CFD Project 2

Submitted by:
Aaditya Nimkar 24250002
Vardan Mittal 24250104

Department of Mechanical Engineering

ME 605 Fall semester

1 Problem statement

Consider the flow of a viscous fluid over a flat plate of length L, as shown below. The fluid enters the simulation domain of height H at a uniform velocity, (U_∞). Assume that the flow is steady, laminar, and in-compressible. Further, assume that the Reynolds number is large enough such that the boundary layer approximation is valid. The resulting governing equations for this specific problem are given below:

The continuity equation is:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0$$

The momentum equation is:

$$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = \nu\frac{\partial^2 u}{\partial y^2}$$

where ν is the kinematic viscosity of the fluid. You are required to write a computer program to solve the above boundary layer equations for this problem for a Reynolds number

$$Re_L = 10^4$$

where,

$$Re_L = \frac{U_\infty L}{\nu}$$

The plate length (L), kinematic viscosity (ν), and free-stream velocity (U_∞) should be chosen such that $Re_L = 10^4$. Use the finite difference method for discretization of the PDEs and solve the PDEs computationally using the following schemes:

1. Euler Explicit scheme. You will have to choose Δx and Δy carefully, as the explicit scheme is not always stable.
2. Euler Implicit scheme.
3. Crank-Nicolson scheme.

For each of the above three schemes, you are required to:

1. Compute the x-velocity (u) and y-velocity (v) fields. Show velocity fields as contour plots. Further, plot the normalized x-velocity (F') as a function of the similarity variable (η) as a line plot and compare with the Blasius solution. The variables are defined below:

   $$F'(\eta) = \frac{u}{U_\infty}$$

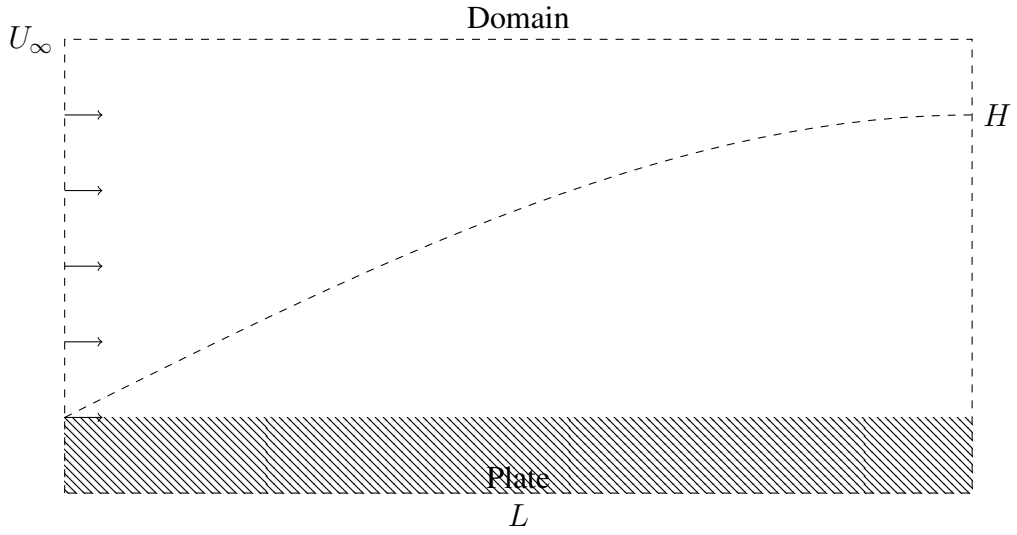
   and

   $$\eta = y\sqrt{\frac{U_\infty}{\nu x}}$$

   Note that you can vary the similarity variable (η) by varying the y coordinate at a specific x location or by varying the x coordinate at a specific y location. You may pick one of the two options to show the comparison between your simulation predictions and the Blasius solution.

2. Compute the boundary layer thickness. Plot the variation of boundary layer thickness with x coordinate. How does your simulation result compare with the Blasius flat plate boundary layer solution given below?

$$\delta(x) = \frac{4.91}{\sqrt{Re_x}}$$



2 Euler explicit scheme

2.1 Grid Structure

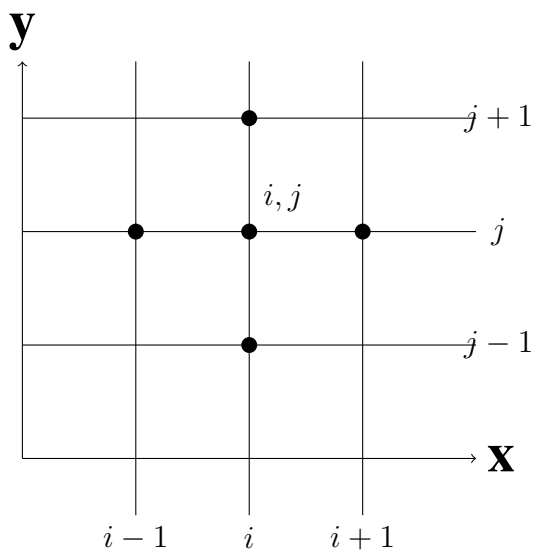


Figure 1: Schematic view of the grid

2.2 Continuity Equation

The continuity equation is:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0$$

This can be approximated as:

$$\left(\frac{\partial u}{\partial x}\right)_{i,j} + \left(\frac{\partial v}{\partial y}\right)_{i,j} = 0$$

Using finite differences, we can express this as:

$$\frac{u_{i+1,j} - u_{i,j}}{\Delta x} + \frac{v_{i,j+1} - v_{i,j-1}}{2\Delta y} = 0$$

From the continuity equation, we can rearrange to solve for v:

$$v_{i+1,j} = v_{i+1,j-1} - \frac{\Delta y}{\Delta x}(u_{i+1,j} - u_{i,j})$$

2.3 Momentum Equation

The momentum equation is:

$$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = \nu\frac{\partial^2 u}{\partial y^2}$$

Discretizing the convection and diffusion terms:

$$u\frac{\partial u}{\partial x} \approx u_{i,j}\frac{u_{i+1,j} - u_{i,j}}{\Delta x}$$

$$v\frac{\partial u}{\partial y} \approx v_{i,j}\frac{u_{i,j+1} - u_{i,j-1}}{2\Delta y}$$

$$\nu\frac{\partial^2 u}{\partial y^2} \approx \nu\frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{(\Delta y)^2}$$

Combining these into the discretized momentum equation:

$$u_{i,j}\frac{u_{i+1,j} - u_{i,j}}{\Delta x} + v_{i,j}\frac{u_{i,j+1} - u_{i,j-1}}{2\Delta y} = \nu\frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta y^2}$$

Thus, the final discretized equation for updating $u_{i,j}$ is:

$$u[i,j] = u_n[j] + \Delta x\left(-v[i,j]\frac{u_n[j+1] - u_n[j-1]}{2\Delta y} + \nu\frac{u_n[j+1] - 2u_n[j] + u_n[j-1]}{\Delta y^2}\right)$$

Where:

- $u_n[j]$ is the previous step value of u at the j-th grid point.
- Δx is the grid spacing in the x-direction.
- Δy is the grid spacing in the y-direction.
- ν is the kinematic viscosity.

2.4 Key Aspects of the Explicit Method in this problem:

The Euler Explicit Method is a numerical technique for solving partial differential equations (PDEs) by discretizing the spatial derivatives. In solving the viscous flow over a flat plate (boundary layer flow), the method is applied to the governing momentum and continuity equations using finite differences.

