EGIM08 COMPUTATIONAL FLUID DYNAMICS

INDIVIDUAL PROJECT 1 REPORT

SHIJTITH BABU SHEEJA (*Student No: 2235777)*

MSc Structural Engineering College of Engineering

Swansea University, Swansea, UK

2235777@swansea.ac.uk

1. AIM:

The aim of the proposed project is to initially propagate a cosine wave having a wave through a given domain of unit length. Followed by it a computer program in MATLAB must be formulated to solve the one-dimensional scalar convection equation using the first order explicit and implicit upwind finite difference schemes. The boundary conditions of the cosine wave are given.

2. INTRODUCTION

The one-dimensional scalar convection equation is a partial differential equation that describes the movement of a scalar quantity (such as temperature, concentration, or density) in a fluid that is undergoing convective motion. This equation can be applied to describe the transport of a scalar quantity in a fluid that is undergoing convective motion. The equation can be written as follows:

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

Where x represents the position, t represents the passage of time, and u represents the velocity of the fluid which is assumed to be and assumed to be independent of ϕ.

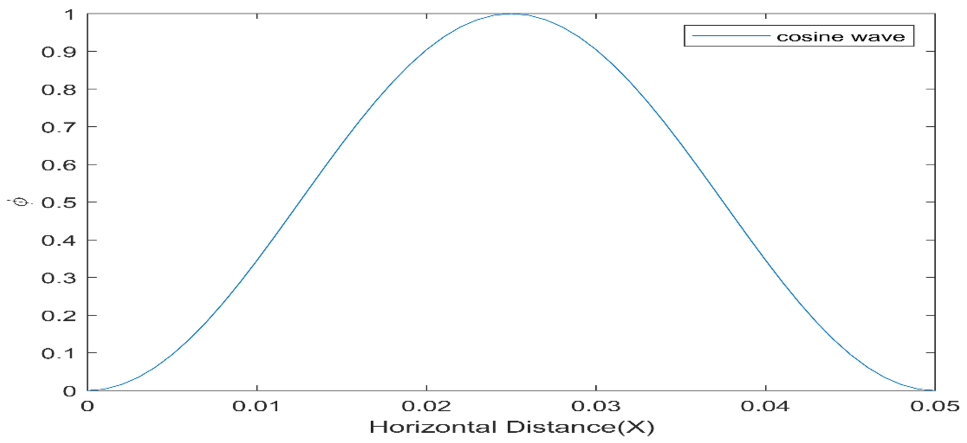


Figure 1: the cosine wave obtained as per boundary conditions as initial input.

This equation is essential in computational fluid dynamics (CFD) because it is used to describe the behavior of fluids in a range of applications, such as the transmission of heat, chemical processes, and the mixing of fluids. In addition, this equation is used to model the behavior of fluids. CFD simulations may be used to optimize the design of engineering systems as well as anticipate the behavior of fluids under a variety of conditions by numerically solving the convection equation. This allows the simulations to be utilized for both of these purposes. In addition, the one-dimensional scalar convection equation is frequently utilized as a benchmark issue in the context of computational fluid dynamics (CFD) in order to evaluate the precision and effectiveness of various numerical approaches. It is possible to solve this equation using any one of a wide variety of numerical methods, such as the finite difference method, the finite volume method, or the finite element method. Comparing the outcomes of these various numerical methods can assist researchers in determining which methods are the most accurate and effective when it comes to modelling fluid flow.

Importance of first order explicit and implicit upwind finite difference schemes.

In the field of computational fluid dynamics (CFD), first order explicit and implicit upwind finite difference schemes are essential components because they offer straightforward and effective solutions to fluid flow problems that require the advection of a scalar quantity. While attempting to model the flow of a fluid, which might be difficult, this kind of difficulty can develop in CFD (such as temperature or concentration). Explicit upwind strategies are important for the resolution of large-scale issues since they are not only straightforward to implement but also very effective from a computational standpoint. However, they are prone to being unstable under certain conditions, which can limit the number of contexts in which they can be used effectively. On the other hand, implicit upwind schemes are not only unconditionally stable, but they also have the potential to be more accurate than explicit schemes. This is because implicit upwind schemes use less information than explicit schemes. But nevertheless, because it is necessary to solve a system of equations at each time step, they call for a greater number of computational resources to be made available. The specific problem that needs to be solved, as well as the level of accuracy and computational efficiency that is desired, are what determine which finite difference scheme should be used in computational fluid dynamics (CFD). In other words, the choice of which finite difference scheme to use is determined by the specific problem. When comparing more complex numerical methods to more basic ones, it is usual practice to use first order explicit and implicit upwind finite difference schemes as baseline methods. This is done for the goal of evaluating the performance of the more advanced numerical methods.

3. METHODOLOGY

The one-dimensional scalar convection equation using the numerical method is solved and compared with the first order explicit and implicit upwind conditions using the Matlab computer program. The given cosine wave has to be plotted to obtain the exact boundary conditions for the curve. The function for the curve is extracted as:

$$\phi = cos^2\left(\left(\frac{\pi}{2}\right) + \left(\frac{\pi}{0.05}\right) * x\right) \tag{1}$$

By applying the initial boundary conditions, CFL value, time step etc .The initial wave form is generated and the wave motion is captured for different time step. For upward explicit and implicit method, the computer program is modified accordingly depending on the theory and the results are obtained.

