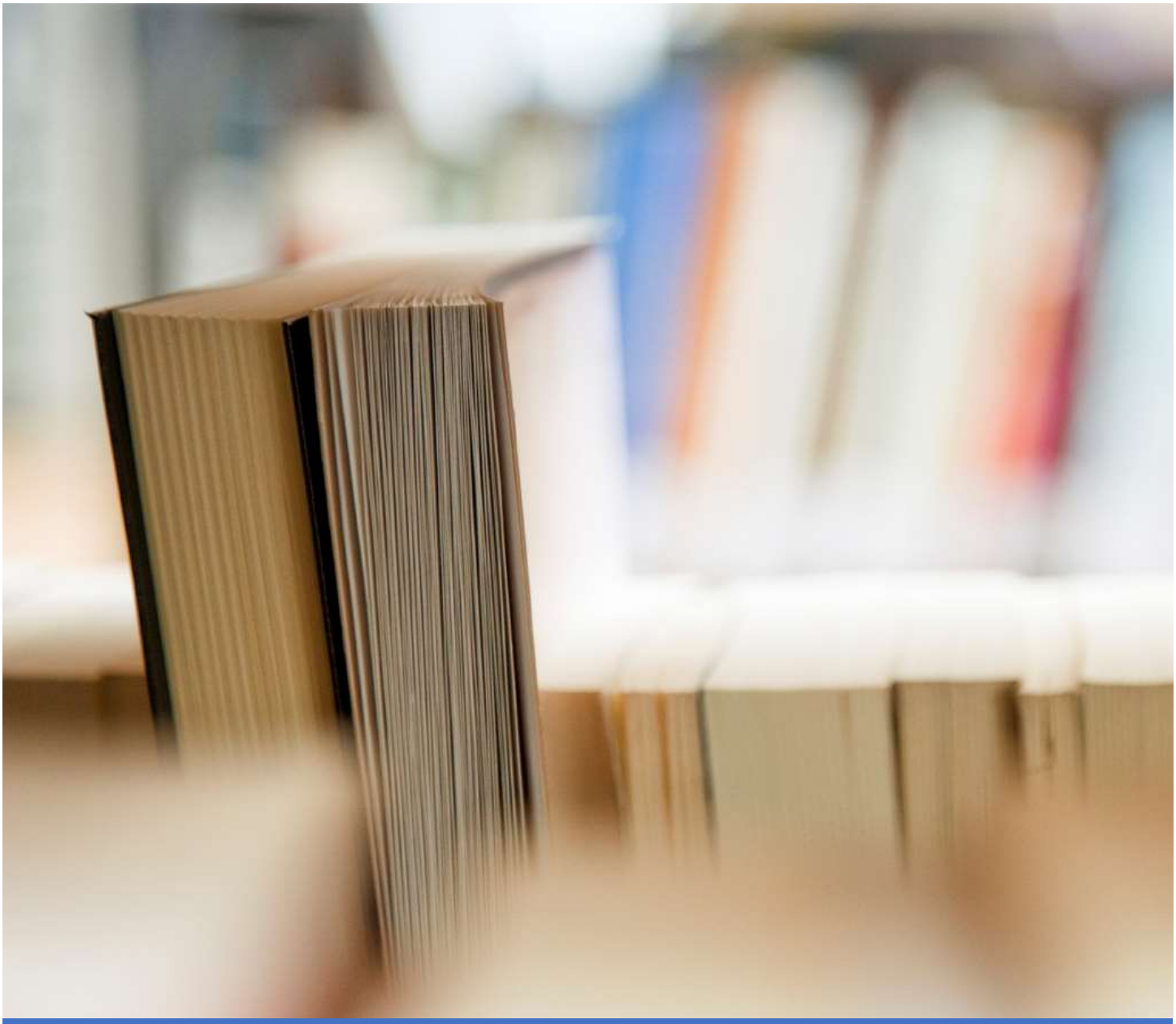


Assignment

Computational Fluid Dynamics

Aditya R. Prajapati

December 4, 2018

aditya12398@gmail.com

Abstract

Numerical techniques are used to solve a variety of equations in the field of engineering. Not all equations are analytically solvable and certain engineering problems can only be handled numerically. One of the most important equations for an aerospace engineer is the Navier Stokes equation. It needs to be solved numerically too. A few of the lower order schemes are the central difference and the backward difference scheme. In this paper we will study the effect of the aforementioned schemes on a relatively smaller equation which has an analytical solution, the linear advection equation. This paper tries to show the dispersion and the diffusive effects of numerical solutions and how it varies with grid sizing, time stepping and changing the Courant-Friedrich-Lewy number, a stability parameter. The used code and resulting charts are attached.

Problem Statement

To solve the linear advection equation numerically for a sine wave as an input using:

1. Forward in time and backward in space discretization.
2. Forward in time and central in space discretization.

Formulation

The linear advection equation is given as:

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

When the forward time stepping and central spatial discretization is applied to the above equation, we get:

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} + c\frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x} = 0$$

When the forward time stepping and backward (upwind) spatial discretization is applied to the above equation, we get:

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} + c\frac{u_j^n - u_{j-1}^n}{\Delta x} = 0$$

Schemes

Upwind Scheme (Backward Euler Method)

The backward Euler method is one of the schemes used to discretize derivatives using the point of interest and the point just before it. The subscript (here; can be superscript too) j is used to denote the point of interest in our formulation. Backward Euler Method is one of the most used methods in numerical techniques as it is unconditionally stable. It is a first order method. Stability analysis can be done as follows:

Let $u_j^n = \sigma^n e^{ikx_j}$

Substituting in the discretized equation, we get:

$$\frac{\sigma^{n+1} e^{ikx_j} - \sigma^n e^{ikx_j}}{\Delta t} + c\frac{\sigma^n e^{ikx_j} - \sigma^n e^{ikx_{j-1}}}{\Delta x} = 0$$

$$\sigma^{n+1}e^{ikx_j} - c\Delta t\frac{\sigma^n e^{ikx_{j-1}}}{\Delta x} = \sigma^n e^{ikx_j}(1 - c\frac{\Delta t}{\Delta x})$$

$$c\frac{\Delta t}{\Delta x} = \gamma$$

$$\boldsymbol{\sigma^{n+1}e^{ikx_j} - \gamma\sigma^n e^{ikx_{j-1}} = \sigma^n e^{ikx_j}(1-\gamma)}$$

$$\boldsymbol{\sigma = \gamma\, e^{-ikh} + (1-\gamma)}$$

We get **0 < γ < 1** for a stable solution. In other cases, the solution blows up.

