Task 2: Numerical Solution of Boundary Layer Equation

Uday Bhaskar Putta, Arjun Lenan Sandhya

December 17, 2023

Abstract

Write MATLAB/Python programs to solve the boundary layer problem of a flow over a flat plate.

1 Given information

The goal of the second deliverable task is to solve the boundary layer equation over a flat plate:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{1}$$

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

The Reynolds number is defined as $Re = u_\infty L/\nu$, where ν is the kinematic viscosity, L is the plate length and u_∞ is the free-stream velocity of the outer flow. Here the velocity components u and v are the dimensionless velocities, $u = \bar{u}/u_\infty$ and $v = \bar{v}/u_\infty$, where $\bar{u}$ and $\bar{v}$ are dimensional velocities. Lengths have been made dimensionless by using the plate length, e.g. $x = \bar{x}/L$.

1.1 Boundary Conditions

$$y = 0: \quad u = v = 0 \quad \text{(No-slip condition)} \tag{3}$$

$$y \to \infty: \quad u \to 1 (\bar{u} \to u_\infty) \quad \text{(Free outer flow)} \tag{4}$$

2 Solutions

2.1 Discretize equation (1)

Given below is the grid scheme we have opted for this problem. Since the boundary layer develops as the fluid moves along x-direction, we need to find the values of $u_{i+1,j}$ and $v_{i+1,j}$ using the known values behind it. For improved accuracy, a possible discretization of $\frac{\partial u}{\partial x}$ can be obtained as the average value between the layers j and $j-1$.

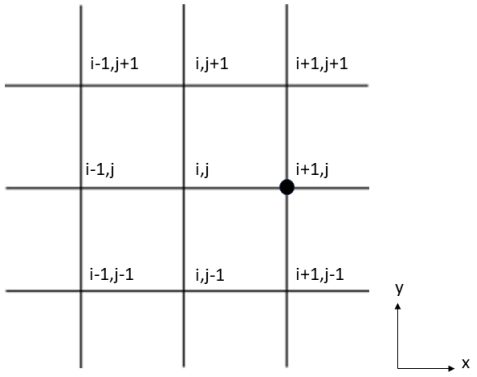

Figure 1: Grid scheme for the domain

Using forward finite difference, $u_{i+1,j}$ can be evaluated as:

$$u_{i+1,j} = u_{i,j} + \Delta x \frac{\partial u}{\partial x}_{i,j} + \frac{(\Delta x)^2}{2!}\frac{\partial^2 u}{\partial x^2} + ... \tag{5}$$

Upon rearrangement, we obtain a first-order accurate approximation of:

$$\frac{\partial u}{\partial x}_{i,j} \approx \frac{u_{i+1,j} - u_{i,j}}{\Delta x} + O(\Delta x) \tag{6}$$

Similarly $\frac{\partial u}{\partial x}_{i,j-1}$ can be evaluated as:

$$\frac{\partial u}{\partial x}_{i,j-1} \approx \frac{u_{i+1,j-1} - u_{i,j-1}}{\Delta x} + O(\Delta x) \tag{7}$$

We can assign the average of these two values as the $\frac{\partial u}{\partial x}$ at (i+1,j):

$$\frac{\partial u}{\partial x}_{i+1,j} \approx \frac{1}{2}(\frac{u_{i+1,j} - u_{i,j}}{\Delta x} + \frac{u_{i+1,j-1} - u_{i,j-1}}{\Delta x}) + O(\Delta x) \tag{8}$$

To approximate $\frac{\partial v}{\partial y}$ at (i+1, j), we use a backward finite difference scheme between points (i+1, j) and (i+1, j-1), since we don't have the data for j+1 but have data for j-1:

$$\frac{\partial v}{\partial y}_{i+1,j} \approx \frac{v_{i+1,j} - v_{i+1,j-1}}{\Delta y} + O(\Delta y) \tag{9}$$

