ASEN 5051 Project 1

Lucas Calvert & Duncan McGough

November 7, 2018

1 Introduction

Conformal mapping allows for the transformation between the complex plane and another defined non-complex plane. The use of the Jokowski transformation

$$F_J(\zeta) = \zeta + \frac{b^2}{\zeta}$$

allows for a circle of radius b in the complex $\zeta - \eta$ plane to be transformed and plotted as a flat plate of length $4b$ in the $x - y$ plane. By changing the location of this cylinder in the complex plane while forcing it to intersect the same point the untranslated cylinder did on the ζ-axis, an airfoil with thickness and camber can be created in the $x - y$ plane.

This is done mathematically as:

$$\Psi(z) = \Psi(F^{-1}(\zeta))$$

2 Derivations

2.1 Complex Potential

We will be deriving the complex potential Ψ for Joukowski airfoil. It should be noted that, to start, the complex potentials for complex potential flow around a flat plate in a flow with an angle of attack α are as such:

Freestream	$\Psi_f(\zeta) = u_\infty e^{-i\alpha}\zeta$
Doublet	$\Psi_d(\zeta) = u_\infty b^2 e^{i\alpha}\zeta^{-1}$
Vortex	$\Psi_v(\zeta) = \frac{\Gamma}{2\pi i}\ln(\frac{\zeta}{b})$

When added together, this is the potential flow over a flat plate:

$$\Psi(\zeta) = u_\infty e^{-i\alpha}\zeta + u_\infty b^2 e^{i\alpha}\zeta^{-1} + \frac{\Gamma}{2\pi i}\ln\left(\frac{\zeta}{b}\right)$$

In order to give dimension to this flat plate, we need to translate the center of the circle in the complex plane using a translational conformal map:

$$F_z(\zeta)^{-1} = \zeta + z$$

This means that all of the ζ terms in the complex potential will be transformed to:

$$\zeta \rightarrow \zeta - \zeta'$$

Where ζ' is equal to $C_x + iC_y$ (the translation of the center of the circle). So, making this change, our complex potential now becomes:

$$\Psi(\zeta) = u_\infty e^{-i\alpha}(\zeta - \zeta') + u_\infty b^2 e^{i\alpha}(\zeta - \zeta')^{-1} + \frac{\Gamma}{2\pi i}\ln\left(\frac{(\zeta - \zeta')}{b}\right)$$

It should also be noted that now since the location of the cylinder in the complex plane has now changed, we cannot simply use b as the radius for the complex potential. In order to satisfy the kutta conditiont, we know that there will always be a stagnation point at the trailing edge of the airfoil - this means that we must keep the original location of the stagnation point on the circle (the point where the circle intersects the positive x-axis when it is centered at (0,0) - let's call this "Point B") constant. This implies that the radius must change according to the following relationship:

$$R = \sqrt{(\mathrm{Re}(\zeta') - b)^2 + (\mathrm{Im}(\zeta'))^2}$$

Our new complex potential is now:

$$\boxed{\Psi(\zeta) = u_\infty e^{-i\alpha}(\zeta - \zeta') + u_\infty R^2 e^{i\alpha}(\zeta - \zeta')^{-1} + \frac{\Gamma}{2\pi i}\ln\left(\frac{(\zeta - \zeta')}{R}\right)}$$

Now, the only thing that is keeping us from obtaining an analytical solution for the complex potential is Γ. However, we can exploit the geometry of our problem setup to determine an expression. First, lets note that a certain circulation is needed to place the stagnation point at the correct location, even when the circle is located at the origin. This circulation precisely the circulation necessary to rotate the stagnation point by the angle α, the angle of attack of the flow.