3.1. UPWIND EXPLICIT METHOD

The governing equation for the proposed problem is:

$$\frac{\partial\phi}{\partial t} + U\frac{\partial\phi}{\partial x} = 0 \tag{2}$$

Approximating the solution of partial differential equations that represent fluid flow can be done numerically with the help of the first-order explicit upwind methodology. In fluid dynamics and other branches of science, hyperbolic equations often need to be solved.

$$\frac{\Phi_i^{n+1}-\Phi_i^n}{\Delta t} + u\left(\frac{\Phi_i^n-\Phi_{i-1}^n}{\Delta x}\right) = 0 \tag{3}$$

Explicit scheme is conditionally stable. A von Neumann stability analysis gives a ourant Friedrichs-Lewy (CFL) condition of $0 \leq c \leq 1$, where $c = u\Delta t/\Delta x$. The stabilized discretization is obtained as:

$$\Phi_i^{n+1} = \Phi_i^n - \left(\frac{u\Delta t}{\Delta x}\right)(\Phi_i^n - \Phi_{i-1}^n) \tag{4}$$

3.2. Stability of explicit method:

$$\Phi_i^{n+1} = \Phi_i^n + \Delta t \frac{\partial \Phi_i^n}{\partial t} + \frac{\Delta t^2}{2}\frac{\partial^2 \Phi_i^n}{\partial t^2} + \cdots \tag{5}$$

$$\Phi_{i-1}^n = \Phi_i^n + \Delta x \frac{\partial \Phi_i^n}{\partial x} - \frac{\Delta x^2}{2}\frac{\partial^2 \Phi_i^n}{\partial x^2} + \cdots \tag{6}$$

Substituting (5) and (6) in (4) gives

$$\Phi_i^n + \Delta t \frac{\partial \Phi_i^n}{\partial t} + \frac{\Delta t^2}{2}\frac{\partial^2 \Phi_i^n}{\partial t^2} = \Phi_i^n - CFL(\Phi_i^n - \Phi_i^n - \Delta x \frac{\partial \Phi_i^n}{\partial x} + \frac{\Delta x^2}{2}\frac{\partial^2 \Phi_i^n}{\partial x^2} + \cdots$$

$$\Delta t \frac{\partial \Phi_i^n}{\partial t} + \frac{\Delta t^2}{2}\frac{\partial^2 \Phi_i^n}{\partial t^2} = -u\Delta t \frac{\partial \Phi_i^n}{\partial x} + \frac{u\,\Delta t\,\Delta x}{2}\frac{\partial^2 \Phi_i^n}{\partial x^2} + \cdots.$$

$$\frac{\partial \Phi_i^n}{\partial t} + \frac{\Delta t}{2}\frac{\partial^2 \Phi_i^n}{\partial t^2} + u\frac{\partial \Phi_i^n}{\partial x} - \frac{u\,\Delta x}{2}\frac{\partial^2 \Phi_i^n}{\partial x^2} + \cdots . = 0$$

$$\frac{\partial \Phi}{\partial t} = -u\frac{\partial \Phi}{\partial x}$$

$$\frac{\partial^2 \Phi_i}{\partial t^2} = -u\frac{\partial}{\partial t}\left(\frac{\partial \Phi}{\partial x}\right) = -u\frac{\partial}{\partial x}\left(\frac{\partial \Phi}{\partial t}\right) = u^2\frac{\partial^2 \Phi}{\partial x^2}$$

$$\frac{\partial \Phi_i^n}{\partial t} + u\frac{\partial \Phi_i^n}{\partial x} + \frac{\Delta t\, u^2}{2}\frac{\partial^2 \Phi_i}{\partial x^2} - \frac{u\,\Delta x}{2}\frac{\partial^2 \Phi_i^n}{\partial x^2} + \cdots = 0 \tag{7}$$

The negative diffusion error is $\frac{\Delta t \mathrm{u}^2}{2}\frac{\partial^2 \varphi_i^n}{\partial x^2}$ and the positive diffusion error is $\frac{u\Delta x}{2}\frac{\partial^2 \varphi_i^n}{\partial \mathrm{x}^2}$.

3.3. Upwind implicit method

The first order explicit upwind scheme is given as:

$$\frac{\Phi_i^{n+1} - \Phi_i^n}{\Delta t} + u\left(\frac{\Phi_i^{n+1} - \Phi_{i-1}^{n+1}}{\Delta x}\right) = 0 \tag{8}$$

$$\Phi_i^{n+1} + \left(\frac{u\Delta t}{\Delta x}\right)(\Phi_i^{n+1} - \Phi_{i-1}^{n+1}) = \Phi_i^n$$

$$\Phi_i^{n+1} + CFL\,(\Phi_i^{n+1} - \Phi_{i-1}^{n+1}) = \Phi_i^n$$

The stability of the upwind method is obtained by :

$$\Phi_i^{n+1} = \Phi_i^n + \Delta t \frac{\partial \Phi_i^n}{\partial t} + \frac{\Delta t^2}{2}\frac{\partial^2 \Phi_i^n}{\partial t^2} + \cdots$$

$$\Phi_i^{n+1} = \Phi_i^n + \Delta t\frac{\partial \Phi_i^n}{\partial t} + \frac{\Delta t^2}{2}u^2\frac{\partial^2 \Phi_i^n}{\partial x^2} + \cdots$$

And,

$$\Phi_{i-1}^{n+1} = \Phi_{i-1}^n + \Delta t\frac{\partial \Phi_{i-1}^n}{\partial t} + \frac{\Delta t^2}{2}\frac{\partial^2 \Phi_{i-1}^n}{\partial t^2} + \cdots$$

$$\Phi_{i-1}^n = \Phi_i^n - \Delta x\frac{\partial \Phi_i^n}{\partial x} + \frac{\Delta x^2}{2}\frac{\partial^2 \Phi_i^n}{\partial x^2} + \cdots$$

