COMPUTATIONAL STUDY OF UNIFORM FLOW AROUND A HALF PLANE

LUKE APPLEBY

ABSTRACT. A computational investigation of uniform flow impinging upon a half plane with a particular focus on the boundary layer thickness as well as viscous stress properties.

1. Introduction

We investigate the boundary layer that develops when uniform flow meets a half plane. A suitable approximation of the Naiver-Stokes equations are used to model this flow. Computational methods are used to find a particular solution of these equations. Additionally we investigate the boundary layer thickness and the viscous stress at the wall. These are key concepts in fluid dynamics important for understanding the drag and heat transfer of components which is critical in aerospace and industrial applications.

Boundary layers can be either laminar or turbulent. Laminar flows are flows in which fluid particles flow in parallel lines whereas in turbulent flows the fluid particles undergo unpredictable and irregular fluctuations. When the flow in the boundary layer is laminar the layer is thinner compared to a turbulent boundary layer which is thicker. Turbulent boundary layers will also have a laminar sub-layer closer to the wall. It is possible for the boundary layer to become detached. This is a phenomenon that occurs when aircraft wings stall. Beyond the separation point there is minimal frictional resistance however there is significantly more pressure resistance. Therefore in aircraft design such an occurrence is considered to be undesirable in most cases.

Viscous stress occurs in both laminar and turbulent boundary layers however it is simpler to model in the laminar case. Understanding the different viscous stress properties in both of these cases is useful in aerospace an industrial applications.

2. Approximating the Naiver-Stokes equations

We can approximately model the development of uniform flow impinging on a half plane using the equations

$$u\partial_x u + w\partial_z u = \partial_{zz},\ \partial_x u + \partial_z w = 0. \tag{1}$$

These are an adequate approximation of the Naiver-Stokes equations. It has been shown that a solution can be found taking the form

$$u = f'(\eta),\ w = \frac{1}{2x^{1/2}}(\eta f'(\eta) - f(\eta)),\ \text{where },\eta = z/x^{1/2}. \tag{2}$$

Where f is a function satisfying the boundary value problem

$$f''' + 1/2f''f = 0,\ f(0) = f'(0) = 0 \text{ and } \lim_{\eta\to\infty} f'(\eta) = 1. \tag{3}$$

