ARISTOTELIO UNIVERSITY OF THESSALONIKI Polytechnic School
Department of Mechanical Engineering
Laboratory of FLUID & TURBINE ENGINEERING

COMPUTATIONAL FLUID MECHANICS

Author : Diogenis Tsichlakis

Thessaloniki, April 2022

CONTENTS

1. INTRODUCTION

The following analysis first presents the problem we have to face and the principles that govern it, citing the appropriate theoretical background. Having analyzed the problem under study, we proceed to analyze the requirements and solve them one by one, both by developing the necessary mathematics and visualizing the results.

2. THE PROBLEM

We are asked to solve a laminar boundary layer on a horizontal plate. As known, the equations that describe the specific flow, for zero pressure gradient, are the Prandtl boundary layer equations (System I).

$$\begin{cases} \dfrac{\partial u}{\partial x} + \dfrac{\partial v}{\partial y} = 0 \\ u\dfrac{\partial u}{\partial x} + v\dfrac{\partial u}{\partial y} = \nu\dfrac{\partial^2 u}{\partial y^2} \end{cases} \quad (\boldsymbol{I})$$

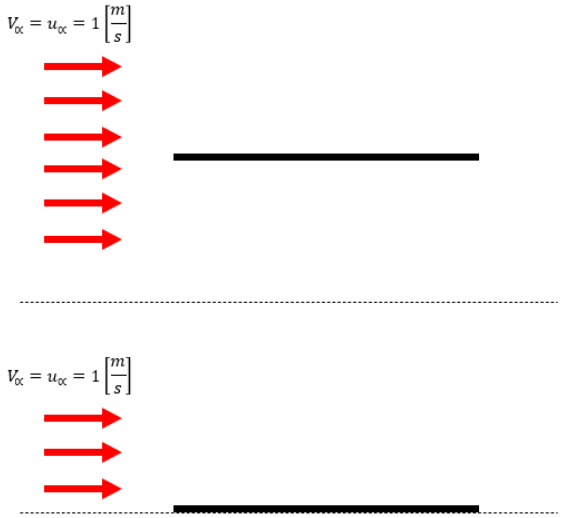
where n is the kinematic viscosity of the air. The growth of the boundary layer over a flat plate is assumed to be two-dimensional, so the layer grows in the (x,y) plane and not in space.$\nu = 1.46\ 10^{-5}\ \left[\frac{m^2}{s}\right]$

As a given we are given that the air velocity, at the beginning of the plate, is uniform and parallel in the direction to the surface of the plate with a value of . This means that the component of the air velocity in the direction perpendicular to the plate is zero in the free stream. It is also given that the length of the plate is 10 m. Based on these, we are asked to calculate the boundary layer of the air on the plate.$1\frac{m}{s}$

Let us examine the system of equations (I) a little. (I) constitutes a system of 2 partial differential equations, the first of which is of the 1st order and the second of the 2nd order. The unknowns in these equations are the velocities in the 2 directions (x and y) u(x,y) and v(x,y). These velocities are functions of only spatial variables because we are looking for the calculation in the steady state and not the transient phenomenon. Therefore the above system to be well defined requires 4 boundary conditions (BCC) for u and v.

For the boundary conditions we have at the beginning of the plate (x=0) and at the surface of the plate (y=0) based on the pronunciation, the following SS:

- $x = 0:\ u = 1\left[\frac{m}{s}\right], v = 0$
- $y = 0:\ u = v = 0$



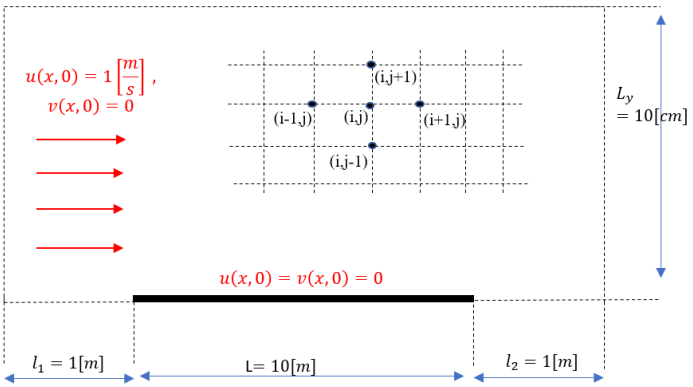
Shape1. Plate model to solve

Having now fully defined the problem under study, we can now proceed to build the numerical solution and analyze the requirements. The problem we are being asked to solve is shown in Figure 1, in the top image. Due to the symmetry of the problem, we can solve the model presented in the bottom image of Figure 1, since the boundary layer with respect to the plate is symmetric.

3. APPLICATION OF EXPLICIT AND IMPLICIT METHOD

In the 1st question we are asked to calculate the boundary layer, applying an explicit and a convoluted method. To achieve this we will first define our grid (geometric discretization) and then discretize the equations (I). Finally, after choosing the geometric discretization step in the 2 directions of the plane (Δx,Δy), we will analyze the code written for the numerical solution and present our results.

In Figure 2, we see the discretization grid for our problem. We notice that the grid starts 1 meter in front of the plate and also ends 1 meter after the plate. In addition, in this figure we distinguish the boundary conditions of the problem, their application point and the dimensions of the discretization grid. In the center of Figure 2, the form of the geometric discretization of our plane and how it is numbered is illustrated. We see that pointers (i,j) are used, where they run in the x and y directions respectively. Based on this figure, the partial derivatives of the system (I) will be discretized.



Shape2.Grid discretization of geometric space

$$\begin{cases} \dfrac{\partial u}{\partial x} + \dfrac{\partial v}{\partial y} = 0 \\ u\dfrac{\partial u}{\partial x} + v\dfrac{\partial u}{\partial y} = \nu\dfrac{\partial^2 u}{\partial y^2} \end{cases} \quad (\boldsymbol{I})$$

We are asked to solve with an explicit and an implicit method. The calculation steps we will follow and the construction of the necessary equations will be presented separately. After describing the 2 methods we will analyze the code that was written, the numerical solution parameters and then we will present the resulting results, where and based on them a comparison of the 2 methods will be made.

3.1 Explicit Method

To solve the equations (I) for the boundary layer of the plate, using the explicit method, we discretize the partial derivatives with the finite difference method. These relations will arise after developing each term in a Taylor series around a point (node) of the grid (eg (i,j)).