Substitute the value of Φ_{i-1}^n in Φ_{i-1}^{n+1}

$$\Phi_{i-1}^{n+1} = \Phi_i^n - \Delta x\frac{\partial \Phi_i^n}{\partial x} + \frac{\Delta x^2}{2}\frac{\partial^2 \Phi_i^n}{\partial x^2} + \Delta t\frac{\partial}{\partial t}\left(\Phi_i^n - \Delta x\frac{\partial \Phi_i^n}{\partial x}\right) + \frac{\Delta t^2}{2}\frac{\partial^2 (\Phi_i^n)}{\partial t^2} + \cdots$$

After simplification

$$\Phi_{i-1}^{n+1} = \Phi_i^n - \Delta x\frac{\partial \Phi_i^n}{\partial x} + \frac{\Delta x^2}{2}\frac{\partial^2 \Phi_i^n}{\partial x^2} + \Delta t\frac{\partial \Phi_i^n}{\partial t} - \Delta x\Delta t\frac{\partial}{\partial t}\left(\frac{\partial \Phi_i^n}{\partial x}\right) + \frac{\Delta t^2}{2}\frac{\partial^2 \Phi_i^n}{\partial t^2} + \cdots$$

$$\frac{\partial \Phi_i^n}{\partial t} + \frac{\Delta t}{2}u^2\frac{\partial^2 \Phi_i^n}{\partial x^2} + \frac{u}{\Delta x}\left(\Phi_i^n + \Delta t\frac{\partial \Phi_i^n}{\partial t} + \frac{\Delta t^2}{2}\frac{\partial^2 \Phi_i^n}{\partial t^2}\right)$$

$$-\frac{u}{\Delta x}\left(\Phi_i^n - \Delta x\frac{\partial \Phi_i^n}{\partial x} + \frac{\Delta x^2}{2}\frac{\partial^2 \Phi_i^n}{\partial x^2} + \Delta t\frac{\partial \Phi_i^n}{\partial t} - \Delta x\Delta t\frac{\partial}{\partial t}\left(\frac{\partial \Phi_i^n}{\partial x}\right) + \frac{\Delta t^2}{2}\frac{\partial^2 \Phi_i^n}{\partial t^2}\right) + \cdots = 0$$

$$\frac{\partial \Phi_i^n}{\partial t} + u\frac{\partial \Phi_i^n}{\partial x} + \frac{\Delta t}{2}u^2\frac{\partial^2 \Phi_i^n}{\partial x^2} - \frac{u\Delta x}{2}\frac{\partial^2 \Phi_i^n}{\partial x^2} + u\Delta t\frac{\partial}{\partial t}\left(\frac{\partial \Phi_i^n}{\partial x}\right) + \cdots = 0$$

$$\frac{\partial \Phi_i^n}{\partial t} + u\frac{\partial \Phi_i^n}{\partial x} + \frac{\Delta t}{2}u^2\frac{\partial^2 \Phi_i^n}{\partial x^2} - \frac{u\Delta x}{2}\frac{\partial^2 \Phi_i^n}{\partial x^2} - u^2\Delta t\frac{\partial^2 \Phi_i^n}{\partial x^2} + \cdots = 0$$

After rearranging and simplification

$$\frac{\partial \Phi_i^n}{\partial t} + u\frac{\partial \Phi_i^n}{\partial x} - \frac{\Delta t}{2}u^2\frac{\partial^2 \Phi_i^n}{\partial x^2} - \frac{u\Delta x}{2}\frac{\partial^2 \Phi_i^n}{\partial x^2} + \cdots = 0 \qquad (9)$$

The error terms are $\frac{u\Delta x}{2}\frac{\partial^2 \varphi_i^n}{\partial \mathrm{x}^2}$ and $\frac{\mathrm{u}^2\Delta t}{2}\frac{\partial^2 \varphi_i^n}{\partial \mathrm{x}^2}$ within the linear convection equation.

4. BOUNDARY CONDITIONS

Boundary conditions are crucial because they are required to completely define the problem and achieve a unique solution in mathematics. These requirements are imposed on the solution of the problem at the limits of the domain across which it is defined. The choice of boundary conditions depends on the physical problem being modeled and can include conditions on the value of the solution, its gradient, or a combination of both. The boundary conditions of the give problem are:

1) The displacement at x=0 and at x= 0.05

2) The velocity u

The choice of boundary conditions depends on the physical problem being modeled and can include conditions on the value of the solution, its gradient, or a combination of both.

5. CODE STRUCTURE

5.1. Explicit method

5.1.1 Pre-Processing:

In order to compare the first order explicit and implicit upwind finite difference schemes the time. The domain length is taken as 1m, and the domain of the curve varies from 0 to 0.05. For the explicit method CFL$\leq$ 1 is chosen and the initial velocity u is taken as 1 m/s. A component of artificial diffusion is given as 'k'. The entire domain is divided into 0.001, so the time step is taken as 0. 001. We have chosen a cosine function as:

$$\phi = cos^2\left(\left(\frac{\pi}{2}\right) + \left(\frac{\pi}{0.05}\right) * x\right)$$

The value of ϕ is contained within the limit of 0 toΔ obtain the end conditions of curve since the curve starts and end at zero. The initial value of phi for the whole domain of the curve is assigned to a new variable phi_prev and the same phi value is assigned to another variable 'phi_initial' for the calculation of first-time step and the proceeding ones. Followed by it, the graph is plotted against displacement 'x' and 'phi_initial' to get the initial curve.

5.1.2 Main processing:

Considering new time step as 'Nt' and using 'for' loop for different time step and by introducing artificial stiffness 'k' the value for phi is obtained. The obtained value of phi is again used for the next time step and the propagation of the wave is obtained for the whole domain.

5.1.3 Post processing:

The propagating wave is obtained. From that it is clear, the CFL number is controlling the stability of the wave. It will be stable for all the CFL ≤ 1 ,the wave will be stable if CFL $\geq$1 becomes unstable.