Using this equation, $v_{i+1,j}$ could be approximated by substituting equations (8) and (9) in the continuity equation (1):

$$\frac{1}{2}(\frac{u_{i+1,j} - u_{i,j}}{\Delta x} + \frac{u_{i+1,j-1} - u_{i,j-1}}{\Delta x}) + \frac{v_{i+1,j} - v_{i+1,j-1}}{\Delta y} = 0 \tag{10}$$

On further rearranging of the terms:

$$v_{i+1,j} = v_{i+1,j-1} - \frac{\Delta y}{2\Delta x}(u_{i+1,j} - u_{i,j} + u_{i+1,j-1} - u_{i,j-1}) \tag{11}$$

The value for $u_{i+1,j}$ can now be found from the momentum equation (2) in x-axis. The discretization for the partial derivative terms in the momentum equations are as follows. For the partial derivatives along y-axis, central finite difference method has been used to approximate the values at (i,j):

$$\frac{\partial u}{\partial y}_{i,j} \approx \frac{u_{i,j+1} - u_{i,j-1}}{2\Delta y} \qquad \text{(Second order approximation)} \tag{12}$$

$$\frac{\partial^2 u}{\partial y^2}_{i,j} \approx \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta y^2} \qquad \text{(Second order approximation)} \tag{13}$$

However, the same scheme cannot be applied in the x direction since data is not available for i+1 step. Hence, a backward finite difference scheme is used to approximate $\frac{\partial u}{\partial x}$.

$$\frac{\partial u}{\partial x}_{i,j} \approx \frac{u_{i+1,j} - u_{i,j}}{\Delta x} + O(\Delta x) \tag{14}$$

Substituting equations (12), (13) and (14) in equation (2) gives:

$$u_{i,j}\frac{u_{i+1,j} - u_{i,j}}{\Delta x} + v_{i,j}\frac{u_{i,j+1} - u_{i,j-1}}{2\Delta y} = \frac{1}{Re}\frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{(\Delta y)^2} \tag{15}$$

$$u_{i,j}\frac{u_{i+1,j} - u_{i,j}}{\Delta x} = \frac{1}{Re}\frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{(\Delta y)^2} - v_{i,j}\frac{u_{i,j+1} - u_{i,j-1}}{2\Delta y} \tag{16}$$

$$\implies u_{i+1,j} = u_{i,j} + \frac{1}{Re}(u_{i,j+1} - 2u_{i,j} + u_{i,j-1})\frac{\Delta x}{(\Delta y)^2}\frac{1}{u_{i,j}} - \frac{v_{i,j}}{u_{i,j}}\frac{u_{i,j+1} - u_{i,j-1}}{2}\frac{\Delta x}{\Delta y} \tag{17}$$

2.2 Is the scheme Explicit or Implicit?

From equations (11) and (17), it can be observed that both $u_{i+1,j}$ and $v_{i+1,j}$ are evaluated using values which are from a previous instance in either x-axis or y-axis. Thus, the scheme we have chosen is *explicit*.

2.3 Order of Accuracy

It can be observed from the steps that equation(17) was derived from that it is a mixed scheme with different orders of approximation. However, in a mixed scheme, the order of approximation would be the lowest order of approximation its source terms are derived from. Hence, the approximation we have formulated is of *first-order approximation*.

2.4 Stability Constraint

The numerical solution we have employed, even though a "time" variable is not present, we can consider the solution to be essentially marching in the x-axis. Hence, the *von Neumann* stability requirement is applicable for the numerical solution[1]:

$$\alpha \frac{\Delta x}{(\Delta y)^2} \leq \frac{1}{2} \tag{18}$$

In the above equation, the "marching" variable is Δx and α is $\frac{1}{Re}$ for our case. Hence the equation for Δx becomes:

$$\Delta x \leq \frac{Re(\Delta y)^2}{2} \tag{19}$$

In the code, we specify the number of grid points required along both axes and calculate Δx and Δy accordingly. For the given problem, the Reynolds number (Re) is set to 10000, and the values of Δx and Δ are determined as 3.3333×10^{-4} and 0.0011 respectively. These values result from utilizing 3000 grid points along the x-axis and 90 grid points along the y-axis.

