Project 1

December 9, 2024

Antonio Pampalone 23586519
Giuseppe Pisante 23610012
Martina Raffaelli 23616907

Task 2.0:

The momentum equation is a parabolic equation since, if we compute the $\Delta = B^2 - 4AC$ from the general form of the partial differential equations:

$$A\frac{\partial^2 u}{\partial x^2} + B\frac{\partial^2 u}{\partial x \partial y} + C\frac{\partial^2 u}{\partial y^2} + D\frac{\partial u}{\partial x} + E\frac{\partial u}{\partial y} + Fu + G = 0 \tag{1}$$

we get $\Delta = 0$, which means that the equation is parabolic.
We now define the additional boundary conditions:

- $u(0, y) = 1$
- $v(0, y) = 0$

For x=1, we do not need boundary conditions, since it would overconstrain the problem.

Task 2.1:

The discretization applied to the x-momentum equation takes into account the parabolic nature of the equation. The discretization is performed using the central difference scheme for the y-direction, to capture the elliptic character of the diffusion process, and the backward difference scheme for the x-direction to better capture the convective term. The discretized equation is as follows:

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

Such equation is a steady-state equation and thus it doesn't require any time-stepping algorithm. For this reason, the costraints for convergence are only applied on the spatial discretization.
The discretization of the first derivative in the x-direction using the Backward Difference Scheme (BDS) is given by:

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

Using Taylor series expansions for $u(x)$ around x_i, we have:

$$u_{i,j-1} = u(x_i - \Delta x) = u(x_i) - \Delta x\frac{\partial u}{\partial x} + \frac{\Delta x^2}{2}\frac{\partial^2 u}{\partial x^2} + O(\Delta x^3)$$

Therefore, the approximation for the derivative becomes:

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

The truncation error for the Backward Difference Scheme is:

$$T_{\text{BDS}} = -\frac{\Delta x}{2}\frac{\partial^2 u}{\partial x^2} + O(\Delta x^2)$$

The discretization of the second derivative in the y-direction using the Central Difference Scheme (CDS) is given by:

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

Using Taylor series expansions for $u(y)$ around y_j, we have:

$$u_{i+1,j} = u(y_j + \Delta y) = u(y_j) + \Delta y\frac{\partial u}{\partial y} + \frac{\Delta y^2}{2}\frac{\partial^2 u}{\partial y^2} + \frac{\Delta y^3}{6}\frac{\partial^3 u}{\partial y^3} + O(\Delta y^4)$$

$$u_{i-1,j} = u(y_j - \Delta y) = u(y_j) - \Delta y\frac{\partial u}{\partial y} + \frac{\Delta y^2}{2}\frac{\partial^2 u}{\partial y^2} - \frac{\Delta y^3}{6}\frac{\partial^3 u}{\partial y^3} + O(\Delta y^4)$$

Subtracting $2u_{i,j}$ from the sum of $u_{i+1,j}$ and $u_{i-1,j}$, we get:

$$u_{i+1,j} - 2u_{i,j} + u_{i-1,j} = 2 \cdot \frac{\Delta y^2}{2}\frac{\partial^2 u}{\partial y^2} + O(\Delta y^4)$$

Thus, the discretized second derivative becomes:

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

The leading truncation error for the Central Difference Scheme is:

$$T_{\text{CDS}} = \frac{\Delta y^2}{6} \frac{\partial^4 u}{\partial y^4} + O(\Delta y^4)$$

Summary of Truncation Errors

Thus, the sum of the truncation errors is:

$$T_{\text{total}} = T_{\text{BDS}} + T_{\text{CDS}} = -\frac{\Delta x}{2} \frac{\partial^2 u}{\partial x^2} + \frac{\Delta y^2}{6} \frac{\partial^4 u}{\partial y^4} + O(\Delta x^2) + O(\Delta y^4)$$

Task 2.2:

This system is solved using a BiCGStab iterative solver, which allows us to solve the system for a non-symmetric A and the non linearities of the x-momentum. The solver updates the values of u and v at each iteration, denoted as $u^{(k+1)}$ and $v^{(k+1)}$, by utilizing the values from the previous iteration, $u^{(k)}$ and $v^{(k)}$, to handle the non-linear terms effectively. The system of partial differential equations is: The continuity equation is discretized using the backward difference scheme (BDS) along the x-direction and the central difference scheme (CDS) along the y-direction. The discretized continuity equation is given by:

$$\begin{cases} \frac{u_{i,j} - u_{i,j-1}}{\Delta x} + \frac{v_{i+1,j} - v_{i-1,j}}{2\Delta y} = 0 \\ u_{i,j} \frac{u_{i,j} - u_{i,j-1}}{\Delta x} + v_{i,j} \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta y} = \frac{1}{Re} \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta y^2} \end{cases}$$