5.2. Implicit method:

5.2.1. Pre-processing: same as that of explicit method

5.2.2. Main processing: The coefficient matrix is defined with the upper diagonal component 1+CFL and the lower diagonal component - CFL. Then inside the loop a new phi value is obtained by multiplying the inverse of coefficient matrix multiplied by the sum of new phi value added to boundary of the problem. Then the value of domain is plotted against phi value.

5.2.3. Post processing*:* The result of the implicit method is obtained as a propagating wave for the given function and the length. The propagating wave shows the variation in the amount of diffusion with respect to time and displacement.

6. Result:

For the explicit upwind scheme the results obtained (x axis displacement, axis phi) for different CFL values and time are:

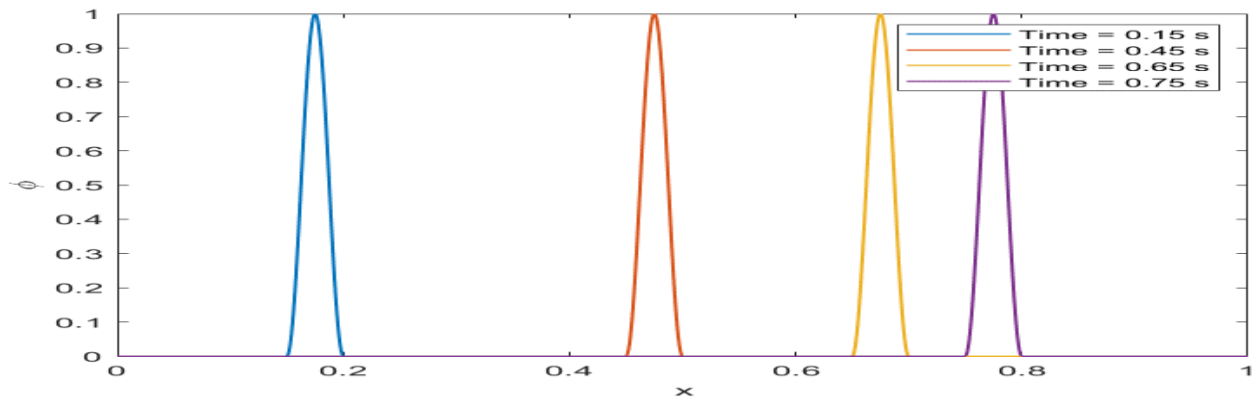


Figure 2: For CFL = 1

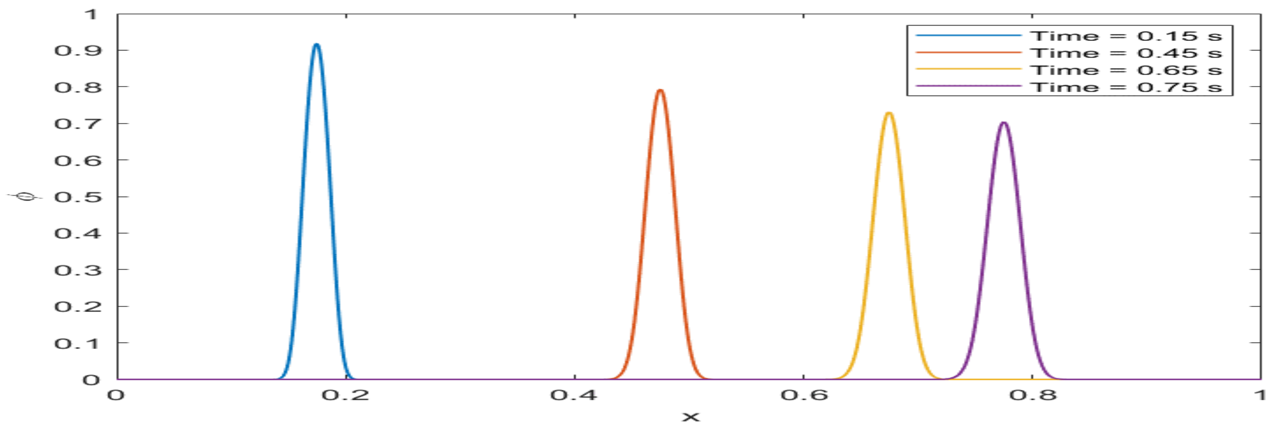


Figure 3: For CFL = 0.85

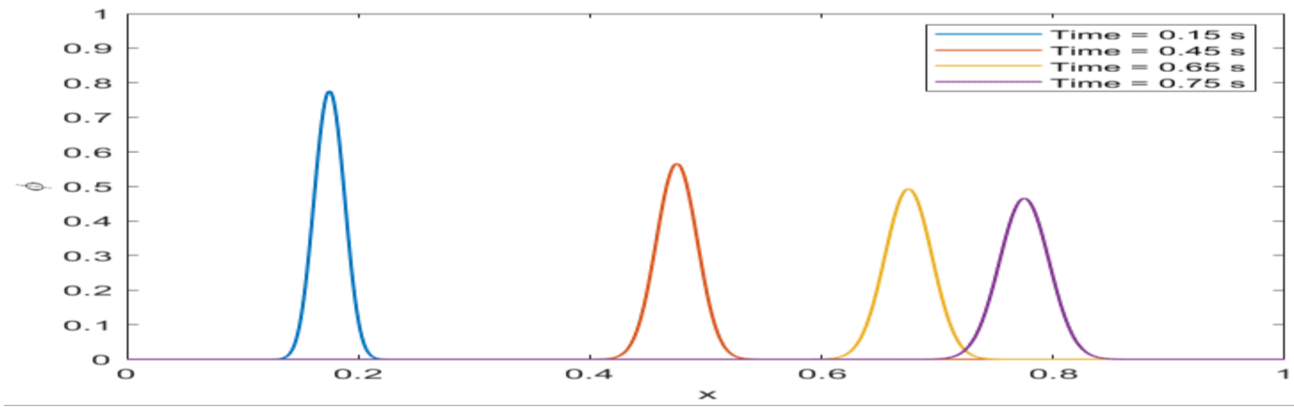


Figure 4: For CFL = 0.5

From the results obtained it is evident that, the solution become unstable if the CFL number is greater than 1 and will be conditionally stable if it is less than 1. The wave obtained satisfies the boundary conditions ass mentioned in the problem.