It was observed that the solution experienced instability when the number of grid points along the y-axis exceeded 100. On the other hand, increasing the grid points along the x-axis beyond 3000 did not significantly alter the solution. In consideration of stability, we have opted to maintain 3000 grid points for the x-axis and 90 grid points for the y-axis in our discretization.

3 MATLAB Solution

3.1 Given conditions

$$0 \leq x \leq 1 \tag{20}$$

$$0 \leq y \leq 2\delta \quad \text{where} \quad \delta = \frac{5}{\sqrt{Re}} \quad \text{at} \quad x = 1 \tag{21}$$

$$Re = 10000 \tag{22}$$

For the given conditions, a MATLAB code was written was to numerically solve the Boundary Layer problem. The output of the code are the contour plots of velocity components along x-axis and y-axis.

3.2 MATLAB Code

Below is the MATLAB code developed to solve the problem.

```
clc;
clear;
%% Numerical Solution
U_inf = 10; %Free stream velocity
L=1;
Re=10000;
nu = 1/Re;
delta = 5/sqrt(Re); %Boundary layer thickness at x=1m
Y =  2*delta; %Size of domain in Y-direction
Nx = 3000; %Number of points along x axis
```

```
Ny = 90; %Number of points along y axis
dx = L/Nx; %Grid size along x axis
dy = Y/Ny; %Grid size along y axis

stability_criterion = 0.5*Re*(dy^2);
if dx <= stability_criterion
    disp("Stability criterion met! Please wait for results.")
    u = zeros(Nx,Ny); %Matrix to store u values
    v = zeros(Nx,Ny); %Matrix to store v values
    x = linspace(0,L,Nx);
    y = linspace(0,Y,Ny);
    %%
    u(:,1) = 0; %No slip condition over flat flate
    u(1,:) = 1; %Free stream velocity condition at inlet since we are solving for non-dimensional
    u
    u(:,Ny) = 1; %Free stream velocity condition at domain top since we are solving for non-
    dimensional u
    u(1,1)=0;

    v(1:Nx) = 0; %No slip condition over flat plate
    v(1,:) = 0; %No vertical velocity component at inlet free stream
    v(:,Ny) = 0; %No vertical velocity component at domain top free stream

    %%
    for t = 1:Nx

        for j = 2:Ny-1
            for i = 1:Nx-1

                u(i+1,j) = u(i,j) + (nu*(u(i,j+1)-2*u(i,j)+u(i,j-1))*(dx))/((u(i,j)*dy^2))-((u(i,j
    +1)-u(i,j-1))*v(i,j)*dx)/(2*u(i,j)*dy);
                v(i+1,j) = v(i+1,j-1) - ((0.5*dy*(u(i+1,j)-u(i,j)+u(i+1,j-1)-u(i,j-1)))/dx);
            end
        end
    end
    figure(1)
    colorbar
    hold on
    contourf(x,y,u',25)
    title("Contour plot of horizontal component of velocity - u")
    xlabel("Distance along the plate")
    ylabel("Vertical distance above the plate")
    legend('u contour')
    hold off
    figure(2)
    colorbar
    hold on
    contourf(x,y,v',25)
    title("Contour plot of vertical component of velocity - v")
    xlabel("Distance along the plate")
    ylabel("Vertical distance above the plate")
    legend('v contour')
    hold off
else
    disp("Stability criterion not met! Recheck your gridpoint values")
end
%%
```

3.3 Results for Re = 10000

From the figure 2, it is evident that the boundary layer forms over the plate and develops as the distance along the flat plate increases. On the surface of the flat plate, the horizontal velocity component is zero due to dominating viscous forces.

