Vorticity Stream Function Applied to Corner Junction in 2D Plane

ME 541, Computational Fluid Dynamics

Jonathan Blisko

Dr. Xin Yong

Date: 17 May, 2020

Contents

Abstract

The vorticity stream function allows for solutions to complex flows that would otherwise be very difficult to solve. This type of modeling is extremely useful in scenarios of high turbulence flows, and can give insight into where the flow tends to coalesce. In this paper, the vorticity stream function will be applied to a 90° square corner junction pipe flow. The problem will be simplified down to two dimensions. The expected results should show an area of high pressure on the bottom of the far side, along with some sort of boundary layer forming on the near side of the vertical channel, originating at the edge. The use of the vorticity-stream function could possibly be more efficient than using a technique like finite volume method or finite difference method. The results obtained from the MATLAB code developed in this paper will be compared to that of fluent in order to verify the accuracy.

Introduction

The flow of fluids describes many phenomena found within physics; from the flow of air over an airfoil, to the flow of water through a pipe. The principles behind fluid flow fields are even analogous to phenomena such as magnetic fields in electrodynamics, or the swirl of plasma in high energy physics. The vastness of this subject lends itself to the rigorous and yet rich foundation of physics and mathematics that it lies on.

However, something that is observed quite abruptly when studying fluid mechanics is the fact that most of the problems observed in real life are quite challenging to model, and in most cases impossible. Even though this field is rigorously explored, it is limited by our current understanding of physics and mathematics. This underlines the double-edged sword aspect to fluid mechanics. It is both elevated and yet limited by its own foundation.

This leads us to try and understand how we can model the continuum properties behind flow from a different perspective, since currently an analytic approach is not possible. Due to the advent of electronic computation and how quickly this field has accelerated, we are now able to model these complex flows through numerical techniques quite easily. This method of solution does not give rise to any analytical expression, but rather solves an array of discrete points. The method is aptly named discretization, for breaking up the initial problem into a set of points, referred to as nodes, that can each be solved for based on the values of the nodes surrounding them. This allows for a simple way to approximate derivatives by taking the differences between these nodes.

This form of analysis leads to an entirely new field within fluid mechanics called computational fluid dynamics, the name of which is quite obviously chosen. Although this method is not perfect, it allows for the characteristics of the complex flows to be known, something only previously understood from experimentation. Most techniques to describing fluid flow utilize the Navier-Stokes equations, a set of equations that have very few analytical solutions due to their inherent complexity. With the inclusion of the continuity equation, and using the definition of the stream function and vorticity of a flow field, we can derive a simpler expression to solve the fluid flow referred to as the Vorticity-Stream Function, otherwise known as the Psi-Omega formulation. For the purposes of this report, we will be utilizing this set of equations in order to solve our flow.

Problem Formulation

In this section, we will be defining each aspect of the model being solved in this report, and providing insight into how they were derived. This includes the fluid domain, governing equations, and boundary conditions. We simplify the model by restricting the space we are working in to two-dimensions.

Fluid Domain

We begin the formulation of the problem with the domain in which we are working. For the purposes of this problem, it is sufficient to choose a 1×1 square domain in which we can embed our problem into. The channel width is given by half of this square domain, as can be seen below.

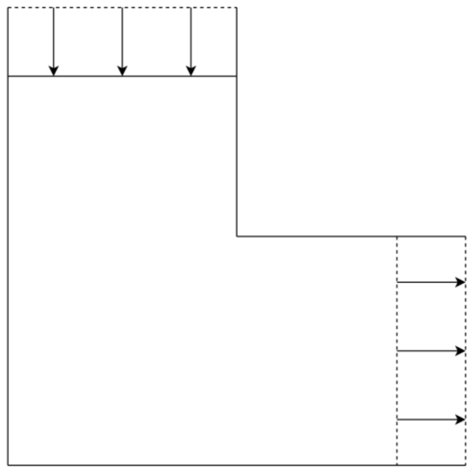

Figure 1. Channel Geometry

In order to solve the governing equations in this domain, the channel has to be split up into two separate areas: the column on the left-hand side, and the rectangular domain on the bottom right.

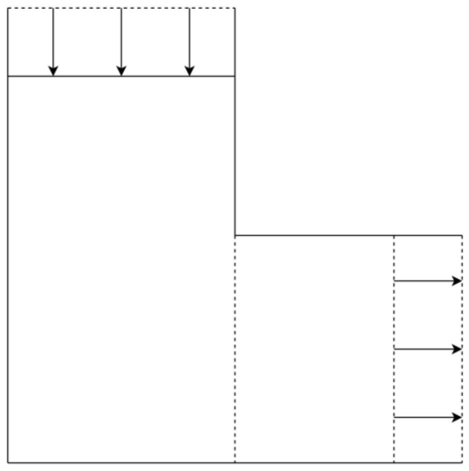

Figure 2. Split Channel Geometry

These two sections of the flow domain will be referred to as the inlet channel and outlet channel respectively.

