HOMEWORK-1

HAZIRLAYAN

ADI SOYADI : ENES ŞAHINER

ÖĞRENCI NUMARASI : 030200018

TESLIM TARIHI : 10/03/2024

DERS ADI : COMPUTATIONAL FLUID DYNAMICS

DERS YÜRÜTÜCÜSÜ : SERTAÇ ÇADIRCI

Table of Contents

What is the Blasius Equation?

The Blasius equation is a fundamental equation in fluid mechanics that describes the boundary layer flow over a flat plate. It is a dimensionless, ordinary differential equation derived from the Navier-Stokes equations and is commonly used to analyze laminar flow regimes. The equation relates the dimensionless velocity gradient to the dimensionless distance from the leading edge of the plate. Solving the Blasius equation provides insight into important parameters such as skin friction coefficient and boundary layer thickness, crucial in engineering applications like aerodynamics and heat transfer analysis.

Conservation of mass:

$$(\delta u/\delta x) + (\delta v/\delta y) = 0$$

Conservation of x-momentum:

$$u * (\delta u/\delta x) + v * (\delta v/\delta y) = -(1/\rho) * (dP/dx) + \nu * \delta^2 u/\delta y^2$$

Conservation of y-momentum:

$$\delta P/\delta y = 0$$

Hydrodynamic solutions:

$$\frac{\delta u}{\delta x} + \frac{\delta v}{\delta y} = 0 \qquad u * (\delta u/\delta x) + v * (\delta v/\delta y) = \nu * \delta^2 u/\delta y^2$$

Defining a stream function, φ:

$$u \equiv \frac{\delta \varphi}{\delta y} \; and \; v \equiv -\frac{\delta \varphi}{\delta x}$$

So,

$$\frac{\delta^2 \varphi}{\delta x * \delta y} - \frac{\delta^2 \varphi}{\delta x * \delta y} = 0$$

Such that mass conservation is guaranteed.

$\varphi = \int u dy$, so, the value of φ at any point above the plate is the volume flow rate between the plate and that point.

Defining similarity parameter:

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

Defining non-dimensional stream function:

$$f(\eta) = \frac{\varphi}{U_\infty * \sqrt{\frac{\nu x}{U_\infty}}}$$

Writing x-momentum in terms of f and η:

u velocity:

$$* u \equiv \frac{\delta\varphi}{\delta y} = \frac{\delta\varphi}{\delta \eta} * \frac{\delta\eta}{\delta y} \qquad * \frac{\delta\eta}{\delta y} = \sqrt{\frac{U_\infty}{\nu x}}$$

$$* \varphi = U_\infty * \sqrt{\frac{\nu x}{U_\infty}} * f(\eta) \qquad * \frac{\delta\varphi}{\delta \eta} = U_\infty * \sqrt{\frac{\nu x}{U_\infty}} * f'(\eta)$$

$$u = U_\infty * \sqrt{\frac{\nu x}{U_\infty}} * f'(\eta)\sqrt{\frac{U_\infty}{\nu x}}$$

So, $u = U_\infty * f'(\eta)$

v velocity:

$$* v \equiv -\frac{\delta\varphi}{\delta x} \qquad * \varphi = U_\infty * \sqrt{\frac{\nu x}{U_\infty}} * f(\eta)$$

$$v = \frac{1}{2}\sqrt{\frac{\nu U_\infty}{x}} * (\eta f' - f)$$

Preparing the equations for Blasius Equation:

$$\frac{\delta u}{\delta x} = -\frac{1}{2}U_{\infty}\eta f''$$

$$\frac{\delta u}{\delta y} = U_{\infty}\sqrt{\frac{U_{\infty}}{\nu x}}f''$$

$$\frac{\delta^2 u}{\delta y^2} = \frac{U_{\infty}^2}{\nu x} * f'''$$

Putting it all together, the equation is Blasius Equation:

$$2f''' + f * f'' = 0$$

Defining Boundary Conditions:

$f(0) = f'(0) = 0$ →No flow and no velocity at the plate surface

$f'(\infty) = 1$ → Free stream velocity for from surface

Solving the Blasius Equation, Part A

To solve the equation with Centered FDD numerical method, the equation

$$f'(\eta) = g(\eta)$$

must be written.

So, new equation is:

$$g''(\eta) + \frac{1}{2}f(\eta) * g'(\eta) = 0$$

Applying Centered FDD:

$$g'' = \frac{g_{i+1} - 2g_i + g_{i-1}}{(\Delta\eta)^2}$$

$$g' = \frac{g_{i+1} - g_{i-1}}{2 * \Delta\eta}$$

When replace these g’ and g’’ in the equation, the equation will be like Thomas Algorithm:

$$g_{i+1}\left(2 + \frac{f_i}{2} * \Delta\eta\right) - 4g_i + g_{i-1}\left(2 - \frac{f_i}{2} * \Delta\eta\right) = 0$$

To write a TDMA, coefficients must be defined:

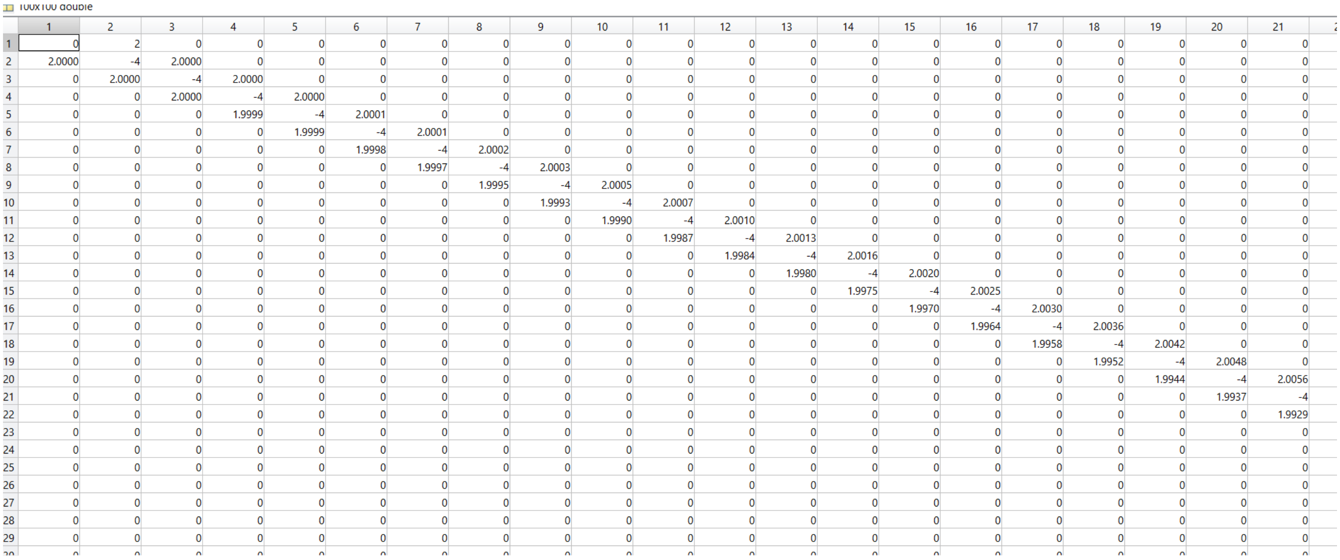
$b = 2 + \frac{f_i}{2} * \Delta\eta$, $a = -4,$ $c = 2 - \frac{f_i}{2} * \Delta\eta,$ $d = 0$

In the lecture notes, TDMA must be like this:

$$\begin{array}{cccccccccccc}
a_1 y_1 & + & b_1 y_2 & & & & & & & = & d_1 & - & c_1 y_0 \\
c_2 y_1 & + & a_2 y_2 & + & b_2 y_3 & & & & & = & d_2 & & \\
 & & c_3 y_2 & + & a_3 y_3 & + & b_3 y_4 & & & = & d_3 & & \\
 & & & & & & \vdots & & & & & & \\
 & & & & & & c_k y_{k-1} & + & a_k y_k & + & b_k y_{k+1} & = & d_k \\
 & & & & & & & & \vdots & & & & \\
 & & & & & & & & c_{n-2} y_{n-3} & + & a_{n-2} y_{n-2} & + & b_{n-2} y_{n-1} & = & d_{n-2} \\
 & & & & & & & & & & c_{n-1} y_{n-2} & + & a_{n-1} y_{n-1} & = & d_{n-1} & - & b_{n-1} \underbrace{y_n}_{known}
\end{array}$$

Figure 1. Tri-diagonal matrix form from lectures notes

Using a for loop, a 100x100 Tri-diagonal matrix is obtained:


100x100 double

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	2.0000	-4	2.0000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	2.0000	-4	2.0000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	2.0000	-4	2.0000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	1.9999	-4	2.0001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	1.9999	-4	2.0001	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	1.9998	-4	2.0002	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	1.9997	-4	2.0003	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	1.9995	-4	2.0005	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	1.9993	-4	2.0007	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	1.9990	-4	2.0010	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	1.9987	-4	2.0013	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	1.9984	-4	2.0016	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	1.9980	-4	2.0020	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	1.9975	-4	2.0025	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.9970	-4	2.0030	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.9964	-4	2.0036	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.9958	-4	2.0042	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.9952	-4	2.0048	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.9944	-4	2.0056
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.9937	-4
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.9929
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2. an image from the calculated Tri-diagonal matrix.

A.1

When the matrix is solved, g function would be found. The below graph represents to comparative plot of g and f functions.