1. **Discretization:** The governing equations are discretized over a grid that covers the plate surface. Finite difference approximations replace spatial derivatives (convective and diffusive terms). This allows the flow variables (e.g., velocity) at each grid point to be updated based on neighbouring points.

2. **Spatial Marching:** Since the problem is steady-state, there is no time dependence. The explicit method marches spatially along the plate in the x-direction, updating the velocity profile in the y-direction at each step.

3. **Convective and Diffusive Terms:** - The convective terms (derivatives of u with respect to x and y) are handled with forward finite differences. - The diffusive term (second derivative of u with respect to y) is treated using central finite differences, representing the viscous effects in the boundary layer.

4. **Iteration and Convergence:** The velocity field is iterated upon, updating $u(x, y)$ at each grid point until convergence is reached (i.e. when the difference between successive iterations is sufficiently small).

5. **Boundary Conditions:** - At the surface of the plate: $u = 0$ (no-slip condition). - At the far-field boundary layer edge: $u = U_\infty$ (free-stream velocity).

6. **Stability criteria**: CFL restriction exists in the spatial domain to ensure stability in the boundary layer flow. The step sizes must satisfy a stability criterion depending on the viscosity and flow velocity

$$\Delta x \leq \frac{\Delta y^2}{2\nu}$$

7. **Advantages and Disadvantages:**

   (a) Advantages: The explicit method is simple to implement and requires minimal computational resources since each update depends only on known values at neighbouring points.

   (b) Disadvantages: The method can be unstable for large step sizes (in x or y), requiring careful grid spacing. It may also converge slowly compared to implicit methods.

2.5 Algorithm

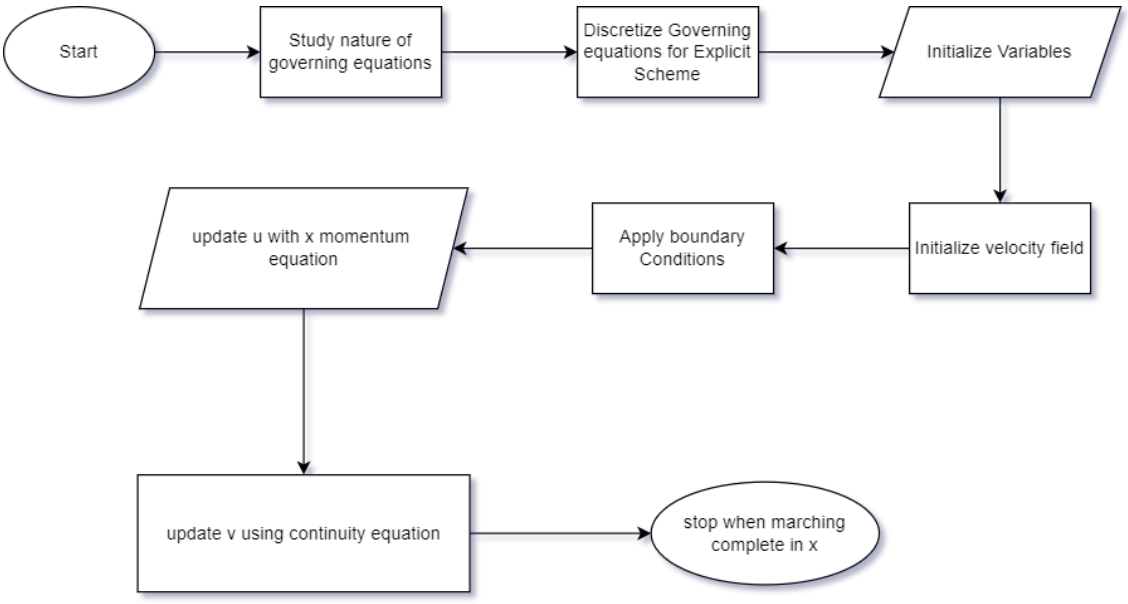


Figure 2: Explicit Scheme Algorithm

2.6 Parameters and Variables

- **Geometry and Flow Parameters:**
  - L_x, L_y: Dimensions of the computational domain in the x and y directions, respectively. Here, $L_x = 1.0$ (length) and $L_y = 0.1$ (height).
  - Re: Reynolds number, set to 1×10^4, which characterizes the flow regime.
  - U_∞: Free stream velocity, set to 1.0, representing the speed of the flow entering the domain.
- **Grid Settings:**
  - n_x, n_y: Number of grid points in the x and y directions. Both are set to 100, creating a grid of 100×100 points.
  - ν: Kinematic viscosity, calculated as $\nu = \frac{U_\infty \cdot L_x}{Re}$.
- **Grid Points:**
  - x: Linearly spaced array from 0 to L_x with n_x points, representing the x-coordinates.
  - y: Linearly spaced array from 0 to L_y with n_y points, representing the y-coordinates.
- **Velocity Fields:**
  - u: A 2D array initialized to U_∞, representing the x-component of velocity across the grid.
  - v: A 2D array initialized to 0, representing the y-component of velocity across the grid.

2.7 Function: *EulerExplicitScheme*

This function implements the explicit Euler time-stepping scheme for solving the Navier-Stokes equations in the context of incompressible flow.

- **Grid Spacing:**
  - dy: Grid spacing in the y-direction, calculated as $dy = \frac{L_y}{n_y - 1}$.
  - dx: Grid spacing in the x-direction, calculated as $dx = 0.9 \cdot \frac{dy^2}{2\nu}$. This ensures stability for the numerical scheme.
- **Boundary Conditions:**
  - $u[:, 0] = 0$: Applies the no-slip condition at the bottom wall ($y = 0$).
  - $u[:, -1] = U_\infty$: Sets the inlet condition at the top edge ($y = L_y$).
  - $v[:, 0] = 0$: No penetration condition at the bottom wall.
  - $v[0, :] = 0$: No penetration condition at the leading edge ($x = 0$).
- **Main Loop:**
  - The outer loop iterates over the x-direction (i from 1 to $n_x - 1$), effectively stepping through time.
  - $un = u[i-1, :].copy()$: Copies the velocity field from the previous time step for computation.
    The inner loop iterates through the y-direction (j from 1 to $n_y - 2$) to update the velocity components:
    - **Updating the x-velocity ($u[i, j]$):**
      - Uses the explicit Euler method, calculating the change in velocity based on advection and diffusion terms.
      - The advection term accounts for the effect of the y-velocity on the x-velocity.
      - The diffusion term represents the viscous effects.
    - **Updating the y-velocity ($v[i, j]$):**
      - Similar to the x-velocity update but considers the change in the x-velocity between time steps.

