Numerical Solver for Airflow

Bogdan Vitoc
William Fairman
Chris Lather

December 12, 2018

Abstract

In this project, we apply concepts from vector calculus to find the airflow around any 2-dimensional shape. We make the assumptions that air is an incompressible and irrotational fluid in order to reduce the scope of our project while maintaining a realistic outcome for subsonic air speeds. We use the relaxation algorithm to solve Laplace's equation and ghost points to enforce a Neumann boundary condition at our surface boundary. We succeeded in producing vector-plots and animations visualizing the air velocity around an object.

1 The Velocity Field

Our velocity field ($\vec{V}$) consists of a fluid and an impermeable region for the fluid to flow around. In order to compute ($\vec{V}$), we have to implement a set of governing equations that produce favorable characteristics. In our case, we decided to make the velocity field irrotational and incompressible by implementing the following equations.

$$\nabla \times \vec{V} = 0 \tag{1}$$

$$\nabla \cdot \vec{V} = 0 \tag{2}$$

By setting the curl of $\vec{V}$ to 0 (Eq. 1), $\vec{V}$ is now considered path-independent and can be referred to as the gradient of a potential field. This potential field is known as the velocity potential field and is referred to as Φ. If the divergence of $\vec{V}$ is also equal to 0 (Eq. 2), then we can claim that Φ is a minimal surface defined by Laplace's equation

$$\nabla^2 \Phi = 0 \tag{3}$$

In this case our velocity potential is now a scalar field that represents the minimal amount of energy across the system: allowing the gradient of such a field to accurately represent fluid flow.

2 Relaxation Method

The numerical solution to solve the Laplacian of a scalar field can be found through various algorithms. One of the most simple ways to solve the Laplacian is to set each point in the field equal to an average of it's neighbors. This method, known as relaxation, is an iterative approach to finding the Laplacian and fully solves the equation at time $= \infty$. For our velocity potential field, the value for each discrete fluid point is equal to

$$\Phi_{i,j} = \frac{\Phi_{i-1,j} + \Phi_{i,j-1} + \Phi_{i,j+1} + \Phi_{i+1,j}}{4} \tag{4}$$

3 Surface Condition

The above equations are able to satisfy the conditions for airflow outside the boundary of our object. However, we must add one more parameter to our solution to account for the airflow at the surface of our boundary. For these models, air does not flow through the object. In order to prevent airflow through an object, we must define the $\vec{V}$ to be tangent to the surface of the object. This is true when the dot product between the velocity vector and normal vector of the surface is equal to 0.

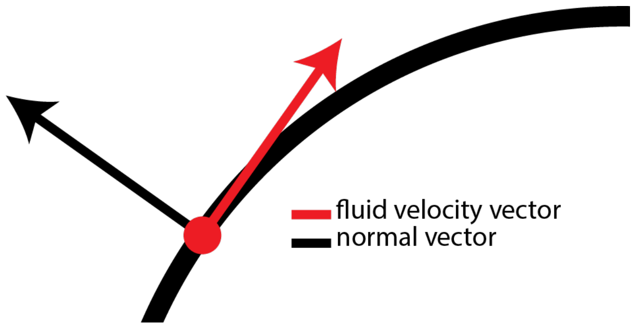

Figure 1: Our simulation is considered realistic when the fluid velocity at the surface is anti-parallel to the normal vector at the surface.

$\vec{V}$ can be re-written as the gradient of Φ and the equation can be reduced to

$$\text{grad}(\Phi) \cdot \hat{\mathbf{n}} = 0 \tag{5}$$

In order to make this condition true, we must set discrete points within the surface to a value that will eventually remove any perpendicular components of airflow when the relaxation method is applied to the surface points. The internal points that are manipulated to satisfy the surface condition are called ghost points.

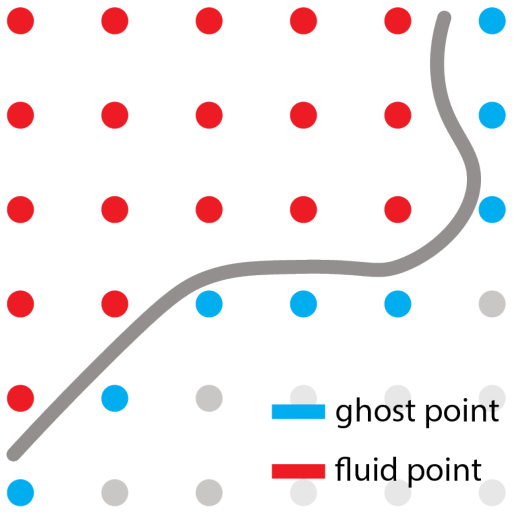

Figure 2: A visualization of ghost points and fluid points bordering a surface. The continuous surface does not need to pass through a point in the numerical grid. Any point located directly on the surface would be classified as a fluid point.

