APL321 : Introduction to Computational Fluid Dynamics
Indian Institute of Technology Delhi
Spring 2024
Prof. Amitabh Bhattacharya

Lab 1

Aditya Agrawal (2021AM10198)

Consider the steady 2D laminar boundary layer over flat plate shown in Fig 1(a) with uniform inlet velocity U along x direction. The velocity field components are $u(x, y)$ and $v(x, y)$. The boundary layer starts forming at $x = 0$, and the upper surface of the plate is located at $y = 0$. This flow allows for a self-similar "Blasius profile" solution for the streamfunction which has the form:

$$\psi(x, y) = U\delta(x)f(\eta) \tag{1}$$

where $\delta(x) = (\nu x/U)^{(1/2)}$ is proportional to the boundary layer thickness, $\eta = y/\delta$ is the similarity variable, and the "similarity solution" satisfies the following ODE:

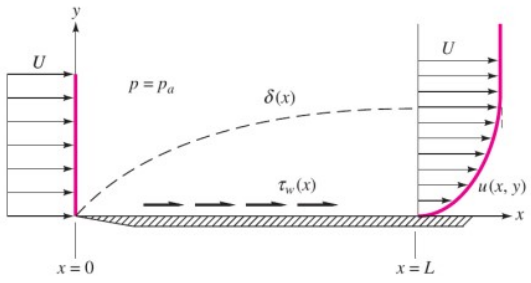
$$\frac{d^3f}{d\eta^3} + \frac{1}{2}f\frac{d^2f}{d\eta^2} = 0 \tag{2}$$

along with the boundary conditions $\frac{df}{d\eta}|_{\eta=0} = 0$, $f(0) = 0$ and

$$\lim_{\eta\to\infty} f' = 1$$

1. Break equation 2 into 3 separate first order ODEs for the variables $g = f'$ and $h = f''$. Use the iterative "shooting method" to solve Eq. 2, in which you have to use a root finding algorithm (e.g. Newton Raphson method or "fzero" function in MATLAB) to set $f''(0)$ so that $lim_{\eta\to\infty}f' = 1$ is satisfied at some large value of η (e.g.$\eta = 5$). While evolving the ODEs along η, make sure that you use $\Delta\eta \ll 1$. Plot u/U vs η over $\eta \in [0, 5]$ and compare your graph with the values tabulated in Fig 1(b). Choose only the values at $\eta = 1, 2, 3, 4$ for comparison, and plot these values from Fig 1(b) along with your numerical solution along the same graph for comparison. Use $u = \frac{\partial\psi}{\partial y}$. In your report, clearly explain how you have solved the boundary value problem using the shooting method.

Figure 1: (a) Schematic of laminar boundary layer over flat plate with zero pressure gradient (b) Table for u/U vs η for Blasius profile (reproduced from Fluid Mechanics by FM White)



(a)

$y[U/(\nu x)]^{1/2}$	u/U	$y[U/(\nu x)]^{1/2}$	u/U
0.0	0.0	2.8	0.81152
0.2	0.06641	3.0	0.84605
0.4	0.13277	3.2	0.87609
0.6	0.19894	3.4	0.90177
0.8	0.26471	3.6	0.92333
1.0	0.32979	3.8	0.94112
1.2	0.39378	4.0	0.95552
1.4	0.45627	4.2	0.96696
1.6	0.51676	4.4	0.97587
1.8	0.57477	4.6	0.98269
2.0	0.62977	4.8	0.98779
2.2	0.68132	5.0	0.99155
2.4	0.72899	∞	1.00000
2.6	0.77246		

(b)

Solution 1:
For solving the system of ODE given by :

$$g = f' \tag{3}$$

$$h = f'' \tag{4}$$

$$h' = -\frac{1}{2}fh \tag{5}$$

we first discretize the continuous range of the similarity variable $\eta \in [0, \infty]$ into points with a step size of $n = 10^{-3}$. We have assumed $\eta = 10$ as infinity which fits well with known solution of the ODEs. The total number of points becomes $N = (10/n) + 1$.