2.8 Results

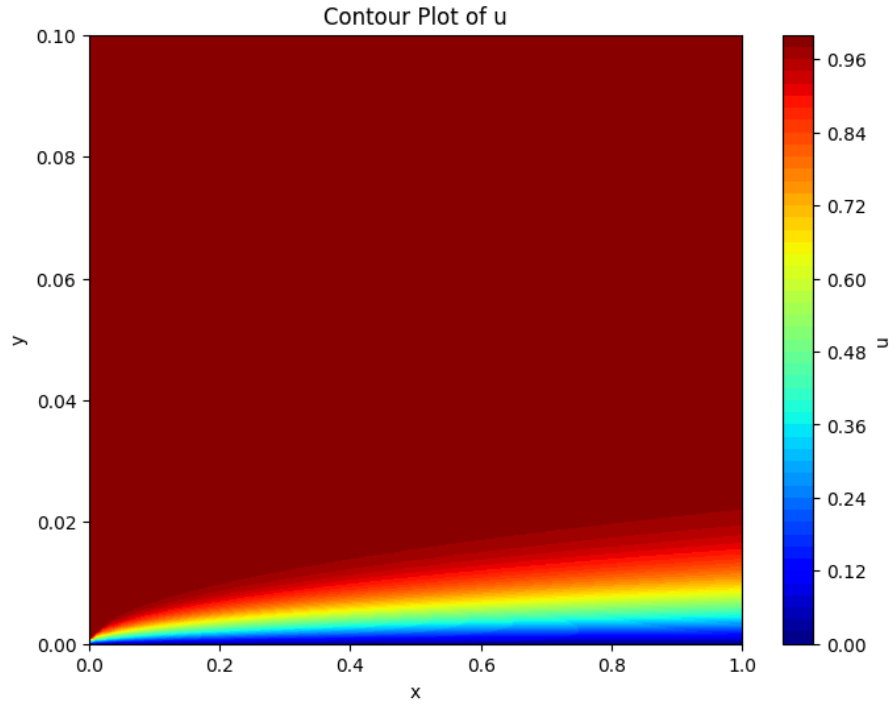


Figure 3: Explicit Scheme Velocity Contour

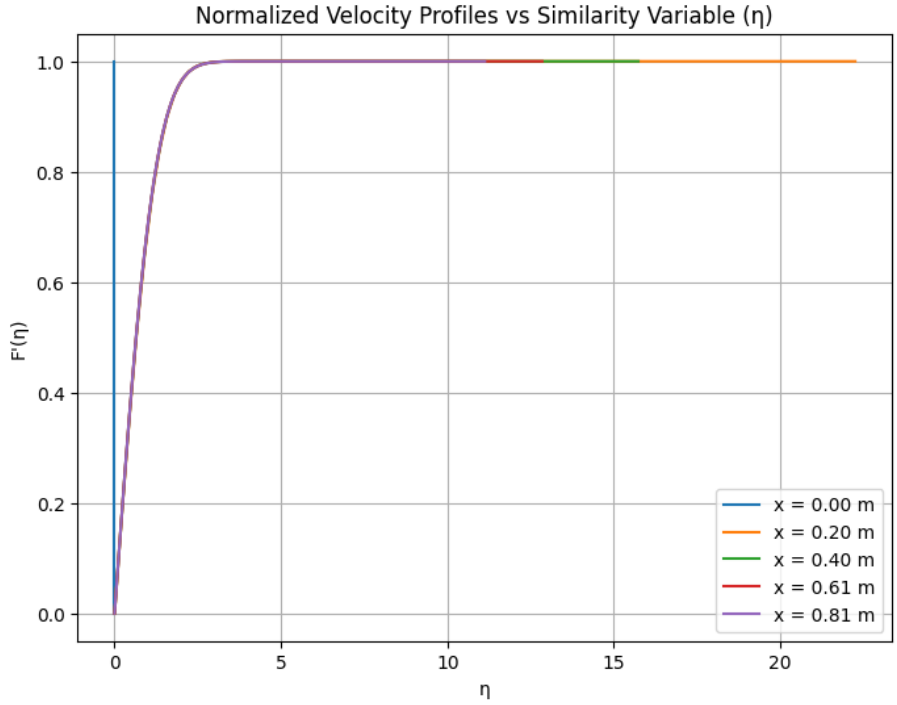


Figure 4: Explicit Scheme Similarity Profile

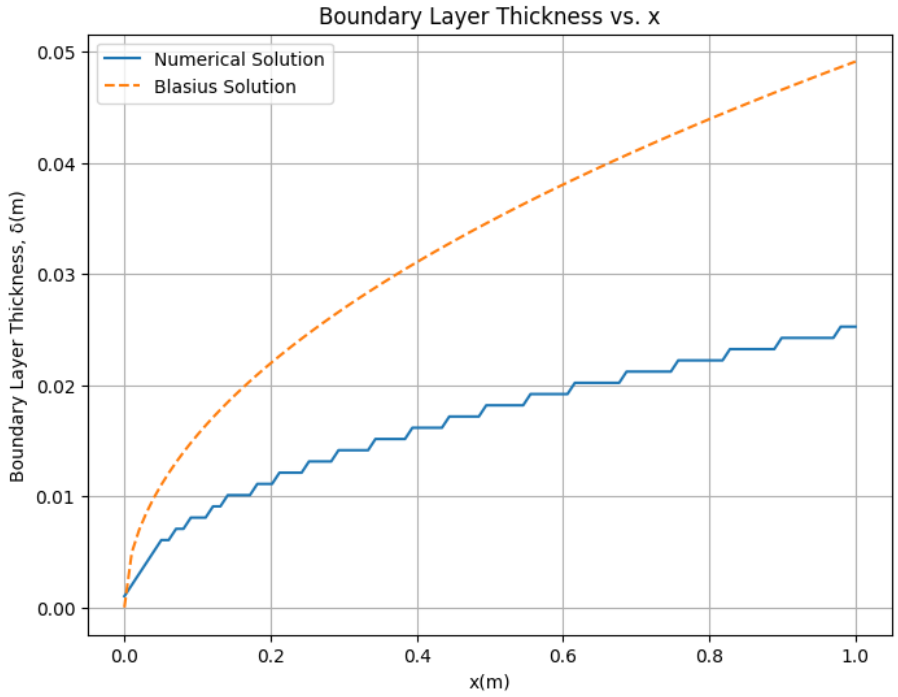


Figure 5: Explicit Scheme Boundary Layer Thickness

3 Euler Implicit Scheme

The governing momentum equation for the viscous flow over a flat plate is:

$$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = \nu\frac{\partial^2 u}{\partial y^2} \tag{1}$$

We discretise the spatial derivatives using finite difference approximations to solve this equation numerically using the Euler Implicit Scheme.

3.1 Discretization

For the discretization, we use backward differencing for the first derivative in the x-direction and central differencing for the second derivative in the y-direction.

Discretization in the x-direction: The first derivative $\frac{\partial u}{\partial x}$ is approximated using the backward difference formula:

$$\frac{\partial u}{\partial x} \approx \frac{u_{i,j} - u_{i-1,j}}{\Delta x} \tag{2}$$

Discretization in the y-direction: The first derivative $\frac{\partial u}{\partial y}$ is approximated as:

$$\frac{\partial u}{\partial y} \approx \frac{u_{i,j} - u_{i,j-1}}{\Delta y} \tag{3}$$

For the second derivative $\frac{\partial^2 u}{\partial y^2}$, we apply central differencing:

$$\frac{\partial^2 u}{\partial y^2} \approx \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta y^2} \tag{4}$$

3.2 Implicit Euler Discretization

We use implicit discretization for the temporal terms in the Euler Implicit Scheme.

The continuity equation at each time step is discretized as:

$$\frac{u_{i,j}^{n+1} - u_{i-1,j}^{n+1}}{\Delta x} + \frac{v_{i,j+1}^{n+1} - v_{i,j}^{n+1}}{\Delta y} = 0 \tag{5}$$

The momentum equation at each time step is discretized as:

$$u_{i,j}^{n+1}\frac{u_{i,j}^{n+1} - u_{i-1,j}^{n+1}}{\Delta x} + v_{i,j}^{n+1}\frac{u_{i,j}^{n+1} - u_{i,j-1}^{n+1}}{\Delta y} = \nu\frac{u_{i,j+1}^{n+1} - 2u_{i,j}^{n+1} + u_{i,j-1}^{n+1}}{\Delta y^2} \tag{6}$$

This equation involves terms from the next time step $n+1$, hence requiring a solution of a system of equations at each time step.