4 Ghost Points

In order for a ghost point to affect a fluid point, the ghost point must touch the fluid point along one of its cardinal sides (North, South, East, West). Flipping this logic around gives us the conditions for selecting our ghost points. If a point internal to the surface comes into contact with a fluid point along one of its cardinal sides, that internal point is a ghost point because it has the ability to make the boundary condition true.

4.1 Setting Values

To find the appropriate value for a ghost point, we must take a closer look at the boundary condition for $\vec{V}$.

$$\text{grad}(\Phi) \cdot \hat{\mathbf{n}} = 0 \tag{6}$$

$$\begin{bmatrix} \frac{\partial \Phi}{\partial x} \\ \frac{\partial \Phi}{\partial y} \end{bmatrix} \cdot \begin{bmatrix} X_n \\ Y_n \end{bmatrix} = 0 \tag{7}$$

If we choose our surface to be a straight horizontal line, the normal vector can be reduced to:

$$\begin{bmatrix} X_n \\ Y_n \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{8}$$

This allows us to further refine our boundary condition:

$$\text{grad}(\Phi) \cdot \hat{\mathbf{n}} = 0 \tag{9}$$

$$\begin{bmatrix} \frac{\partial \Phi}{\partial x} \\ \frac{\partial \Phi}{\partial y} \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = 0 \tag{10}$$

$$\frac{\partial \Phi}{\partial y} = 0 \tag{11}$$

When we implement a discrete representation of $\frac{d\Phi}{dy}$,we can now claim that:

$$\frac{\partial \Phi}{\partial y} = \frac{\Phi_{i,j+1} - \Phi_{i,j-1}}{2} = 0 \tag{12}$$

$$\Phi_{i,j-1} = \Phi_{i,j+1} \tag{13}$$

The value of each ghost point is selected by copying the value of a point mirrored directly across the surface, called the mirror point. If we make the ghost point equal to its mirrored fluid point, the normal component of the gradient between the two values of Φ will be 0, satisfying the boundary condition. Conceptually, by setting an inner ring of ghost points equal to an outer ring of fluid points, we can guarantee the slope across the surface to be tangential (Fig. 1).

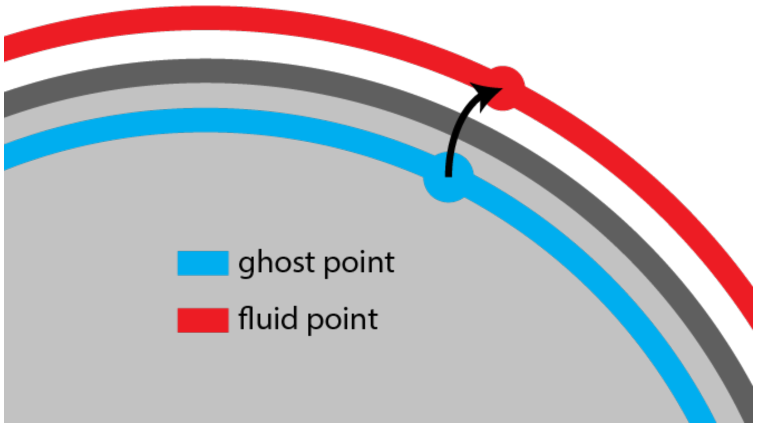

Figure 3: Setting the ghost point equal to the value of its mirrored fluid point will satisfy the boundary condition.

However, in a discrete setting, the number of ghost points will always be less than the number of exterior fluid points because the perimeter of the fluid points coming into contact with the surface is larger than the perimeter of ghost points (at least on a surface without concavities).

