Fluid Flow around a Cylinder & the Pix2Pix Neural Network

Dariyan Khan

Imperial College London

Oral: `https://imperial.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=157502b1-6667-4596-8a0f-ad4801520b65`

Laplace's Equation

We show how Laplace's equation ($\nabla^2\phi = 0$) applies to fluid flow without viscosity. Assuming irrotationality and incompressibility, $\nabla \times \mathbf{u} = 0$ and $\nabla \cdot \mathbf{u} = 0$ via the continuity of mass equation, where $\mathbf{u}$ is the flow velocity vector field. Writing $\mathbf{u}$ as the sum of a curl-free and divergence-free vector (the Helmholtz decomposition), $\mathbf{u} = -\nabla\phi + \nabla \times \mathbf{A}$. From what we've found, it's easy to show that $\nabla \times \mathbf{A} = 0$ and then that $\nabla^2\phi = 0$. ϕ is our velocity potential function. In polar form: $\frac{\partial^2\phi}{\partial r^2} + \frac{1}{r}\frac{\partial \phi}{\partial r} + \frac{1}{r^2}\frac{\partial^2 \phi}{\partial \theta^2} = 0$ (1) [1]

Neural Network: Setup

My neural network was designed to take a heatmap for the boundary conditions of a 64x64 square, and predict what the heat map of the whole square should be.

My training set consisted of 5996 heat maps of boundary conditions (generated by a normal distribution) and the expected output (generated by the method of relaxation for 1000 iterations, where each interior pixel value is the average of the 4 touching pixels). My test set contained 1496 of similarly generated pairs. All images were in greyscale.

Neural Network: Architecture

I used a Pix2Pix model, which is a Generative Adversarial Network (GAN). So, my model actually consists of two neural networks, a generator (G) and discriminator (D). Colloquially, G produces a heat map given the boundary conditions, and then D is passed the boundary conditions and either the real or generated image. D then has to decide whether the image is real or fake. As training occurs, G makes more realistic images to deceive D, and D gets better at finding the fake ones.

Mathematically, G and D play a min-max game where the objective is to minimise

$$G^* = arg\min{}_G \max{}_D \mathcal{L}_{cGAN}(G,D) + \lambda\mathcal{L}_{L1}(G)$$

where:

$\mathcal{L}_{cGAN}(G,D) = \mathbb{E}_{x,y}[\log D(x,y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x,z)))]$

$\mathcal{L}_{L1} = \mathbb{E}_{x,y,z}[||y - G(x,z)||_1]$.

λ is a scaling hyperparameter, usually set to 100.

G is a U-Net auto-encoder, and D is a PatchGAN, which assesses square patches of the images at a time, thereby modelling the image as a Markov random field with pixels in different patches being independent.[6]

A Rotating Cylinder

We want a solution to (1). We assume a solution of the form $R(r)H(\theta)$. Substituting into (1) and rearranging gives $\frac{r^2}{R}(R^{(2)} + \frac{R'}{r}) = -\frac{H^{(2)}}{H}$. These must equal λ^2, where $\lambda \in \mathbb{N}$ for ϕ to be continuous.

Solving both ODEs gives the general solution:

$$\phi_{GS}(r,\theta) = \sum_{n=1}^{\infty}\left[(a_n r^n + b_n r^{-n})(A_n cos(n\theta) + B_n sin(n\theta))\right] + Clog(r) + D\theta + E$$