1st-order partial derivatives will be approximated by 2nd-order central differences, which show better accuracy than 1st-order methods because their error is 2nd-order, i.e. the omitted terms are up to 2nd-order in step. The 2nd order partial derivative will be approximated by 2nd order central second difference. After writing the system (I) for the point (i,j), we develop the derivatives in a Taylor series around this point, and then with appropriate additions and subtractions the following finite difference relations arise:

$$\begin{cases} \left(\frac{\partial u}{\partial x}\right)_{i,j} + \left(\frac{\partial v}{\partial y}\right)_{i,j} = 0 \\ (u_{i,j})\left(\frac{\partial u}{\partial x}\right)_{i,j} + (v_{i,j})\left(\frac{\partial u}{\partial y}\right)_{i,j} = \nu\left(\frac{\partial^2 u}{\partial y^2}\right)_{i,j} \end{cases} \quad (\boldsymbol{II})$$

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

By replacing these in system (II), the following equations result:

$$\begin{cases} \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} + \frac{v_{i,j+1} - v_{i,j-1}}{2\Delta y} = 0 \\ (u_{i,j})\frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} + (v_{i,j})\frac{u_{i,j+1} - u_{i,j-1}}{2\Delta y} = \nu\left(\frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{(\Delta y)^2}\right) \end{cases}$$

The velocity term in the x direction by which we will divide is non-zero since the above equations will be applied from the 2nd row of nodes onwards (above the plate). In the numerical procedure that we will follow, we will need to use a finite difference for the lower boundary, to calculate the velocity v in the second row of nodes. Approaching this partial derivative with a 2nd degree polynomial, after operations we have:

$$\left(\frac{\partial v}{\partial y}\right)_{i,0} = \frac{-3v_{i,0} + 4v_{i,1} - v_{i,2}}{2\Delta y} + O(\Delta y)^2 \ (\mathbf{1})$$

of Thus we have by substituting the differences:

$$\begin{cases} v_{i,j+1} = v_{i,j-1} - \frac{\Delta y}{\Delta x}(u_{i+1,j} - u_{i-1,j}) \quad (\mathbf{2}) \\ u_{i+1,j} = u_{i-1,j} + \frac{2\Delta x}{(\Delta y)^2}\nu\left(\frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{u_{i,j}}\right) - \frac{\Delta x}{\Delta y}\left(\frac{v_{i,j}}{u_{i,j}}(u_{i,j+1} - u_{i,j-1})\right) \quad (\mathbf{3}) \end{cases}$$

and to calculate the term we replace the differences only for the 1st equation so we have:$v_{i,1}$

$$\left(\frac{\partial u}{\partial x}\right)_{i,0} + \left(\frac{\partial v}{\partial y}\right)_{i,0} = 0 \overset{(1)}{\Rightarrow} \frac{u_{i+1,0} - u_{i-1,0}}{2\Delta x} + \frac{-3v_{i,0} + 4v_{i,1} - v_{i,2}}{2\Delta y} = 0$$

but because of the SS we have:$u_{i+1,0} - u_{i-1,0} = 0 \ \kappa\alpha\iota \ v_{i,0} = 0$

Therefore

$$\Rightarrow v_{i,1} = \frac{1}{2} \Delta y \; v_{i,2} \quad (\mathbf{4})$$

3.1.1 Explicit Method Algorithm

The calculation process we follow for the application of the explicit method is as follows:

1. We define the Boundary conditions on the free flow (and on the plate (, where k is the number of the node from which the plate starts. The numbering of the discretization cells starts from zero. We see that in the SS marked in red for the component of velocity in x we define the first 2 columns of nodes while for the component in y we define all the nodes of the first row, since we have a uniform flow.$u_{0\ \kappa\alpha\iota\ 1,j} = 1\frac{m}{s}\ \kappa\alpha\iota\ v_{0,j} = 0) u_{k,j} = v_{0\ \acute{\varepsilon}\omega\varsigma\ end,j} = 0)$

2. We calculate with equation (3) the term of the velocity at the position and thus the velocity at the position (i+1 column, j row) is obtained.$i = 1\ , j = 1 u_{2,1}$

3. Then with equation (2) we calculate for the same point (since as a system of equations they must be solved simultaneously for the same point), the term (i column, j+1 row).$v_{1,2}$

4. From Eq.(4) for j=1 we calculate (i column, 2nd row) using the velocity v we calculated in the previous step.$v_{i,1}$

5. We repeat steps 2 to 3 for the same i but $.j = j_{\pi\rho\iota\nu} + 1$

6. Having calculated for all j of the same column we now have all v velocities for column i and u velocities for column i+1.

7. We continue to the next point (i+1,j+1) so that we calculate the velocities u in column i+2 and the velocities v in column i+1 and apply steps 2 to 5 from the beginning.

8. Having calculated the entire velocity field in the boundary layer, we visualize the desired results.

With the above procedure we have calculated the entire velocity field above the plate. Having completed the description of the explicit method for solving the boundary layer, we proceed to the description of the second method.

3.2 Implicit Method

We rewrite the system of equations for recall and better tracking of the problem.

$$\begin{cases} \dfrac{\partial u}{\partial x}+\dfrac{\partial v}{\partial y}=0 \\ u\dfrac{\partial u}{\partial x}+v\dfrac{\partial u}{\partial y}=\nu\dfrac{\partial^2 u}{\partial y^2} \end{cases} \quad (\boldsymbol{II})$$

After developing the Taylor series for the discrete points of the derivatives, the 1st and 2nd order finite differences for the partial derivatives in the 2 directions are obtained:

- $\left(\frac{\partial u}{\partial x}\right)_{i,j} = \frac{u_{i+1,j}-u_{i,j}}{\Delta x} + O(\Delta x)$(Forward Difference)
- $\left(\frac{\partial u}{\partial y}\right)_{i,j} = \frac{u_{i,j+1}-u_{i,j-1}}{2\Delta y} + O(\Delta y)^2$(Central Difference)
- $\left(\frac{\partial v}{\partial y}\right)_{i,j} = \frac{v_{i,j+1}-v_{i,j}}{\Delta y} + O(\Delta y)$(Forward Difference)
- $\left(\frac{\partial^2 u}{\partial y^2}\right)_{i,j} = \frac{u_{i,j+1}-2u_{i,j}+u_{i,j-1}}{(\Delta y)^2} + O(\Delta y)^2$ (2nd order finite difference)

To increase the accuracy of the calculations and approximations, we write the derivatives of the velocity with respect to y, as a half-sum between the positions , so we have The above derivative approach is called 'Crank Nicolson Method' :$(i+1)$ $\kappa\alpha\iota$ (i)

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

By substituting the above differences in system (II), the discretized equations result:

1st Equation: $\frac{u_{i+1,j}-u_{i,j}}{\Delta x}+\frac{1}{2}\left(\frac{v_{i+1,j+1}-v_{i+1,j}}{\Delta y}+\frac{v_{i,j+1}-v_{i,j}}{\Delta y}\right)=0$

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

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

In compact form, we have:

$$v_{i+1,j+1} = \Lambda \quad (\mathbf{5})$$

Where:

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

The term Λ from the moment the velocities u at position i+1 are calculated, is considered known.

