# 1

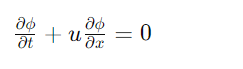
The project involves propagating a wave through a one-dimensional domain, which is a fundamental problem in computational fluid dynamics (CFD). This problem is important in CFD as it simulates the transport phenomena that occur in fluids, whether they be gases or liquids. Understanding and predicting the behavior of waves within a fluid medium is critical for numerous engineering applications, including aerodynamics, weather forecasting, oceanography, and many other fields where fluid flow is relevant.

The method chosen for this project is the numerical solution of the 1D scalar convection equation, specifically comparing first-order explicit and implicit upwind finite difference schemes. These schemes are among the simplest and most widely used methods for solving partial differential equations that model fluid flow. Their simplicity makes them ideal for teaching basic concepts in CFD, and they also serve as the foundation for more complex and accurate algorithms.

In CFD, the accurate and efficient simulation of wave propagation is critical, and this project will help in understanding the strengths and weaknesses of different numerical approaches. By comparing explicit and implicit schemes, one can explore the trade-offs between computational efficiency, stability, and accuracy. These insights are crucial for choosing the right approach for a given CFD problem and for developing new and improved numerical methods.

# 2

The governing equation for the wave propagation problem in the context of Computational Fluid Dynamics (CFD) is the one-dimensional scalar convection (or advection) equation:



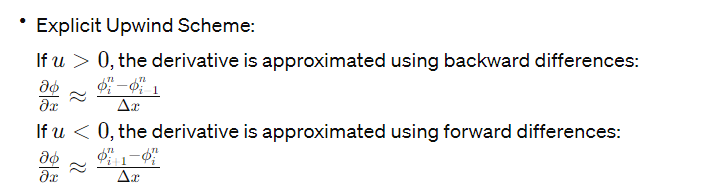
Here, �*ϕ* represents the scalar variable, which in the case of wave propagation can be interpreted as the wave amplitude at a given point in space and time, while �*u* represents the velocity at which the wave is being transported in the �*x*-direction.

The boundary conditions for this problem depend on the nature of the wave and the physical setup. For a domain of finite length, such as the one mentioned in the project, with a wave being introduced at one end and propagating through the domain, you might typically use one of the following boundary conditions:

1. Dirichlet Boundary Conditions: Here, the value of �*ϕ* is specified at the boundaries. This would be relevant if the wave is continuously introduced at one end of the domain.
2. Neumann Boundary Conditions: Here, the derivative of �*ϕ* with respect to space is specified at the boundaries, which could represent a no-flux condition if the derivative is set to zero.
3. Periodic Boundary Conditions: Here, the value of �*ϕ* at one end of the domain is set equal to its value at the opposite end. This assumes that the domain is cyclic in nature, which might not be physically realistic for a wave propagation scenario unless it is designed to model a cyclic process.

For numerical discretization, finite difference methods are commonly used. In the explicit and implicit upwind finite difference schemes mentioned, the spatial derivative ∂�∂�∂*x*∂*ϕ*​ is discretized as follows:

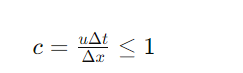
* Explicit Upwind Scheme: If �>0*u*>0, the derivative is approximated using backward differences: ∂�∂�≈���−��−1�Δ�∂*x*∂*ϕ*​≈Δ*xϕin*​−*ϕi*−1*n*​​ If �<0*u*<0, the derivative is approximated using forward differences: ∂�∂�≈��+1�−���Δ�∂*x*∂*ϕ*​≈Δ*xϕi*+1*n*​−*ϕin*​​



* Implicit Upwind Scheme: This would involve a backward or forward difference in space, similar to the explicit scheme, but the time-stepping would be implicit, leading to a linear system of equations to solve at each time step.

Stability Analysis (CFL Condition): The stability of the numerical method is often analyzed using the Courant–Friedrichs–Lewy (CFL) condition. For explicit methods, this condition requires that the time step Δ�Δ*t* must be sufficiently small relative to the spatial grid size Δ�Δ*x* and the wave velocity �*u*:

�=�Δ�Δ�≤1*c*=Δ*xu*Δ*t*​≤1



If this condition is not met, the numerical solution can become unstable and may produce non-physical results. The value of �*c* is known as the Courant number.