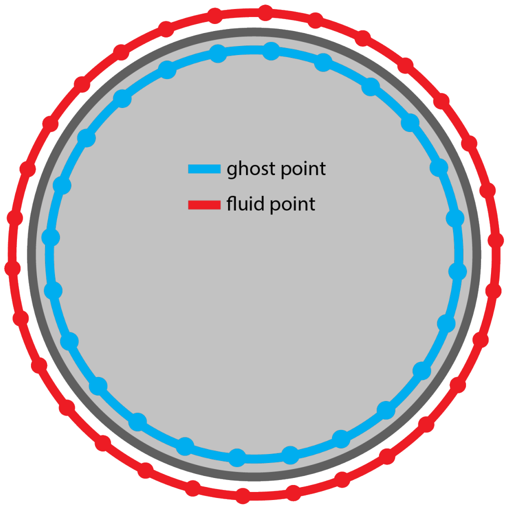

Figure 4: The perimeter of fluid and ghost points is represented by two rings with discrete points equally spaced along both rings. The number of fluid points is larger than the number of ghost points

This inequality makes it difficult to find a fluid point that is a direct mirror of each ghost point. However, if we make the ghost points equal to a *set* of fluid points nearby, we can force that area to be approximately equipotential.

4.2 Stencil Average

To approximate the value of our ghost points, we decided to equate each ghost point to the average value of all the fluid points it touches in a 25-point stencil. Other mirroring techniques are discussed in Lekven (2017)

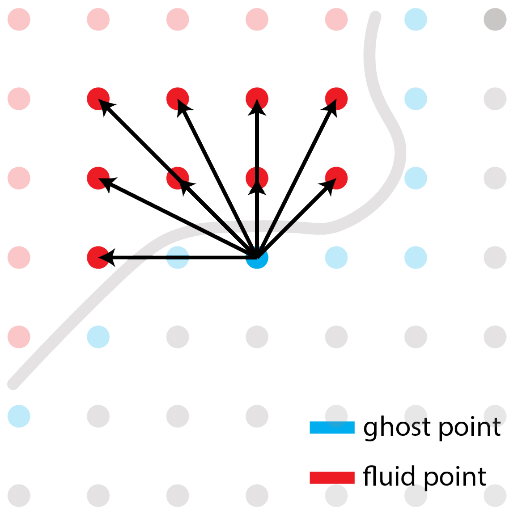

Figure 5: Each ghost point mirrored from an average of its surrounding fluid points. One might accurately assume a 9-point stencil to be more reasonable. The 25-point average is an artifact of a flawed POV.

4.3 Initial Conditions

We need an exterior airflow to give our surface something to disrupt. We create this fluid flow by enforcing Dirichlet boundary conditions on all sides of our field. Since we want the air to be flowing to the right, the left side holds a positive velocity potential while the right side is held at 0. The top and bottom must be held at some value since the relaxation method does not work at a boundary. We choose to hold the top and bottom at zero as well.

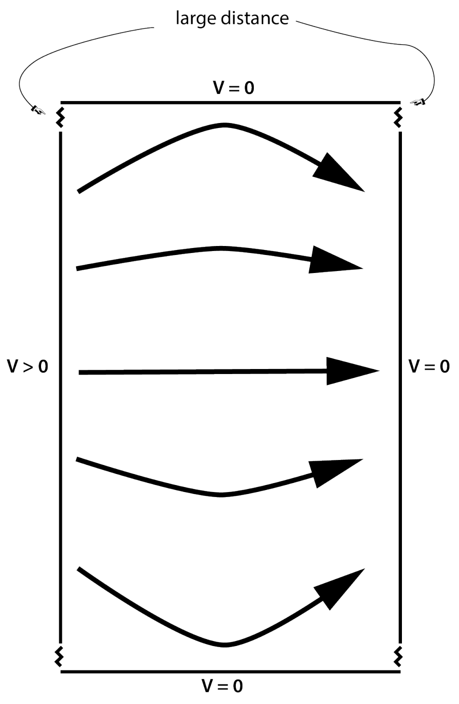

Figure 6: Initial air flow (e.g. from wind).

This does result in a non-parallel airflow most noticeably near the top and bottom. To counteract this effect we made our fluid domain very tall relative to our object. Alternatively the top and bottom sides could be given the same boundary condition as the shape, enforcing a parallel airflow, as if in a wind tunnel. Not only would this increase accuracy, but it would bring big savings in computation time.

5 Results

We decided to setup our simulation environment in Matlab over Mathematica due to its heightened abilities to handle numerical calculations. In order to take advantage of our numerical solver's ability to work for almost any surface, we needed an easy way to define and import complex surfaces. Our final program can convert black and white images into a discrete surface.