2nd Equation:

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

$$\Rightarrow u_{i,j}\left(\frac{u_{i+1,j}}{\Delta x}\right) - u_{i,j}\left(\frac{u_{i,j}}{\Delta x}\right) + \left(\frac{v_{i,j}}{4\Delta y}\right)\left(u_{i+1,j+1} - u_{i+1,j-1}\right) + \left(\frac{v_{i,j}}{4\Delta y}\right)\left(u_{i,j+1} - u_{i,j-1}\right) = \frac{\nu}{2(\Delta y)^2}\left(u_{i+1,j+1} - 2u_{i+1,j} + u_{i+1,j-1}\right) + \frac{\nu}{2(\Delta y)^2}\left(u_{i,j+1} - 2u_{i,j} + u_{i,j-1}\right)$$

$$\Rightarrow u_{i+1,j+1}\left(\frac{v_{i,j}}{4\Delta y} - \frac{\nu}{2(\Delta y)^2}\right) + u_{i+1,j}\left(\frac{u_{i,j}}{\Delta x} + \frac{\nu}{(\Delta y)^2}\right) - u_{i+1,j-1}\left(\frac{v_{i,j}}{4\Delta y} + \frac{\nu}{2(\Delta y)^2}\right) = u_{i,j}\left(\frac{u_{i,j}}{\Delta x}\right) - \left(\frac{v_{i,j}}{4\Delta y}\right)\left(u_{i,j+1} - u_{i,j-1}\right) + \frac{\nu}{2(\Delta y)^2}\left(u_{i,j+1} - 2u_{i,j} + u_{i,j-1}\right)$$

$$\Rightarrow u_{i+1,j+1}\left(\frac{v_{i,j}\Delta y - 2\nu}{4(\Delta y)^2}\right) + u_{i+1,j}\left(\frac{u_{i,j}(\Delta y)^2 + \nu\Delta x}{(\Delta y)^2\Delta x}\right) - u_{i+1,j-1}\left(\frac{v_{i,j}\Delta y + 2\nu}{4(\Delta y)^2}\right) = u_{i,j}\left(\frac{u_{i,j}}{\Delta x}\right) - \left(\frac{v_{i,j}}{4\Delta y}\right)\left(u_{i,j+1} - u_{i,j-1}\right) + \frac{\nu}{2(\Delta y)^2}\left(u_{i,j+1} - 2u_{i,j} + u_{i,j-1}\right)$$

I multiply all terms by to get rid of the denominators and make the process computationally easier.$(4(\Delta y)^2\Delta x)$

$$\Rightarrow u_{i+1,j+1}\left(v_{i,j}\Delta y\Delta x - 2\nu\Delta x\right) + u_{i+1,j}\left(4u_{i,j}(\Delta y)^2 + 4\nu\Delta x\right) + u_{i+1,j-1}\left(-v_{i,j}\Delta y\Delta x - 2\nu\Delta x\right) = u_{i,j}\left(4(\Delta y)^2 u_{i,j}\right) - \left(v_{i,j}\Delta y\Delta x\right)\left(u_{i,j+1} - u_{i,j-1}\right) + 2\nu\Delta x\left(u_{i,j+1} - 2u_{i,j} + u_{i,j-1}\right)$$

In compact form, we have:

$$\Rightarrow A_2 u_{i+1,j+1} + B_2 u_{i+1,j} + C_2 u_{i+1,j-1} = K \quad (\mathbf{6})$$

Where:

$$A_2 = \left(v_{i,j}\Delta y\Delta x - 2\nu\Delta x\right)$$

$$B_2 = \left(4u_{i,j}(\Delta y)^2 + 4\nu\Delta x\right)$$

$$C_2 = \left(-v_{i,j}\Delta y\Delta x - 2\nu\Delta x\right)$$

$$K = u_{i,j}\left(4(\Delta y)^2 u_{i,j}\right) - \left(v_{i,j}\Delta y\Delta x\right)\left(u_{i,j+1} - u_{i,j-1}\right) + 2\nu\Delta x\left(u_{i,j+1} - 2u_{i,j} + u_{i,j-1}\right)$$

$$\Rightarrow K = u_{i,j}\left(4(\Delta y)^2 u_{i,j} - 4\nu\Delta x\right) - u_{i,j+1}\left(-2\nu\Delta x + v_{i,j}\Delta y\Delta x\right) + u_{i,j-1}\left(2\nu\Delta x + v_{i,j}\Delta y\Delta x\right)$$

$$K = D_2 u_{i,j} - A_2 u_{i,j+1} - C_2 u_{i,j-1}$$

$with D_2 = 4(\Delta y)^2 u_{i,j} - 4\nu\Delta x$

A,B,C,D,K are known constants, since they include terms calculated in previous steps. So we ended up with a system of 2 equations of the form:

$$\begin{cases} v_{i+1,j+1} = \Lambda \quad (\mathbf{5}) \\ A_2 u_{i+1,j+1} + B_2 u_{i+1,j} + C_2 u_{i+1,j-1} = K \quad (\mathbf{6}) \end{cases} \qquad (III)$$

Observing the system (III) we see that we have the 1st equation which is solved directly if the quantities that include Λ are known and the 2nd equation which is in complex form. To solve the 2nd equation, you need to build a system. More specifically, following the path we have followed so far, we calculate the discretized equation at the point so as to find the velocities for all j, using the values we calculated in the previous steps. Then we calculate from the 1st equation the velocities for all j. System (III) is a system of 2 equations with 4 unknowns:$(i) u_{i+1}\ v_{i+1} \boldsymbol{u_{i+1,j+1}}, \boldsymbol{u_{i+1,j}}\ ,\ \boldsymbol{u_{i+1,j-1}}\ , \boldsymbol{v_{i+1,j+1}}$

In fact, from a purely mathematical point of view, such systems have either no or infinite solutions. But by using the boundary conditions and simultaneously solving a system of equations for the 2nd equation, it is possible to calculate the 4th unknowns.