Governing Equations

As talked about previously, the governing equations that we will be dealing with are the vorticity-stream functions. This formulation is characterized by two variables, ψ and ω, which are the stream function and vorticity respectively. The equations that govern these variables is given by two coupled PDE's:

$$\nabla^2 \psi = -\omega$$

$$\frac{\partial \omega}{\partial t} + v \cdot \nabla \omega = \frac{1}{Re} \nabla^2 \omega$$

Taking note that these are the nondimensional forms of the equations. Using a FTCS scheme, we obtain a discretized form of the equations.

$$\frac{\psi^n_{i-1,j} - 2\psi^n_{i,j} + \psi^n_{i+1,j}}{\Delta x^2} + \frac{\psi^n_{i,j-1} - 2\psi^n_{i,j} + \psi^n_{i,j+1}}{\Delta y^2} = -\omega^n_{i,j}$$

$$\frac{\omega^{n+1}_{i,j} - \omega^n_{i,j}}{\Delta t} = -u^n_{i,j}\left(\frac{\omega^n_{i+1,j} - \omega^n_{i-1,j}}{2\Delta x}\right) - v^n_{i,j}\left(\frac{\omega^n_{i,j+1} - \omega^n_{i,j-1}}{2\Delta y}\right) + \frac{1}{Re}\left[\frac{\omega^n_{i+1,j} - 2\omega^n_{i,j} + \omega^n_{i-1,j}}{\Delta x^2} + \frac{\omega^n_{i,j+1} - 2\omega^n_{i,j} + \omega^n_{i,j-1}}{\Delta y^2}\right]$$

If we choose the same spacing in both the x and y-directions, denoted by h, we can simplify these further. The derivation of these equations is found in the Appendices at the end of the paper.

$$\psi^n_{i-1,j} + \psi^n_{i+1,j} + \psi^n_{i,j-1} + \psi^n_{i,j+1} - 4\psi^n_{i,j} = -h^2\omega^n_{i,j}$$

$$\omega^{n+1}_{i,j} = \omega^n_{i,j} - \frac{u^n_{i,j}\Delta t}{2h}\left(\omega^n_{i+1,j} - \omega^n_{i-1,j}\right) - \frac{v^n_{i,j}\Delta t}{2h}\left(\omega^n_{i,j+1} - \omega^n_{i,j-1}\right) + \frac{\Delta t}{h^2 Re}\left[\omega^n_{i+1,j} + \omega^n_{i-1,j} + \omega^n_{i,j+1} + \omega^n_{i,j-1} - 4\omega^n_{i,j}\right]$$

We can now apply an SOR–Method scheme to the Poisson equation, the derivation of which is found in Appendix A.

$$\psi^{n+1}_{i,j} = \frac{\alpha}{4}\left(\psi^{n+1}_{i+1,j} + \psi^n_{i-1,j} + \psi^{n+1}_{i,j+1} + \psi^n_{i,j-1} + h^2\omega^{n+1}_{i,j}\right) + (1-\alpha)\psi^n_{i,j}$$

Where we have the relaxation factor $\alpha \in (1,2)$. This relaxation factor can be optimized for discretized Poisson equations.

$$\alpha_{opt} = \frac{2}{1 + \sqrt{1 - p^2}}, \qquad p = \frac{\cos(\pi/n_x) + (\Delta x/\Delta y)^2 \cos(\pi/n_y)}{1 + (\Delta x/\Delta y)^2}$$

The horizontal and vertical velocities at each node can now be updated using the newly found values for the stream function.

$$u^{n+1}_{i,j} = \frac{\left(\psi^{n+1}_{i,j+1} - \psi^{n+1}_{i,j-1}\right)}{2h}, \qquad v^{n+1}_{i,j} = -\frac{\left(\psi^{n+1}_{i+1,j} - \psi^{n+1}_{i-1,j}\right)}{2h}$$

Boundary Conditions

We begin establishing boundary conditions for the problem with the inlet velocity, which for this model will be a constant velocity across the gap.

$$v_{i,ny} = V$$

For which $i \in [1, hf]$ gives the nodes along the inlet. We can now define the stream function along the inlet, the derivation of which is found in Appendix A.

$$\psi_{i,ny} = Vx$$

Since there is no pressure term within the vorticity-stream function, there must be some other component that is driving the flow. What we find is that the flow is driven by the vorticity, and more specifically, the boundary conditions specified for the vorticity. We obtain these boundary conditions through the Taylor series expansion of the stream function, utilizing the Poisson equation along with the no-slip and no-penetration assumptions. We can simply establish the wall boundary conditions generally as follows.