3.3 Key Aspects of the Implict Method in this problem:

The implicit scheme is a numerical approach used to solve partial differential equations (PDEs), particularly beneficial for handling diffusion-dominated problems such as boundary layer flow. This method is advantageous in steady-state simulations due to its inherent stability properties.

1. **Governing Equations:** In boundary layer flow over a flat plate, the problem involves solving the steady-state forms of the continuity and momentum equations. These equations describe the conservation of mass and momentum within the boundary layer.

2. **Discretization:** The implicit scheme approximates the governing equations' spatial derivatives using discrete grid points. Unlike explicit schemes, which rely on previous time steps, the implicit scheme uses information from current and future grid points, allowing for improved stability.

    - **Grid Setup:** The computational domain is divided into a grid with spacing in both the x (streamwise) and y (normal) directions.
    - **Numerical Approximation:** The derivatives in the governing equations are approximated using central differences, where the unknown values are treated implicitly. This approach results in a system of linear equations that must be solved simultaneously.
    - **Linear System Formation:** The discretized equations lead to a set of linear equations for each grid point. These equations are generally coupled, meaning that solving for one grid point requires information from neighboring points.
    - **Solution Method:** To solve the system of linear equations, numerical methods such as Gaussian elimination or iterative solvers (e.g., Conjugate Gradient) are used. These methods solve the system efficiently and handle the implicit nature of the discretization.
    - **Boundary Conditions:** Proper boundary conditions must be applied to ensure that the solution meets the physical constraints at the domain's boundaries. This involves setting appropriate values for velocities at the edges of the computational domain.

3. **Advantages and Disadvantages:**

    - **Stability:** The implicit scheme is unconditionally stable with respect to the grid size, which allows for more flexibility in choosing grid spacing and solving larger systems.
    - **Robustness:** It effectively handles high diffusion and boundary layer problems, providing a stable solution even for larger grid sizes.
    - **Accuracy:** Implicit scheme is second-order accurate in space but only first-order accurate in time. This is a drawback of this scheme.

3.4 Algorithm

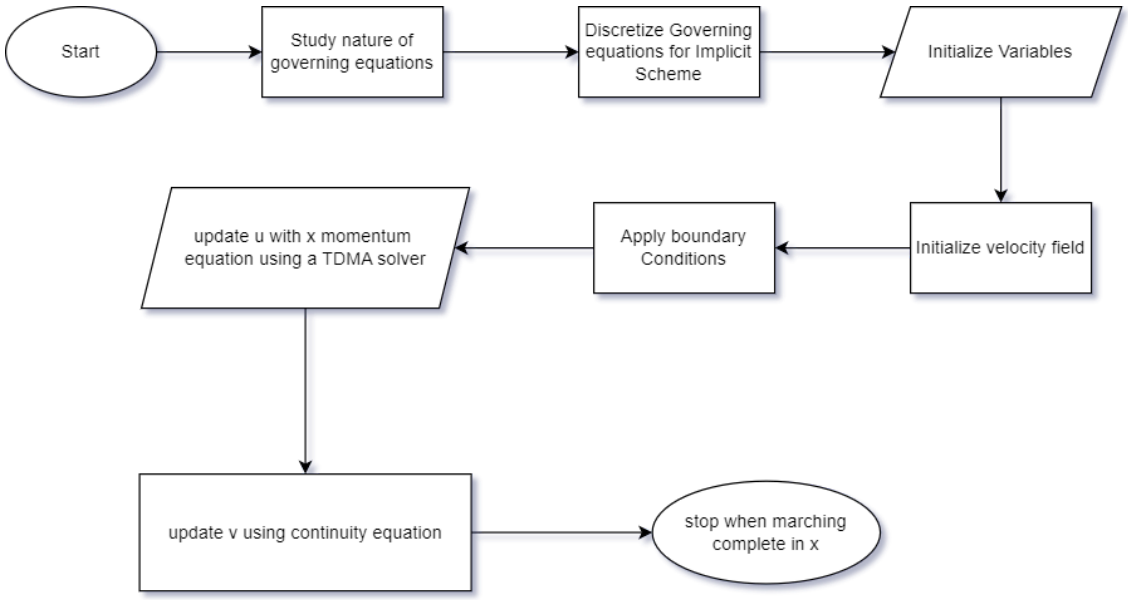


Figure 6: Implicit Scheme Algorithm

3.5 Parameters and Variables

- **Geometry and Flow Parameters:**
  - L_x, L_y: Dimensions of the computational domain. Here, $L_x = 10.0$ (length) and $L_y = 1.0$ (height).
  - Re: Reynolds number, set to 1×10^4.
  - U_∞: Free stream velocity, set to 1.0.
- **Grid Settings:**
  - n_x, n_y: Number of grid points in the x and y directions, set to 200 and 100, respectively.
  - ν: Kinematic viscosity, calculated as $\nu = \frac{U_\infty \cdot L_x}{Re}$.
- **Grid Points:**
  - x: Linearly spaced array from 0 to L_x with n_x points.
  - y: Linearly spaced array from 0 to L_y with n_y points.
- **Velocity Fields:**
  - u: A 2D array initialized to U_∞, representing the x-component of velocity across the grid.
  - v: A 2D array initialized to 0, representing the y-component of velocity across the grid.

3.6 Function: *EulerImplicitScheme*

This function implements the Euler implicit time-stepping scheme for solving the Navier-Stokes equations, focusing on incompressible flow in a 2D domain.

Main Loop

1. **Boundary Conditions Initialization:**

- $u[:, 0] = 0$: Applies no-slip condition at the bottom wall ($y = 0$).
- $u[:, -1] = U_\infty$: Sets inlet condition at the top edge ($y = L_y$).
- $v[:, 0] = 0$: No penetration condition at the bottom wall.
- $v[0, :] = 0$: No penetration condition at the leading edge ($x = 0$).

2. **Time Step Loop:**

- The outer loop runs from $i = 1$ to $n_x - 1$, stepping through each time level.
- $u[i, :] = u[i - 1, :]$: Initializes the current time step's x-velocity with values from the previous time step.

3. **Inner Iteration for Convergence:**

- The inner loop runs up to 10,000 times to achieve convergence for each time step.
- Arrays for the tridiagonal matrix system:
  - A_{lower}: Lower diagonal of the coefficient matrix.
  - A_{main}: Main diagonal of the coefficient matrix.
  - A_{upper}: Upper diagonal of the coefficient matrix.
  - b: Right-hand side vector for the linear system.
- The inner loop iterates over the y-direction (j from 1 to $n_y - 2$) to compute the coefficients and right-hand side values:
  - **Coefficient Calculations:**
    - $A_{\text{lower}}[j - 1] = -\frac{\nu}{dy^2} + \frac{v[i,j]}{2dy}$: Coefficient for the lower diagonal.
    - $A_{\text{main}}[j] = 2 \cdot \frac{\nu}{dy^2} + \frac{u[i,j]}{dx}$: Coefficient for the main diagonal.
    - $A_{\text{upper}}[j] = -\frac{\nu}{dy^2} - \frac{v[i,j]}{2dy}$: Coefficient for the upper diagonal.
    - $b[j] = \frac{u[i-1,j] \cdot u[i,j]}{dx}$: Right-hand side vector.
- **Boundary Conditions:**
  - $A_{\text{main}}[0] = 1$: Dirichlet boundary condition at the bottom wall.
  - $b[0] = 0$: No-slip condition at the bottom wall.
  - $A_{\text{main}}[-1] = 1$: Dirichlet boundary condition at the top wall.