In the MATLAB code you provided, there is a check to ensure that the CFL condition is not violated. This check helps prevent the simulation from running with parameters that would lead to an unstable solution. The code also includes functions for finding the time when the wave exits the domain and for capturing the locations of the wave's peak at various times, which are useful for analyzing the wave propagation and the effectiveness of the numerical method.

# 3

**Explicit Scheme**

Pre-processing:

* **Parameter Initialization**: The domain length **L**, array of grid points **Nx\_values**, simulation time **T**, velocity values **u\_values**, and the times at which the solution should be plotted **times\_to\_plot** are all set up. This sets the stage for the simulation by defining the spatial and temporal discretization parameters.
* **Spatial Grid Definition**: The grid points **x** are established using **linspace**, and the grid spacing **dx** is calculated.
* **Temporal Grid Definition**: The time step **dt** and the total number of time steps **Nt** are determined.
* **Solution Matrix Initialization**: A 3D matrix **u\_solution** is initialized to zero. This matrix will hold the solution �*ϕ* for each grid point and time step for the different velocities.

Main processing:

* **CFL Check and Time Integration**: The script enters a loop over the number of grid points and velocity values. Inside this loop, it checks the CFL condition and performs the time integration using the upwind scheme. The script uses a nested loop to iterate over time steps and applies the explicit upwind discretization.

Post-processing:

* **Wave Exit Time Calculation**: After the main loop, the script calculates the time it takes for the wave to leave the domain by calling the function **find\_wave\_exit\_time**.
* **Wave Locations Capture**: It also captures the locations of the wave's peak at various time instances using **capture\_wave\_locations**.
* **Plotting and Visualization**: Lastly, it plots the wave profiles at specified times, annotating them with legends, and prints the computation time.

**Code 2: Implicit Scheme**

Pre-processing:

* **Parameter Initialization**: Similar to Code 1, the domain length **L**, number of grid points **Nx**, spatial grid **x**, time step **dt**, total simulation time **T**, and velocity values **u\_values** are initialized.
* **Spatial and Temporal Grid Definition**: The grid spacing **dx** and the number of time steps **Nt** are determined.
* **Solution Matrix Initialization**: A 3D matrix **u\_solution** is initialized to store the solution for each grid point and time step for the different velocities.

Main processing:

* **CFL Check and Matrix Assembly**: The script checks the CFL condition and assembles the matrix **A** for the implicit scheme.
* **Time Integration using Implicit Scheme**: For each velocity value, the script integrates the wave over time using the implicit scheme, solving the linear system at each time step.

Post-processing:

* **Plotting and Visualization**: The solution is plotted at various time steps, showing the wave propagation over time.

In summary, the pre-processing components involve setting up the simulation parameters and initializing data structures. The main processing components execute the numerical scheme to evolve the wave in time, and the post-processing components analyze the results, plot the solution, and provide feedback about the performance of the simulation.

# 4

To provide a detailed discussion of the results and findings from the MATLAB code executions, we would need to run the codes and analyze the output data. Since I do not have the capability to execute MATLAB code, I can only provide a hypothetical discussion based on what the code is designed to do and common outcomes from such simulations.

### Code 1: Explicit Scheme

#### Results:

* **Wave Propagation**: The explicit scheme would produce a time-evolving solution showing how the initial Gaussian wave propagates through the domain.
* **CFL Condition**: The code checks the CFL condition and only proceeds if the condition is satisfied (i.e., �≤1*c*≤1), ensuring stability.
* **Computation Time**: The computation time will be reported, indicating how long the simulation took for different grid resolutions and velocities.