Now, lets also note that there will be an additional angle formed between the new center of the circle and "Point B" whenever the circle is translated. This angle is found using the following:

$$\theta = \arccos\left(\frac{-C_x + b}{R}\right)$$

This angle also has an associated circulation necessary to keep the stagnation point at the correct location. The total circulation about the body will be equal to the circulation necessary to rotate the stagnation point an angle Θ, equal to the sum of α and θ. To find the actual expression for the circulation that will rotate the stagnation point by Θ, we can use our knowledge of the physical properties of the flow around the cylinder. First, since we know the flow does not pass through the cylinder, the radial velocity will equal zero everywhere along the surface of the cylinder. Second, we know that that the tangential velocity will be equal to twice the free stream velocity multiplied by the sin of the angle around the cylinder. Last, we know the effect of the circulation around the body is to effectively "induce" a velocity equal to the circulation over two times the radius of the body multiplied by π. These three properties are summarized below:

$$u_r = 0$$

$$u_\theta = -2u_\infty sin(\theta)$$

$$u_{induced} = \frac{\Gamma}{2\pi R}$$

In order to put the stagnation point at the desired location, we know that we want both the radial and tangential velocities evaluated at the angle $-\Theta$ to be equal to zero. From this, we get the following expression:

$$-2u_\infty sin(\theta) + \frac{\Gamma}{2\pi R} = 0$$

Solving for Γ results in:

$$\boxed{\Gamma = -4\pi R u_\infty sin(\Theta)}$$

This equation for Γ can then be used in the above equation for the complex potential. Taking just the imaginary part of this updated complex potential and using a contour plotting function allows us to visualize the analytical solution for the streamlines around the cylinder. An example of this is shown below.

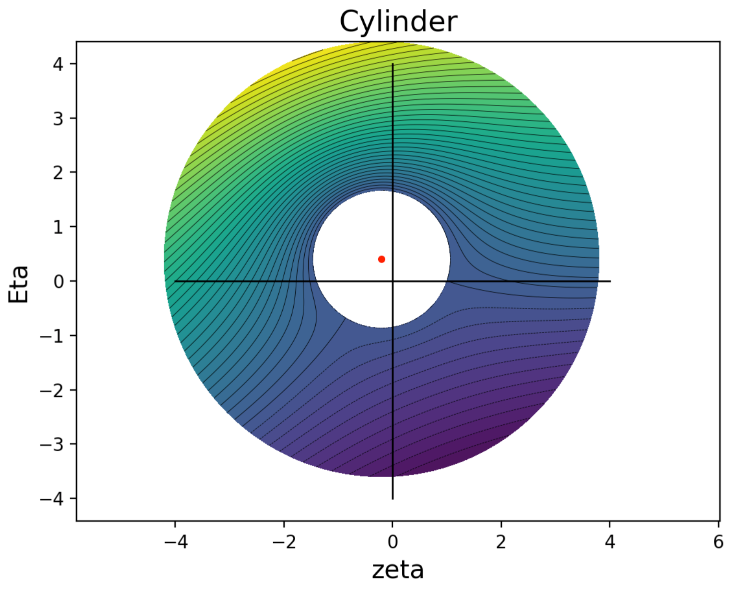


Figure 1: Streamlines Around a Cylinder in the Complex Plane

From here, we can use the Joukowski Transform to change our domain from the ζ plane to the Z plane. By construction, this special conformal map will turn cylinders into "Joukowski Airfoils". Applying this map to the cylinder shown above results in the following airfoil and streamline plot:

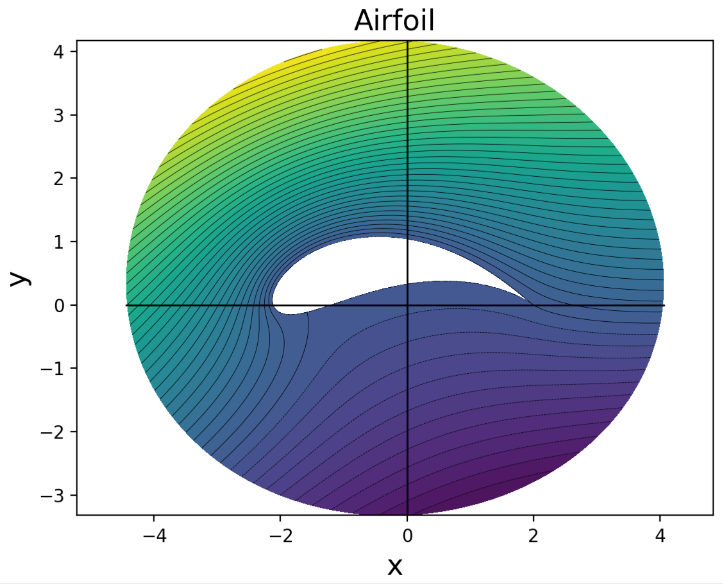


Figure 2: Streamlines Around an Airfoil

Next, we'd like to find an analytical solution for both the x and y velocities around the airfoil. To do this, we can simply take the derivative of the complex potential in the Z plane, being careful to note that the chain rule must be used:

$$\frac{d\Phi}{dz} = \frac{\frac{d\Psi}{d\zeta}}{\frac{dF_J}{d\zeta}}$$

This differentiation was performed symbolically in MATLAB (script included in submission). Since:

$$\frac{d\Phi}{dz} = u - iv$$

We can find the x and y velocities by simply plotting the real and imaginary parts of $\frac{d\Phi}{dz}$, respectively. Examples are shown below for the cylinder above with setup conditions of $U_\infty = 100$, $P_\infty = 1 * 10^6$, $\rho_\infty = 1.0$, $b = 1$ and $\alpha = 20°$.