Central Difference Scheme

The central difference scheme is a 2nd order accurate scheme. It uses points one ahead and one before the point of interest to calculate the derivatives at the point of interest. This scheme, even though 2nd order accurate, is not used since the scheme is unstable and will blow up at any value of γ. Stability analysis can be done as follows:

Let $\boldsymbol{u_j^n = \sigma^n e^{ikx_j}}$

Substituting in the discretized equation, we get:

$$\boldsymbol{\frac{\sigma^{n+1}e^{ikx_j} - \sigma^n e^{ikx_j}}{\Delta t} + c\frac{\sigma^n e^{ikx_{j+1}} - \sigma^n e^{ikx_{j-1}}}{2\Delta x} = 0}$$

$$\boldsymbol{\sigma^{n+1}e^{ikx_j} + c\Delta t\frac{\sigma^n e^{ikx_{j+1}} - \sigma^n e^{ikx_{j-1}}}{2\Delta x} = \sigma^n e^{ikx_j}}$$

$$\boldsymbol{c\frac{\Delta t}{\Delta x} = \gamma}$$

$$\boldsymbol{\sigma = 1 - \frac{\gamma}{2}\,(e^{ikh} - e^{-ikh})}$$

$$\boldsymbol{\sigma = 1 - \gamma\, i\, sin(kh)}$$

$$\boldsymbol{|\sigma|^2 = 1 + \gamma^2 sin^2(kh)}$$

The obtained expression's magnitude is never less than one. This means that the scheme is inherently unstable. We might get stable solutions for small time steps and extremely low values of gamma, but as we move ahead in time the solution starts to oscillate and eventually blows up unboundedly.

About the Code

The numerical scheme is coded in java with the following functionalities:

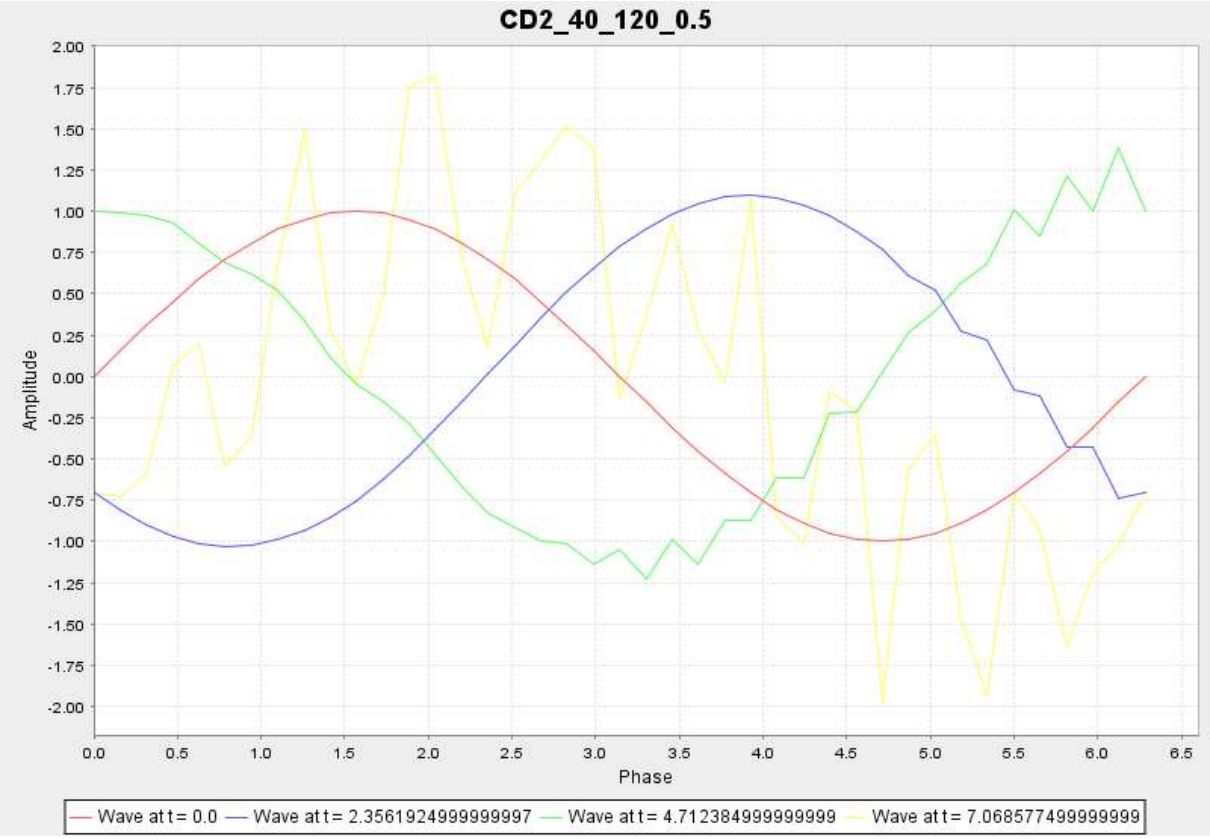
1. Saving the data in a .csv file.
2. Plotting the graph, displaying on a frame and saving it in the assignment directory in .png format.