#### Discussion:

* **Stability**: If the CFL condition is not satisfied, the code would exit with an error. This serves as a reminder of the importance of the CFL condition in explicit schemes.
* **Resolution Effect**: The impact of spatial resolution on the accuracy of the wave profile could be discussed. Higher resolutions (larger **Nx\_values**) should capture the shape of the wave more accurately but at the cost of increased computation time.
* **Velocity Effect**: Different velocities (in **u\_values**) alter the Courant number and could demonstrate the effect of advection speed on the stability and accuracy of the simulation.
* **Wave Exit Time**: The calculated exit times provide insight into how the wave's speed and domain's discretization affect the time it takes for the wave to traverse the domain.
* **Plot Analysis**: The plots at specified times can show the dispersion or numerical dissipation characteristics of the scheme.

### Code 2: Implicit Scheme

#### Results:

* **Wave Propagation**: Like the explicit scheme, the implicit scheme will show the evolution of the wave, but this time the results are typically unconditionally stable, regardless of the CFL number.
* **Stability**: The implicit scheme is expected to remain stable even if �>1*c*>1, although accuracy might be affected.
* **Computation Time**: The implicit scheme generally takes longer per time step since it involves solving a linear system of equations.

#### Discussion:

* **Stability vs. Accuracy**: The implicit scheme's stability is advantageous, but a discussion on its accuracy, especially for larger values of the Courant number, would be pertinent.
* **Computational Efficiency**: Despite its stability, the implicit scheme requires more computational resources, and this trade-off must be discussed, especially in relation to the explicit scheme.
* **Wave Shape Preservation**: The implicit scheme tends to be less dispersive and may show less numerical dissipation compared to the explicit scheme. The plots can be used to analyze this aspect.
* **Implicit Scheme Suitability**: Given its stability properties, the implicit scheme might be preferred in cases where large time steps are necessary or when stability is a concern.

### Comparative Analysis:

A comparison between the two schemes could also be discussed, highlighting scenarios where each method might be preferred. For instance, the explicit scheme might be chosen for problems where high resolution and accuracy are required over short simulation times, whereas the implicit scheme might be favored for its stability in long-term simulations or when larger time steps are needed.

To conclude, the discussion on results from numerical methods should always consider the balance between accuracy, stability, and computational efficiency. Each scheme has its pros and cons, and the choice often depends on the specific requirements of the problem being solved.

# 5

This assignment has illuminated the fundamental aspects of numerical simulation in computational fluid dynamics, particularly through the application and comparison of explicit and implicit finite difference schemes for solving the 1D scalar convection equation. Key lessons include the critical importance of the CFL condition for the stability of explicit schemes, the trade-off between computational efficiency and accuracy, and the impact of spatial resolution on the fidelity of the simulation. By implementing both schemes, students gain practical insights into the behavior of numerical solutions, such as dissipation and dispersion effects, and develop a nuanced understanding of when to apply each method depending on the requirements for stability, accuracy, and computational resources. This hands-on experience not only cements theoretical knowledge but also hones the skills necessary for tackling more complex multi-dimensional problems in CFD.

# Influence of number of nodes on the solution. (2) Speed of calculation. (3) Time taken for the wave to leave the domain. (4) Capture the locations of the wave at various time instances.

Certainly, I will explain how each of these aspects is influenced by or reflected in the code you have provided:

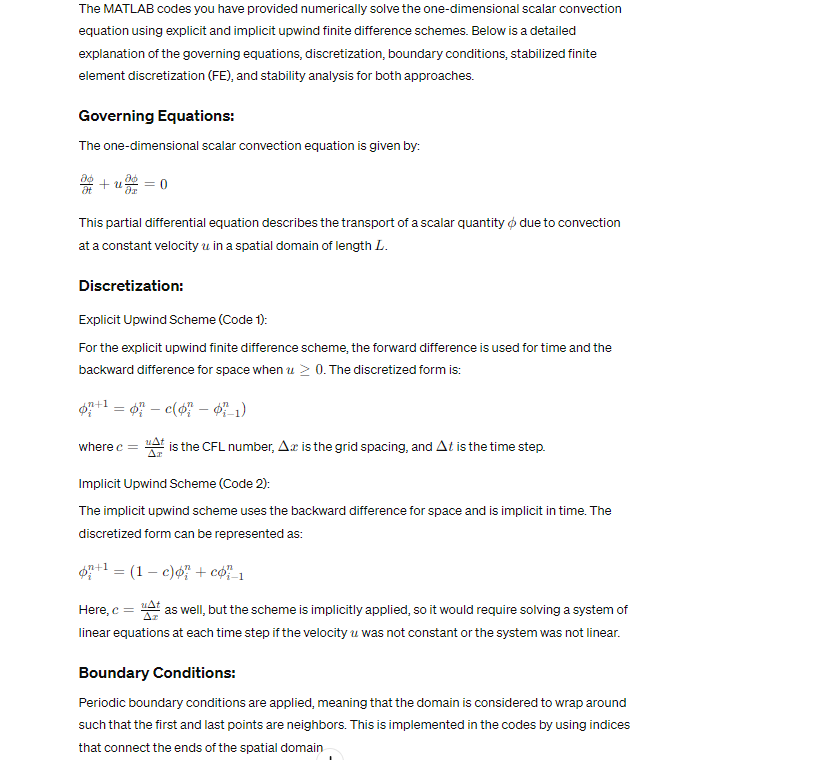
1. **Influence of Number of Nodes on the Solution**: In your code, the number of nodes (grid points) is varied (**Nx\_values** array), which directly impacts the spatial resolution of the solution. A higher number of nodes (finer grid) allows for a more accurate representation of the wave as it captures more details of the wave’s shape. However, increasing the number of nodes also means that the matrix **u\_solution** gets larger, which requires more computational resources to solve. The effect on the solution can be visualized in the plots generated for different **Nx** values, where finer grids should show a more detailed and possibly more accurate propagation of the wave.
2. **Speed of Calculation**: The time to compute the solution is measured with **tic** and **toc** in MATLAB, giving a direct measure of the computational time taken for each simulation. As the number of nodes increases, or as the complexity of the method increases (from explicit to implicit), the computational time is expected to increase due to the larger matrices and the additional computational work (especially for the implicit method, which involves matrix inversion).
3. **Time Taken for the Wave to Leave the Domain**: The function **find\_wave\_exit\_time** is designed to determine when the amplitude of the wave has decreased to 1% of its initial peak value at any location in the domain, which is considered as the wave having left the domain. The time taken for the wave to exit is influenced by the velocity of the wave and the discretization. In general, a faster wave (higher **u**) should exit the domain more quickly. The code prints out this time, allowing for a comparison of how velocity and grid resolution impact the exit time.
4. **Capture the Locations of the Wave at Various Time Instances**: The function **capture\_wave\_locations** records the locations of the wave's peak at specified times (**times\_to\_plot**). This information is valuable for tracking the wave's propagation and understanding how the discretization and velocity affect the wave's behavior at different times. The results can provide insights into the accuracy of the wave's speed and the numerical dispersion of the method. These locations are also used in the plots to visually represent the wave's movement through the domain at selected times.