- $b[-1] = U_\infty$: Inlet condition at the top wall.

- The system of equations is solved using a tridiagonal matrix algorithm (TDMA):

$$u_{\text{new}} = \text{tdma\_solver}(A_{\text{lower}}, A_{\text{main}}, A_{\text{upper}}, b)$$

- **Convergence Check:**
  - The algorithm checks for convergence using the norm:

$$\text{if } \|u_{\text{new}} - u[i,:]\| < 1 \times 10^{-6} :$$

    If converged, updates the velocity field and breaks the inner loop; otherwise, it continues updating.

4. **Updating the Y-Velocity:**

- After updating the x-velocity, the y-velocity is computed for the current time step:

$$v[i,j] = v[i,j-1] - \frac{dy}{dx}(u[i,j] - u[i-1,j])$$

3.7 Results

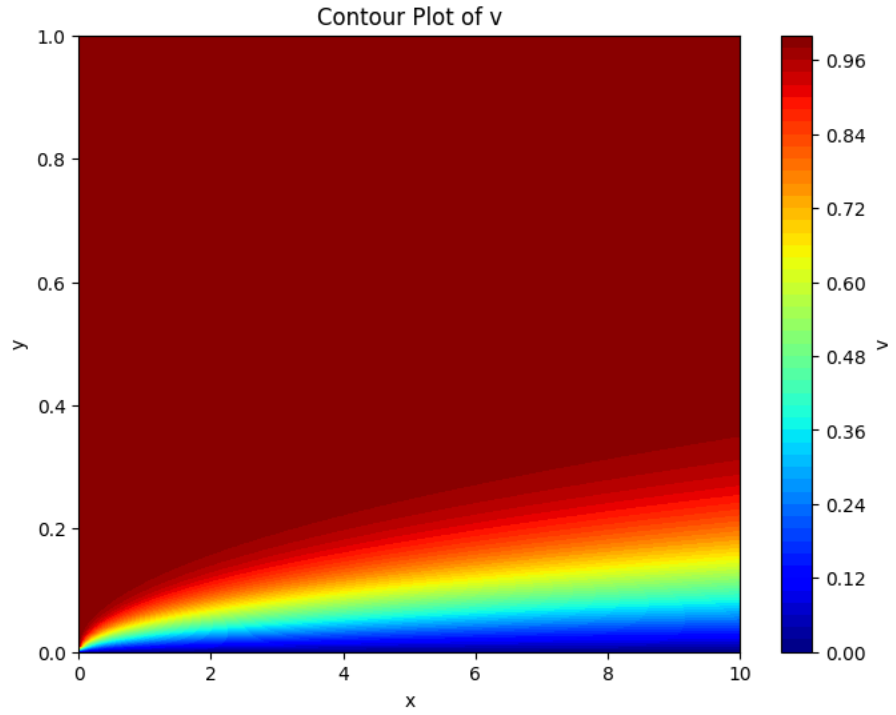


Figure 7: Implicit Scheme Velocity Contour

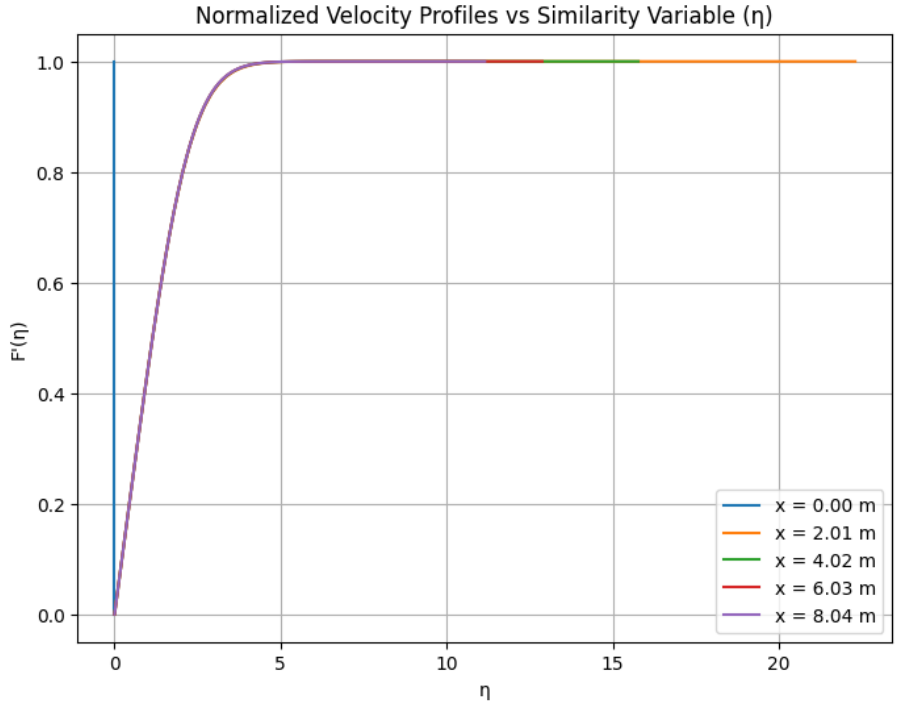


Figure 8: Implicit Scheme Similarity Profile

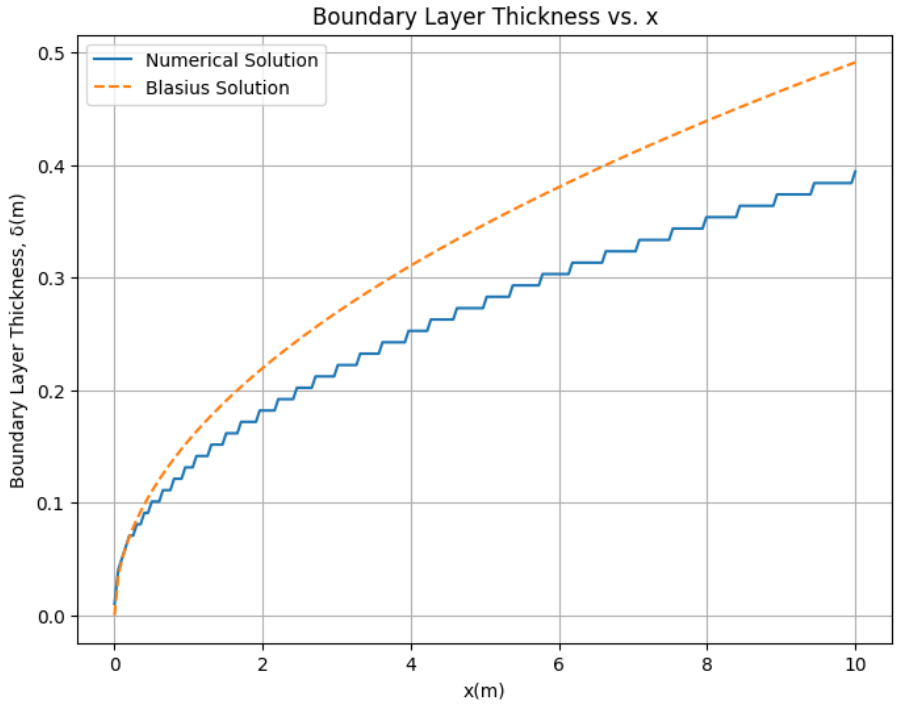


Figure 9: Implicit Scheme Boundary Layer Thickness

4 Crank-Nicholson scheme

4.1 Continuity Equation

The continuity equation is:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0$$