For the implicit upwind scheme, the results obtained (x axis displacement, axis phi) for different CFL values and time are:

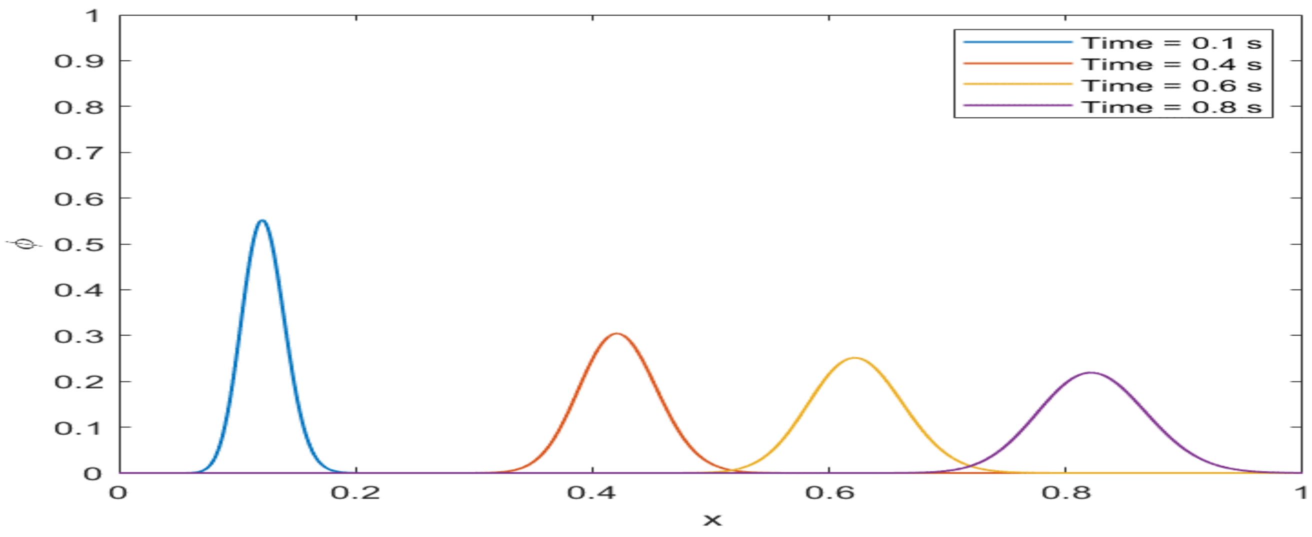


Figure 5: For CFL = 1.5

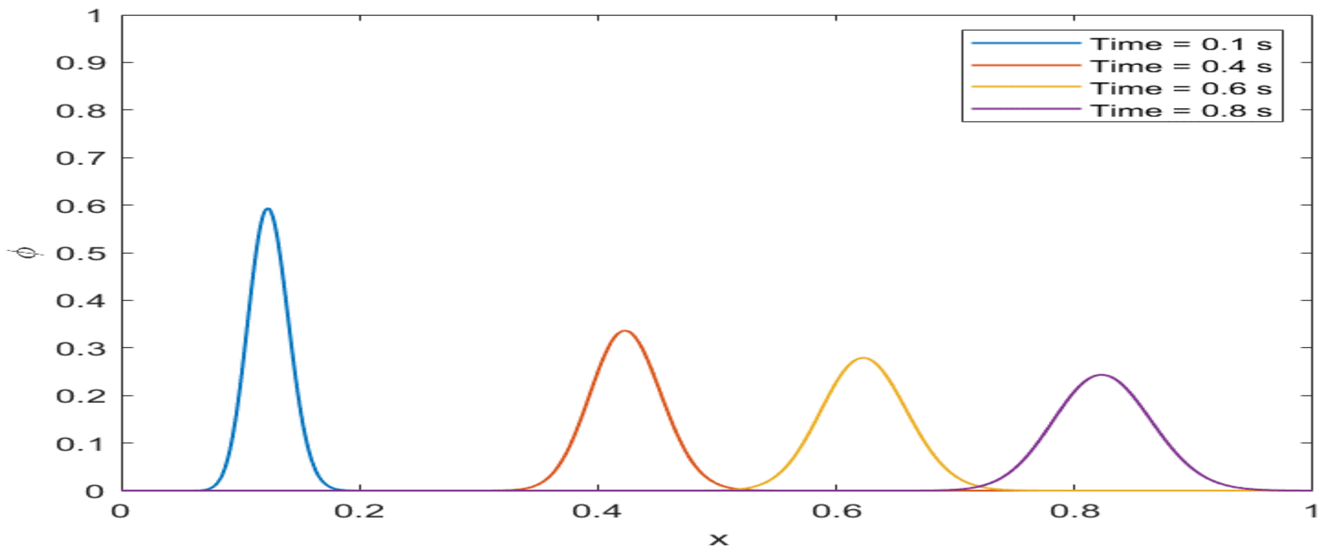


Figure 6: For CFL = 1

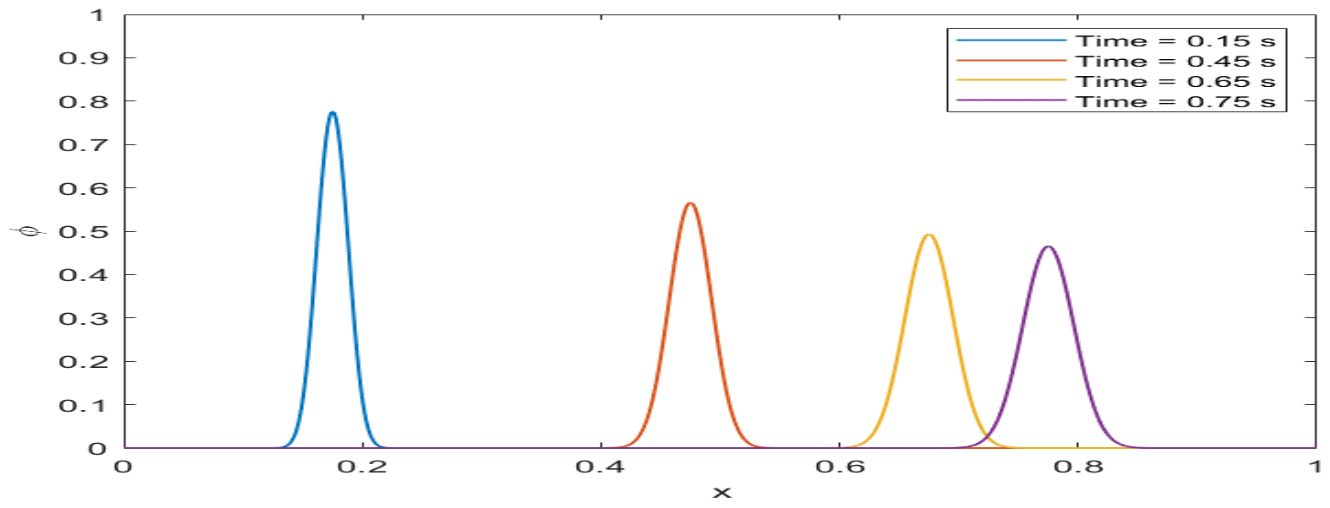


Figure 7: For CFL = 0.5

Implicit methods are numerical methods in which the time stepping is performed by solving a system of equations, typically through matrix inversion or iterative techniques. In contrast to explicit methods, implicit methods are unconditionally stable, which means that they do not have a CFL condition. However, the size of the time step used in implicit methods can still affect the accuracy and efficiency of the solution. If the time step used in an implicit method is too large (i.e., the CFL number is much greater than 1), the solution may become unstable and diverge. In other words, the numerical scheme may not be able to keep up with the changes in the system, leading to numerical errors and loss of accuracy. On the other hand, if the time step is too small (i.e., the CFL number is much less than 1), the solution may be unnecessarily slow and computationally expensive, without providing significant improvements in accuracy as in figure (5, 6, 7).

7.DISCUSSION:

The propagating wave form is obtained for both implicit and explicit methods. From the results it is very clear that the CFL value is the one that is controlling the stability of the wave. For explicit method if the CFL≤ 1 , the wave will be stable if CFL ≥ 1 the wave become unstable. In case of implicit method, it doesn't matter if the CFL value is less than or equal to one, it will be unconditionally stable.

Depending on the results obtained   we have found the following discussion topics:

1)   Influence of number of nodes on the solution.

In the explicit method, the solution at each time step is determined using only the previous time step's values. The stability of the scheme is dependent on the Courant-Friedrichs-Lewy (CFL) condition, which relates the time step size to the spatial grid size and solution velocity. In particular, the CFL condition requires that the time step size be chosen to ensure that information is not propagated across more than one spatial grid cell per time step. Consequently, as the number of nodes increases, both the spatial grid size and time step size must decrease to satisfy the CFL condition. This can result in longer computation times, as additional time steps are required to achieve the same final time.