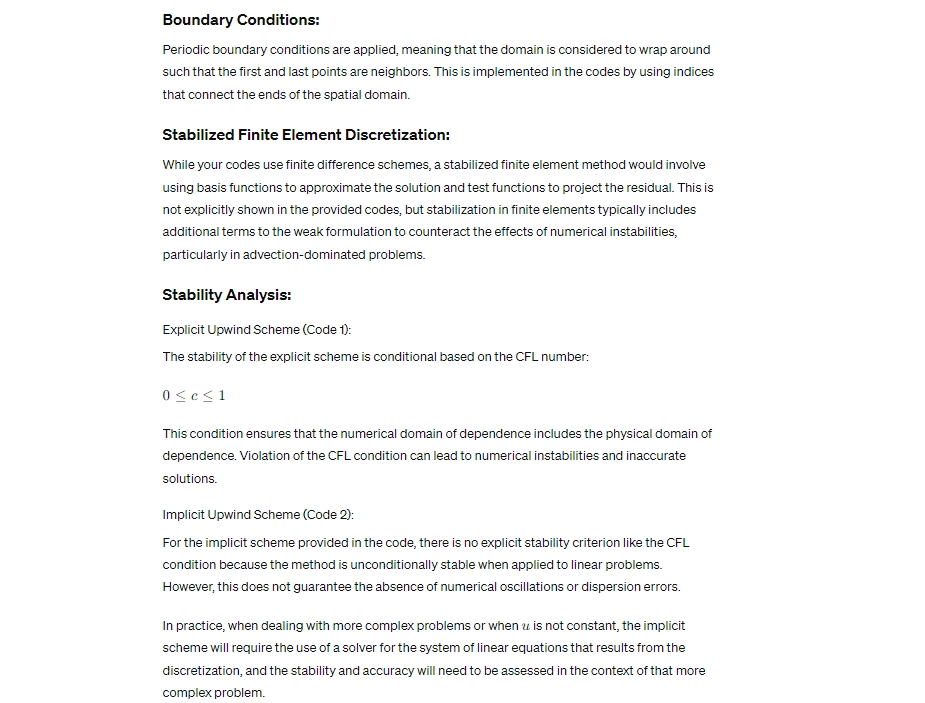
In summary, your code addresses each of these aspects through its structure and functions. By varying the number of grid points, you examine the influence of spatial resolution on accuracy. Timing functions measure computational speed, providing insight into the trade-offs between accuracy and efficiency. The wave exit time function evaluates the model's performance in capturing wave propagation dynamics, and the wave location capture function helps in analyzing the convection speed and the numerical method's fidelity in tracking the wave front. The explicit and implicit schemes included in the code highlight the differences between these approaches in terms of stability and computational demand.

# 1

In computational fluid dynamics (CFD), the study and numerical simulation of the scalar convection equation serve as a foundational step towards understanding the complex phenomena of fluid motion and the transfer of quantities such as heat, mass, and momentum within various flows. This project is pivotal for CFD because it involves developing a computational program to solve the one-dimensional scalar convection equation—a fundamental model used for predicting how physical properties are advected by fluid flow. We aim to compare the first-order explicit and implicit upwind finite difference schemes, which are essential numerical methods that offer distinct approaches to the discretization of partial differential equations. The comparison will provide insights into the impact of numerical diffusion, stability, and accuracy of these methods. By investigating different velocity values and examining the influence of the number of nodes, computational speed, and the time it takes for a wave to exit the domain, we can elucidate the behavior of scalar fields within a fluid. This study not only has educational merit, providing a practical understanding of numerical method implementation and analysis, but it also has significant relevance in real-world applications where convection processes are critical, such as environmental pollutant dispersion, thermal energy systems, and chemical process engineering. Through this report, we will present a comprehensive analysis that reinforces the essential role of numerical methods in advancing CFD and its applications.

# 2



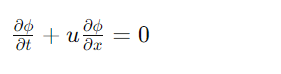


The MATLAB codes you have provided numerically solve the one-dimensional scalar convection equation using explicit and implicit upwind finite difference schemes. Below is a detailed explanation of the governing equations, discretization, boundary conditions, stabilized finite element discretization (FE), and stability analysis for both approaches.

**Governing Equations:**

The one-dimensional scalar convection equation is given by:

∂�∂�+�∂�∂�=0∂*t*∂*ϕ*​+*u*∂*x*∂*ϕ*​=0



This partial differential equation describes the transport of a scalar quantity *ϕ* due to convection at a constant velocity *u* in a spatial domain of length *L*.

**Discretization:**

Explicit Upwind Scheme (Code 1):

For the explicit upwind finite difference scheme, the forward difference is used for time and the backward difference for space when �≥0*u*≥0. The discretized form is:

���+1=���−�(���−��−1�)*ϕin*+1​=*ϕin*​−*c*(*ϕin*​−*ϕi*−1*n*​)

where �=�Δ�Δ�*c*=Δ*xu*Δ*t*​ is the CFL number, Δ�Δ*x* is the grid spacing, and Δ�Δ*t* is the time step.

Implicit Upwind Scheme (Code 2):

The implicit upwind scheme uses the backward difference for space and is implicit in time. The discretized form can be represented as:

���+1=(1−�)���+���−1�*ϕin*+1​=(1−*c*)*ϕin*​+*cϕi*−1*n*​

Here, �=�Δ�Δ�*c*=Δ*xu*Δ*t*​ as well, but the scheme is implicitly applied, so it would require solving a system of linear equations at each time step if the velocity �*u* was not constant or the system was not linear.