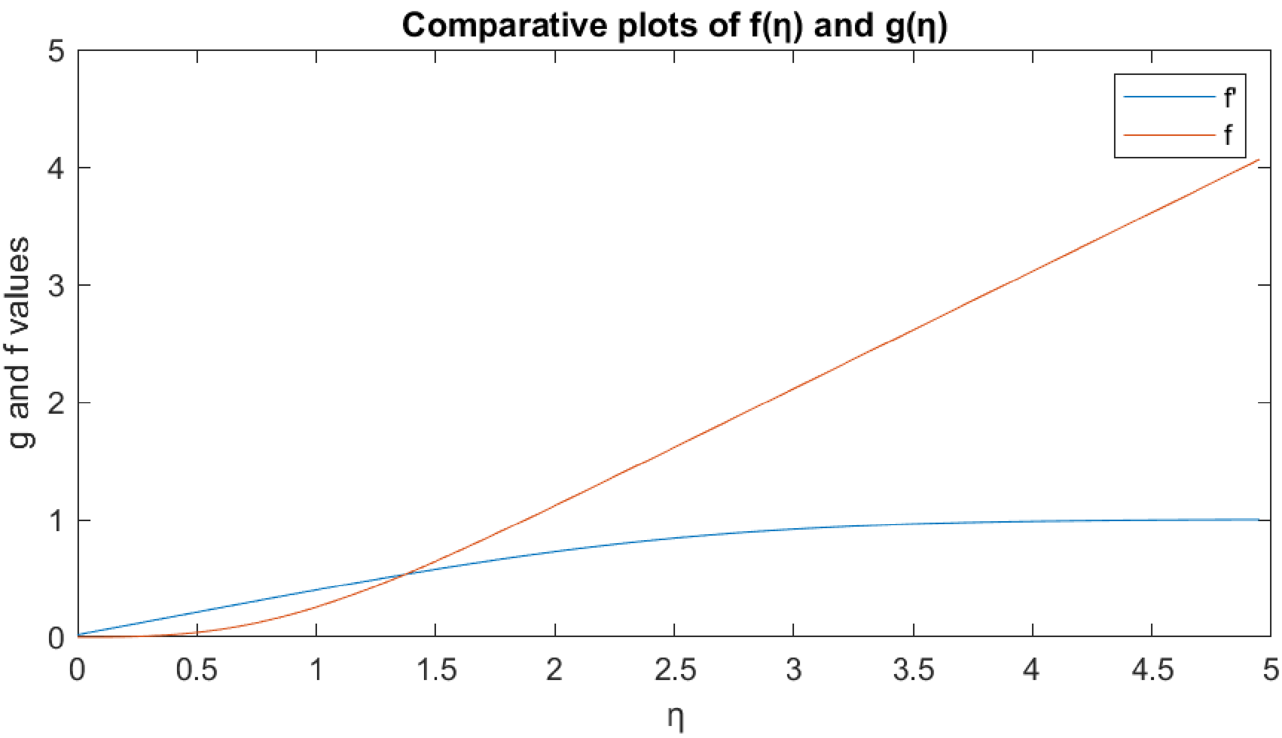


Figure 3. Comparative plots of f and f'.

The graph shows that f(η) starts at zero and increases monotonically to a positive value as η approaches infinity. This indicates that the velocity of the fluid increases from zero at the wall to a constant value far away from the wall. f'(η), on the other hand, starts at zero and then decreases monotonically to zero as η approaches infinity. This means that the rate of change of the velocity (acceleration) is highest at the wall and decreases as moving away from the wall.

Also, boundary conditions are satisfied.

$$f(0) = f'(0) = 0 \qquad\qquad f'(\infty) = 1$$

A.2

As mentioned, the Boundary Layer Theory Lectures' Notes by Hasan Güneş, the velocity profile must be like this:

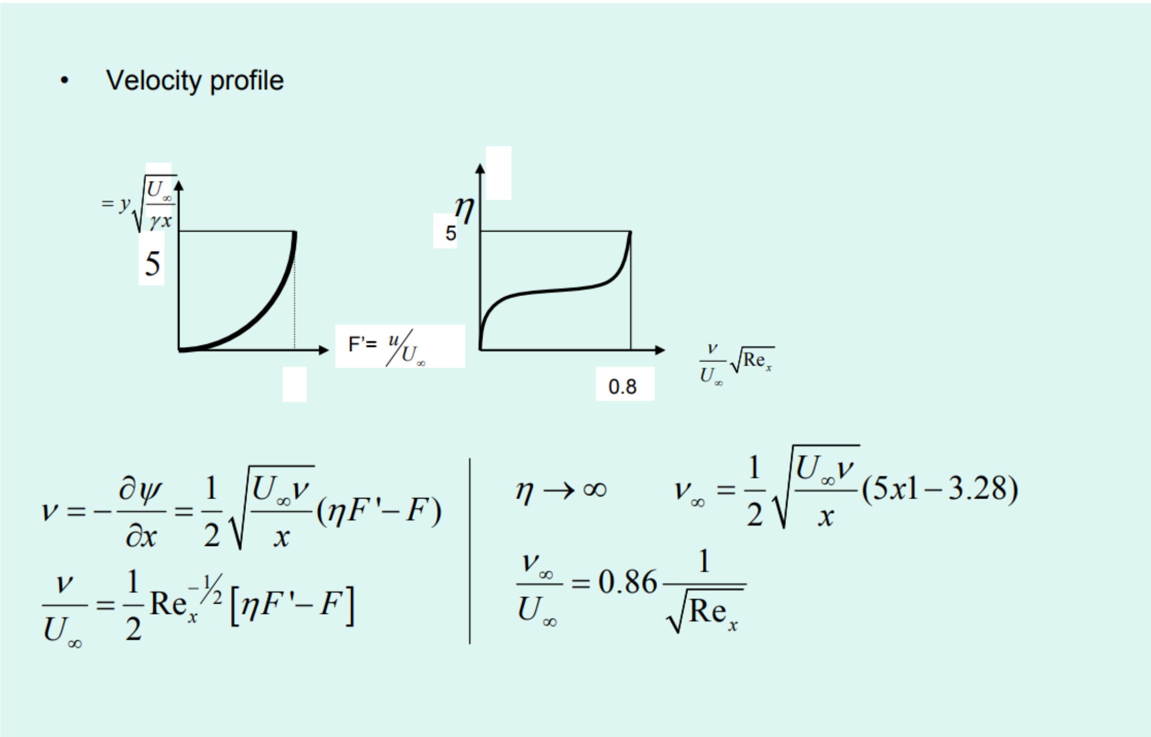


Figure 4. Example graph of velocity profile.

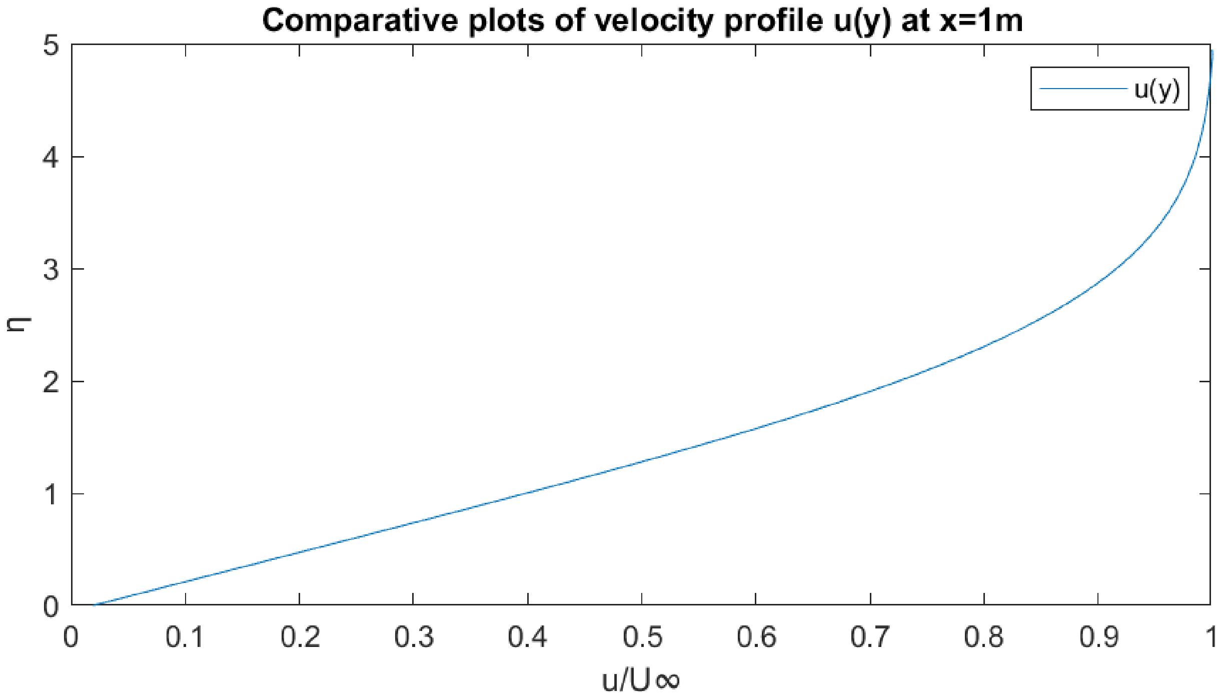


Figure 5. Velocity profile graph.

The graph shows that the velocity (u/U_∞) starts at zero at the wall ($\eta = 0$) and increases towards one as η approaches infinity. This indicates that the fluid velocity increases from zero at the wall to the freestream velocity (U_∞=5m/s) far away from the wall. This is the expected behavior for a boundary layer flow over a flat plate.

A.3

For the shear stress, this equation is used:

$$\tau = \mu * U_\infty \sqrt{\frac{U_\infty}{\nu x}} * f''$$

Here, $\mu = 1.78 * 10^{-5}, U_\infty = 5\frac{m}{s}, \nu = \frac{\mu}{\rho} = \frac{1.78*10^{-5}}{1.225} = 1.418 * 10^{-5}\ [m^2/s]\ , f''(0) =\ 0.3827$

.