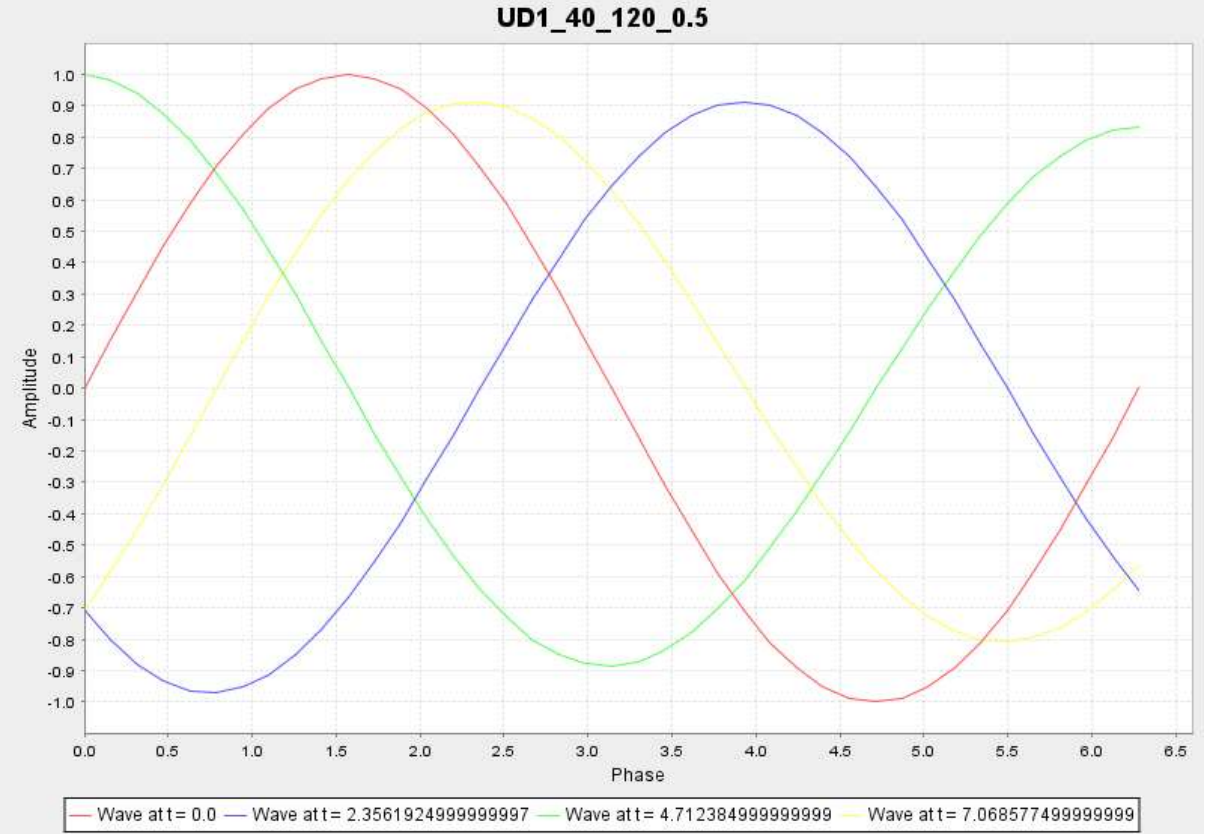
The program is written in Java. The built-in library JFrame is used to generate the frame which displays the chart. Another library 'JFreechart' is used to generate charts. The charts are saved and titled as 'scheme_subdomains_iterations_CFL'. The number of subdomains, iterations and test CFL number are taken as inputs from the user. The graphs at every quarter of total iterations are generated.

Results

Case 1: Low subdomain number, Low iteration count, stable CFL number

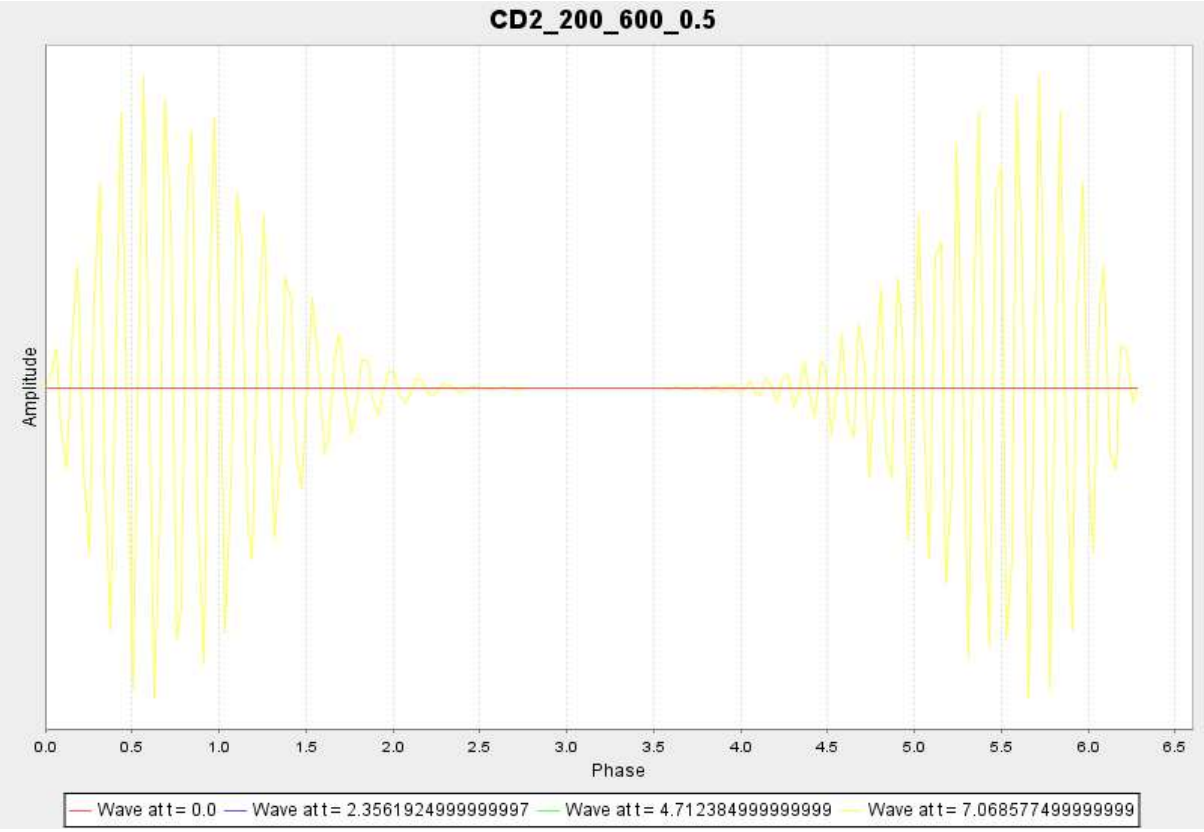
Basic case to show the dispersion and diffusing effect more clearly.