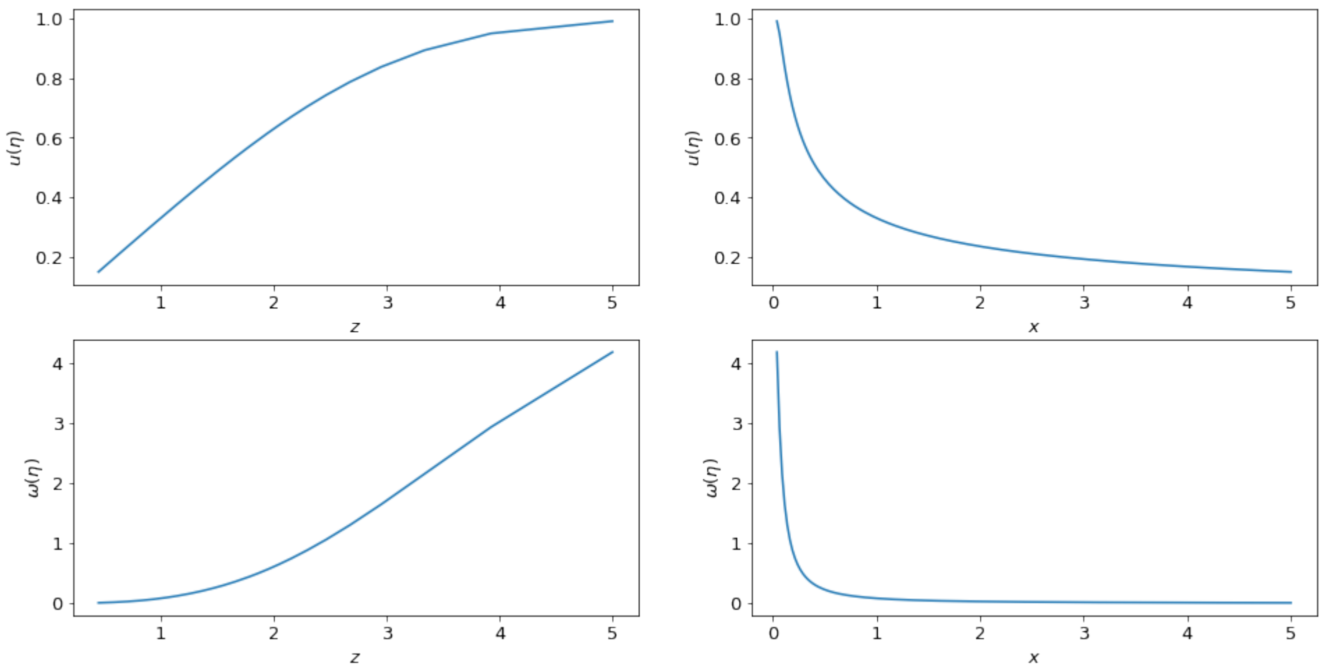

FIGURE 1. Plots of u and w against x and z

We are able to solve this boundary value problem by first converting into a system of first order ordinary differential equations where $g(\eta) = f'(\eta)$ and $h(\eta) = g'(\eta)$. Then using the `solve_bvp` function from `scipy.integrate` we are able to solve this system numerically in our code A. The behaviour of u and w are plotted against x and z in figure 1

3. Boundary Layer Thickness

The boundary layer refers to a thin layer of fluid that sits next to the solid. When we model systems such as uniform flow impinging on a half plane we assume the no slip condition. That is fluid particles immediately adjacent to the surface will have velocity of zero. Due to the viscous forces between fluid particles this velocity increases smoothly as we move further from the surface.

$\delta(x)$ approximates the boundary layer thickness which is precisely defined as the distance between the solid boundary and the point at which the flow is 99% of the final stream velocity 2.

$$\delta(x) = \int_0^\infty 1 - u(x,z)\,dz \tag{4}$$

By changing the variable of integration from z to η via $dz = x^{1/2}d\eta$ see that the integral becomes

$$\begin{aligned} \delta(x) &= \int_0^\infty x^{1/2}(1-u(x,z))\,d\eta \qquad (5)\\ &= \lim_{L\to\infty}[x^{1/2}(\eta - f(\eta))]_0^L. \qquad (6)\end{aligned}$$

Evaluating numerically using our code A we have that $\delta(x) \approx 1.72x^{1/2}$.

4. Viscous Stress

Viscous stress is the internal force arising from the viscosity of a given fluid. When we have a non uniform flow layers of fluid particles flow along side each other and the velocity gradients between these generate viscous stress. Viscous stress depends upon the velocity gradients in the fluid, the directions of the flow and the viscous properties of the fluid.

In laminar flow it is fairly simply to model the viscous stress as the fluid particles move in parallel layers and the motion is smooth and predictable. In turbulent flows it is far more difficult to model the viscous stress. One common approach is to decompose the two components, firstly the viscous stress due to the mean flow and second the viscous stress due to the fluctuations in the flow. One way of approximating the second component is with the Reynolds-averaged Naiver-Stokes equations which demonstrates the broad utility of these equations.