The implicit scheme calculates the solution at each time step using a system of equations that incorporates values from both the current and previous time steps. Even for larger time step sizes, this can result in more precise and stable solutions. Since the system of equations must be solved at each time step, the computation time required for the implicit scheme may be significantly longer than that for the explicit scheme. Consequently, as the number of nodes increases, so too does the size of the system of equations, resulting in longer computation times.

Overall, the choice of the number of nodes in a finite difference scheme is a trade-off between accuracy, stability, and computational efficiency. A larger number of nodes can lead to more accurate solutions, but can also result in longer computation times for both explicit and implicit schemes.

2) Speed of calculation.

The speed of calculation in implicit and explicit 1D scalar upwind finite difference schemes can differ significantly, depending on several factors.

In the explicit method, the solution at each time step is determined using only the previous time step's values. Since the solution is computed using a straightforward formula, this makes the explicit scheme computationally faster than the implicit scheme. However, the time step size in the explicit scheme must be sufficiently small to satisfy the CFL condition, which can result in a large number of time steps being required to reach a given final time. Therefore, the computation time for the explicit scheme can significantly.

The implicit scheme calculates the solution at each time step using a system of equations that incorporates values from both the current and previous time steps. Each time step requires finding the solution to the system of equations, which can be computationally intensive. The implicit scheme, however, is unconditionally stable and can tolerate larger time step sizes than the explicit scheme. This implies that the implicit scheme can require fewer time steps to reach a given final time, which can offset the additional computational cost of solving the system of equations at each time step..

In conclusion, the computational speed of implicit and explicit 1D scalar upwind finite difference schemes depends on the time step size, the number of nodes, and the final time required. The explicit scheme is computationally faster but requires smaller time steps, whereas the implicit scheme is slower but can accommodate larger time steps. The choice of algorithm is determined by the desired precision, stability, and computational efficiency.