The theoretical boundary layer thickness is defined as the vertical distance above the plate where the horizontal component of velocity becomes 0.99 times U_∞. At $x = 1$, the theoretical boundary layer thickness, denoted as δ is given by 0.05. However, the boundary layer calculated from the numerical solution is 0.04222. The difference, which is in the order of 10^{-3}, and it is expected since numerical methods doesn't give an exact solution but only an approximation. Error could be due to numerical errors (discretization of governing equations), round-off errors

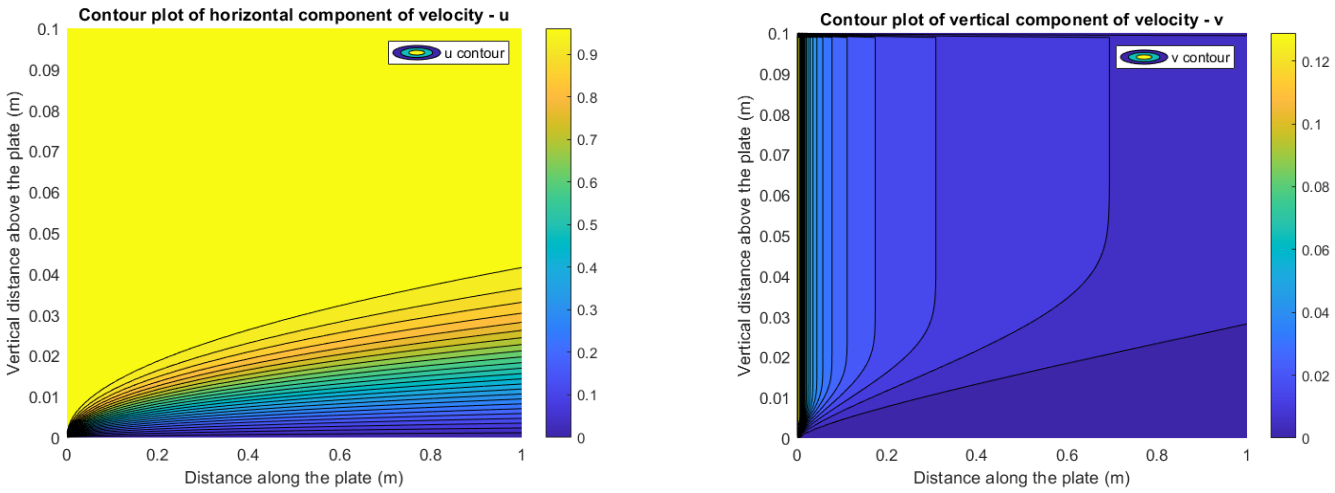

Figure 2: Contour plots of velocity components

and solver accuracy.

From the figure 2, it can be seen that the magnitude of vertical component of velocity is very small compared to the magnitude of free stream velocity. Similar to the horizontal velocity, the vertical velocity is zero on the surface of the flat plate due to dominating viscous terms. The magnitude of the vertical velocity increases gradually although not at as steadily as horizontal component, finally attaining a constant value after the boundary layer is crossed.

3.4 Results for $U_\infty = 10ms^{-1}$ at $x = 0.0005$ and $x = 0.5$

For this part of the exercise, we were told to obtain the values for u and v at $x = 0.0005$ and $x = 0.5$ and compare it with the Blasius equation results.

3.4.1 A bit about Blasius Equation

Since the governing equations for boundary layer problems are partial differential equations, H. Blasius in 1908, transformed the independent and dependent variables to obtain a ordinary differential equation which can be solved much easily. The Blasius equation is given as:

$$2f''' + ff'' = 0 \tag{23}$$

However, the equation is still non-linear and it must be solved numerically with adequate boundary conditions. To solve the Blasius equation[2] for boundary layer problems, the function f is defined as a stream function ψ such that:

$$\psi = \sqrt{\nu x U_\infty} f(\eta) \tag{24}$$

Where,

$$\eta = y\sqrt{\frac{U_\infty}{\nu x}} \tag{25}$$

Then we can find the value of the flow parameters as,

$$u = U_\infty f'(\eta) \tag{26}$$

$$v = U_\infty \frac{\eta f' - f}{2\sqrt{Re_x}} \quad \text{,where } Re_x \text{ is the local Reynolds number} \tag{27}$$