We then require 3 boundary conditions to solve the system of ODEs. Two of them are already provided in the question given by $\frac{df}{d\eta}|_{\eta=0} = 0$ and $f(0) = 0$. We need to find the third boundary condition for $h(0)$ by using the fact that $\lim_{\eta\to\infty} f' = 1$. This is done by using the Newton Raphson method to find the root (i.e. $h(0)$) for the equation:

$$g(N) - 1 = 0 \tag{6}$$

The iterative equation for Newton Raphson method gives us:

$$h(0)_{k+1} = h(0)_k - \frac{(g(N)_k - 1)}{(\frac{dg}{dh(0)})_k} \tag{7}$$

We can then use the following approximation:

$$(\frac{dg}{dh(0)})_k \approx \frac{g(N)_k - g(N)_{k-1}}{h(0)_k - h(0)_{k-1}} \tag{8}$$

Therefore, equation (7) becomes:

$$h(0)_{k+1} = h(0)_k - (g(N)_k - 1)\frac{h(0)_k - h(0)_{k-1}}{g(N)_k - g(N)_{k-1}} \tag{9}$$

However, we require two initial guesses for the values of h(0). In our model we have taken the guesses to be $h(0)_0 = 0.1$ and $h(0)_1 = 0.2$.

We use the Euler's method of solving ODE on each of the three equations 3, 4 and 5 for the values of $g(N)_0$ and $g(N)_1$ using the guessed values of $h(0)$. According to Euler's method, the iterative equations are given by:

$$f_{i+1} = f_i + ng_i \tag{10}$$

$$g_{i+1} = g_i + nh_i \tag{11}$$

$$h_{i+1} = h_i - \frac{1}{2}f_ih_i \tag{12}$$

After computing $g(N)_0$ and $g(N)_1$ we can solve the iterative equation (9) to get the optimal value of the boundary condition $h(0)$ which turns out to be equal to 0.3319 (which is very close to the accepted value of 0.332).

We now have three linear ODEs with 3 boundary conditions. Thus, we can now use the same Euler's method given by equations 10, 11 and 12 to obtain the values of f , g and h at all the discretized points of η.

The MATLAB code for the solution and the plots of $\frac{u}{U_\infty}$ Vs η for the existing and computed values are given below.

```
% Question 1

n = 10^(-3);                        % step size
eta_0 = 0;                          % initial eta
eta_f = 10;                         % final_eta
N = (eta_f - eta_0) / n;            % Number of iterations

% Boundary Conditions

f_0 = 0;
g_0 = 0;
h0_0 = 0.1;                         % 1st guess for f"(0)

f0 = zeros(N+1, 1);
g0 = zeros(N+1, 1);
h0 = zeros(N+1, 1);

f0(1) = f_0;
g0(1) = g_0;
h0(1) = h0_0;

% Euler's Method to solve system of ODEs

for i = 1:N
    f0(i+1) = f0(i) + n*g0(i);
    g0(i+1) = g0(i) + n*h0(i);
    h0(i+1) = h0(i) + n*(-0.5)*f0(i)*h0(i);
end

h1_0 = 0.2;                         % 2nd guess for f"(0)

f1 = zeros(N+1, 1);
g1 = zeros(N+1, 1);
h1 = zeros(N+1, 1);

f1(1) = f_0;
g1(1) = g_0;
h1(1) = h1_0;

% Euler's Method to solve system of ODEs

for i = 1:N
    f1(i+1) = f1(i) + n*g1(i);
    g1(i+1) = g1(i) + n*h1(i);
    h1(i+1) = h1(i) + n*(-0.5)*f1(i)*h1(i);
end

```

```
% Newton-Raphson to optimize h_0

hk_1 = h0_0;
hk = h1_0;
gk_1 = g0(N+1);
gk = g1(N+1);
tol = 1e-5;             % tolerance value for convergence
err = abs(gk - 1);      % error after each iteration

while (err > tol)

    h_next = hk - ((gk - 1)*(hk - hk_1))/(gk - gk_1);

    f = zeros(N+1, 1);
    g = zeros(N+1, 1);
    h = zeros(N+1, 1);

    f(1) = f_0;
    g(1) = g_0;
    h(1) = h_next;

    % Euler's Method to solve system of ODEs
    for i = 1:N
        f(i+1) = f(i) + n*g(i);
        g(i+1) = g(i) + n*h(i);
        h(i+1) = h(i) + n*(-0.5)*f(i)*h(i);
    end

    hk_1 = hk;
    hk = h_next;
    gk_1 = gk;
    gk = g(N+1);
    err = abs(gk - 1);

end

h_0 = hk;               % Optimized guess value of h_0

f = zeros(N+1, 1);
g = zeros(N+1, 1);
h = zeros(N+1, 1);
eta = zeros(N+1, 1);
eta(1) = 0;

f(1) = f_0;
g(1) = g_0;
h(1) = h_0;