8.CONCLUSION:

Methods that are implicit and those that are explicit each have their own set of benefits and drawbacks. The obtained results make it abundantly clear that CFL no plays a significant role in the stability of both methods. In order to obtain solutions that are more accurate, the number of time steps used is another factor that plays a significant role. In addition, taking shorter time steps can result in a more accurate solution being found. First and foremost, the problem's initial conditions as well as its boundary conditions need to be clearly defined in order to obtain the accurate and consistent results that are desired.

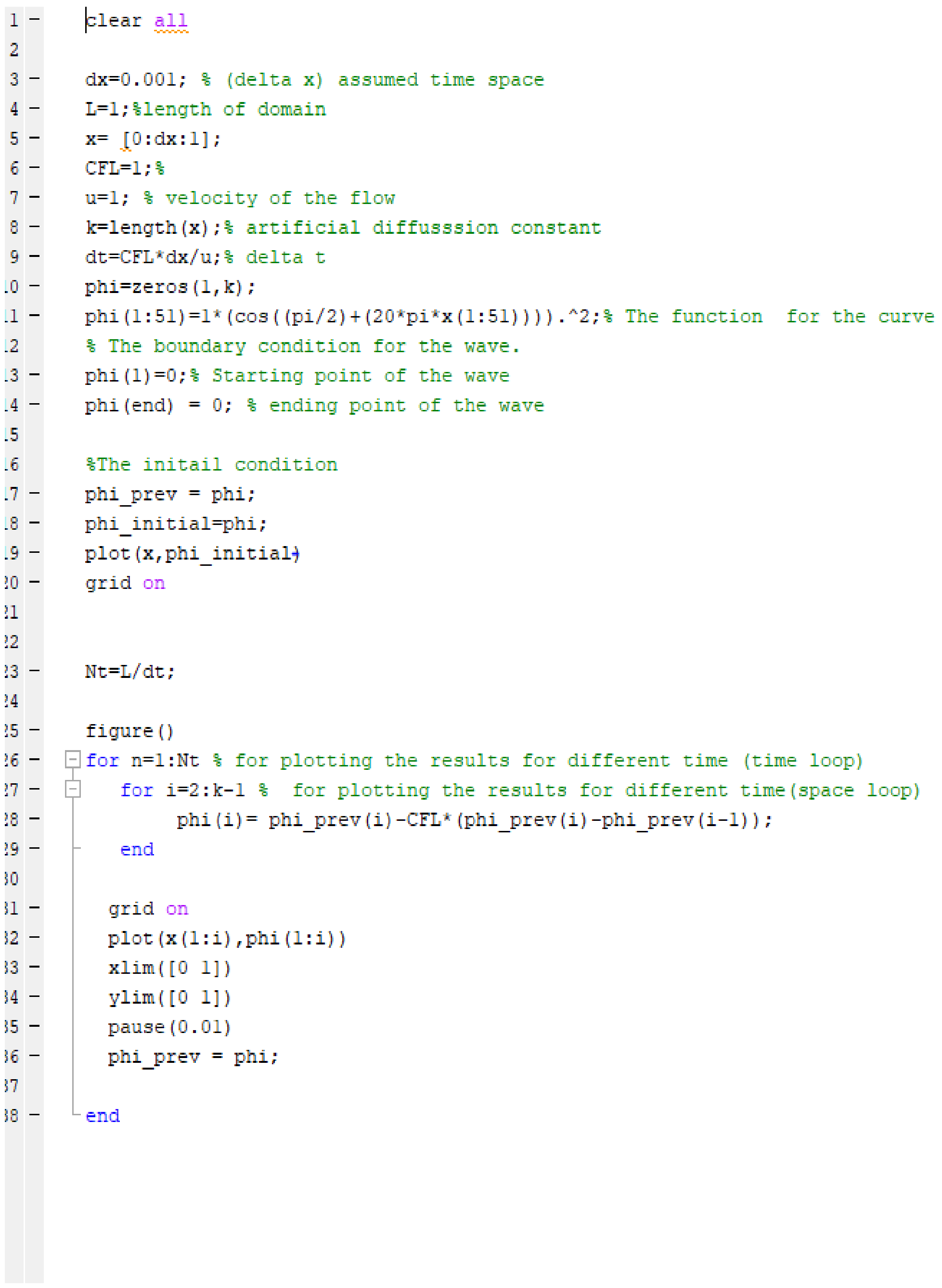
9.REFERENCES:

1) Computational fluid dynamics, P. Nithiarasu, Swansea University, lecture videos and notes

2) COMPUTATIONAL FLUID DYNAMICS 1 ,P. Nithiarasu Room 127, Engineering Central, Civil Engineering, Zienkiewicz Institute for Data, AI and Modelling College of Engineering, Swansea University, Swansea, SA2 8PP, United Kingdom.