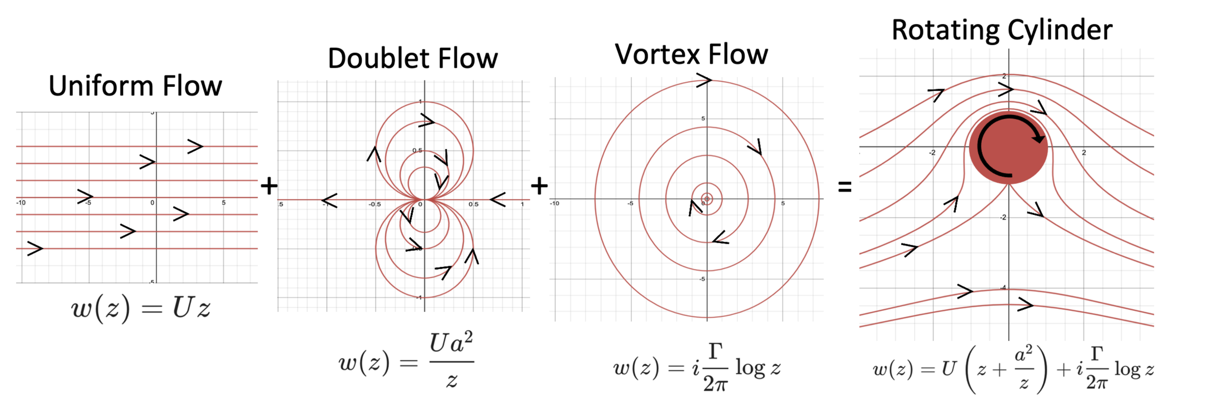
For a uniform flow around a cylinder of radius a with 0 angle of attack, our boundary conditions are:

1. As $r \to \infty$, $\mathbf{u} = Ui + 0j$ for some U so $\phi = Urcos(\theta)$.
2. $\frac{\partial\phi}{\partial r} = 0$ at $r = a$ so no fluid enters the cylinder.

Invoking these, our general solution then becomes

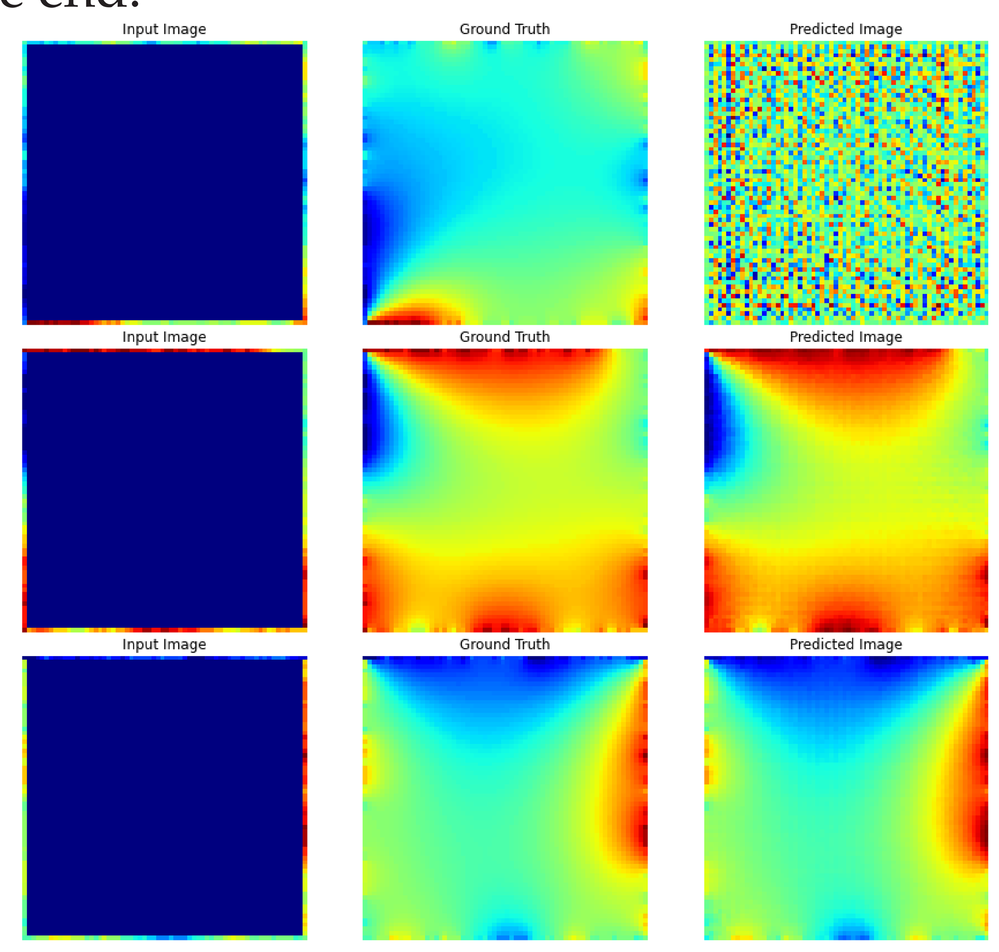
$$\phi = U(r + a^2/r)cos(\theta) + D\theta \ [2]$$