It is important to note that at the boundaries, we cannot use this discretization directly. Depending on the boundary we have to apply different schemes:

1. Left boundary ($x = 0$): we cannot apply BDS on the x-direction. We thus use FDS on such boundary:

$$\begin{cases} \frac{u_{i,j+1} - u_{i,j}}{\Delta x} + \frac{v_{i+1,j} - v_{i-1,j}}{2\Delta y} = 0 \\ u_{i,j} \frac{u_{i,j+1} - u_{i,j}}{\Delta x} + v_{i,j} \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta y} = \frac{1}{Re} \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta y^2} \end{cases}$$

2. Bottom boundary ($y = 0$): CDS cannot be applied on the y-direction. We thus use FDS on such boundary:

$$\begin{cases} \frac{u_{i,j} - u_{i,j-1}}{\Delta x} + \frac{v_{i,j} - v_{i-1,j}}{\Delta y} = 0 \\ u_{i,j} \frac{u_{i,j} - u_{i,j-1}}{\Delta x} + v_{i,j} \frac{u_{i,j} - u_{i-1,j}}{\Delta y} = \frac{1}{Re} \frac{u_{i-2,j} - 2u_{i-1,j} + u_{i,j}}{\Delta y^2} \end{cases}$$

Since we only apply Dirichlet Boundary Conditions, a different discretization at the boundaries is not relevant for solving the problem. But for generality, we decided to add this information. To solve this, we first tried to construct a linear system of equations in the form $\mathbf{Ax} = \mathbf{b}$, where $A \in \mathbb{R}^{2n^2 \times 2n^2}$, $b \in \mathbb{R}^{2n^2}$, and $x \in \mathbb{R}^{2n^2 \times 2n^2}$. The structure of $\mathbf{x}$ is as follows:

$$\mathbf{x} = \begin{bmatrix} u_{1,1} \\ u_{1,2} \\ \vdots \\ u_{n,n} \\ v_{1,1} \\ v_{1,2} \\ \vdots \\ v_{n,n} \end{bmatrix}$$

The structure of the matrix $\mathbf{A}$ is as follows:

$$
A = \begin{bmatrix}
 & -\frac{1}{\Delta x} & & \frac{1}{\Delta x} & & & & & & & & & & -\frac{1}{\Delta y} \cdots & \frac{1}{\Delta y} \cdots \\
 & & & \ddots & & & & & & & & & & \ddots & \\
 & \cdots -\frac{1}{\Delta x} & & \frac{1}{\Delta x} & & & & & & & & & & \cdots -\frac{1}{2\Delta y} & \\
 & & & \vdots & & \ddots & & & & & & & & & \\
 & & & & & & & & & & & & & & \cdots -\frac{1}{\Delta y} \\
 & \cdots & & \cdots & & \cdots & & \cdots & & \cdots & & \cdots & \cdots & \cdots & \\
 & -\frac{v_{i,j}^{(k)}}{\Delta y} + \frac{1}{Re\Delta y^2} - \frac{u_{i,j}^{(k)}}{\Delta x} & & \frac{u_{i,j}^{(k)}}{\Delta x} & & \cdots \frac{v_{i,j}^{(k)}}{\Delta y} + \frac{2}{Re\Delta y^2} & & \frac{1}{Re\Delta y^2} & & \cdots & & & & \\
 & \cdots & & -\frac{1}{Re\Delta y^2} \cdots & & -\frac{v_{i,j}^{(k)}}{\Delta y} + \frac{2}{Re\Delta y^2} \cdots & & -\frac{u_{i,j}^{(k)}}{\Delta x} & & \frac{v_{i,j}^{(k)}}{\Delta y} - \frac{1}{Re\Delta y^2} + \frac{u_{i,j}^{(k)}}{\Delta x} & & \cdots & & \\
 & & & \vdots & & \vdots & & \vdots & & \vdots & & \vdots & & \\
 & & & \cdots & & \cdots -\frac{v_{i,j}^{(k)}}{2\Delta y} + \frac{1}{Re\Delta y^2} \cdots & & -\frac{u_{i,j}^{(k)}}{\Delta x} & & \frac{u_{i,j}^{(k)}}{\Delta x} - \frac{2}{Re\Delta y^2} & & \cdots \frac{v_{i,j}^{(k)}}{2\Delta y} + \frac{1}{Re\Delta y^2} \cdots & \\
 & & & \vdots & & \vdots & & \vdots & & \vdots & & \vdots & & \\
 & & & \cdots \frac{1}{Re\Delta y^2} \cdots & & \frac{v_{i,j}^{(k)}}{\Delta y} + \frac{1}{Re\Delta y^2} \cdots & & -\frac{u_{i,j}^{(k)}}{\Delta x} & & \frac{u_{i,j}^{(k)}}{\Delta x} - \frac{1}{Re\Delta y^2} - \frac{v_{i,j}^{(k)}}{\Delta y} & & \cdots \frac{1}{Re\Delta y^2} & \\
\end{bmatrix}
$$