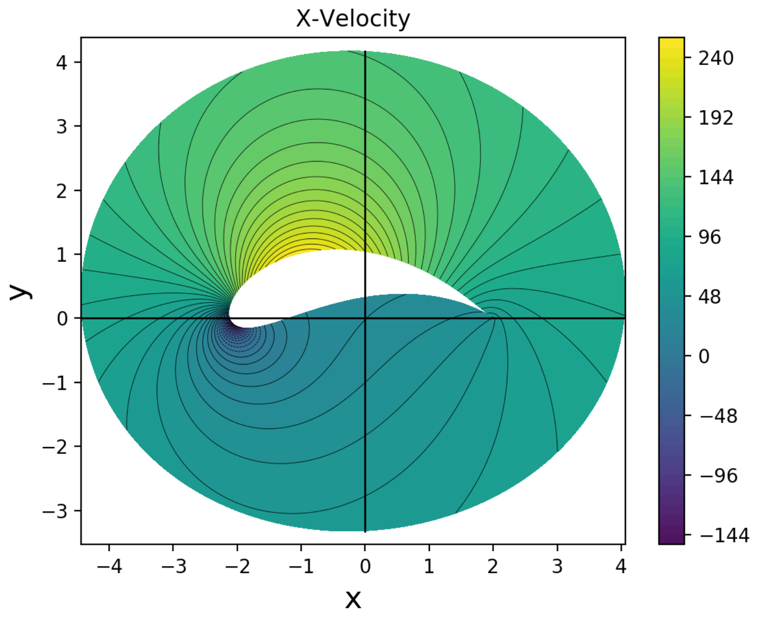


Figure 3: X-Velocity around an Airfoil

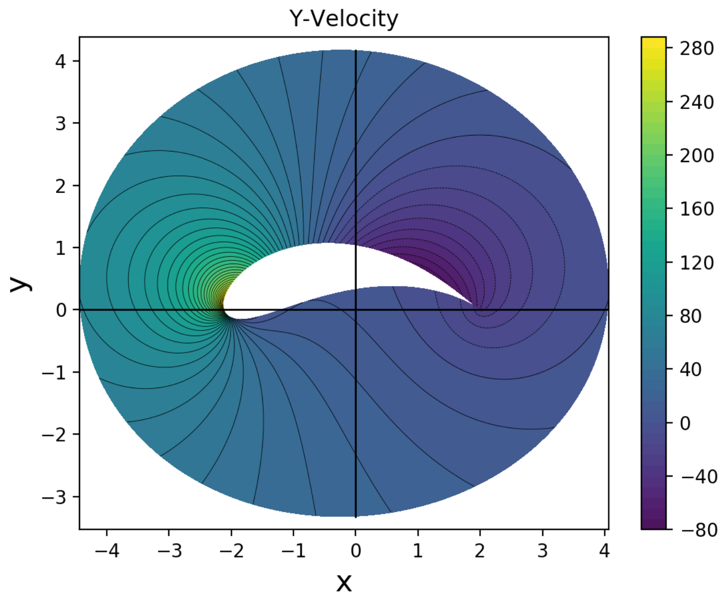


Figure 4: Y-Velocity around an Airfoil

The magnitude of the velocity can be found by simply using $|U| = \sqrt{u_x^2 + u_y^2}$ and the pressure around the body can the be found by employing the Steady Bernoulli relation:

$$P_\infty + \frac{1}{2}\rho_\infty |U|^2 = \text{constant along streamlines}$$

But since the free stream is assumed to be uniform, this constant is the same everywhere. This implies that:

$$P = C - \frac{1}{2}\rho_\infty |U|^2$$

Where C is the constant calculated using the free stream values for pressure, density and velocity. Examples of both are again shown below.

