Indian Institute of Technology Gandhinagar

—---

MA 202: Numerical Methods

Project Report

Project: 12

Group: 42

Professor: Prof. Uddipta Ghosh

Title of the proposed project: Solution of the Blasius equation (Boundary Layer Theory) using the Shooting Method.

Ruchika Sonagote	21110212
Harshal Sonawane	21110213
Animesh Tumne	21110227
Prashant Malav	19110056

Introduction:

Boundary layer theory is an important issue in fluid dynamics since it provides a framework for understanding the behavior of fluid flows near boundaries. To find the laminar flow of an incompressible fluid over a flat plate, we use the Blasius equation. It is a third-order ordinary differential equation in boundary layer theory. It is difficult to discover analytical solutions for the Blasius equation, so numerical approaches such as the shooting method are used to identify approximations. In the shooting method, the differential equation is repeatedly solved under various initial conditions until the solution satisfies the required boundary condition. So, here we will be finding the solution of Blasius equation (boundary layer theory) using the shooting method and providing velocity profile and boundary layer thickness, which is useful in the design of the fluid systems by the Runge-Kutta fourth order (RK4) method.

Blasius equation:

The Blasius equation is a famous nonlinear ordinary differential equation that explains the incompressible, laminar flow of a fluid over a flat plate.

Blasius equation:

$$f''' + \frac{1}{2} f f'' = 0$$

where f signifies differentiation with respect to the dimensionless variable η which is defined as:

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

Where y = distance from the plate surface,
U_∞ = free-stream velocity,
ν = fluid's kinematic viscosity, and
x = the distance along the plate surface.

Shooting method:

The shooting method is a numerical methodology for solving ordinary differential equations (ODEs) which have boundary conditions. It converts a boundary value problem (BVP) to an initial value problem (IVP) and predicts an initial value for the solution and then repeatedly changes the original guess until the necessary boundary conditions are satisfied.

The following are the basic steps in the shooting method:

- Convert the BVP to an IVP by assuming an initial solution value at one of the boundary points.
- Solve the resulting IVP using a numerical integration method such as the Runge-Kutta method.
- Check if the resulting solution meets the other boundary condition. If it does, the issue is resolved. If not, revise the initial guess and continue the process until we attain the desired accuracy iteratively.

Interpolation:

At some x = p_1, y = a_1 and x = p_2, y = a_2, then to calculate the x value at some y = a_3, we do linear interpolation as

$$(p_2 - p_1)/(a_2 - a_1) = (p_2 - p_3)/(a_2 - x)$$

$$x = (p_2 - p_1)/(a_2 - a_1) * (1 - a_1) + p_1$$

where $a_1 < a_3 < a_2$

Solution:

Blasius equation:

$$f''' + \frac{1}{2}ff'' = 0$$

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

Boundary conditions $f(0) = 0$, $f'(0) = 0$ and $f'(\eta) = 1$ $as\, \eta \to \infty$

Here $f \to x\ and\ \eta \to t$

We can write this equation as

$$\frac{d^3x}{dt^3} + \frac{1}{2}x\frac{d^2x}{dt^2} = 0$$

To convert this equation into a system of three first-order equations, we define three new variables

$$x = x_1\ ,\ \frac{dx_1}{dt} = x_2\ ,\ \frac{dx_2}{dt} = x_3$$

Now we can write Blasius equation as $\frac{dx_3}{dt} + \frac{1}{2}x_1x_3 = 0$

Now we have a system of three first-order equations:

$$\frac{dx_1}{dt} = x_2\ ,$$
$$\frac{dx_2}{dt} = x_3\ ,$$
And
$$\frac{dx_3}{dt} = -\frac{1}{2}x_1x_3$$

where $x_1(0) = 0,\ x_2(0) = 0,\ x_3(0) = ?$

We will guess two initial values for $x_3(0)$, and then do interpolation to satisfy $x_2(\infty) = 1$.

Code Explanation:

Initial guesses: η = 5
We take the first two initial guesses (for $x_3(0)$) as p1 = 0.1 and p2 = 1. From these, the values of $x_2(\eta)$ comes out to be 0.4025 and 2.0853, where η = 5. Then we use the linear interpolation technique repetitively until $x_2(\eta) \cong 1$ (tolerance = 0.001). To calculate the value of $x_2(\eta)$, we use the RK method of the fourth order: listed below

RK method of the fourth order:

In this method, we find out the solution to n-coupled differential equations of the first order (given their initial values) iteratively. For the given statement, we don't have n coupled linear differential equations, so we have to convert the Blasius equation into n-coupled linear equations, which can be done by:

$$f''' + \frac{1}{2}ff'' = 0$$

- Blasius Equation

-

Which, converted to 3 coupled linear differential equations:

$$y_1 = f,\ y_1' = y_2$$
$$y_2 = f',\ y_2' = y_3$$
$$y_3 = y_2',\ y_3' = -\frac{1}{2} * y_1 * y_3$$

Where y' = the differential of y with respect to t (here, η) at the current iteration

Initial values:

$$y_1(\eta = 0) = 0$$
$$y_2(\eta = 0) = 0$$
$$y_3(\eta = 0) = ?$$

Now, since we have three coupled linear differential equations, we can easily apply the Runge Kutta method to these, provided that we have the initial values to the equations.