3.2.1 Implicit Method Algorithm

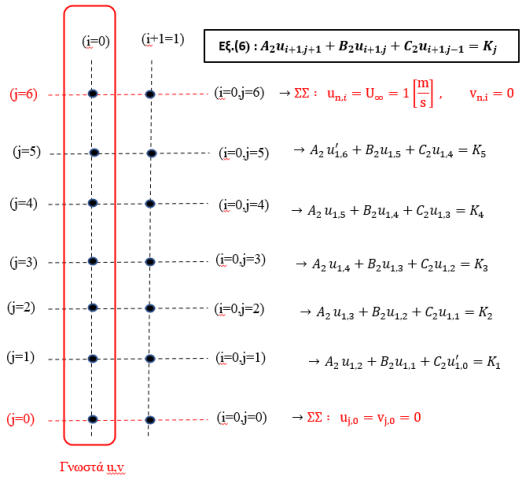
The calculation process (Algorithm) we follow has the following form:

1. First we define all the necessary parameters for our problem: Plate length, viscosity, mesh length, mesh height, discretization step (Δx, Δy)

2. We construct the registers of coordinates (x,y) and velocities (u,v) for all nodes.

3. We write the equation (5) for the column (starting from the 2nd column onwards) and the lines j=1(2nd gr.) to n-1(penultimate gr.). For a better understanding of the path we

follow, the process and the resulting equations are illustrated in Figure 4, for a mesh with 7 nodes in the vertical.i



We observe in figure 4, the resulting equations. The terms with a tone are known because of the SS. So we ended up with a linear system of 5 equations with 5 unknowns. Writing this system in register form we get the system (IV), with a constant tridiagonal register of

Figure 4. Example of building equations, for speed calculation

coefficients.$(u_{1,1}, u_{1,2}, u_{1,3}, u_{1,4}, u_{1,5})$

$$\begin{bmatrix} 3 & 2 & 0 & 0 & 0 \\ 2 & 3 & 2 & 0 & 0 \\ 0 & 2 & 3 & 2 & 0 \\ 0 & 0 & 2 & 3 & 2 \\ 0 & 0 & 0 & 2 & 3 \end{bmatrix} \begin{bmatrix} u_{2,2} \\ u_{2,3} \\ u_{2,4} \\ u_{2,5} \\ u_{2,6} \end{bmatrix} = \begin{bmatrix} K_2 - 2u'_{2,1} \\ K_3 \\ K_4 \\ K_5 \\ K_6 - 2u'_{2,7} \end{bmatrix} \Rightarrow A\underline{u}_{2,j} = \underline{K} \quad (\boldsymbol{IV})$$

To solve linear systems with such behavior, we apply the Thomas algorithm. The steps of this algorithm are analyzed after the description of the computational process for the calculation of the boundary layer.

4. After applying step 3 and solving system of the form (IV) we have the velocities in column i+1. We return to equation (6) and calculate the velocity in column i+1 for the node lines , where n is the number of nodes in the y direction.$v_{i+1,j+1} j = 1$ έως $n-1$
5. Having calculated all the u and v velocities of column i+1, we proceed to the next column of nodes and apply steps 2 to 4.

6. By calculating for all our nodes the velocities we have fully defined the velocity field in the boundary layer on the plate. This is how we finally visualize the desired results.

3.3 Thomas algorithm

A tridiagonal array consists of elements only on the main diagonal and the diagonals on either side of it. Thomas' algorithm takes a system of linear algebraic equations and after linear operations, converts the tridiagonal register of coefficients (Register A in the above analysis)

into a triangular register, which is finally solved from top to bottom with simple arithmetic operations. Next we describe this algorithm.

Let be a triangular register with rows and columns of the form , the elements , constitute the main, lower and upper diagonal respectively. The algorithm follows the following steps:$i\begin{bmatrix} d_1 & a_1 & 0 & 0 \\ b_2 & d_2 & a_3 & 0 \\ 0 & b_3 & d_3 & a_4 \\ 0 & 0 & b_4 & d_4 \end{bmatrix}\underline{u} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} d_i, b_i\ \kappa\alpha\iota\ a_i$

1. We multiply the 1st line by the lower diagonal element of the 2nd line(b_2).
2. We multiply the 2nd line by the diagonal element of the 1st line(d_1).
3. We subtract from the equation obtained in the 2nd step, the equation of the 1st step and divide the new equation by the diagonal element of the 1st line.
4. We replace the coefficients of the variables resulting from the above polynomials with a new constant, so that we have a more compact form of the equations in the last expression.$\left(d_2' = d_2 - \frac{b_2 a_1}{d_1}\ , c_2' = c_2 - \frac{c_1 b_2}{d_1}\right)$
5. The resulting equation is the new 2nd equation of our system
6. Repeat steps 1 to 5 for the other lines.
7. We end up in upper triangular register, and calculate the unknowns by working the equations from the bottom up. The last term is calculated from the relation and all the others from the relation.$\begin{bmatrix} d_1 & a_1 & 0 & 0 \\ 0 & d_2' & a_3 & 0 \\ 0 & 0 & d_3' & a_4 \\ 0 & 0 & 0 & d_4' \end{bmatrix}\underline{u} = \begin{bmatrix} c_1' \\ c_2' \\ c_3' \\ c_4' \end{bmatrix} u_n = \frac{c_n'}{d_n'} u_{n-i} = \frac{c_{n-i}' - a_{n-i} u_n}{d_{n-1}'}$

3.4 Select step(Δx,Δy)

We have now laid out the 2 methods we will use to calculate the boundary layer above the slab. Before describing the code, we need to define the discretization step in the 2 directions (Δx, Δy). The choice of the step has a large impact on the stability of the numerical calculation of the scheme we described above for the 2 methods. Of course, it has a greater impact on the explicit method, since the entangled methods show greater stability. A bad choice of Δx, Δy can lead to terms with a much larger order of magnitude difference and thus whatever software we use fails to solve the problem. In addition, due to the large number of calculations, and the rounding that the computer does by itself, they can lead to a large accumulation of errors and finally to a failure of the calculation. Of course, an important factor for choosing the discretization step is the computing power (the means) we have at our disposal.