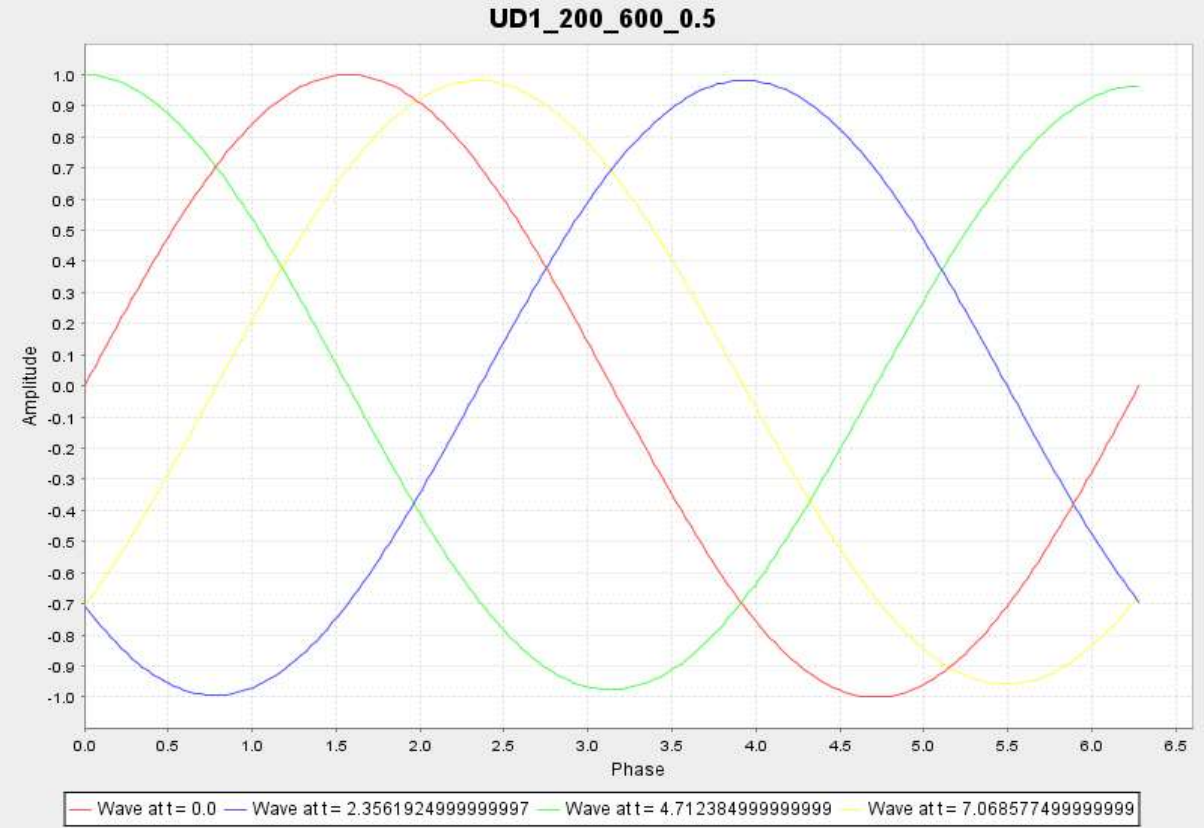




Case 2: Higher spatial and time steps, Same CFL number

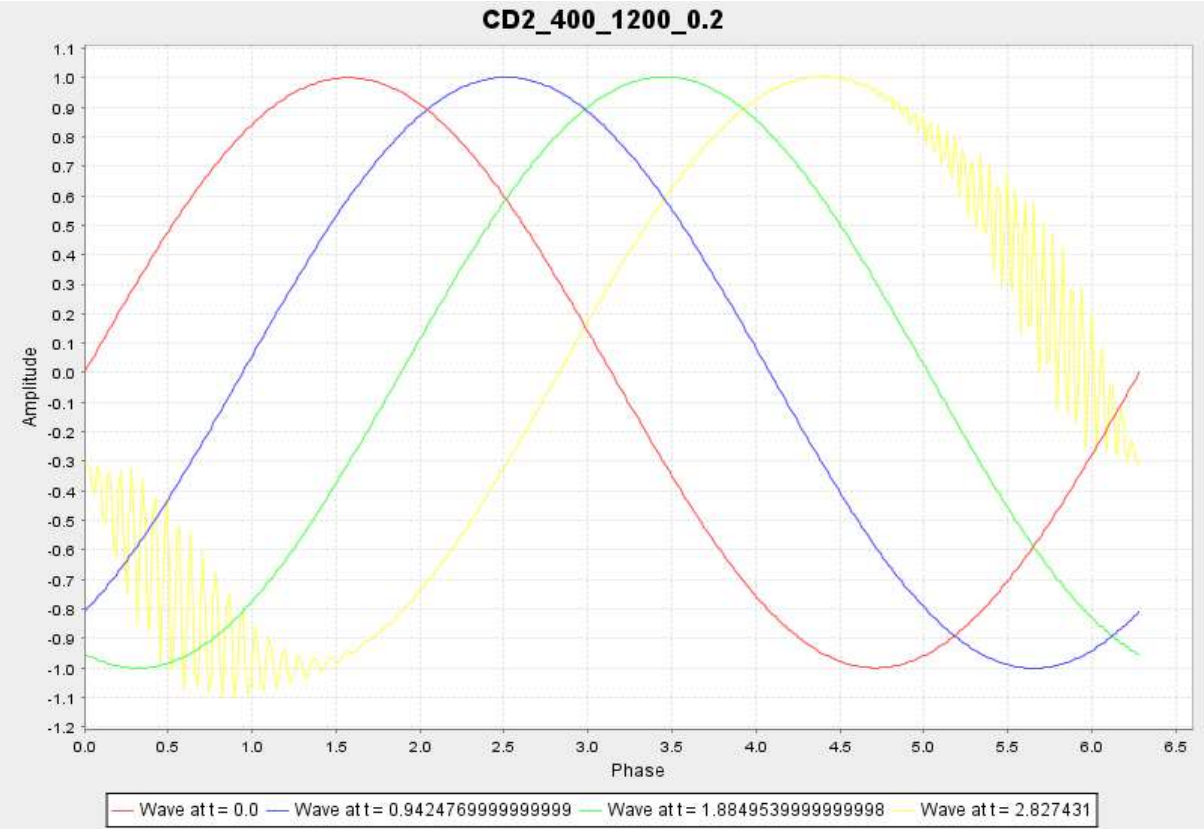
A higher number of grid points for both time steps and spatial steps are shown to illustrate the reduction of diffusive effect in UD1 by increasing spatial grid numbers and the increased travelling of the instability in the CD2 scheme