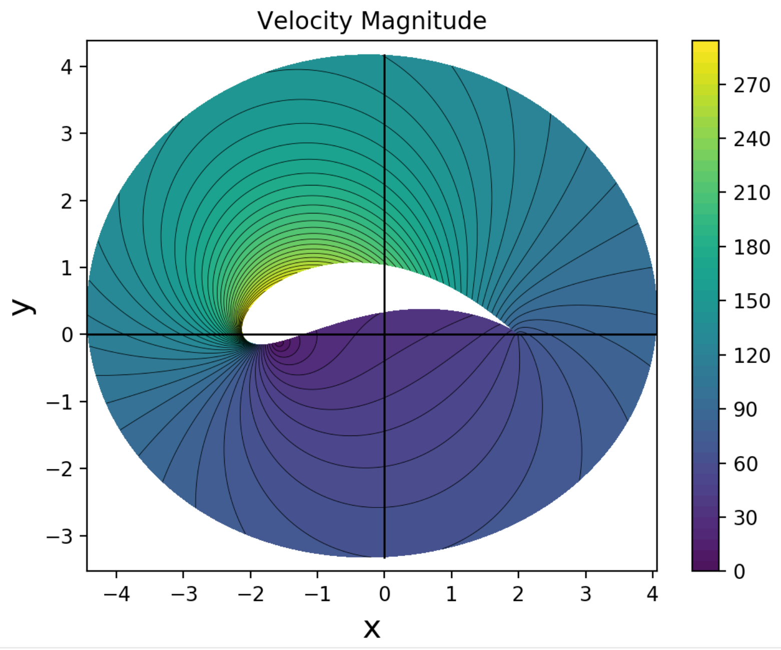


Figure 5: Velocity Magnitude around an Airfoil

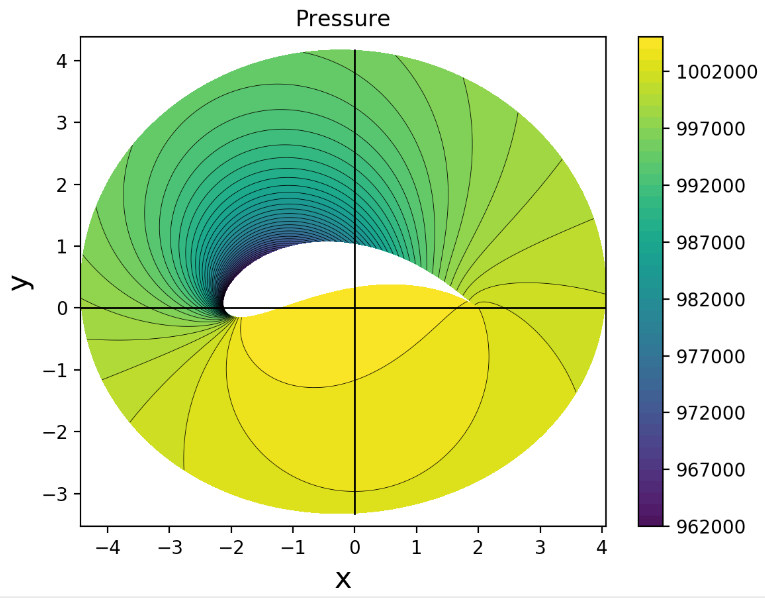
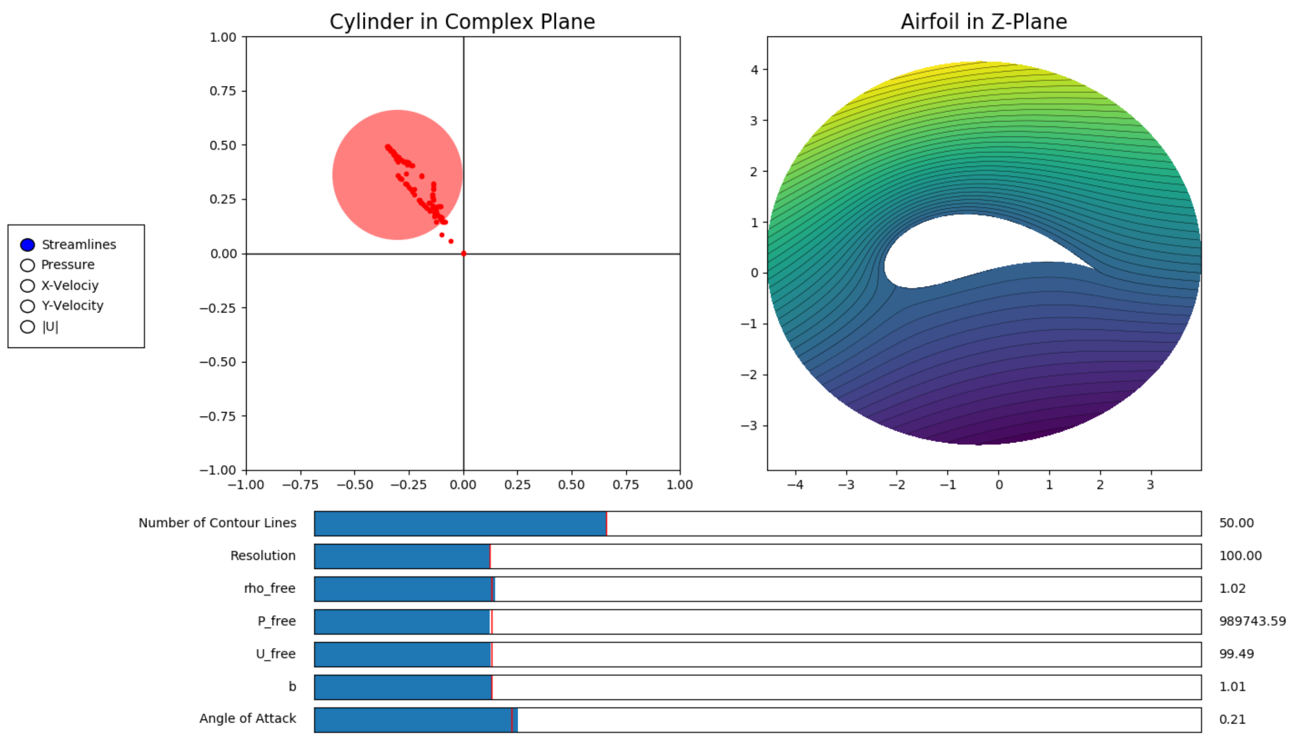


Figure 6: Pressure around an Airfoil

3 Python Code Walkthrough

3.1 Walkthrough

This can be visualized through the use of Python or MATLAB. Python was used because it is free and open source. The GUI and output of the code can be see in Figure 7.