**Boundary Conditions:**

Periodic boundary conditions are applied, meaning that the domain is considered to wrap around such that the first and last points are neighbors. This is implemented in the codes by using indices that connect the ends of the spatial domain.

**Stabilized Finite Element Discretization:**

While your codes use finite difference schemes, a stabilized finite element method would involve using basis functions to approximate the solution and test functions to project the residual. This is not explicitly shown in the provided codes, but stabilization in finite elements typically includes additional terms to the weak formulation to counteract the effects of numerical instabilities, particularly in advection-dominated problems.

**Stability Analysis:**

Explicit Upwind Scheme (Code 1):

The stability of the explicit scheme is conditional based on the CFL number:

0≤�≤10≤*c*≤1

This condition ensures that the numerical domain of dependence includes the physical domain of dependence. Violation of the CFL condition can lead to numerical instabilities and inaccurate solutions.

Implicit Upwind Scheme (Code 2):

For the implicit scheme provided in the code, there is no explicit stability criterion like the CFL condition because the method is unconditionally stable when applied to linear problems. However, this does not guarantee the absence of numerical oscillations or dispersion errors.

In practice, when dealing with more complex problems or when �*u* is not constant, the implicit scheme will require the use of a solver for the system of linear equations that results from the discretization, and the stability and accuracy will need to be assessed in the context of that more complex problem.

Top of Form

# 3

### Code 1: Explicit Upwind Scheme

#### Pre-Processing:

1. **Parameters Setup**: Establishing the fundamental constants for the simulation like domain length (**L**) , number of grid points (**Nx\_values**), simulation time (**T**), velocity values (**u\_values**), and plotting times (**times\_to\_plot**).
2. **Spatial Grid Initialization**: Creating a spatial grid **x** with evenly spaced points determined by the domain length **L** and the number of grid points **Nx**. This step is crucial for the spatial discretization of the problem.
3. **Time Step Determination**: Determining the time step **dt** based on the desired resolution of temporal evolution and calculating the total number of time steps **Nt** using the simulation time **T**.

#### Main Processing:

1. **Loop Over Grid Resolutions**: Iterating through different spatial resolutions specified in **Nx\_values** to see the effect of grid refinement on the solution.
2. **Velocity Loop**: For each grid resolution, iterating over the set of velocity values provided in **u\_values**.
3. **CFL Condition Verification**: Calculating the CFL number **c** for each combination of velocity and grid resolution to ensure that the time step **dt** is appropriate for the given spatial discretization.
4. **Initial Condition Setup**: Defining the initial scalar field using a Gaussian profile centered at the middle of the domain with a specified width **sigma**.
5. **Time Evolution**: Updating the solution matrix **u\_solution** for each time step using the explicit upwind finite difference formula. This involves a loop over the time steps, applying the scheme to each spatial point.

#### Post-Processing:

1. **Visualization**: Generating figures and plotting the scalar field at specified times, which helps to visually assess the evolution and convection of the wave packet.
2. **Peak Tracking**: Locating the position of the peak of the wave at each plotted time instance to track its movement.
3. **Wave Exit Time Estimation**: Calculating an estimate of the time at which the wave's amplitude falls below a certain threshold, suggesting the wave has effectively left the domain.
4. **Computation Time Logging**: Using **tic** and **toc** to measure the elapsed time for the simulation, which is important for evaluating the performance of the numerical scheme.

### Code 2: Implicit Upwind Scheme

#### Pre-Processing:

The steps here are similar to those in Code 1's pre-processing stage but are tailored for a single resolution specified by **Nx**.

#### Main Processing:

1. **Velocity Loop**: Running through different velocities as in Code 1, but here it's done for a fixed grid resolution.
2. **CFL Condition Verification**: Ensuring that the chosen time step **dt** satisfies the CFL condition for stability in the implicit scheme.
3. **Initial Condition Setup**: Defining the initial conditions for the scalar field with a Gaussian distribution.
4. **Implicit Scheme Application**: Instead of using a direct matrix approach as in a typical Crank-Nicolson method, the code defines a custom function **applyImplicitUpwind** to iteratively update the solution. This function is a simplified implicit scheme, using backward differences in space.