But, since we only have two initial values, we have to use the shooting method (explained above) to interpolate the third initial value.
We initially interpolate between p_1 and p_2:

$$y_3(\eta = 0) = p$$
$$p_1 < p < p_2$$
$$p_1 = 0.1$$
$$p_2 = 1$$

Now, the RK4 method is written as a function, whose inputs are the interpolated values of p each iteration, and as the iterations march on, we get the plotting according to these values, which can be seen to be converging to

$$p = 0.3362$$

Now, we have all the initial guesses and all the coupled equations ready, we just now need to plug these values into the functional code and get the output.

How does the RK4 method work?
The RK4 method was developed in a way that it takes in n coupled linear equations, and produces the solution in a particular time interval that we need them to be solved, i.e., the range of the independent variable must be defined.

This method is iterative; the value of the function, $\overline{y}$, at the (i+1)th iteration is calculated using:

$$\overline{y}(t_{i+1}) = \overline{y}(t_i) + \frac{1}{6}(\overline{k}_1 + 2\overline{k}_2 + 2\overline{k}_3 + \overline{k}_4)$$
$$\overline{k}_1 = \overline{f}(t_i, \overline{y}_i)$$
$$\overline{k}_2 = f(\overline{t}_i + dt/2, \overline{y}_i + (\overline{k}_1 * dt)/2)$$
$$\overline{k}_3 = f(\overline{t}_i + dt/2, \overline{y}_i + (\overline{k}_2 * dt)/2)$$
$$\overline{k}_4 = f(\overline{t}_i + dt, \overline{y}_i + (\overline{k}_3 * dt))$$

Where:

$$\overline{y}_i = \overline{y}(t_i)$$
$$\overline{y}' = \overline{f}(\overline{y}, t)$$

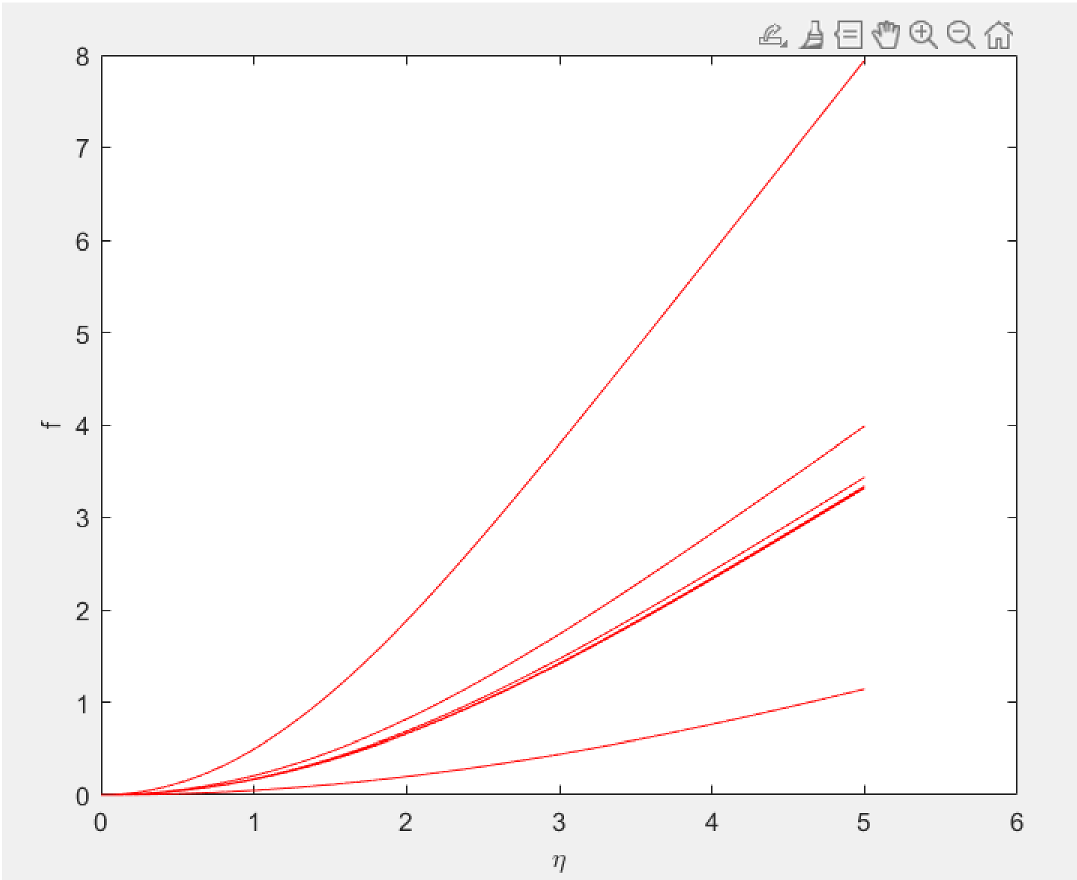
Note -

- In general, $\overline{v}$ is a vector array.
- $\overline{f}$ is the system of linear ODE's.