```

```
% Euler's Method to solve system of ODEs
for i = 1:N
    eta(i+1) = i*n;
    f(i+1) = f(i) + n*g(i);
    g(i+1) = g(i) + n*h(i);
    h(i+1) = h(i) + n*(-0.5)*f(i)*h(i);
end

% values of f' = u/U for eta = 1,2,3,4
solution = zeros(4, 1);

for i = 1:4
    solution(i) = g(i/n + 1);
end

figure('Position',[10 10 1600 600]);

% Plot of u/U Vs eta
subplot(1,2,1);
plot(eta, g);
title('Plot of u/U Vs eta')
ylabel("u/U");
xlabel("eta");
set(gca,'FontSize',16);

Expected = [0.32979; 0.62977; 0.84605; 0.95552];
eta_ = [1; 2; 3; 4];

% Expected Vs Computed plots of u/U Vs eta
subplot(1,2,2);
plot(eta_, Expected)
hold on
plot(eta_, solution)
legend('Expected', 'Computed');
customTicks = [1, 2, 3, 4];
xticks(customTicks);
ylabel("u/U");
xlabel("eta");
title('Expected Vs Computed Solution')
set(gca,'FontSize',16);
```

MATLAB code for the numerical solution of the Blasius equation

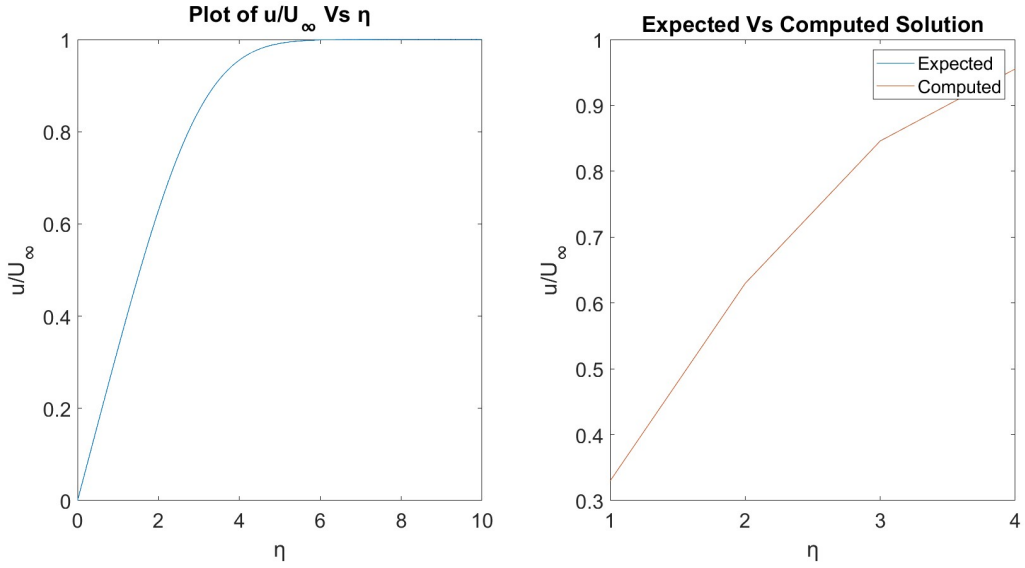


Figure 2: Plots for $\frac{u}{U_\infty} Vs\ \eta$

2. Consider flow over flat plate in which $U = 10$ m/s and $\nu = 10^{-5}\ m^2/\text{s}$. Calculate u/U on a uniform 20×20 Cartesian grid over the flow domain $x \in [1m, 10m]$, using the Blasius solution above. Store these values in a 20×20 array and then make a filled isocontour plot of u/U over the flow domain in which the x and y values should be used for the axes. Clearly show colorbar, axes labels in the figure. In your report, mention clearly how you managed to calculate u/U on a structured Cartesian grid. Hint: You can solve this problem by using interpolation (see interp1 function in matlab) or by judiciously choosing $\Delta\eta$ at every x location.

   Solution 2:
   First of all, we need to divide the given region into a 20×20 mesh grid. For this, we divide the x-axis and y-axis into $nx = 19$ and $ny = 19$ points since we also need to accomodate the extreme limits of the range. Therefore, the separation between two abscissas and ordinates is given by $dx = \frac{(10-1)}{nx}$ and $dy = \frac{0.01}{ny}$.