$$f'(\eta) = \frac{u}{U_\infty} \tag{28}$$

3.4.2 Comparison with Blasius Equation results

For comparison with the Blasius equation results, we selected two points along the plate length, specifically at $x = 0.0005$ and $x = 0.5$. Since the Blasius equation provides results in terms of η, we calculated the corresponding y values using equation(25). Subsequently, the horizontal and vertical velocity values at these points were plotted against the vertical distance above the plate to visualize their distribution.

Figure 3 clearly illustrates that the velocity values obtained from both the numerical method and the Blasius equation are in excellent agreement at the point $x = 0.5$. At this point, both results agree each other because the point taken is in the middle of the domain where the values have lesser errors.

However, a noticeable difference exists at $x = 0.0005$, for the vertical velocity component. $x = 0.0005$ corresponds to a point just after the start of the boundary of the domain, where the values would have errors associated with them due to factors like grid size, number of iterations etc. which are inherent to numerical method solutions.

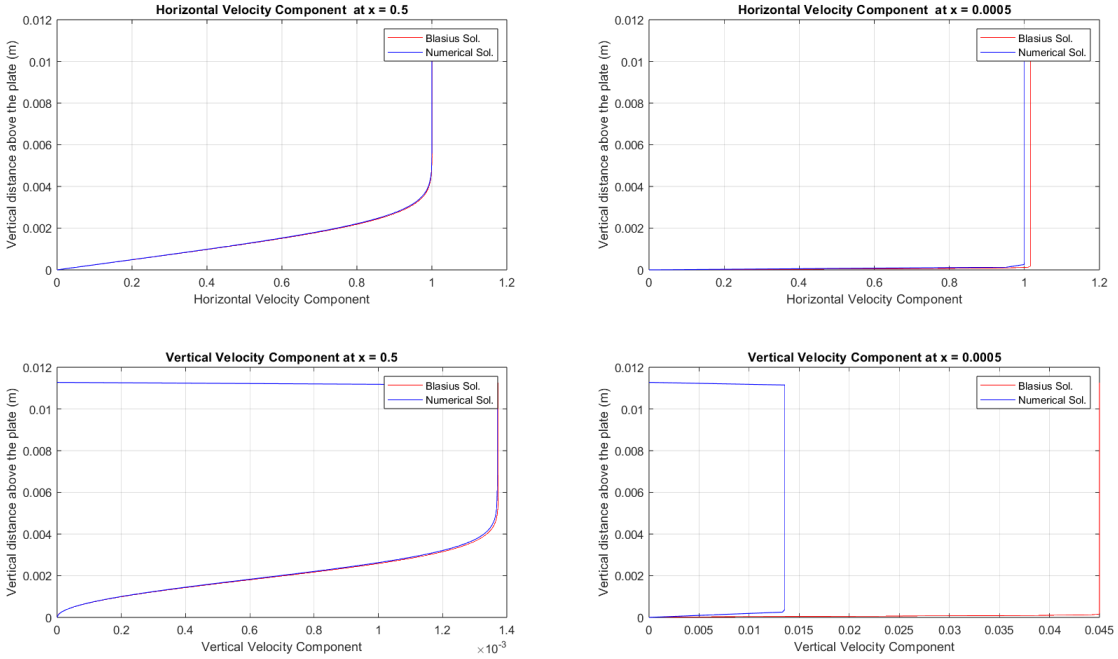

Figure 3: Horizontal and Vertical velocity components plotted against Y-distance for points $x = 0.0005$ and $x = 0.5$

References

[1] John D. Anderson Jr. *Computational Fluid Dynamics*. McGraw-Hill, 1995.

[2] John D. Anderson Jr. *Fundamentals of Aerodynamics*. McGraw-Hill, 2011.