This can be approximated as:

$$\frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} + \frac{v_{i,j+1} - v_{i,j-1}}{2\Delta y} = 0$$

Solving for v:

$$v_{i,j} = v_{i,j-1} - \frac{\Delta y}{\Delta x}\left(u_{i+1,j} - u_{i-1,j}\right)$$

4.2 Momentum equation

The x-momentum equation is:

$$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = \nu\frac{\partial^2 u}{\partial y^2}$$

The derivative $\frac{\partial u}{\partial x}$ is approximated using central differences as:

$$\left(\frac{\partial u}{\partial x}\right)_{i,j} = \frac{u_{i+1,j}^{n+1} - u_{i-1,j}^{n}}{2\Delta x}$$

The convection term $u\frac{\partial u}{\partial x}$ is then discretized as:

$$\left(u\frac{\partial u}{\partial x}\right)_{i,j}^{n+1/2} = \frac{1}{2}\left(u_{i,j}^{n} + u_{i,j}^{n+1}\right)\frac{u_{i+1,j}^{n+1} - u_{i-1,j}^{n}}{2\Delta x}$$

The derivative $\frac{\partial u}{\partial y}$ is approximated as:

$$\left(\frac{\partial u}{\partial y}\right)_{i,j} = \frac{u_{i,j+1}^{n+1} - u_{i,j-1}^{n}}{2\Delta y}$$

The convection term $v\frac{\partial u}{\partial y}$ becomes:

$$\left(v\frac{\partial u}{\partial y}\right)_{i,j}^{n+1/2} = \frac{1}{2}\left(v_{i,j}^{n+1} + v_{i,j}^{n}\right)\frac{u_{i,j+1}^{n+1} - u_{i,j-1}^{n}}{2\Delta y}$$

The second derivative $\frac{\partial^2 u}{\partial y^2}$ is approximated as:

$$\frac{\partial^2 u}{\partial y^2} \approx \frac{u_{i,j+1}^{n+1} - 2u_{i,j}^{n+1} + u_{i,j-1}^{n+1}}{\Delta y^2}$$

Combining all the terms, we obtain the fully discretized x-momentum equation:

$$\frac{1}{2}\left(u_{i,j}^{n}+u_{i,j}^{n+1}\right)\frac{u_{i+1,j}^{n+1}-u_{i-1,j}^{n}}{2\Delta x}+\frac{1}{2}\left(v_{i,j}^{n}+v_{i,j}^{n+1}\right)\frac{u_{i,j+1}^{n+1}-u_{i,j-1}^{n}}{2\Delta y}=\frac{\nu}{\Delta y^{2}}\left[u_{i,j+1}^{n+1}-2u_{i,j}^{n+1}+u_{i,j-1}^{n+1}\right]$$

Simplifying:

$$\frac{u_{i,j}^{n+1}-u_{i,j}^{n}}{\Delta x}=-\frac{u_{i,j}^{n+1}\left(u_{i+1,j}^{n+1}-u_{i-1,j}^{n}\right)}{2\Delta x}-\frac{v_{i,j}^{n+1}\left(u_{i,j+1}^{n+1}-u_{i,j-1}^{n}\right)}{2\Delta y}+\frac{\nu}{\Delta y^{2}}\left(u_{i,j+1}^{n+1}-2u_{i,j}^{n+1}+u_{i,j-1}^{n+1}\right)$$

After discretizing the x-momentum equation, the resulting system of equations can be written in a tridiagonal form. To solve this efficiently, we use the Tridiagonal Matrix Algorithm (TDMA), also known as the Thomas Algorithm. The tridiagonal system for each i-position can be written as:

$$a_j u_{i,j-1}^{n+1}+b_j u_{i,j}^{n+1}+c_j u_{i,j+1}^{n+1}=d_j$$

where:

$$\begin{aligned}
a_j &= -\frac{\nu}{\Delta y^{2}},\\
b_j &= \frac{1}{\Delta x}+\frac{2\nu}{\Delta y^{2}},\\
c_j &= -\frac{\nu}{\Delta y^{2}},\\
d_j &= -\frac{u_{i,j}^{n}}{\Delta x}-\frac{u_{i,j}^{n+1}\left(u_{i+1,j}^{n+1}-u_{i-1,j}^{n}\right)}{2\Delta x}-\frac{v_{i,j}^{n+1}\left(u_{i,j+1}^{n+1}-u_{i,j-1}^{n}\right)}{2\Delta y}
\end{aligned}$$

4.3 Key Aspects of the Crank - Nicholson scheme in this problem:

The Crank-Nicholson scheme is a numerical method widely used in computational fluid dynamics (CFD) to solve partial differential equations, particularly the Navier-Stokes equations governing fluid flow. It is an implicit method, meaning that it requires solving a system of equations at each step.

1. **Spatial discretization:** The domain is divided into a grid where both x- and y-directions are discretized using central differences.
2. **Solution method:** The Crank-Nicholson method is implicit, requiring the solution of a system of equations at each step. The method simultaneously accounts for both continuity and momentum equations, ensuring mass and momentum conservation.
3. **Convergence:** The iterative solution process continues until a predefined tolerance level is met. The method often requires multiple iterations within each time step to refine the solution.
4. **Stability:** The Crank-Nicholson scheme is generally unconditionally stable, meaning it can handle a wide range of step sizes without becoming unstable. However, improper parameter choices can lead to numerical oscillations.
5. **Advantages:**
   (a) Accuracy: Provides second-order accuracy in both space and time.
   (b) Flexibility: Suitable for a wide range of problems.
6. **Disadvantages:**
   (a) Computational complexity: Requires solving a system of equations at each time step.
   (b) Implementation difficulty: Setting up equations and boundary conditions can be complex.
   (c) Memory usage: Storing coefficients and managing iterations can increase memory requirements.