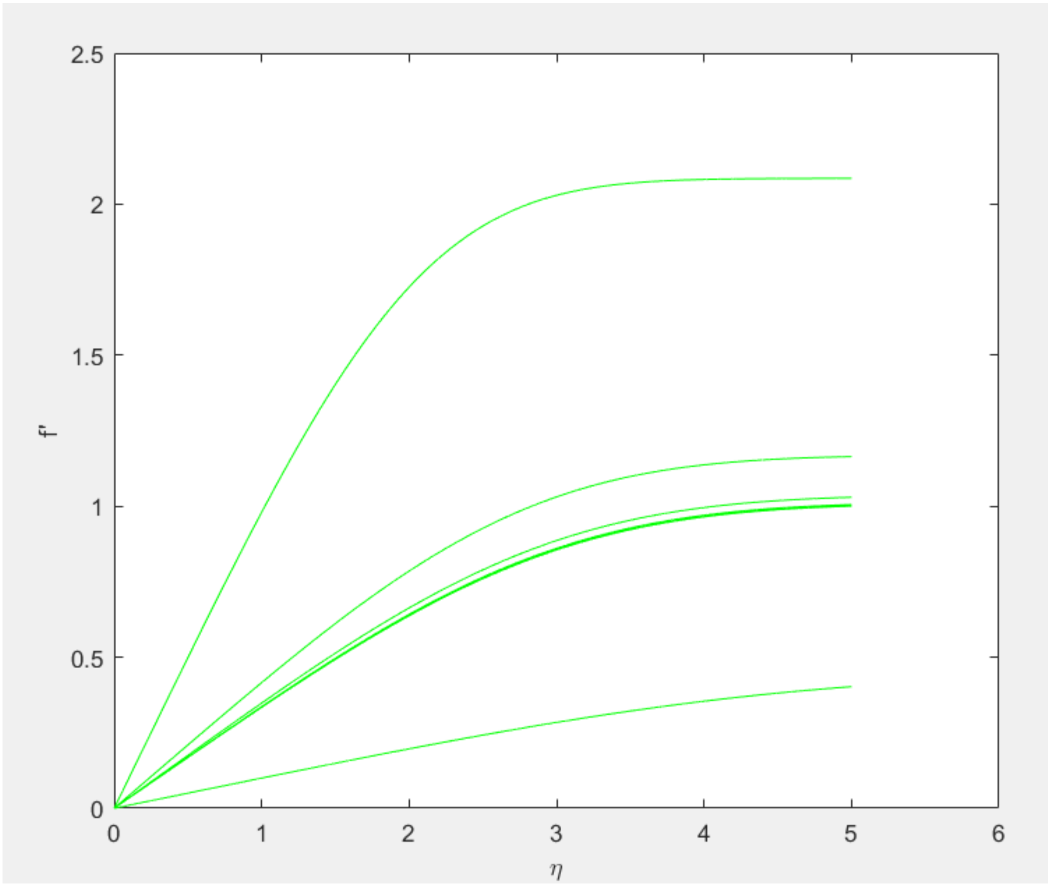
We have inputted the initial y as y = [0 0 p], where p is the value using the shooting method, and we subsequently get the value we need as the output.

The code outputs the following graphs:

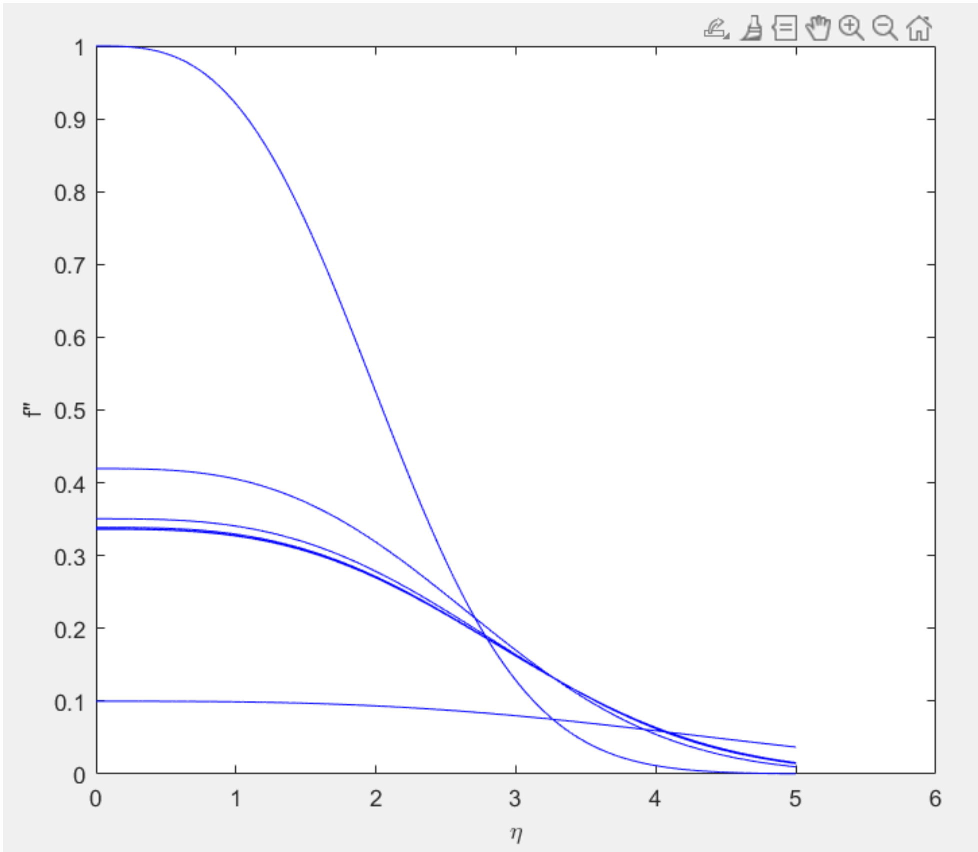
1. **f vs η**



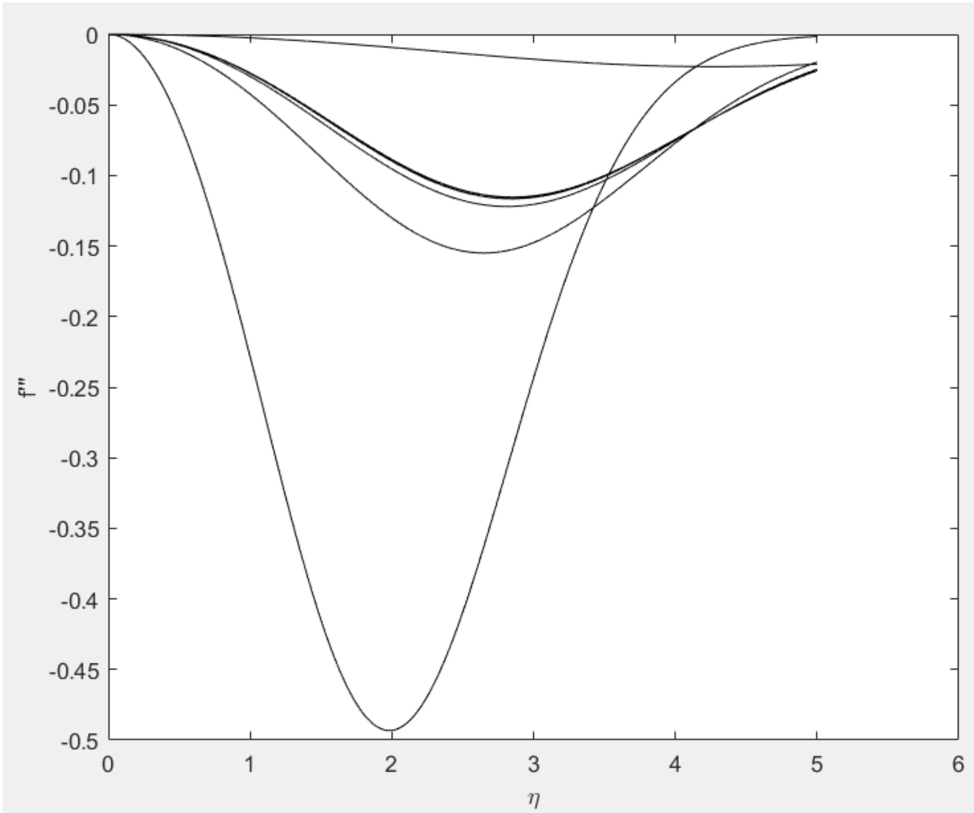
2. f' **vs** η



3. f'' **vs** η



4. f''' **vs** η



References:

1. A. Bandopadhyay, “[GNU OCTAVE] L4 Blasius equation and shooting method,” 05-Dec-2018. [Online]. Available: https://youtu.be/95ATrAZdIqA. [Accessed: 26-Apr-2023].