From the literature we find for the explicit method, since it is the most critical, that the steps in the 2 directions must be chosen so as to satisfy the equation:

$$\Delta x \leq \frac{\frac{1}{2} u_\infty (\Delta y)^2}{\nu} \quad (7)$$

For the results that will be presented immediately after, for the first questions of the work and for the questions that we will present later, the files, 'CFD_main1.py' and 'CFD_main2.py' were

written in Python programming language for the application of the expression and of the entangled method respectively.

Specifically, the code structure for the explicit method was built with classes, while for the convoluted method it was done exclusively with the 'numpy' package. This differentiation was due to the increased computational difficulty displayed by the entangled method compared to the explicit one.

In addition, it is important to emphasize that due to limited computing power, it was not possible to discretize and solve the problem with very small steps Δx and Δy, because we had a large accumulation of error and instability of the code (values at infinity).

3.5 Results

Knowing that we are studying laminar flow, over a straight plate, we expect to get a parabolic velocity profile. We expect this profile to stabilize as it extends along the x-direction of the plate, and for the velocities to approach each other the farther we are from the plate.

In Figures 5 and 6, the velocity profiles u, as a function of the height y, for different positions of the plate x are presented. For the application of the 2 methods we used the set of parameters:

- Slab length: 10m
- Mesh length\Height: 12\0.09 m
- Discretization step Explicit Method: $\Delta x=0.001$, $\Delta y=0.01$
- Discretization step Complex Method : $\Delta x=0.1$, $\Delta y=0.01$

Observing these 2 figures, we see that with both methods we approach the parabolic velocity profile. It is of course obvious that the values given by these 2 methods differ. This is best understood by looking at Table 1, in which we have given the values of the velocities u, which we get from the 2 methods for specific positions x. From this Table, we see that the values differ, now which method is more accurate, will be considered in the last part of the paper where the 2 methods will be compared with Blasius' solutions.

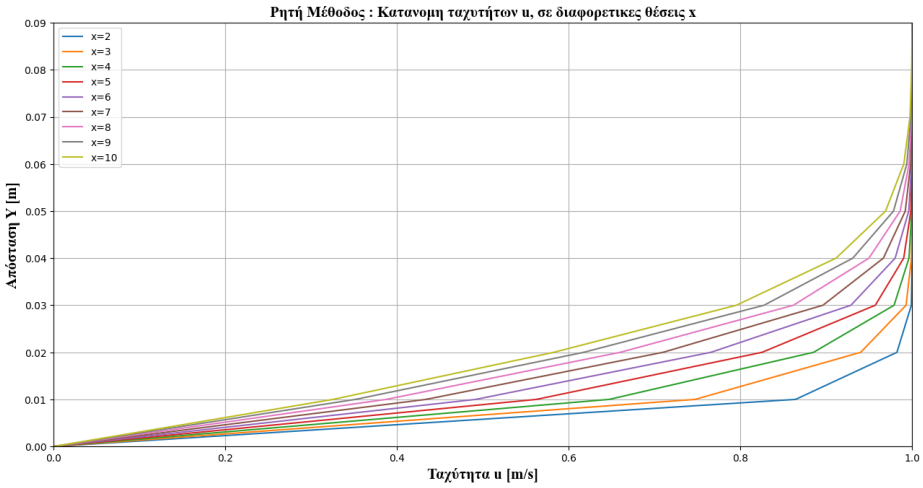


Figure 5. Velocity profiles at selected positions x, with the application of the Exact Numerical Method

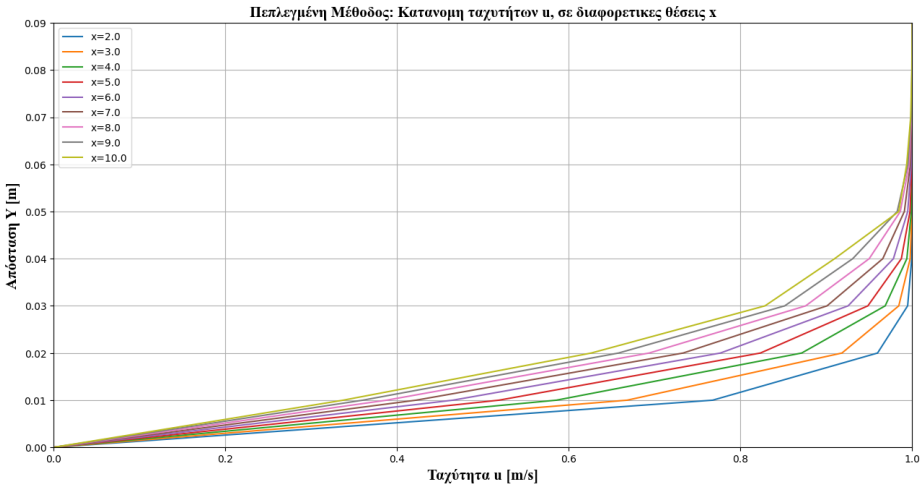


Figure 6. Velocity profiles at selected positions x, with application of the Complex Numerical Method

Panel1. Velocity values for the 2 methods at specific positions x

	Explicit (x=3)	Interlaced (x=3)	Explicit (x=7)	Interlaced (x=7)
y = 0	0	0	0	0
y = 0.01	0.7472	0.6680	0.4337	0.4222
y = 0.02	0.94	0.9182	0.7098	0.7332
y = 0.03	0.9928	0.9843	0.8960	0.9
y = 0.04	0.9991	0.9976	0.9664	0.9657
y = 0.05	0.9999	0.9997	0.99194	0.9908
y = 0.06	0.9999	0.9999	0.99816	0.9978
y = 0.07	1	0.99999	0.999675	0.9995
y = 0.08	1	1	0.9999	0.9999
y = 0.09	1	1	1	1

Obviously, if we had a better computer at our disposal so that we could reduce the discretization steps, or if we chose different finite differences, we could achieve greater accuracy. Also as a last measure of comparison of the 2 methods, we mention the times it took to solve them:

$$t_{Ρητής} = 4.584\ [\text{sec}]\ ,\qquad t_{Πεπλεγμένης} = 3.278\ [sec]$$

We see that despite the many more points we have obtained in the explicit method (2 orders of magnitude smaller Δx), the solution time is only slightly longer than the convoluted method. This is yet another proof that by using a convoluted method we increase the computational difficulty.

4. GRID INDEPENDENCE

In the next part of the work we are asked to check the independence of the grid. When we refer to grid independence we mean tracking the solution as we change the shape and dimension of the discretization grid. To check independence we will apply 2 procedures. First we will change the grid dimension (Length and Height) and then we will change the discretization step in both directions (Δx and Δy). For each change we will compare the resulting diagrams and where possible the numerical values of the speed.