Our initial proof of concept was to model the airflow around an ellipse. The velocity potential field (Φ) is shown below:

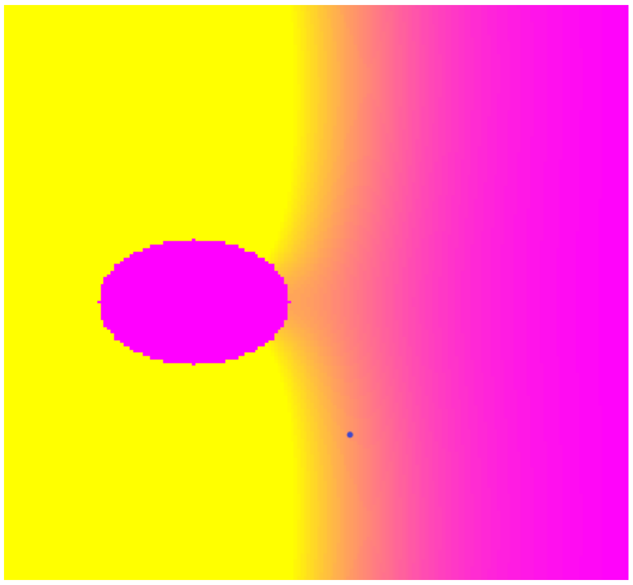

Figure 7: Velocity potential around an ellipse.

In this example, Φ is represented as a color range between bright yellow and purple: where yellow is an area of high potential and purple is an area of low potential. The outline of our object can be seen as an area of low potential because the internal points of the object are set to 0.

The velocity vector $\vec{V} = \nabla \cdot \Phi$ can be shown when taking the gradient of the field.

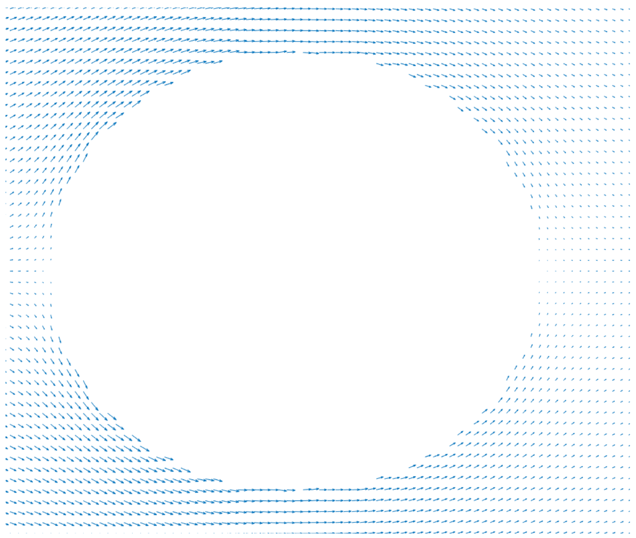

Figure 8: Vector field representing the gradient of the velocity potential field.

While the figure shows an accurate representation of airflow, the airflow as a whole is hard to visualize due to the size of the vector arrows. After modifying our code, we settled on representing airflow as a stream-line plot with Φ overlaid in the background.

5.1 Airflow Graphics

Below is a selection of figures that reflect the ability of our code to parse a black and white image and calculate the accompanying airflow. Each image has an accompanying video that can be found here: https://drive.google.com/drive/folders/1YatS7map3rJvWgA_mO1nS_kQ8kbgIf8t

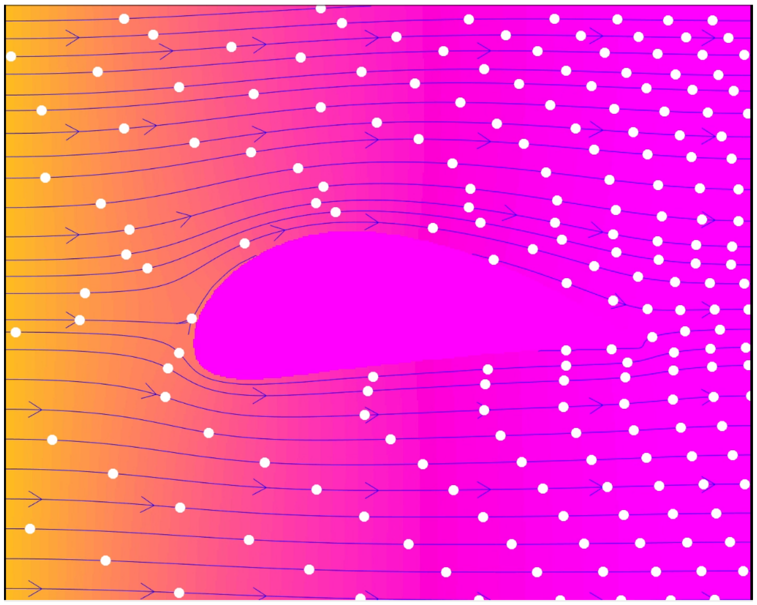

Figure 9: Airflow around an airfoil.