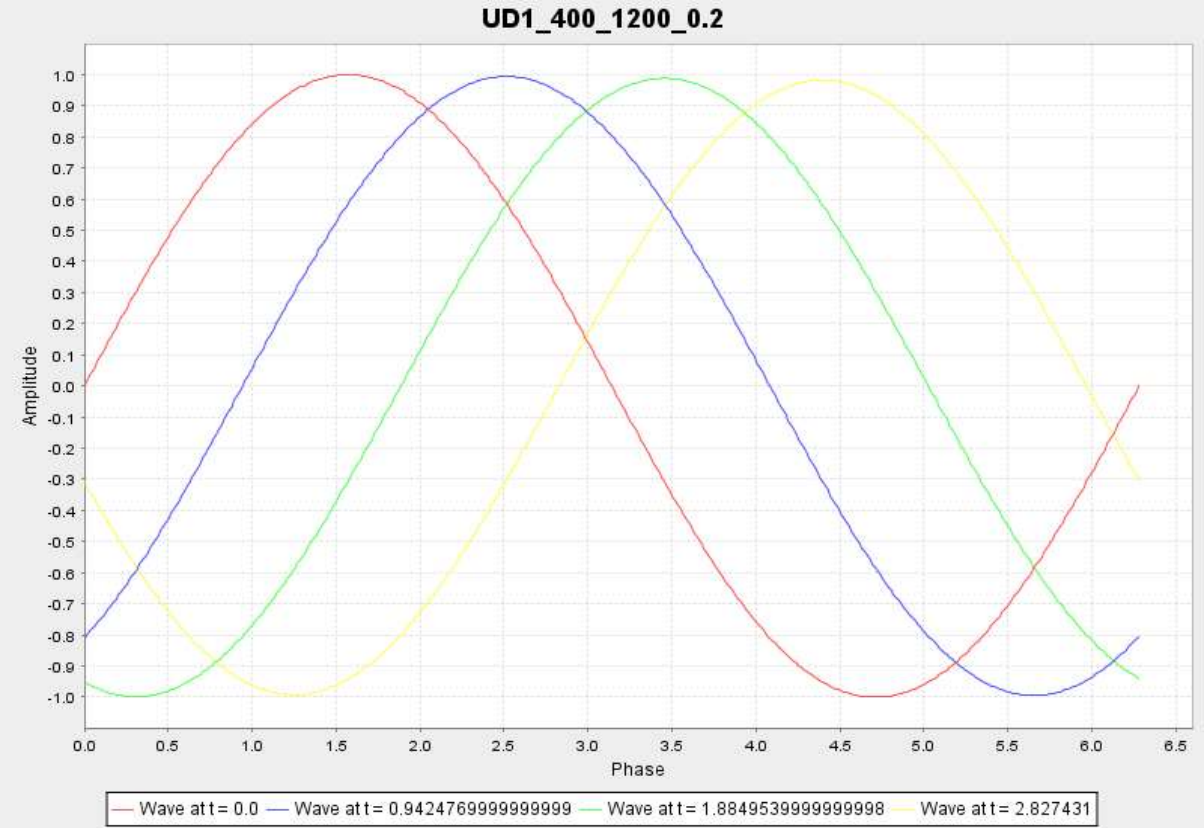




Case 3: Higher Time and Spatial steps, Lower CFL number

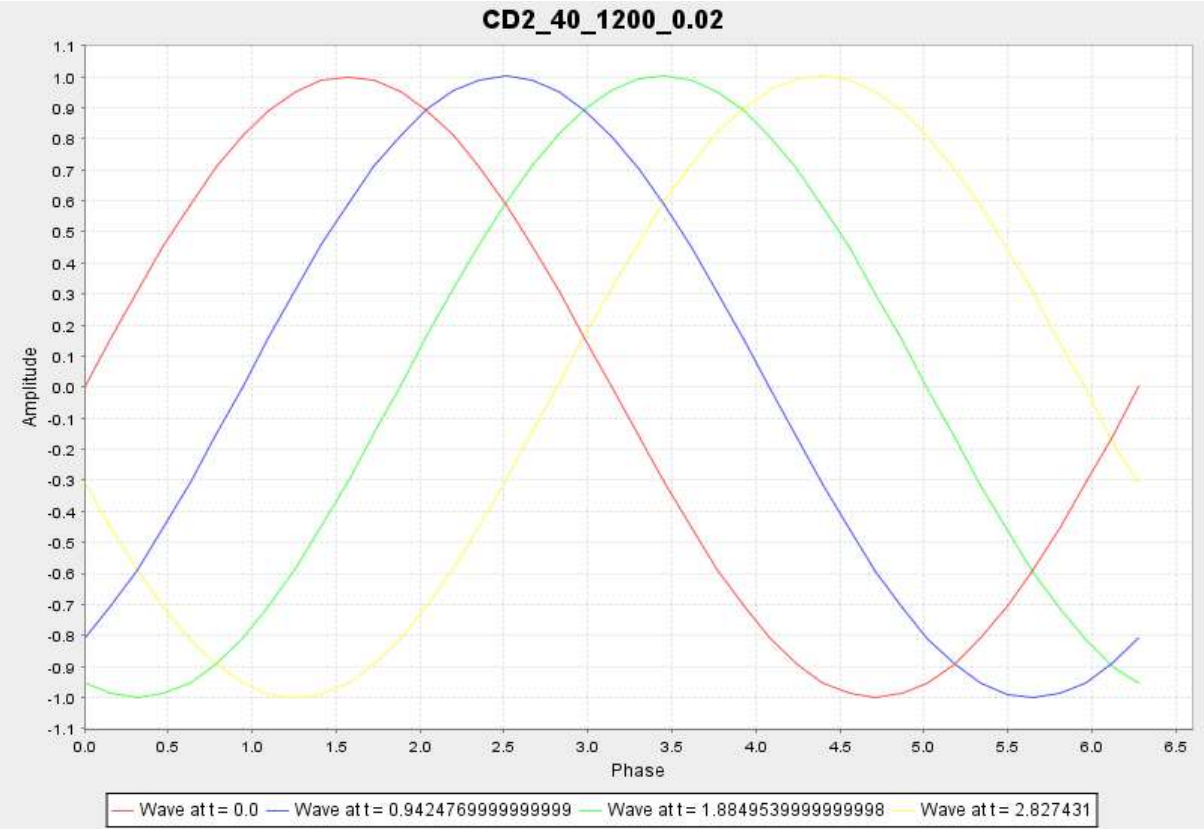
A higher number of spatial steps ensures almost negligible diffusivity in UD1 scheme and delays the onset of blowing up of the solution in CD2 scheme. The fluctuations in he solution are observed at the ends of the domain. As time passes by, this is carried to the other parts of the domain as well.