4.1 Changing the Grid Dimension

In the above analysis, as mentioned above, the grid started 1 meter in front and ended 1 meter behind the plate P1 (Problem 1). We change our grid to start 7 meters in front and end 7 meters after the plate and also increase the height of the grid to 0.14 meters P2(Problem 2) .

For the Explicit Method the results are presented in Table 2 and Figure 7.

Panel2. Comparison of velocity values at the same locations, for differentiated grid

	P1	P2	P1	P2
	Explicit (x=3)	Explicit (x=3)	Explicit (x=7)	Explicit (x=7)
y = 0	0	0	0	0
y = 0.01	0.7472	0.6471	0.4337	0.3874
y = 0.02	0.94	0.8850	0.7098	0.66
y = 0.03	0.9928	0.9789	0.8960	0.8616
y = 0.04	0.9991	0.9962	0.9664	0.9496
y = 0.05	0.9999	0.9995	0.99194	0.9859
y = 0.06	0.9999	0.9999	0.99816	0.9963
y = 0.07	0.9999	0.9999	0.999675	0.9992
y = 0.08	0.9999	0.9999	0.9999	0.9999
y = 0.09	1	0.9999	1	0.9999

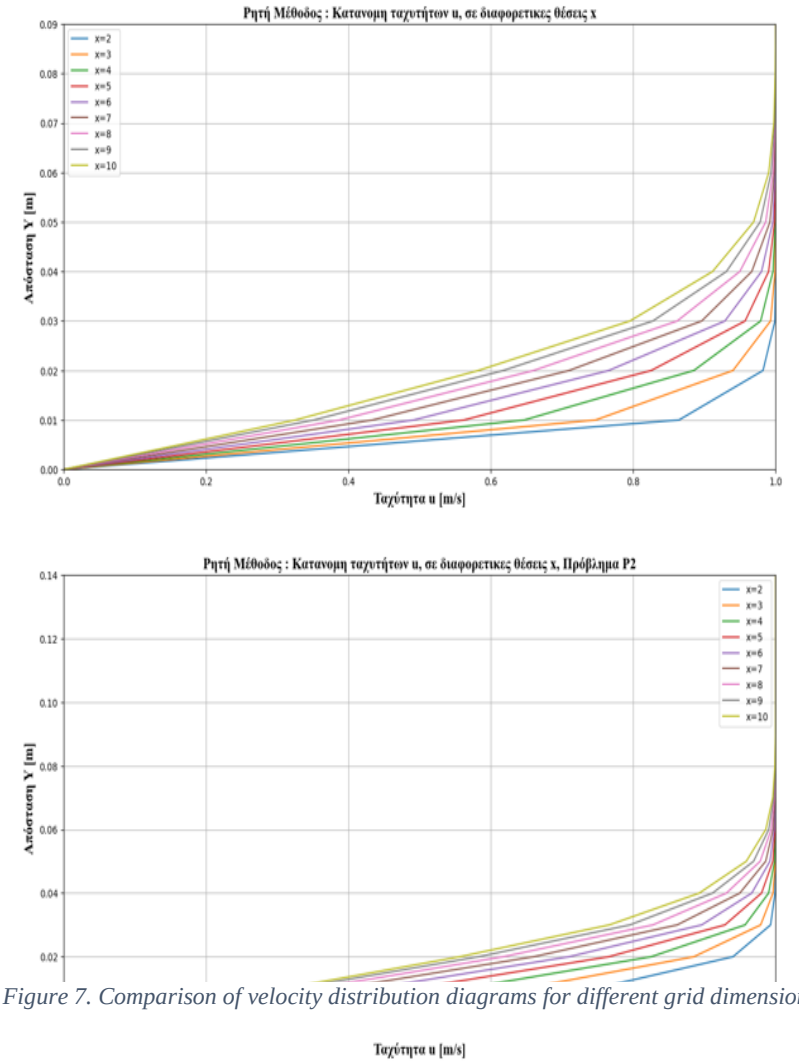


Figure 7. Comparison of velocity distribution diagrams for different grid dimensions

We see from the above Table and from Figure 7, that the grid differentiation for the explicit method affects the values we calculate. Of course, the differences in prices are small, so we can say that for a smaller difference in dimensions, we could consider our grid independent.

On the other hand, for the entangled method our code leads us to instability. Due to a large concentration of error in the successive operations, the error becomes infinite and we fail to calculate the velocities. So we can say that for the entangled method, the mesh is particularly sensitive to its dimensions. Of course, the way the equations are written in the programming environment, as well as the computing capacity of the machine, play a big role in this result.

4.2 Change of Pitch (Δx, Δy)

As mentioned above, my codes are very sensitive to the discretization step. In the above results, we recall that we had defined as steps:

- Discretization step Explicit Method: Δx=0.001, Δy=0.01
- Discretization step Complex Method : Δx=0.1 , Δy=0.01

We will duplicate the 2 steps for both methods and implement them. Therefore the new steps are:

- Discretization step Explicit Method : Δx=0.0005 , Δy=0.005
- Discretization step Complex Method : Δx=0.05 , Δy=0.005