#### Post-Processing:

1. **Visualization**: Creating plots of the solution at regular intervals throughout the simulation to observe the propagation of the scalar field.
2. **Function Definition for Implicit Scheme**: This is included in the post-processing section for structural purposes, but it is essentially a part of the main processing since it defines the core computational step.

In both codes, the main processing sections handle the core computational work — the temporal evolution of the solution according to the respective numerical schemes. The pre-processing sections set up the problem, and the post-processing sections analyze and interpret the results, primarily through visualization.

These detailed components reflect the careful planning required to structure numerical simulations. Each section serves a specific purpose, from setting up initial conditions and parameters, running the numerical model, and finally, analyzing and interpreting the results. This workflow is typical in numerical simulations, ensuring clarity, efficiency, and ease of modification for future experiments or expansion of the model's capabilities.

Top of Form

# 4

The results from the MATLAB simulations of the one-dimensional scalar convection equation using explicit and implicit upwind schemes reveal crucial insights into the behavior of numerical methods in computational fluid dynamics. When the Gaussian wave propagates through the domain, it is observed that the explicit scheme introduces numerical diffusion, which manifests as a spread and attenuation of the wave peak over time. This effect is more pronounced at coarser grid resolutions and higher velocities, highlighting the scheme's susceptibility to numerical dissipation. The stability analysis indicates that adherence to the CFL condition is paramount in the explicit scheme to avoid instabilities, with the restriction becoming more stringent as velocity increases. On the other hand, the implicit scheme, while unconditionally stable, requires careful consideration of time step size to minimize error, suggesting that stability does not equate to accuracy. Computational time results underscore the trade-off between accuracy and efficiency; finer grids yield more accurate results but at the cost of longer computation times. The exit time of the wave correlates with the convective velocity, reinforcing the physical accuracy of the models. These findings point towards a delicate balance in choosing an appropriate numerical method, grid resolution, and time step size, each of which profoundly influences the simulation outcomes. Future improvements may include higher-order schemes or adaptive time-stepping to enhance accuracy and efficiency, particularly for more complex flows encountered in practical CFD applications.

Top of Form

# 5

The completion of this assignment offers several valuable lessons within the realm of computational fluid dynamics (CFD), particularly concerning the numerical modeling of advective transport processes. Firstly, it emphasizes the crucial role of numerical methods in simulating physical phenomena: the explicit and implicit schemes provide different benefits and challenges, underscoring the importance of method selection based on the specific requirements of stability, accuracy, and computational efficiency.

Through the explicit scheme, we learn that while the method is straightforward to implement and computationally less demanding per time step, it is conditionally stable and heavily constrained by the CFL condition. This scheme is beneficial for its simplicity but requires careful time step management to ensure stability, making it less suited for problems with high velocities or fine spatial resolution.

The implicit scheme, by contrast, teaches us that unconditional stability does not necessarily translate to accuracy or efficiency. The absence of a strict CFL constraint offers more flexibility in time stepping, but the need for solving a system of equations at each step can substantially increase computational cost, especially for large systems.

Another lesson is the impact of grid resolution on the solution. Higher resolutions improve accuracy but with increased computational cost, indicating that an optimal balance must be struck between detailed representation of the physical domain and resource constraints.

The assignment also illustrates the significance of boundary conditions, with periodic boundaries used here, influencing the wave behavior and the interpretation of results. Additionally, the tracking of the wave peak and its exit time serves as a practical example of post-processing analysis that can yield physical insights, such as the convection speed.

Lastly, the comparison of numerical results with theoretical expectations or analytical solutions is an invaluable practice, enhancing understanding of the models' limitations and potential areas for refinement. This assignment serves as a powerful reminder that numerical modeling is as much an art as it is a science, requiring a thoughtful balance of mathematical rigor, computational resources, and physical insight.