[● Streamlines] ○ Pressure ○ X-Velociy ○ Y-Velocity ○ |U|

Number of Contour Lines	50.00
Resolution	100.00
rho_free	1.02
P_free	989743.59
U_free	99.49
b	1.01
Angle of Attack	0.21

Figure 7: Airfoil Conformal Mapping Plotting GUI

The GUI plots both the location of the cylinder in the complex plane and the airfoil shape in the Z-plane. There is a radio button menu to the left that when double-clicked allows for the changing of plot types for the airfoil. A number of sliders exist on the bottom of the GUI allowing for the tweaking of parameters live in the GUI. The red lines in the sliders denote the default values. The red dots plotted on the cylinder axis denote the locations in which the cylinder has been. A user can use their mouse to click and drag the cylinder around the plot, and see the airfoil update in real-time as the cylinder is dragged around.

The code consists of several sections. Packages are first imported, allowing for plotting to be done as well as the fun GUI interactions. Next, the `DraggablePoints` class is defined and allows for the user control of the cylinder on the GUI using the mouse. This class will not be explained in detail as it is out of the scope of this project document.

Next, the `reDrawAirfoil` function is defined. This function is the meat and bones of the GUI, performing the calculations necessary. This function first defines the radius of the complex cylinder as well as the angle of attack of the flow. Complex locations of the cylinder are developed from this, and ζ is created using these complex locations. Γ and Ψ are defined as derived in this document. The conformal map is applied and the Z-plane locations are then extracted from this. The contour plot can then use the Z-plane locations as well as the imaginary parts of Ψ to plot. $\frac{d\Psi}{dJ}$ is is then defined as derived previously in this document. The real and imaginary parts of that define the u and v components of velocity, respectively. We can then use Bernoulli's to find the pressure. Then, the contours for each can be plotted.

The next function `drawAirfoil` does the initial draw of the airfoil when the GUI comes up. It is only used once and is identical to `reDrawAirfoil` in nearly every way. Lastly, the main function defines global variables and initial starting values for the plots and the GUI. The sliders and radio buttons are defined here as well, including the update information as well that tells the GUI to redraw the contours whenever the radio buttons are double clicked or if a slider is moved.

3.2 ReadMe

In order to run this Python GUI, one must have Python 3 installed. While all computers usually have Python 2 installed with their operating systems, Python 3 is arguably a better product in terms of features and maturity as a programming language. Python 3 can be installed in a variety of ways. The two most popular ways to obtain it are either through the Python Software Foundation at `https://www.python.org/` or usually a smarter method for engineers is through Anaconda at `https://www.anaconda.com/download/`. Anaconda is a scientific computing package manager for Python, and includes many of the popular science and engineering packages for Python. Once Anaconda (for Python 3) or Python 3 are installed on the local computer, the GUI `joukowski.py` can be run via the Terminal on MacOS and Linux or Command Prompt on Windows using `python joukowski.py`. Assuming that one has downloaded the necessary imported packages or just used Anaconda for the install, the GUI will start and user interaction can happen.

It should be noted that this software was developed on Linux (Kubuntu/KDE Neon) with up-to-date QT. If the GUI does not appear to be responsive, such as the slider not responding to mouse clicks or the cylinder not moving when dragged, it is possible that the operating system the user is using is out-of-date or suffering from increasing levels of lack of support from the manufacturer (i.e. Apple or Microsoft) for modern GUI tools. One should contact the authors if this is the case, as there are methods for compiling the source code for native binaries for MacOS and Windows if necessary. Or, a virtual machine running a modern Linux distribution can be used.