Alternatively, because our required complex potential, w(z), is a uniform, doublet, and vortex flow superimposed, and the Laplacian operator is linear, $w(z) = U(z + a^2/z) + i\frac{\Gamma}{2\pi}\log(z)$ where Γ = 'circulation'.[3]



Neural Network: Training

I trained my GAN for 30 epochs, using Adam and a learning rate of 0.0002, on a Tesla K80 GPU using Google Colab. Here is what the output looked like at the beginning of training, halfway through, and at the end:



D'Alembert's Paradox

The Theorem of Blasius

The force (X,Y) around a fixed 2D body with bounding contour C in a uniform harmonic flow is

$$X - iY = \frac{i\rho}{2}\oint_C \left(\frac{dw}{dz}\right)^2 dz$$

where ρ is fluid density.

For a cylinder with $w(z) = U(z + \frac{a^2}{z}) + i\frac{\Gamma}{2\pi}\log(z)$, the force is

$$X - iY = \frac{i\rho}{2}\oint_{|z|=a}\left(U - \frac{Ua^2}{z^2} - \frac{i\Gamma}{2\pi z}\right)^2 dz$$

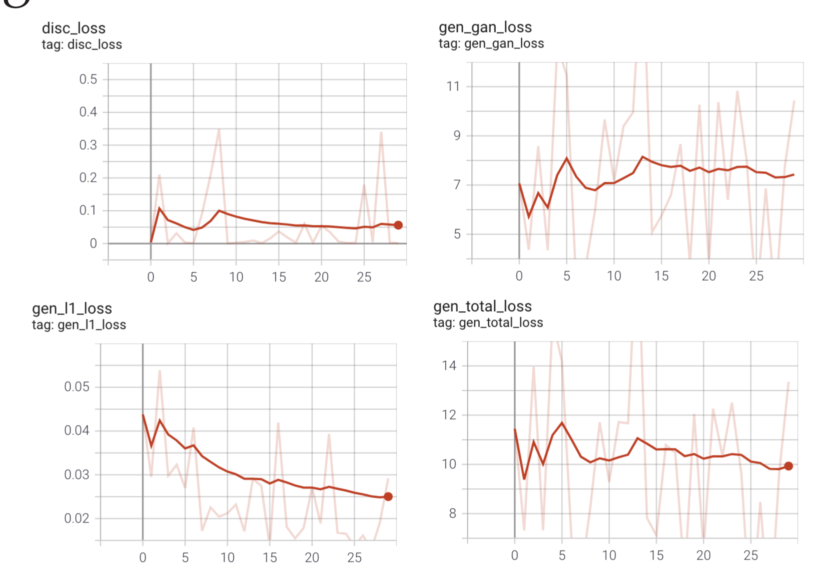
The residue of the integrand is $\frac{-iU\Gamma}{\pi}$, the coefficient of the z^{-1} term. Thus by Cauchy's Reside Theorem, the integral is $2U\Gamma$. So X (our drag force) is 0 and Y (our lift force) is $-\rho U\Gamma$.

But in reality, drag>0. This is D'Alembert's paradox, and it turns out this occurs because we dismissed viscosity.

Our lift value is a special case of the **Kutta Joukowski theorem**, which states that a 2-D body in ambient fluid with velocity U has a lift perpendicular to U of magnitude $\rho U\Gamma$ where $\Gamma = \oint_C \mathbf{u} \cdot d\mathbf{l}$.[3]

Neural Network: Conclusion

Here are the graphs of D's adversarial loss, and G's adverserial loss, L1 loss, and total loss, which I plotted using Tensorflow's TensorBoard:



Clearly, the losses seem to be starting to taper off. However, if I trained for more epochs, I might have been able to lower the L1 loss even more. (Lower L1 loss means generated images are closer to the ground truth). But, I might have let the learning rate decay exponentially over time too, and done more hyperparameter tuning for the batch size, the dropout rate etc.