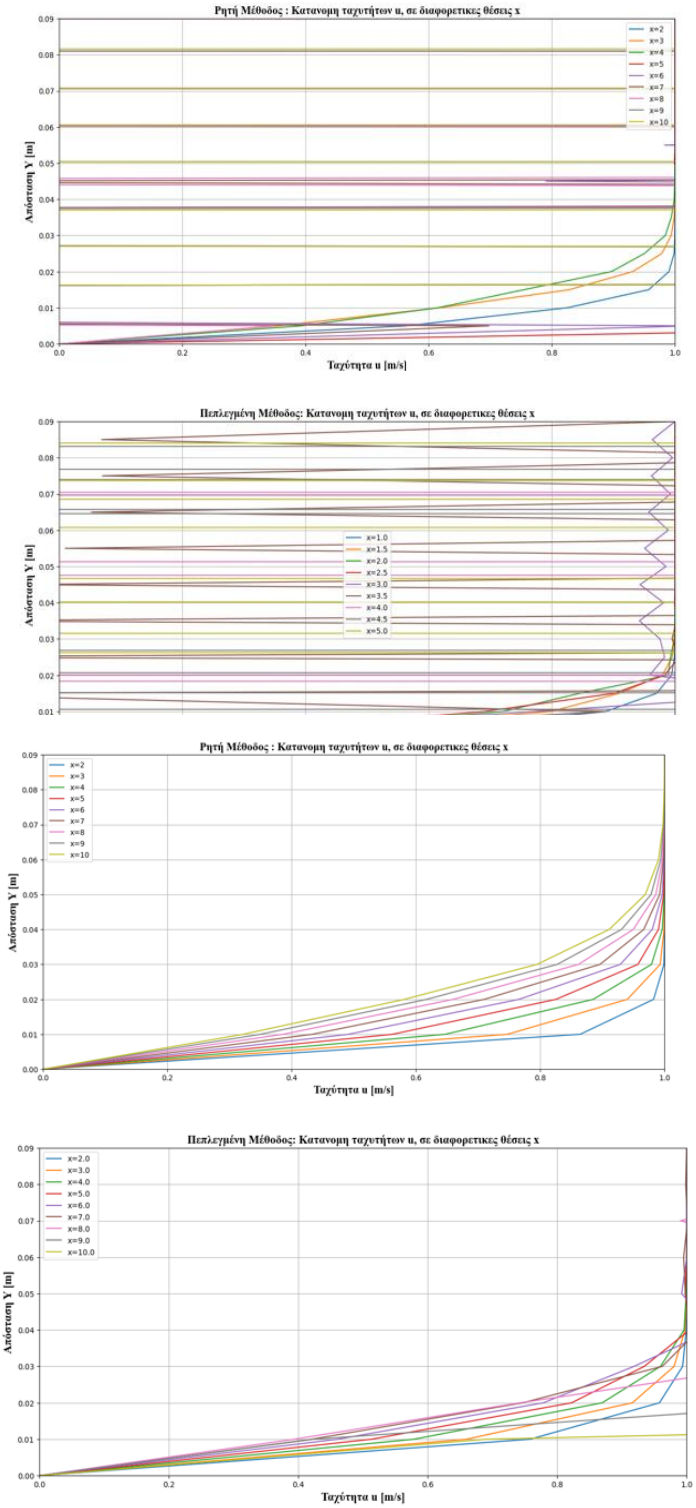


Figure 9. Comparison of velocity distribution diagrams for different discretization step in the x direction

For the given steps neither of the 2 methods can resolve the boundary layer. Specifically we get the distributions shown in Figure 8. The horizontal lines are due to infinite values for the velocities. We see that both the explicit and the convoluted method can only give a solution up to the position x=3 meters. From this point onwards, errors grow and the code becomes unstable. So we can confidently say that the code has a high dependence on the discretization step.

ation

Now doubling only Δx and keeping Δy constant we get the diagrams of Figure 9. We see that with the explicit method, we end up with a solution in the same expensive form as before. But for the tangled method, from an x and above, the code cannot give us a solution and gives us the same strange behavior as before.

5. COMPARISON WITH BLASIUS

In the last question of the paper, we are asked to compare the values we calculate for the boundary layer with the quantities calculated from the Blasius equations, for the following quantities:

- Boundary layer thickness$\delta = \frac{5x}{\sqrt{Re_x}}$
- Offset thickness$\delta_1 = \frac{1.72x}{\sqrt{Re_x}}$
- Momentum loss thickness$\delta_2 = \frac{0.664x}{\sqrt{Re_x}}$
- Coefficient of Friction : Blasius$C_f = \frac{0.664}{\sqrt{Re_x}}$

The last equation of the coefficient of friction, to be compared with the value of the coefficient of friction that will result from the equation.$C_f = \frac{\tau_{wall}}{0.5\rho U_\infty^2}$

First we will create the equations with which we will calculate the quantities we want to compare.

We know that the thickness of the boundary layer is the height at which the velocity u first reaches 99% of the free stream velocity. Therefore we construct within the code for both methods, an iterative process by which we find in each column of nodes, the first node with speed , and define the height y of this node as the thickness of the boundary layer$99\% U_\infty$

$$\delta_{0.99} = y(u = 0.99U)$$

The boundary layer displacement thickness is calculated from the integral:δ_1

$$\delta_1 = \int_0^\infty \left(1 - \frac{u}{U_\infty}\right) dy$$

Since we have discrete points and not a continuous field of variables, we convert the integral into a sum and thus calculate it from the equation:δ_1

$$\delta_1 = \sum_{j=0}^{n} \left(\left(1 - \frac{u_{j,i}}{U_\infty}\right) \Delta y\right)$$

Thus we calculate for each position x the displacement thickness.δ_1

The momentum loss thickness of the boundary layer is also calculated from an integral which, due to discrete points, is also converted into a sum, so we have:

$$\delta_2 = \int_0^\infty \frac{u}{U_\infty}\left(1 - \frac{u}{U_\infty}\right) dy \Rightarrow \delta_2 = \sum_{j=0}^{n} \left(\frac{u_{j,i}}{U_\infty}\left(1 - \frac{u_{j,i}}{U_\infty}\right) \Delta y\right)$$

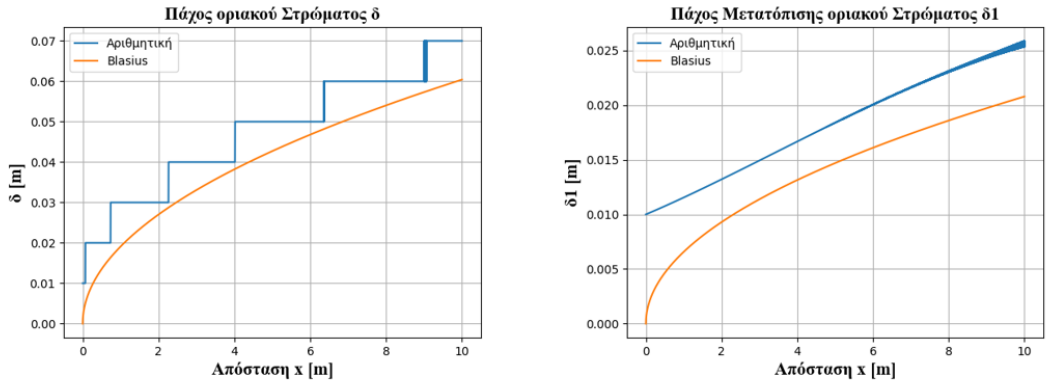
Therefore with the same procedure as for the displacement thickness of the boundary layer, we calculate for each position x, of the plate.δ_2

The coefficient of friction is calculated from Eq$C_f = \frac{\tau_{wall}}{0.5\rho U_\infty^2}$which is given to us in the utterance.