4 Variable Exploration

4.1 Impact of ρ_∞

Using the `rho_free` slider on the plotting GUI, we note that changing the freestream density does not change the shape of the streamlines. However, it does change the pressure contours. Increasing ρ_∞ expands the contours outward and away from the airfoil whereas decreasing ρ_∞ retracts the contours and brings them closer to the airfoil.

Using the terminal output, we see that the magnitude of the pressure is directly proportional to ρ_∞. We can also see that it has no change on the magnitude of the velocity.

4.2 Impact of P_∞

Using the `P_free` slider on the plotting GUI, we note that changing the freestream pressure does not change the shape of the streamlines. However, it does change the pressure contours. Increasing P_∞ retracts the contours and brings them closer to the airfoil whereas decreasing P_∞ expands the contours outward and away from the airfoil.

Using the terminal output, we see that the magnitude of the pressure is directly proportional to P_∞. We can also see that it has no change on the magnitude of the velocity.

4.3 Impact of U_∞

Using the `U_free` slider on the plotting GUI, we note that changing the freestream velocity does change the streamlines and the pressure contours. Relative to the airfoil, increasing U_∞ brings the streamline contours closer and pushes the pressure contours away whereas decreasing U_∞ brings the pressure contours closer and pushes the streamline contours away.

Using the terminal output, we see that the magnitude of the pressure and the magnitude of the velocity are directly proportional to U_∞.

4.4 Impact of α

Using the `Angle of Attack` slider on the plotting GUI, we note that changing the freestream angle of attack relative to the airfoil changes both the pressure and freestream contours. Increasing the angle of attack increases the angle at which the streamlines are coincident to the airfoil whereas decreasing the angle of attack decreases that angle. With respect to the pressure contours, increasing the angle of attack expands the pressure contours outward from the airfoil and decreasing α moves the pressure contours towards the airfoil.

The magnitude of the pressure and velocity are a little more complicated to related to angle of attack depending on how extreme the angle is. Increasing or decreasing the angle of attack does not change the pressure magnitude much for either small or large angles of attack. It does however change the magnitude of the maximum velocity seen in the field, being directly proportional for both small and large angles of attack.

4.5 Impact of b

Using the `Angle of Attack` slider on the plotting GUI, we note that changing b changes both the streamlines and pressure contours. The most noticeable effect is that the entire scale of the streamline and pressure contours changes as b changes. The scale of the contours is directly proportional to pressure contours and inversely proportional to the streamline contours.

We see from the terminal output that pressure magnitude is directly proportional to b and velocity magnitude is inversely proportional to b.

4.6 Impact of ζ'

Using the plotting GUI, we can move the location of the complex cylinder using the mouse. This is actively changing ζ'. Moving the cylinder in the vertical direction increases and decreases the camber of the airfoil whereas moving the cylinder in the horizontal direction changes the thickness of the airfoil. The pressure and contours reflect this as they will change shape accordingly, with the streamlines and pressure contours having more curvature as the camber is increased as well as the thickness of the airfoil.

Using the terminal output we see that pressure magnitude and velocity magnitude are directly proportional to ζ'.

5 Discussion

The conformal mapping of an airfoil is surprisingly similar to that of a flat plate. The only necessary changes required were to Ψ and Γ. In fact, just changing the radius of the flat plate example produces contours that look surprisingly like an airfoil which is quite interesting. It is pretty neat that the entire analytical solution of the flow field around an airfoil can be solved for and plotted using just a few lines of code. Granted, we used quite a bit more lines to fulfill the GUI aspect of the project. Another interesting aspect of the project is how easily this method can be adapted to other flow scenarios. The skeleton of the code produced here can be used to plot other potential flow problems, if appropriate conformal maps are chosen.