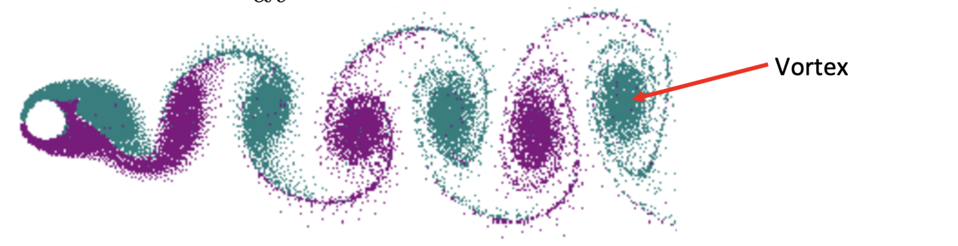
I also trained a model that works with RGB images. This could be adapted to predict fluid flow in a square given the velocity on the boundaries, if we let each RGB layer represent horizontal velocity, vertical velocity, and the pressure field respectively.

What If We Include Viscosity?

Our flow will then be made of two layers: close to the cylinder, we will have a boundary layer where the flow is laminar and viscosity is important, while further out we'll have turbulent inviscid flow.[4]

More interestingly, however, for a stationary cylinder where the Reynold's number ($\rho|\mathbf{u}|L/\mu$ where L is the most important length in the system and μ=viscosity) is between about 60 and 5000, vortices are shed alternately from the top and bottom of the cylinder, and their position z_j can be modelled using the equation for N vortices: $\frac{\overline{dz_j}}{dt} = \frac{-i}{2\pi}\sum_{k=1,k\neq j}^{N}\frac{\Gamma_k}{z - z_k}$.

This is done by considering small time steps τ, and setting $z_k := z_k + \frac{dz_j}{dt}\tau$ each time step.[3]



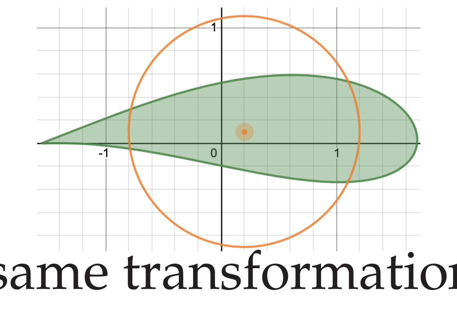
From: https://en.wikipedia.org/wiki/K%C3%A1rm%C3%A1n_vortex_street

Joukowsky Transformation

Conformal mapping allows us to transform complicated geometries into simpler ones that preserve angles and orientation. The Joukowsky transform is a special mapping, that lets us map a circle with equation $z = be^{i\theta}$ to an aerofoil. The transform is

$$w(z) = z + \frac{\lambda^2}{z}$$

However, to create an aerofoil we want the circle to be slightly off centre. Let our circle z have centre s. We then set $\lambda = b - |s|$. So, a unit circle with centre (0.2,0.1) would be mapped to:



If we apply the same transformation to the complex flow for a cylinder, we get the complex flow for the aerofoil above![5]

References

[1] L.D.Landau; E.M.Lifshitz. *Fluid Mechanics (2nd ed.).* U.S.S.R Academy of Sciences.PERGAMON PRESS. 1984. Available at: `https://users-phys.au.dk/srf/hydro/Landau+Lifschitz.pdf` [accessed 29th May 2021]

[2] Imperial College London *Flow around a 2D cylinder.* Available at: `https://www.ma.imperial.ac.uk/~jdg/AECYL.PDF` [accessed 4th June 2021]

[3] S.Childres *Chapter 4 Potential flow.* Available at: `https://www.math.nyu.edu/~childres/chpfour.PDF` [accessed 5th June 2021]

[4] Epifanov,V. M. *BOUNDARY LAYER.* Available at: `https://www.thermopedia.com/content/595/` [accessed 8th June 2021]

[5] *Modelling the Fluid Flow around Aerofoils using Conformal Mapping* N.R.Kapania; K.Terracciano; S.Taylor Available at: `http://www.iiserpune.ac.in/~p.subramanian/conformal_mapping2.pdf` [accessed 10th June]

[6] P.Isola; J.Zhu; T.Zhou; A.A.Efros. *Image-to-Image Translation with Conditional Adversarial Networks* Available at: `https://arxiv.org/pdf/1611.07004.pdf` [accessed 28th May]

Fluid flow plots were made using desmos.com

Thank you to Professor Mestel for his guidance throughout this project. I thoroughly enjoyed it.