Having constructed all our equations, we proceed to present the comparison plots between the Blasius solutions and the numerical solutions. We will present the comparison charts for both methods separately.

5.1 Comparison of explicit Method – Blasius

In Figures 10 and 11, we see the comparison of the 2 solutions.



Shape10. Comparison of Numerical solutions – with Blasius solutions for andδ δ_1

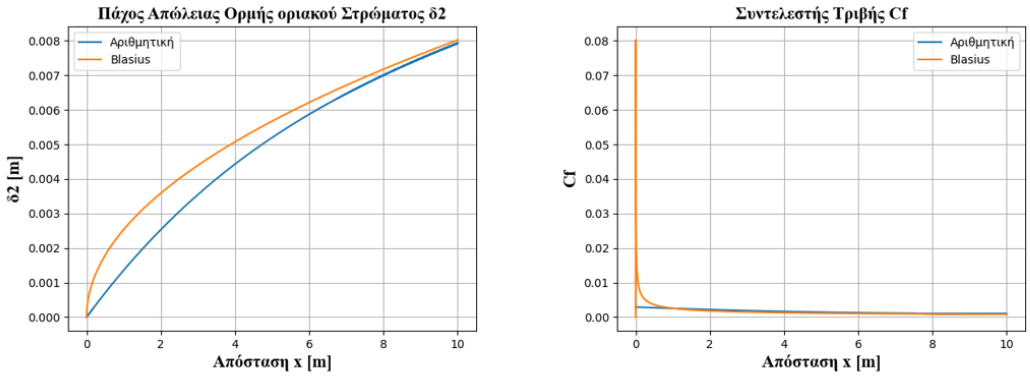
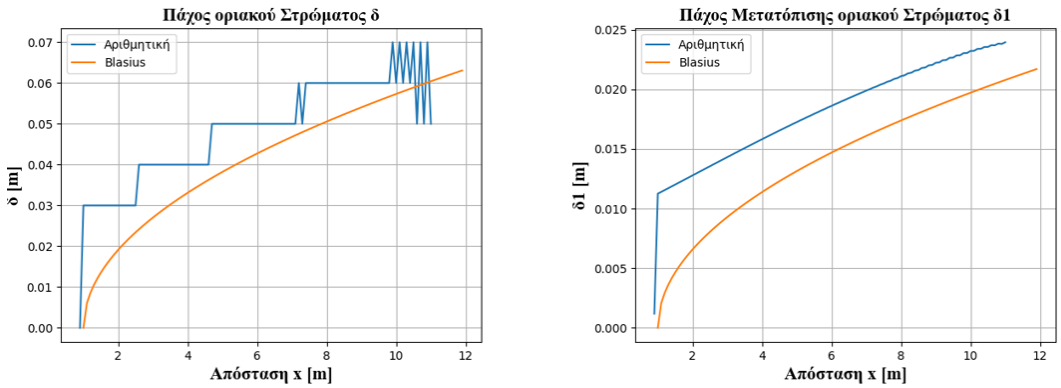


Figure 11. Comparison of Numerical solutions – with Blasius solutions for andδ_2 C_f

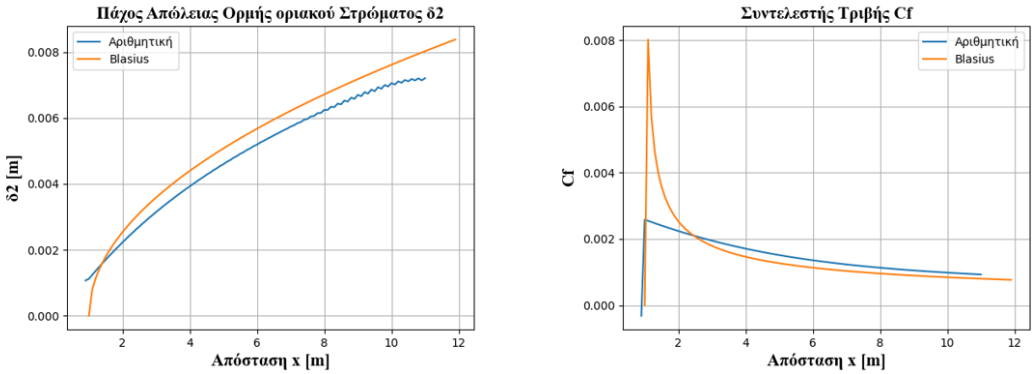
Observing the 2 figures above, we see that with the explicit method we satisfactorily approximate the Blasius solution for the momentum loss thickness and for the friction coefficient. For the thickness of the boundary layer from the numerical solution we have a graph of the digital signal form, this behavior is due to the size of the discretization and the condition we define to find these values of the thickness inside the code. The biggest deviation for all the quantities under study is found in the displacement thickness of the boundary layer. By choosing smaller discretization steps, particularly in the y direction, we could better approximate the Blasius solutions. In general, however, we can say that the algorithm satisfactorily approximates the Blasius solutions and in addition we get a confirmation of the correct operation of our code.($\acute{\epsilon}\chi\omega\ \delta\ \acute{o}\tau\alpha\nu\ u < 0.99$)

5.2 Comparison of Implicit Method – Blasius

In Figures 12 and 13, we see the comparison of the solutions of the Convoluted Method and the Blasius solution. To reduce the noise of the numerical system and see as few perturbations as possible, we run the code for a grid that extends one meter in front of the plate and ends at the end of the plate.



Shape22. Comparison of Numerical solutions – with Blasius solutions for and$\delta\,\delta_1$



Shape33. Comparison of Numerical solutions – with Blasius solutions for and$\delta_2 C_f$

In all figures we notice at the end of the plate that some form of oscillations appear in the values (noise). This behavior is due to the increase in errors and the onset of code instability. Furthermore, even though our grid extends 1 meter in front of the plate, this part is not included in the diagrams since the equations do not apply. Observing now the rest of the plate we can see that here too we have a satisfactory approximation of the values of the Blasius solution. In comparison, however, for the 2 methods from the 4 previous figures, we can say that the explicit method exhibits more satisfactory behavior, in terms of convergence with the Blasius solution and also in terms of numerical stability.

As we mentioned above, in general the entangled methods show better stability and accuracy, but they are more computationally demanding and depend in particular on the discretization of the equations. Therefore with more computing power we can say that any instabilities that appear now could be avoided. We would also have the possibility of better and more

extensive discretization in our model. Finally, we can say that both methods can give us a first picture of the boundary layer above the slab.