However, given the complexity of the system, we were unable to solve it through such method and we thus decided to move towards a different route. In particular, given the physical nature of the problem, we decided to trat the x spatial variable as a temporal variable, in order to solve the problem in a 1D environment using Explicit Euler. This was in fact a very good way to solve the problem, as it is clearly more simple and doesn't require the construction of a complex system of equations.
In the context of Explicit Euler, the left boundary condition was thus treated as an initial condition.

Python Code

Our Python implementation can be found at the following GitHub repository [1]. In the src folder there can be found two main scripts and two solvers. One script is related to the 1D solver using Explicit Euler, which works correctly. The other script pertains to the 2D solver, which is still a work in progress. The solutions for both u and v are included in this report, as requested.

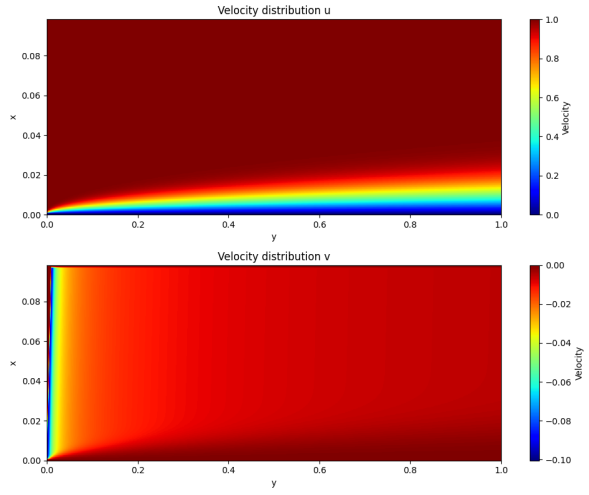


Figure 1: Velocity distribution along the domain

Task 2.5:

Starting from the non-dimensional result obtained in the previous task, we can now obtain the dimensional results by multiplying the non-dimensional results by the characteristic scales of the problem, as follows:

$$u_{dim} = u \cdot U_\infty, \qquad v_{dim} = v \cdot U_\infty, \qquad x_{dim} = x \cdot L, \qquad y_{dim} = y \cdot L$$

assuming for y the same characteristic scale of x. To compute the carachteristic length L we can use the formula: $Re_L = \frac{U_\infty L}{\nu}$ from which we get: $L = \frac{Re_L \cdot \nu}{U_\infty}$, and substituting the values of $Re_L = 10000$ and $\nu = 1.5 \times 10^{-5}$ m^2/s we get $L = 0.0075m$.

In conclusion the dimensional solution is reported in the following picture:

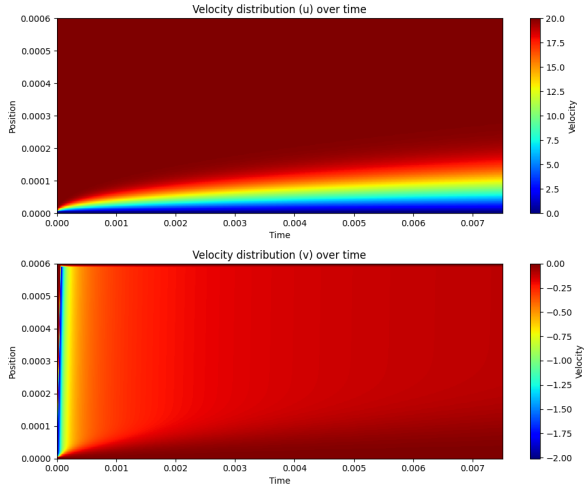


Figure 2: Dimensional solution

Task 2.6:

The stream function $\psi(x, y)$ is related to the velocity components as follows:

$$\frac{\partial \psi}{\partial y} = u, \qquad \frac{\partial \psi}{\partial x} = -v. \tag{2}$$