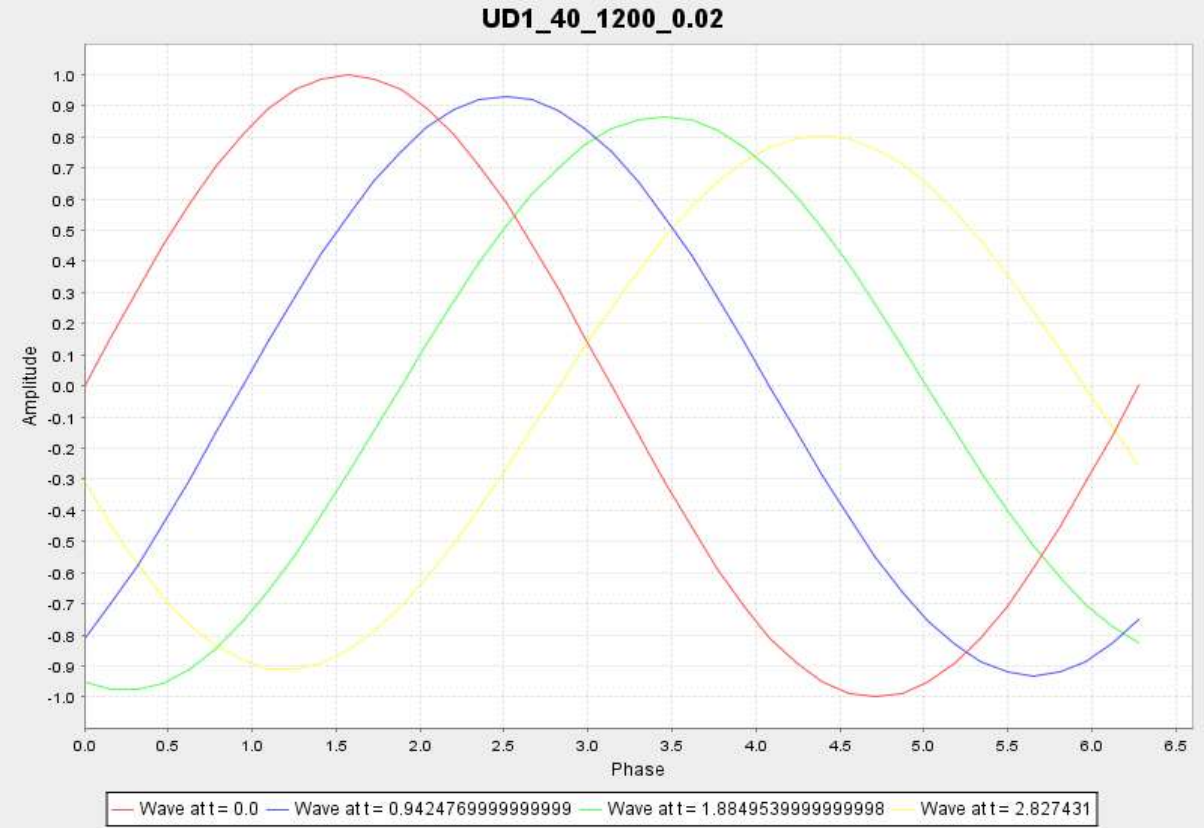




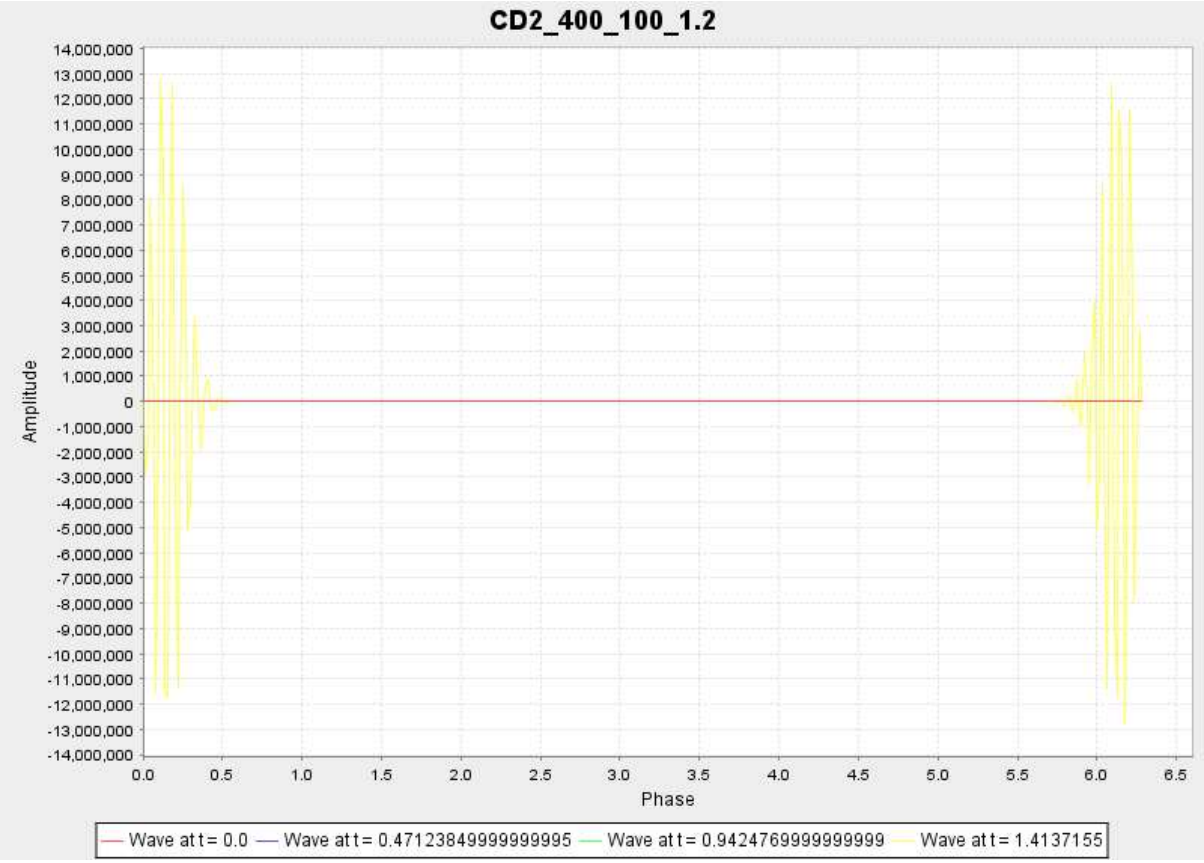
Case 4: Lesser spatial steps, stable (lower) CFL