4.4 Algorithm

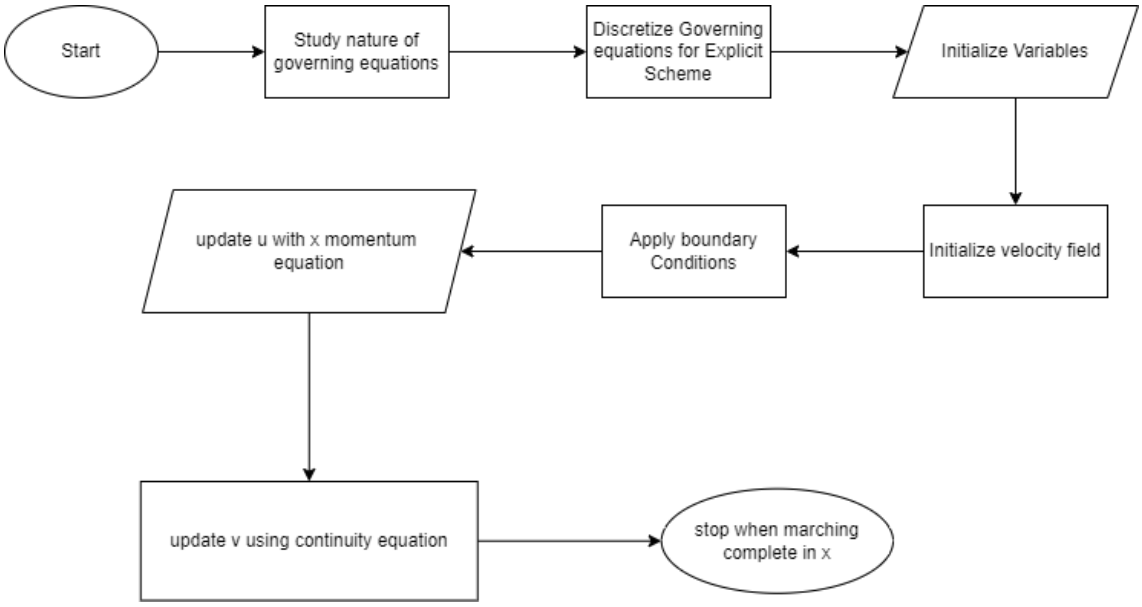


Figure 10: Crank-Nicolson Scheme Algorithm

4.5 Parameters and Variables

- **Geometry and Flow Parameters:**
  - L_x, L_y: Dimensions of the computational domain. Here, $L_x = 10.0$ (length) and $L_y = 1.0$ (height).
  - Re: Reynolds number, set to 1×10^4.
  - U_∞: Free stream velocity, set to 1.0.
- **Grid Settings:**
  - n_x, n_y: Number of grid points in the x and y directions, set to 200 and 100, respectively.
  - ν: Kinematic viscosity, calculated as $\nu = \frac{U_\infty \cdot L_x}{Re}$.
- **Grid Points:**
  - x: Linearly spaced array from 0 to L_x with n_x points.
  - y: Linearly spaced array from 0 to L_y with n_y points.
- **Velocity Fields:**
  - u: A 2D array initialized to U_∞, representing the x-component of velocity across the grid.
  - v: A 2D array initialized to 0, representing the y-component of velocity across the grid.

4.6 Function: *CrankNicolsonScheme*

This function implements the Crank-Nicolson time-stepping scheme for solving the Navier-Stokes equations, focusing on the incompressible flow in a 2D domain.

Variables

- **Loop Variables:**
  - i: Index variable iterating over the x-direction (time steps).
- **Velocity Fields:**
  - $u[i, :]$: The x-component of velocity for the current time step, initialized to the previous time step's values.
  - $v[i, j]$: The y-component of velocity, which will be updated later in the function.

Main Loop

1. **Time Step Initialization:**

- The outer loop runs from $i = 1$ to $n_x - 1$, stepping through each time level.
- The x-velocity array $u[i, :]$ is initialized to the values from the previous time step $u[i - 1, :]$.

2. **Inner Iteration for Convergence:**

- A loop runs up to 10,000 times to achieve convergence for each time step.
- Arrays for the tridiagonal matrix system:
  - A_{lower}: Lower diagonal of the coefficient matrix.
  - A_{main}: Main diagonal of the coefficient matrix.
  - A_{upper}: Upper diagonal of the coefficient matrix.
  - b: Right-hand side vector for the linear system.
- The inner loop iterates over the y-direction (j from 1 to $n_y - 2$) to compute the coefficients and right-hand side values:
  - **Crank-Nicolson Coefficients:**
    - $A_{\text{lower}}[j-1] = -\frac{\nu}{2dy^2} + \frac{v[i,j]}{4dy}$: Coefficient for the lower diagonal.
    - $A_{\text{main}}[j] = \frac{1}{dx} + \frac{\nu}{dy^2}$: Coefficient for the main diagonal.
    - $A_{\text{upper}}[j] = -\frac{\nu}{2dy^2} - \frac{v[i,j]}{4dy}$: Coefficient for the upper diagonal.
    - $b[j] = \frac{u[i-1,j]}{dx} + \left(\frac{u[i,j+1]-2u[i,j]+u[i,j-1]}{dy^2}\right) \cdot \frac{\nu}{2}$: Right-hand side vector.
- **Boundary Conditions:**
  - $A_{\text{main}}[0] = 1$: Dirichlet boundary condition at the bottom wall.

- $b[0] = 0$: No-slip condition at the bottom wall.
- $A_{\text{main}}[-1] = 1$: Dirichlet boundary condition at the top wall.
- $b[-1] = U_\infty$: Inlet condition at the top wall.

- The system of equations is solved using a tridiagonal matrix algorithm (TDMA):

$$u_{\text{new}} = \text{tdma\_solver}(A_{\text{lower}}, A_{\text{main}}, A_{\text{upper}}, b)$$

- **Convergence Check:**
  - The algorithm checks for convergence using the norm:

$$\text{if } \|u_{\text{new}} - u[i,:]\| < 1 \times 10^{-6} :$$

    If converged, updates the velocity field and breaks the inner loop; otherwise, it continues updating.

3. **Updating the Y-Velocity:**

- After updating the x-velocity, the y-velocity is computed for the current time step:

$$v[i,j] = v[i,j-1] - \frac{dy}{dx}(u[i,j] - u[i-1,j])$$

4.7 Results

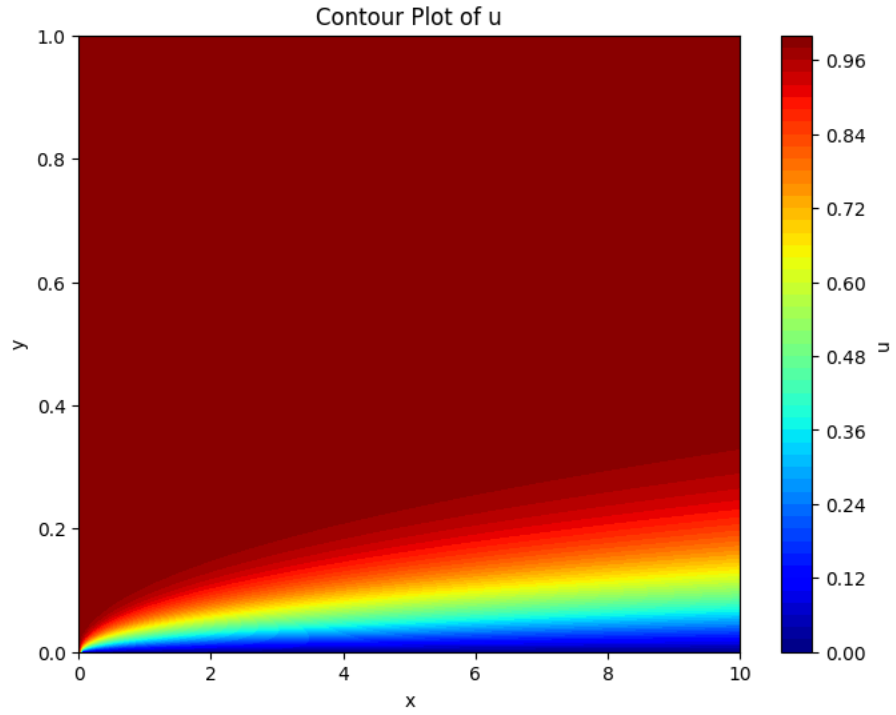


Figure 11: Crank-Nicolson Scheme Velocity Contour

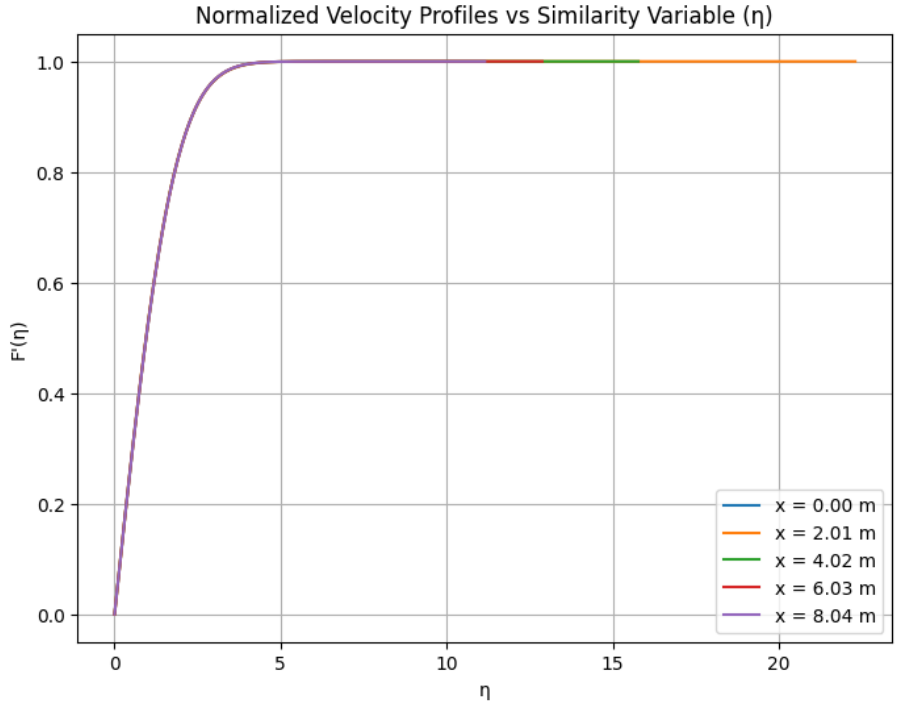


Figure 12: Crank-Nicolson Scheme Similarity Profile

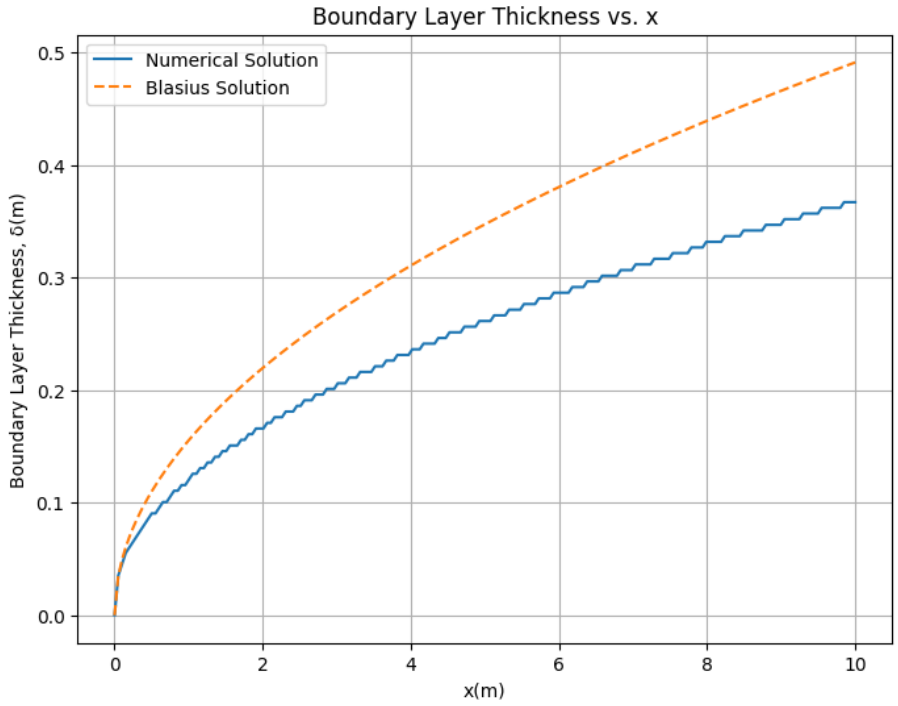


Figure 13: Crank-Nicolson Scheme Boundary Layer Thickness

5 Discussion

The numerical solutions for the boundary layer flow over a flat plate were obtained using three different finite difference schemes: Euler Explicit, Euler Implicit, and Crank-Nicolson. Each method was implemented to solve the governing equations for a Reynolds number of $Re_L = 10^4$. The results provide insights into the performance and characteristics of these numerical schemes for this specific flow problem.

5.1 Velocity Contours

Examining the velocity contour plots (Figures 3, 7, and 11) for all three schemes, we observe that they qualitatively capture the expected boundary layer development over the flat plate. The gradual thickening of the boundary layer along the streamwise direction is evident in all cases. However, subtle differences can be noted:

- The Euler Explicit scheme (Figure 3) shows a slightly more diffuse boundary layer, which may be attributed to the scheme's inherent numerical diffusion.
- The Euler Implicit (Figure 7) and Crank-Nicolson (Figure 11) schemes produce sharper contours, indicating better resolution of the boundary layer profile.

5.2 Similarity Profiles

The similarity profiles (Figures 4, 8, and 12) provide a quantitative comparison of the numerical solutions with the Blasius solution. All three schemes show good agreement with the theoretical profile, but some differences are apparent:

- The Euler Explicit scheme (Figure 4) shows slight deviations from the Blasius solution, particularly in the outer region of the boundary layer.
- The Euler Implicit (Figure 8) and Crank-Nicolson (Figure 12) schemes demonstrate excellent agreement with the Blasius solution throughout the boundary layer thickness.

These observations suggest that the implicit schemes (Euler Implicit and Crank-Nicolson) provide more accurate representations of the velocity profile compared to the explicit scheme.

5.3 Boundary Layer Thickness

The plots of boundary layer thickness variation along the plate (Figures 5, 9, and 13) offer insights into the growth of the boundary layer:

- All three schemes capture the general trend of boundary layer growth, showing good agreement with the Blasius solution ($\delta(x) = \frac{4.91}{\sqrt{Re_x}}$).
- The Euler Explicit scheme (Figure 5) shows some oscillations in the boundary layer thickness, particularly near the leading edge. This may be due to the scheme's sensitivity to grid spacing and stability constraints.
- The Euler Implicit (Figure 9) and Crank-Nicolson (Figure 13) schemes produce smoother variations in boundary layer thickness, closely matching the theoretical prediction.

5.4 Numerical Considerations

The implementation of these schemes revealed important numerical considerations:

- The Euler Explicit scheme required careful selection of grid spacing (Δx and Δy) to ensure stability, as noted in the problem statement. This constraint limits the flexibility in choosing grid parameters.
- The Euler Implicit and Crank-Nicolson schemes, being implicit methods, demonstrated better stability characteristics. They allowed for larger grid spacings without compromising stability, although at the cost of solving a system of equations at each step.
- The Crank-Nicolson scheme, being second-order accurate in both space and time, theoretically offers the highest accuracy among the three methods. This is reflected in its close agreement with the Blasius solution.

6 Conclusion

This study implemented and compared three finite difference schemes for solving the boundary layer flow over a flat plate. The following conclusions can be drawn:

1. All three numerical schemes (Euler Explicit, Euler Implicit, and Crank-Nicolson) successfully captured the essential features of the boundary layer flow, including velocity profiles and boundary layer growth.
2. The implicit schemes (Euler Implicit and Crank-Nicolson) demonstrated superior performance in terms of accuracy and stability compared to the Euler Explicit scheme.
3. The Crank-Nicolson scheme, with its second-order accuracy, provided the closest agreement with the Blasius solution, making it the most suitable choice for this boundary layer flow problem.
4. The study highlighted the trade-offs between computational simplicity (Euler Explicit) and numerical accuracy/stability (implicit schemes).
5. The results underscore the importance of choosing appropriate numerical methods and grid parameters in computational fluid dynamics simulations to balance accuracy, stability, and computational efficiency.