 2*p_n(2:end-1,2:end-1) - p_nm1(2:end-1,2:end-1);
    X = p_n(2:end-1,1:end-2) - 2*p_n(2:end-1,2:end-1)
```

```
        + p_n(2:end-1,3:end);
    Y = p_n(1:end-2,2:end-1) - 2*p_n(2:end-1,2:end-1) + p_n(3:end,2:end-1);

    %calculate pressure at n+1 in 2D using the above three terms
    p_np1(2:end-1,2:end-1) = T + ( c0^2 * dt^2 * (X/dx^2 + Y/dy^2) );

    %update the pressure matrix values
    p_nm1 = p_n;
    p_n = p_np1;

    %to save the value of the pressure field at in vector p_out_reduced_var
    p_out_reduced_var(n)=p_np1(40,40);

end

%To plot p_out_reduced_var against time
figure('units','normalized','outerposition',[0 0 1 1])
plot(time,p_out_reduced_var,'color','b','linewidth',LW)
title({'Pressure at Grid Position (40,40)'; ...
        'Gaussian Initial Pressure Variance = (dx)^{2}'},'FontSize',fs+1 ...
        ,'FontWeight','bold')
xlabel('Time / s','FontWeight','bold','FontSize',fs-1)
ylabel('Pressure / Pa','FontWeight','bold','FontSize',fs-1)
%plot formatting
grid minor
ax = gca;
ax.FontSize = fs-3;
xlim([min(time) max(time)]) %tight x-axis fit
line(xlim,[0 0],'color','k');  % plot x-axis
%to save plot
saveas(figure(4),'Pressure_Field_40_40_Reduced_Variance','jpg');

%close all open figures
close all
```