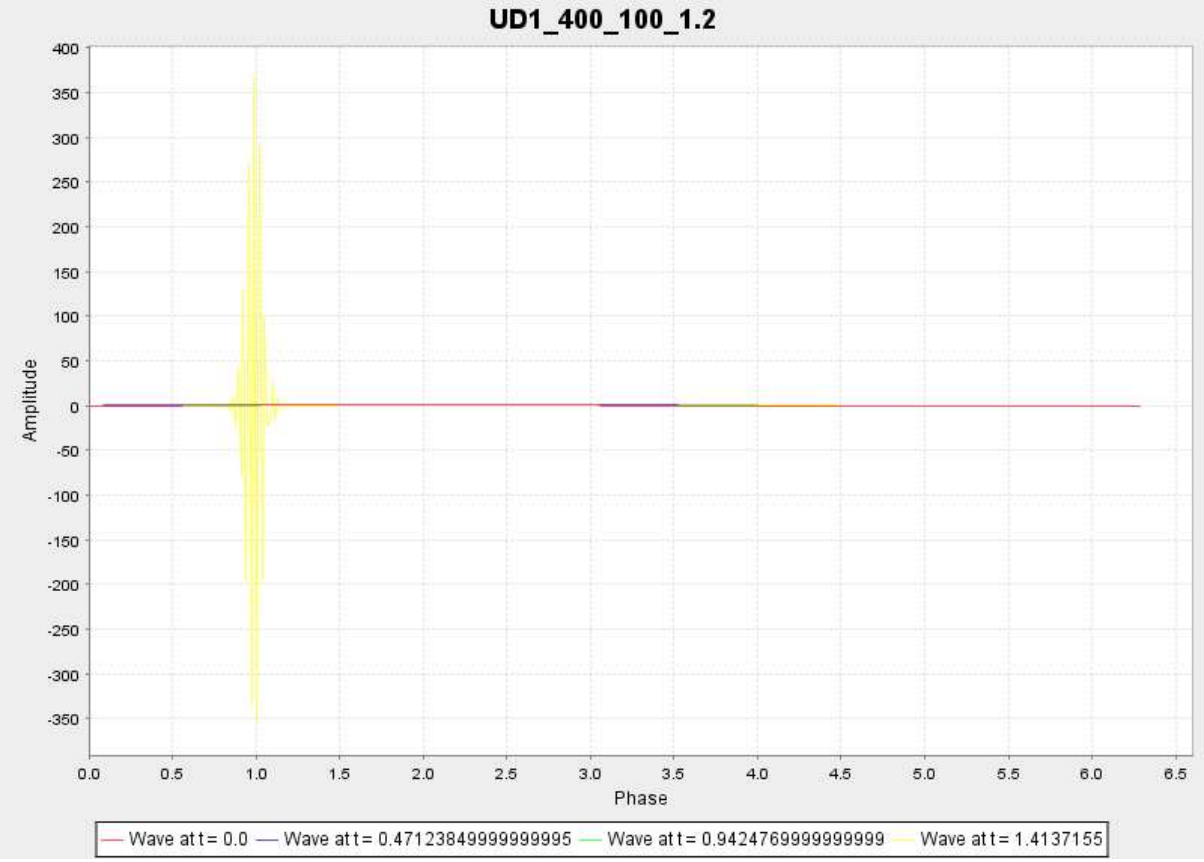
The last case demonstrates the importance of the CFL number in numerical solutions. We can see that the CD2, even after being unstable, has not shown oscillations at either ends of the boundary. A lower number of spatial steps is, however, used to show the diffusive effect in UD1 scheme over time.





Case 5: Demonstration of not satisfying the CFL criteria.





Inference

- The solution tends to have lesser diffusion when the number of spatial grids is increased significantly in a scheme. It also ensures a closer to analytical solution.
- The increase of the time steps showed us how rapidly the fluctuations in an unstable scheme travel and blow the solution up.
- The importance of the CFL number is demonstrated and how a number closer to 0 delays fluctuations in an inherently unstable scheme.