So, $\tau = 0.0175 * \sqrt{\frac{1}{x}}$

The graph is:

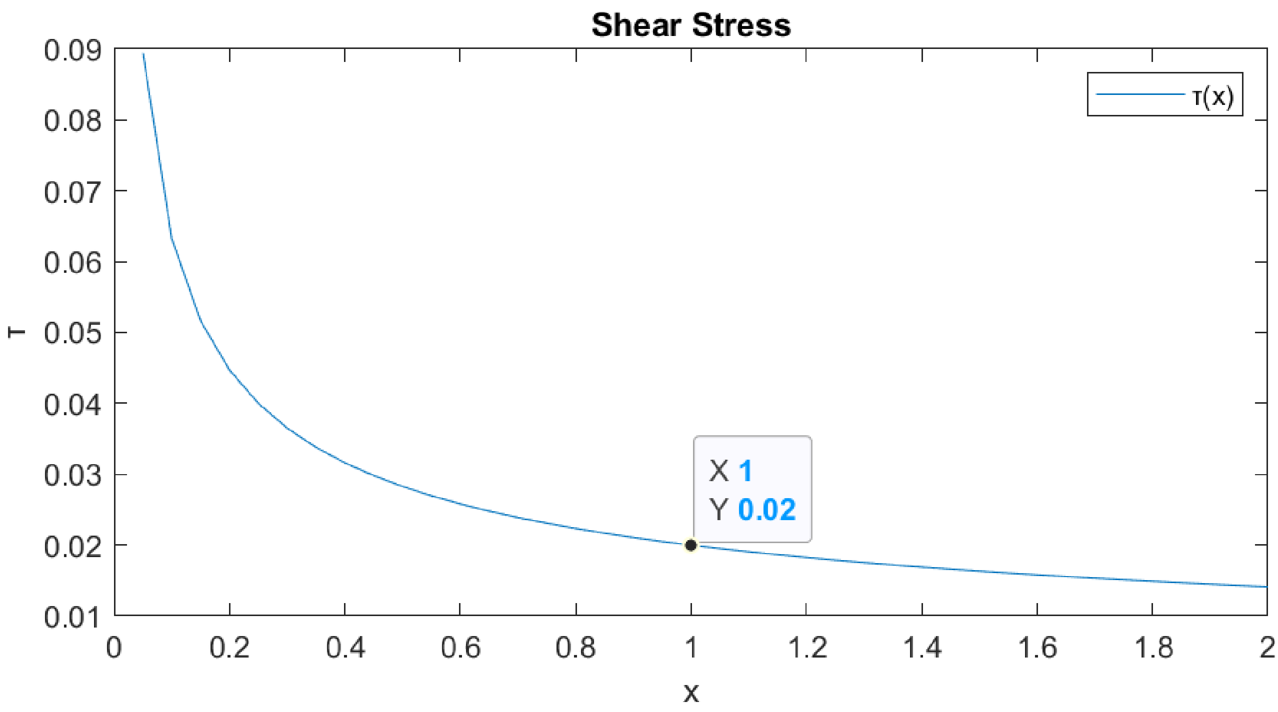


Figure 6. Shear stress distribution.

The friction experienced by the fluid in the boundary layer results in a shear stress. This stress acts tangentially to the wall, opposing the flow of the fluid.

For the skin friction coefficients, this equation is used:

$$C_{Df} = \frac{\tau}{0.5 * \rho * U^2{}_{\infty}}$$

So,

$$C_{Df} = \frac{\tau}{0.5 * 1.225 * 25}$$

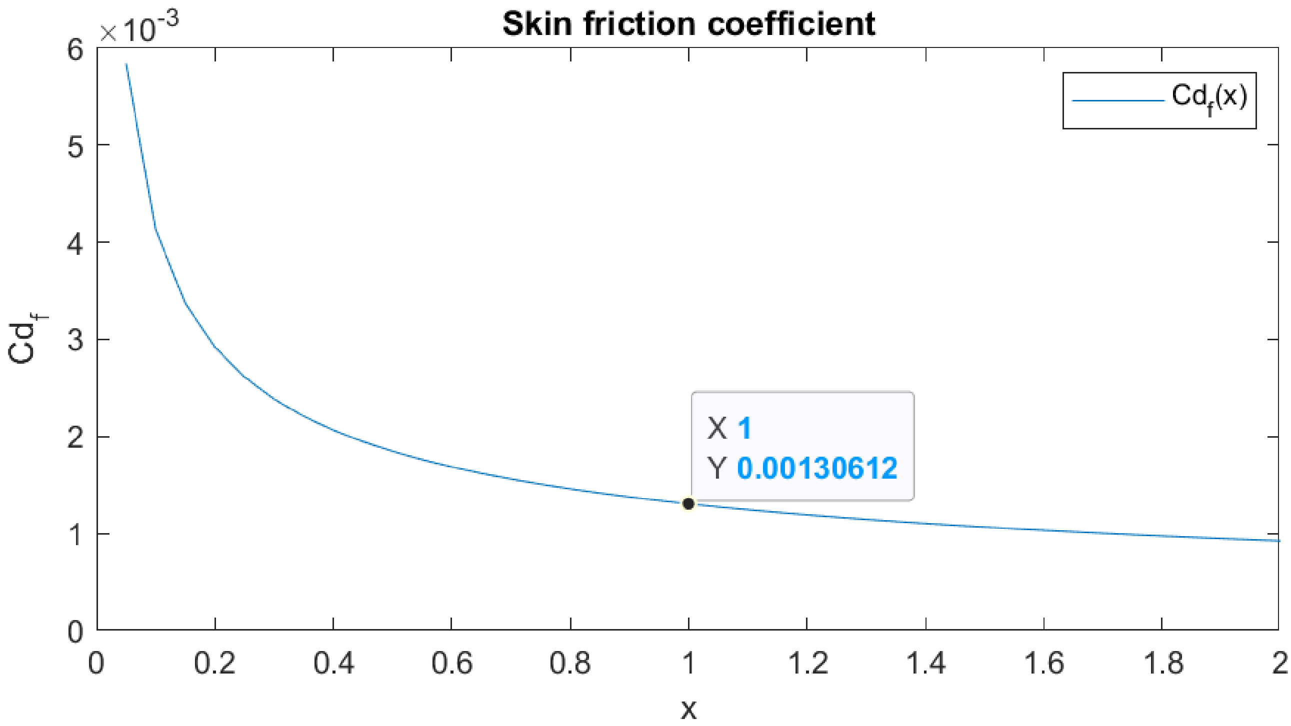


Figure 7. Skin friction coefficient distribution.

Solving the Blasius Equation, Part B

The shooting method is a numerical technique used to solve a specific type of differential equation called a boundary value problem (BVP). Unlike initial value problems (IVPs) where you have initial conditions at a single point, BVPs have conditions specified at two different points within the solution domain

Unlike IVPs where you start with initial conditions and solve forward, directly solving BVPs can be more complex due to the two boundary conditions at different locations.

B.1

When the ODE is solved with Runge-Kutta Method, g function would be found. The below graph represents to comparative plot of g and f functions.

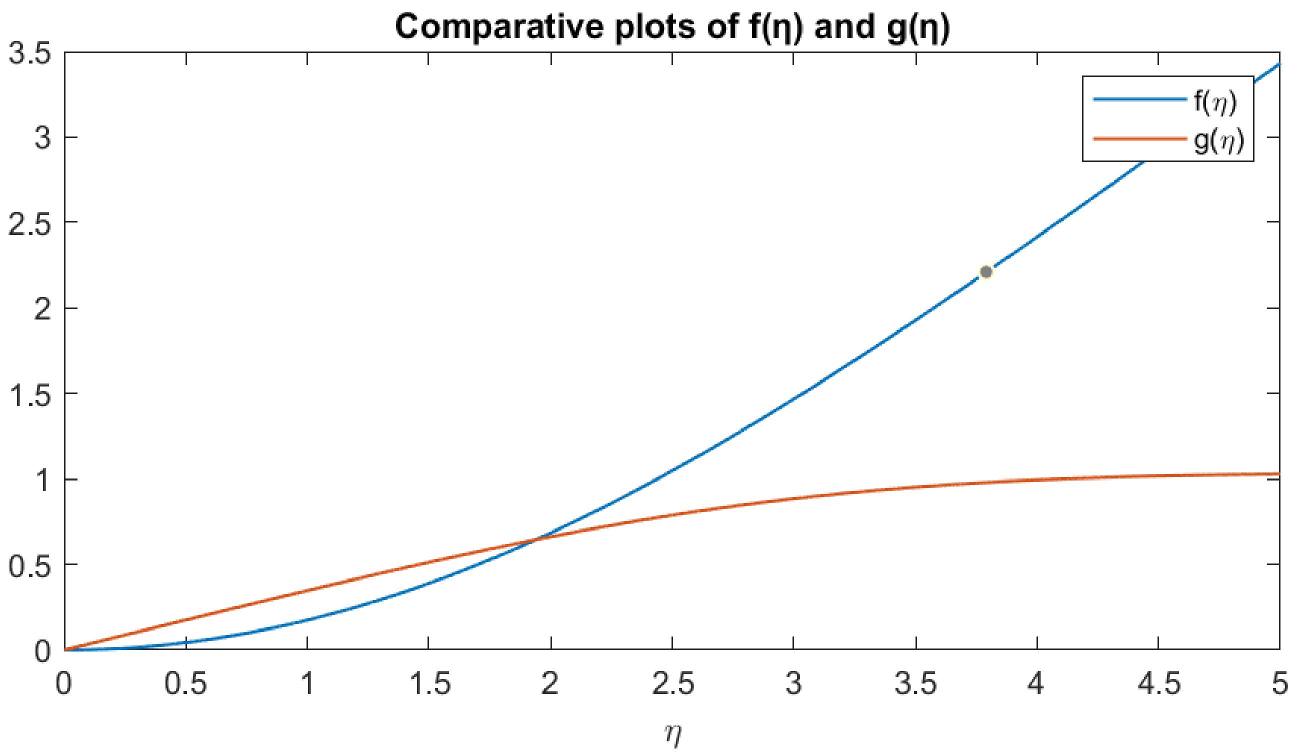


Figure 8.Comparative plots of f and f'.