To compute ψ numerically, we integrate numerically over the spatial domain. We use a cumulative sum to integrate u along the y direction, imposing the boundary condition $\psi(0, y) = 0$. The plot we get is the following:

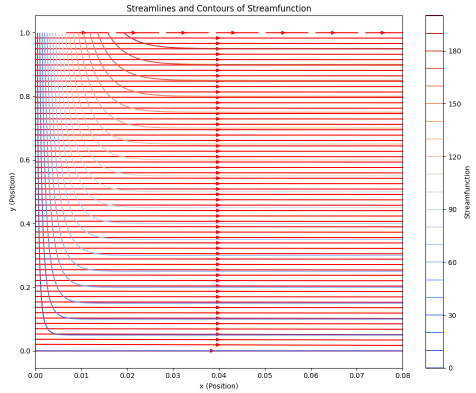
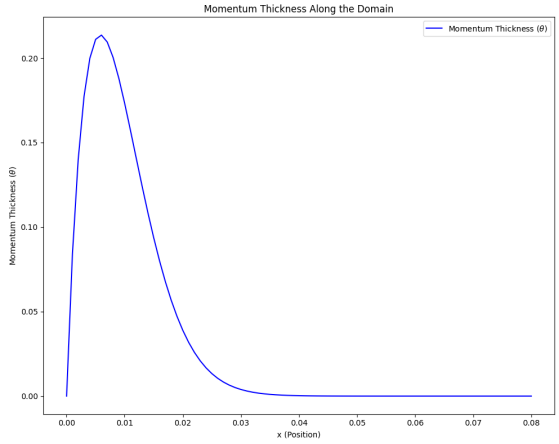


Figure 3: Streamlines

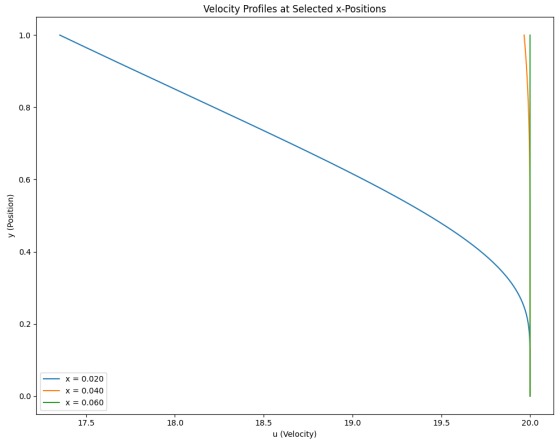
The streamlines crossing the boundary layer edge indicate interaction between the external flow and the boundary layer. This can occur due to numerical errors in streamline calculation or physical effects like entrainment. If the boundary layer definition is well-resolved and no-slip conditions are enforced accurately, such behavior might suggest deviations in the streamline plot accuracy or physical flow features, depending on the resolution and method used.

Task 2.7:

The plots obtained by the code we implemented are shown in the following figures: Momentum thickness provides



(a) Momentum thickness



(b) Velocity profiles

insight into the development of the boundary layer. It is a measure of the velocity distribution across the boundary layer and indicates how much momentum is being transported. By plotting the momentum thickness along the domain, we can observe how it evolves with increasing x (the streamwise position). Here we observe the typical behavior: momentum thickness increases initially as the boundary layer develops and then stabilizes as the flow becomes fully developed.

These profiles allow us to analyze the shape of the boundary layer at different streamwise locations. At lower x-values (near the leading edge), the profiles are steeper, indicating that the boundary layer is still developing. As x increases, the profiles tend to become more gradual, representing the fully developed boundary layer.

Streamlines represent the actual flow trajectories of fluid particles, showing the path the fluid takes under the influence of the velocity field. They indicate the direction of flow and provide insights into the flow patterns like vortices or separation. Iso-lines of the streamfunction epresent regions of constant flow and are mathematically derived from the velocity field. They indicate the same flow features but are typically smoother and more evenly spaced in a well-behaved flow.

Considering their meaning we expect to generally coincide, as both describe the same physical flow. However, differences might appear due to numerical errors, interpolation methods, or grid resolution, especially in regions with complex flow patterns or sharp gradients. In particular in this case we were not able to print them due to some issues in the code.

References

[1] *CFD Repository*,
Available at: `https://github.com/GiuseppePisante/CFD.git`

[2] *GitHub Copilot*,
GitHub. Available at: `https://github.com/features/copilot`