Code

```
1. import org.jfree.chart.*;
2. import org.jfree.chart.plot.*;
3. import org.jfree.data.xy.*;
4. import org.jfree.ui.*;
5. import java.io.*;
6. import java.awt.event.*;
7. import javax.swing.*;

8. class CFDassignment
9. {
10. static String title, directory = "D:\\Dropbox\\Documents\\IIT
    Kharagpur\\Academics\\Aerospace Engineering\\Semester 5\\Computational Fluid
    Dynamics\\Assignment";
11. static double udata_CD2[][],udata_UD1[][];
12. int x, y;
13. double PI=3.141590, dt, dx, c=1, CFL;
14. XYSeriesCollection data[]=new XYSeriesCollection[2];

15. public CFDassignment(final String title1)throws IOException
16. {
17. int ch;
18. BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
19. getValues();
20. setBoundary();
21. calculate();
22. store(title);
23. System.out.println("dx = " + dx + " dt = " + dt + " CFL = " + CFL);
24. System.out.println("Do you want to display a graph?(1/0)");
25. ch=Integer.parseInt(in.readLine());
26. if(ch==1)
27. printCD2(title1);
28. }

29. public void printCD2(String title1)throws IOException
30. {
31. JFrame f=new JFrame(title1);
32. int i,j;
33. Boolean cs;
34. data[0] = new XYSeriesCollection();
35. XYSeries series[]= new XYSeries[y];
36. for(i=0;i<y;i+=(y/4))
```

```
{
series[i]=new XYSeries("Wave at t = " + (i*dt) + "\t");
for(j=0;j<=x;j++)
{
series[i].add(j*dx, udata_CD2[i][j]);
}
data[0].addSeries(series[i]);
}
final JFreeChart chart =
ChartFactory.createXYLineChart("CD2_"+title,"Phase","Amplitude",data[0],PlotOrientation.VERTI
CAL,true,true,false);
final ChartPanel chartPanel = new ChartPanel(chart);
ChartUtilities.saveChartAsPNG(new File(directory + "\\Graphs\\CD2_" + title + ".png"), chart,
800, 570);
chartPanel.setPreferredSize(new java.awt.Dimension(800, 570));
f.setContentPane(chartPanel);
f.pack();
RefineryUtilities.centerFrameOnScreen(f);
f.addWindowListener(new WindowAdapter(){ @Override public void
windowClosing(WindowEvent e){try{printUD1(title1);}catch(IOException
exp){System.out.println("Error.");}}});
f.setVisible(true);
}

public void printUD1(String title1)throws IOException
{
JFrame f=new JFrame(title1);
int i,j;
Boolean cs;
data[1] = new XYSeriesCollection();
XYSeries series[]= new XYSeries[y];
for(i=0;i<y;i+=(y/4))
{
series[i]=new XYSeries("Wave at t = " + (i*dt) + "\t");
for(j=0;j<=x;j++)
{
series[i].add(j*dx, udata_UD1[i][j]);
}
data[1].addSeries(series[i]);
}
final JFreeChart chart =
ChartFactory.createXYLineChart("UD1_"+title,"Phase","Amplitude",data[1],PlotOrientation.VERTI
CAL,true,true,false);
final ChartPanel chartPanel = new ChartPanel(chart);
ChartUtilities.saveChartAsPNG(new File(directory + "\\Graphs\\UD1_" + title + ".png"), chart,
800, 570);
chartPanel.setPreferredSize(new java.awt.Dimension(800, 570));
f.setContentPane(chartPanel);
f.pack();
RefineryUtilities.centerFrameOnScreen(f);
f.setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
f.setVisible(true);
```

```
80.  }

81.  public void getValues()throws IOException
82.  {
83.  BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
84.  System.out.print("Specify the number of subdomains: ");
85.  x=Integer.parseInt(in.readLine());
86.  System.out.print("Specify the number of iterations: ");
87.  y=Integer.parseInt(in.readLine());
88.  System.out.print("Specify the Courant Number (0-1 for stable solution): ");
89.  CFL=Double.parseDouble(in.readLine());
90.  udata_CD2=new double[y][x+1];
91.  udata_UD1=new double[y][x+1];
92.  if(x<=0)
93.  {
94.  System.out.print("Incorrect input for subdomains, using the default value, 200");
95.  x=200;
96.  }
97.  if(y<=0)
98.  {
99.  System.out.print("Incorrect input for iterations, using the default value, 50");
100.y=50;
101.}
102.if(CFL<=0)
103.{
104.System.out.print("Incorrect input for CFL, using the default value, 0.05");
105.CFL=0.05;
106.}
107.title=Integer.valueOf(x).toString() + "_" + Integer.valueOf(y).toString() + "_" +
     Double.valueOf(CFL).toString();
108.dx=2*PI/x;
109.dt=dx*CFL/c;
110.}

111.public static void main(String args[])throws IOException
112.{
113.CFDassignment obj=new CFDassignment("Graph");
114.}

115.public void setBoundary()
116.{
117.int i, j;
118.for (i=0;i<=x;i++)
119.{
120.udata_CD2[0][i]=Math.sin(i*dx);
121.udata_UD1[0][i]=Math.sin(i*dx);
122.}
123.for(j=1;j<y;j++)
124.{
125.udata_CD2[j][0]=-Math.sin(j*dt);
126.udata_CD2[j][x]=Math.sin(2*PI-j*dt);
127.udata_UD1[j][0]=-Math.sin(j*dt);
```

```
128.}
129.}

130.public void calculate()
131.{
132.int i, j=1;
133.for(i=1;i<y;i++)
134.{
135.for(j=1;j<x;j++)
136.{
137.udata_CD2[i][j]=udata_CD2[i-1][j] + (dt*c)/(2*dx)*(udata_CD2[i-1][j-1]-udata_CD2[i-1][j+1]);
138.udata_UD1[i][j]=udata_UD1[i-1][j] + (dt*c)/(dx)*(udata_UD1[i-1][j-1]-udata_UD1[i-1][j]);
139.}
140.udata_UD1[i][j]=udata_UD1[i-1][j] + (dt*c)/(dx)*(udata_UD1[i-1][j-1]-udata_UD1[i-1][j]);
141.}
142.}

143.public void store(String title)throws IOException
144.{ int i, j;
145.double a;
146.PrintWriter out_CD2 = new PrintWriter(directory + "\\Data Sheets\\CD2_" + title + ".csv");
147.PrintWriter out_UD1 = new PrintWriter(directory + "\\Data Sheets\\UD1_" + title + ".csv");
148.out_CD2.write("dt\\dx, ");
149.out_UD1.write("dt\\dx, ");
150.for(j=0;j<=x;j++)
151.{
152.out_CD2.write(Double.valueOf(j*dx).toString() + ", ");
153.out_UD1.write(Double.valueOf(j*dx).toString() + ", ");
154.}
155.for(i=0;i<y;i++)
156.{
157.out_CD2.write("\n" + Double.valueOf(i*dt).toString() + ", ");
158.out_UD1.write("\n" + Double.valueOf(i*dt).toString() + ", ");
159.for(j=0;j<=x;j++)
160.{
161.out_CD2.write(Double.valueOf(udata_CD2[i][j]).toString() + ", ");
162.out_UD1.write(Double.valueOf(udata_UD1[i][j]).toString() + ", ");
163.}
164.}
165.out_CD2.close();
166.out_UD1.close();
167.}
168.}
```