The comments made in section A.1 are also valid here.

B.2

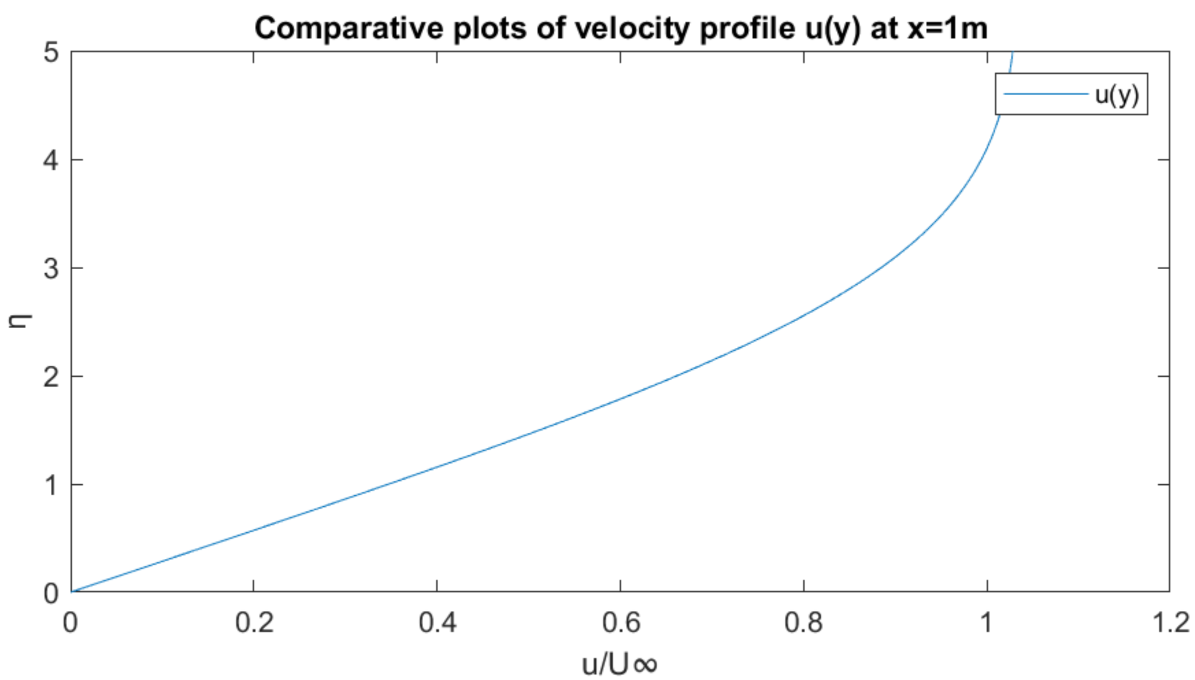


Figure 9.Velocity profile graph.

The equations used in part A.2 are also used here. The graph was similar to the graph shown as the source.

B.3

Here:

$$f''(0) = .35$$

However, this value was not found by the Regula-Falsi method as requested in the homework, but by trial and error by checking the convergence of the g graph to 1. Other values and equations are same with part A.3.

For the shear stress:

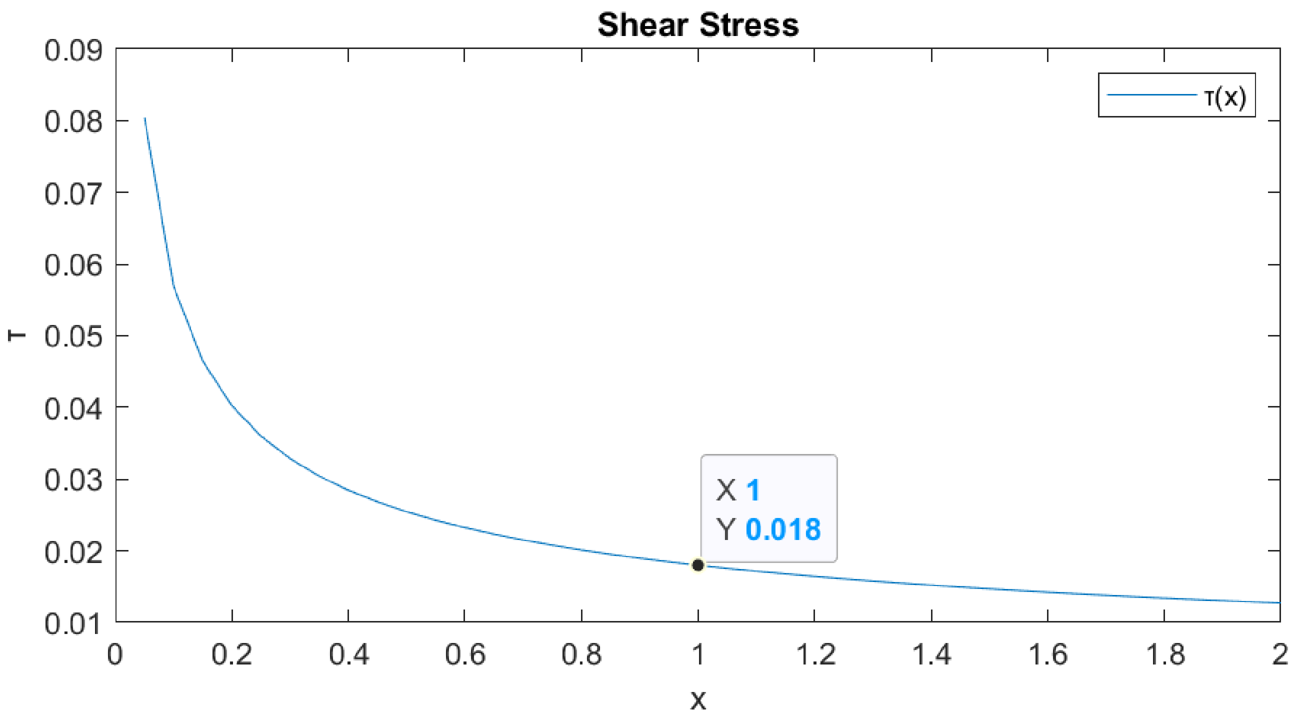


Figure 10. Shear stress distribution.

For the skin friction coefficients:

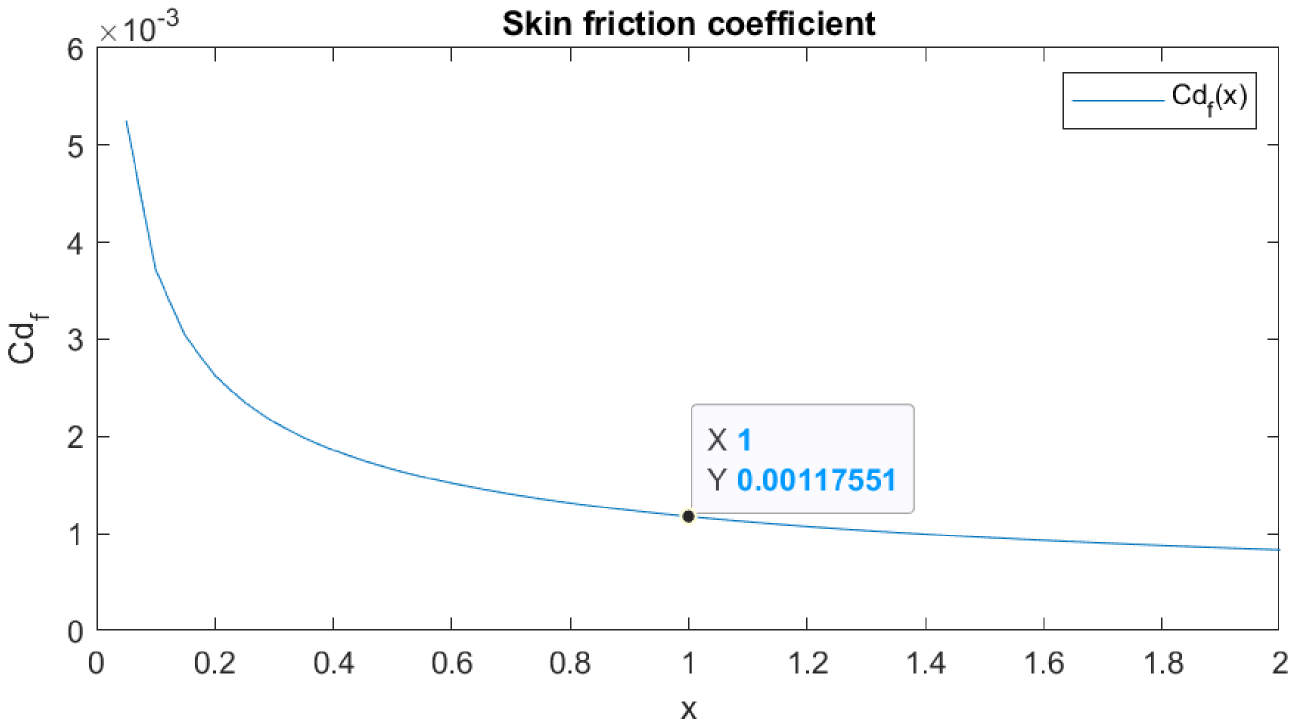


Figure 11. Skin friction coefficient distribution.

References

1. Boundary Layer Theory Lecture Notes, Güneş, H.
   www.web.itu.edu.tr/~guneshasa/viscous/blt.pdf
2. CFD Lecture Notes, Çadırcı, S.
3. *Solving Blasius Equation Using RK-4 Numerical method*. (2021, December 16). Solving Blasius Equation Using RK-4 Numerical Method - File Exchange - MATLAB CentralFile Exchange - MATLAB Central. https://www.mathworks.com/matlabcentral/fileexchange/102189-solving-blasius-equation-using-rk-4-numerical-method