The viscous stress at the wall can be given by

$$\sigma = \partial_z u(x,z)|_{z=0} \tag{7}$$

$$= \frac{\partial}{\partial z} f'(\eta)|_{z=0} \tag{8}$$

$$= \frac{\partial}{\partial z} f'(z/x^{1/2})|_{z=0} \tag{9}$$

$$= x^{-1/2} f''(0). \tag{10}$$

Using our code A we see that $\sigma(x,z) = 0.33x^{-1/2}$

5. Discussion and conclusions

Using computational methods and an approximation for the Naiver-Stokes equations we found suitable formulae for both the boundary thickness and the viscous stress properties when uniform flow impinges on a half plane. Additionally the theory behind theses properties and equations that are used to model them were discussed. These equations and methods could be used to model flow around components in industry and aerospace to monitor the stress such components would be dealing with to predict when they may be damaged or need to be replaced. Further study into how the geometry of a surface affects these properties is useful. An example of this is components to disrupt the flow over the top of the wing to create more favourable conditions to produce lift.

References

[1] Maths in Action lecture notes.
[2] KSB website
`https://www.ksb.com/en-global/centrifugal-pump-lexicon/article/boundary-layer-1118362`

Appendix A. Codes

```
from scipy.integrate import solve_bvp
from scipy import integrate as Int
import numpy as np
import matplotlib.pyplot as plt
import scipy as sp

# f''' + 1/2ff'' = 0, f(0) = f'(0) = 0,
# g = f', h = g', h' + 1/2fg' = 0, f(0) = g(0) = 0

def fun(n, y):
    return np.vstack((y[1], y[2], -1/2*y[0]*y[2]))

def bc(ya, yb):
    return np.array([ya[0], ya[1], yb[1]-1])

n = np.linspace(0, 10, 10)

# Obtain array of square of each element in x
neg_exp = lambda n: np.exp(-n)
y_a = np.array([neg_exp(ni) for ni in n])
y_a = np.vstack((y_a, y_a, y_a))

res_a = solve_bvp(fun, bc, n, y_a)

def eta(z, x):
    return z * x**(-1/2)

def omega(f, g, x):
    eta = x**(-1/2)
    return 0.5*eta * (eta * g(eta) - f(eta))


minus_one = sp.interpolate.PPoly([[0],[-1]], [-1, -2], extrapolate=None, axis=0)
R = sp.interpolate.PPoly(res_a.sol.c, res_a.sol.x, extrapolate=None, axis=0)

Arr = res_a.sol.c.copy()
Arr[3] += -1
R_minus_one = sp.interpolate.PPoly(-1*Arr, res_a.sol.x, extrapolate=None, axis=0)

x = np.linspace(0,10, 100)
e = eta(x, 1)

y0 = minus_one(e)
y1 = res_a.sol(e)[1]
y2 = R(x)[:,1]
y3 = R_minus_one(e)[:,1]

plt.plot(x,y1, label = r'u')
plt.plot(x,y3, color = 'r', label = r'1-u')
plt.legend()
plt.show()

I = R_minus_one.integrate(0, 20)

sigma = res_a.sol.derivative(nu=1)
```

```
y4 = sigma(x)
plt.plot(x, y4[1])
plt.plot(x, res_a.sol(e)[2])

res_a.sol(15)[0] - 15
fig, ax = plt.subplots(2,2, figsize = (16,8))
x_plot = np.linspace(0.04, 5, 200)
n = eta(1, x_plot)

y_plot_a = res_a.sol(n)[1]
y_plot_b = 0.5*eta(1, x_plot)*(eta(1, x_plot)*res_a.sol(n)[1] - res_a.sol(n)[0])

ax[0, 0].plot(n, y_plot_a)
ax[0, 0].set_xlabel(r"$\eta$")
ax[0, 0].set_ylabel(r"$u(\eta)$")
ax[1, 0].plot(n, y_plot_b)
ax[1, 0].set_xlabel(r"$\eta$")
ax[1, 0].set_ylabel(r"$\omega(\eta)$")

ax[0, 1].plot(x_plot, y_plot_a)
ax[0, 1].set_xlabel(r"$x$")
ax[0, 1].set_ylabel(r"$u(\eta)$")
ax[1, 1].plot(x_plot, y_plot_b)
ax[1, 1].set_xlabel(r"$x$")
ax[1, 1].set_ylabel(r"$\omega(\eta)$")
plt.show()
```