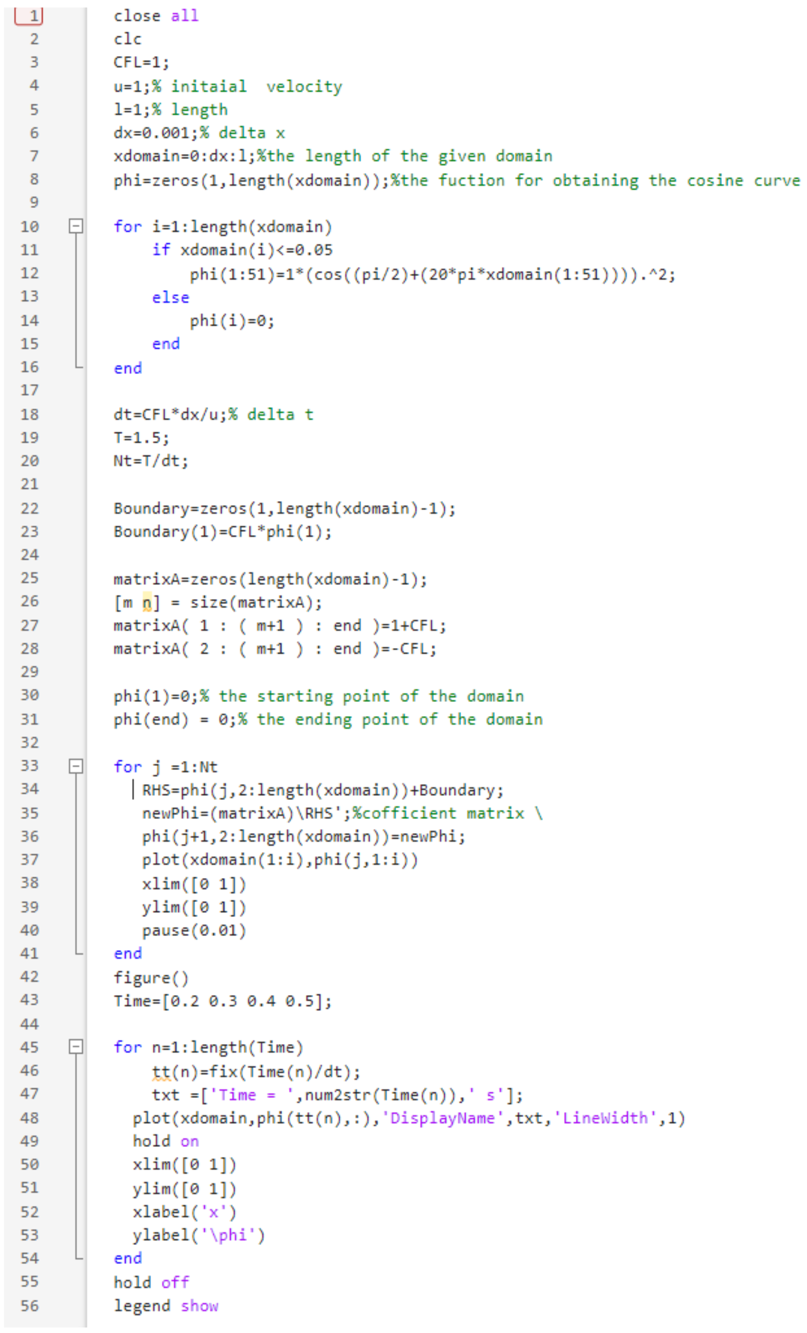
10.Appendix

The computer program for implicit and explicit methods are:



```
clear all

dx=0.001; % (delta x) assumed time space
L=1;%length of domain
x= [0:dx:1];
CFL=1;%
u=1; % velocity of the flow
k=length(x);% artificial diffusssion constant
dt=CFL*dx/u;% delta t
phi=zeros(1,k);
phi(1:51)=1*(cos((pi/2)+(20*pi*x(1:51)))).^2;% The function  for the curve
% The boundary condition for the wave.
phi(1)=0;% Starting point of the wave
phi(end) = 0; % ending point of the wave

%The initail condition
phi_prev = phi;
phi_initial=phi;
plot(x,phi_initial)
grid on


Nt=L/dt;

figure()
for n=1:Nt % for plotting the results for different time (time loop)
   for i=2:k-1 %  for plotting the results for different time(space loop)
        phi(i)= phi_prev(i)-CFL*(phi_prev(i)-phi_prev(i-1));
   end

  grid on
  plot(x(1:i),phi(1:i))
  xlim([0 1])
  ylim([0 1])
  pause(0.01)
  phi_prev = phi;

end
```

Figure 8: Explicit method MATLAB program.



```matlab
close all
clc
CFL=1;
u=1;% initaial  velocity
l=1;% length
dx=0.001;% delta x
xdomain=0:dx:l;%the length of the given domain
phi=zeros(1,length(xdomain));%the fuction for obtaining the cosine curve

for i=1:length(xdomain)
    if xdomain(i)<=0.05
        phi(1:51)=1*(cos((pi/2)+(20*pi*xdomain(1:51)))).^2;
    else
        phi(i)=0;
    end
end

dt=CFL*dx/u;% delta t
T=1.5;
Nt=T/dt;

Boundary=zeros(1,length(xdomain)-1);
Boundary(1)=CFL*phi(1);

matrixA=zeros(length(xdomain)-1);
[m n] = size(matrixA);
matrixA( 1 : ( m+1 ) : end )=1+CFL;
matrixA( 2 : ( m+1 ) : end )=-CFL;

phi(1)=0;% the starting point of the domain
phi(end) = 0;% the ending point of the domain

for j =1:Nt
   RHS=phi(j,2:length(xdomain))+Boundary;
   newPhi=(matrixA)\RHS';%cofficient matrix \
   phi(j+1,2:length(xdomain))=newPhi;
   plot(xdomain(1:i),phi(j,1:i))
   xlim([0 1])
   ylim([0 1])
   pause(0.01)
end
figure()
Time=[0.2 0.3 0.4 0.5];

for n=1:length(Time)
    tt(n)=fix(Time(n)/dt);
    txt =['Time = ',num2str(Time(n)),' s'];
  plot(xdomain,phi(tt(n),:),'DisplayName',txt,'LineWidth',1)
  hold on
  xlim([0 1])
  ylim([0 1])
  xlabel('x')
  ylabel('\phi')
end
hold off
legend show
```

Figure 9: Implicit method MATLAB program.