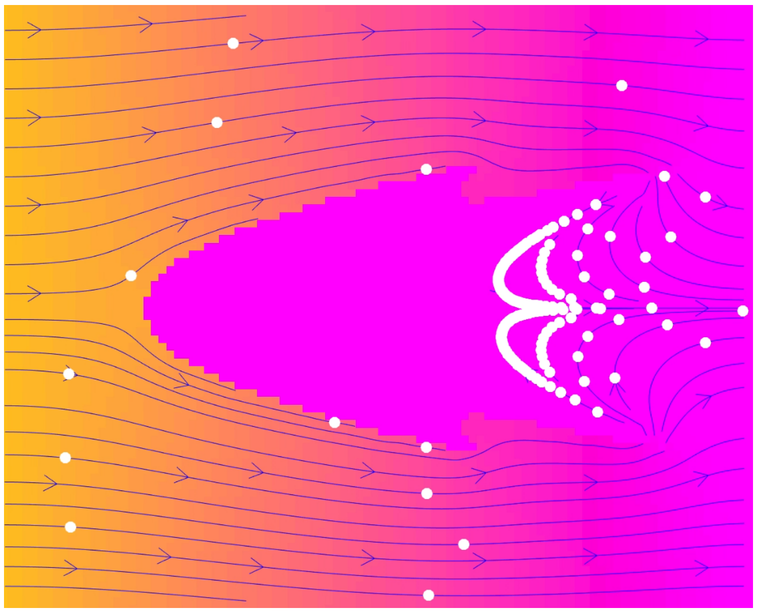

Figure 10: Airflow around an arrowhead.

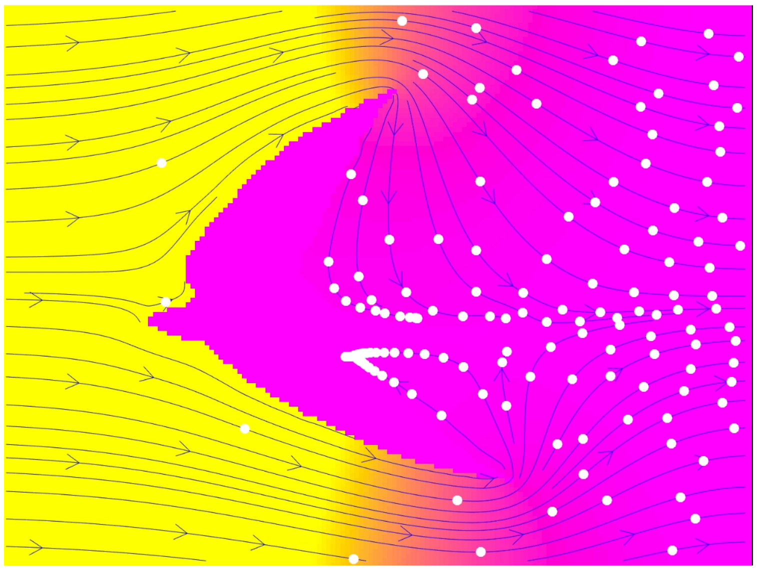

Figure 11: Airflow around an African Swallow carrying a coconut.

References

M. L. Lekven. Verification of a cartesian grid method for compressible flow over moving structures. Master's thesis, NTNU, 2017.

Further Reading

Alexander H-D Cheng and Daisy T Cheng. Heritage and early history of the boundary element method. *Engineering Analysis with Boundary Elements*, 29(3):268–302, 2005.

Armando Coco and Giovanni Russo. A finite difference ghost-cell multigrid approach for poisson equation with mixed boundary conditions in arbitrary domain. *arXiv preprint arXiv:1111.0983*, 2011.

Katya Donovan. Examining Air Flow around a Cylinder. 2017. URL `https://drive.google.com/file/d/1DQr2NhFleutVF-HaHQpKX-SQuHO9A2NW/view`.

Ásdís Helgadóttir, Yen Ting Ng, Chohong Min, and Frédéric Gibou. Imposing mixed dirichlet–neumann–robin boundary conditions in a level-set framework. *Computers & Fluids*, 121:68–80, 2015.