   Next we use the inbuilt MATLAB function "meshgrid" to create a mesh grid of 20×20 points for the x and y coordinates.

   We use the function "fxy" to calculate the η values at each point (x, y) in the mesh grid. The function "fxy" is defined by:

$$f(x, y) = y\sqrt{\frac{U_\infty}{(\nu x)}} \tag{13}$$

   Once we get the η values at every mesh grid point, we now use the values of g calculated in Question 1 by interpolating the respective (η, g) values to get $\frac{u}{U_\infty}$ at these points. For interpolation, we use the inbuilt MATLAB function "interp1" which interpolates a function based on the linear method.

   Finally, we plot the isocontour lines for $\frac{u}{U_\infty} = c$, for any constant c. Given below is the isocontour plot obtained and the MATLAB code implemented to get the plot.

```
% Question 2

U = 10; nu = 1e-5;        % constants U and nu

lx = 1;                   % lower limit of x coordinate
Lx = 10;                  % upper limit of x coordinate
Ly = 1e-2;                % upper limit of y coordinate

% number of points along x and y
nx = 19; ny = 19;

% distance between two consecutive abscissa and ordinate
dx = (Lx-lx)/nx; dy = Ly/ny;

% coordinates along x and y directions
xcoor = (0:nx)*dx + lx;
ycoor = (0:ny)*dy;

% coordinates in a ny-by-nx mesh
[xm,ym] = meshgrid(xcoor,ycoor);

% eta values in the meshgrid
eta_mesh = fxy(xm,ym,U,nu);

% finding values of u/U on the mesh using interpolation
fmesh = interp1(eta,g,eta_mesh);

% new figure
figure('Position',[10 10 1600 600]);

% isocontour plot of u/U
contourf(xm,ym,fmesh);
xlabel('x');
ylabel('y');
colorbar;
title('Isocontour of u/U');
set(gca,'FontSize',16);
xlim([1.00 10.00])
ylim([0.0000 0.0100])

% function to evaluate eta at (x,y)
function f = fxy(x,y,U,nu)
    f = y .* sqrt(U ./ (nu .* x));
end
```

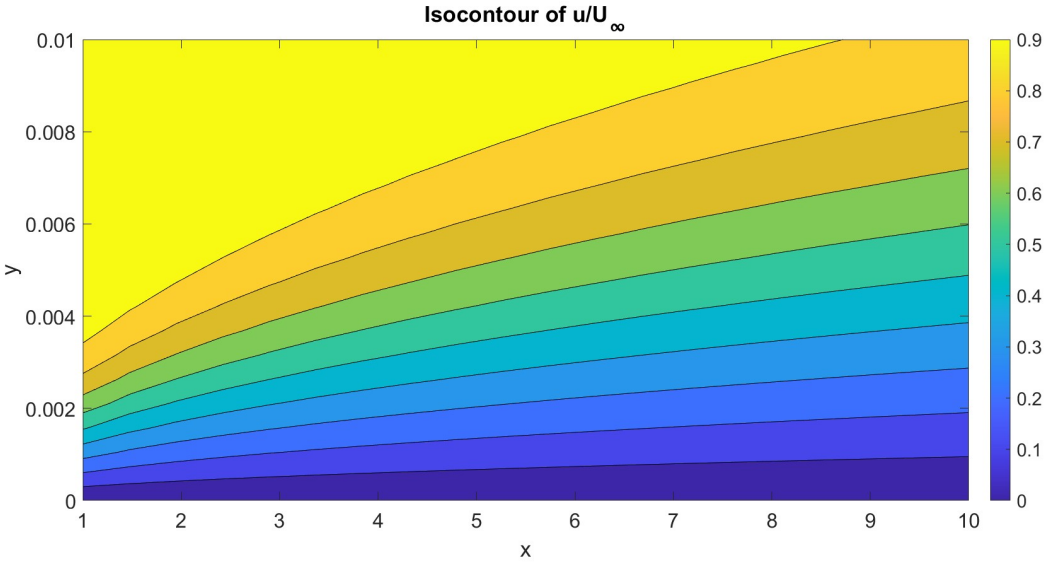


Figure 3: Isocontour plots of $\frac{u}{U_\infty}$