$$\omega_N = \frac{2}{h^2}(\psi_N - \psi_n)$$

Where N denotes the node on the boundary and n denotes the node adjacent to the boundary in the flow domain. For the inlet, we have a general form for the vorticity.

$$\omega_N = \frac{2}{h^2}(\psi_N - \psi_n) - \frac{\partial^2\psi}{\partial x^2}$$

This can be simplified due to the velocity boundary condition we had established earlier at the inlet. Since we know that $\frac{\partial\psi}{\partial x} = -v$, and we know that the velocity is constant along the gap, then we must have that $\frac{\partial^2\psi}{\partial x^2} \equiv 0$. Therefore, the inlet vorticity boundary condition simplifies to the wall boundary condition.

Results (MATLAB)

This section gives the results that were found from using the MATLAB code that was written specifically for this problem. We will be investigating the flow for a Reynolds of $Re = 200$. This Reynolds was chosen due to the reasonable runtime of the code, as well as the fact that it captures the characteristic shape of the flow. The mesh size used throughout this section will be the same in both the x and y-directions, and due to long runtimes was chosen to be $h = 0.01$. The inlet velocity, which is constant across the gap, was chosen to be $V = 1$ and $V = 2$ in order to see what the effect the inlet velocity has on the resulting flow.

Results for $Re = 200, V = 1$

The first result that we will be looking at is the flow for a Reynolds of $Re = 200$ and a velocity at the inlet of $V = 1$.

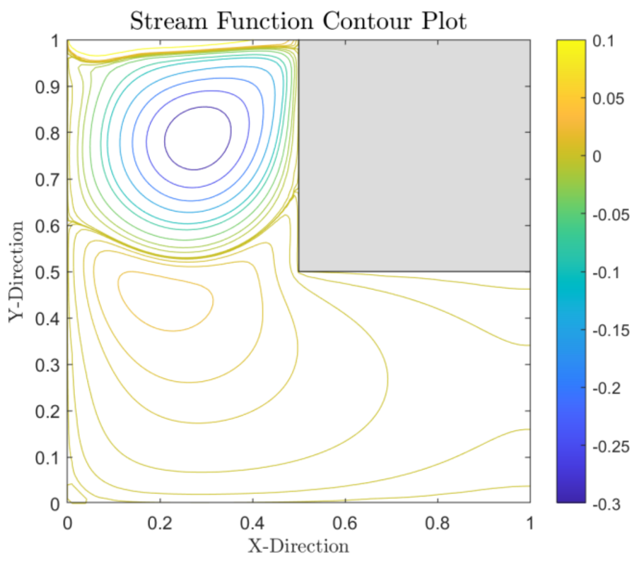

Figure 3. Stream Function from MATLAB

This plot gives the contours of the stream function for the given inlet boundary conditions. We can see how there is a vortice forming at the inlet, with another area of flow concentration forming below it. This is in contrast with what was expected, which would have had a vortice forming in the bottom left corner. A possible explanation for this could arise from the fact that we have undeveloped flow entering the system, but rather a constant velocity across the inlet. A more in-depth explanation for this phenomenon is found within the conclusion. We can now look at the vorticity results in order to get a better idea of the characteristics of the flow.

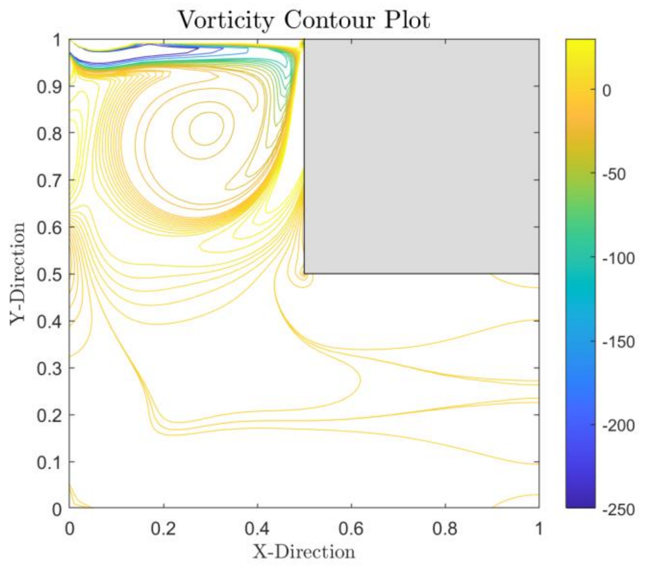

Figure 4. Vorticity Contours from MATLAB

This plot gives the contours of the vorticity of the flow. We can see the vortice at the inlet that was noticed to have formed from looking at the stream function. We can also see some vortices forming at the corners edge, which is expected due to the jarring nature of a square 90° corner. We now look at the vertical velocity flow field.

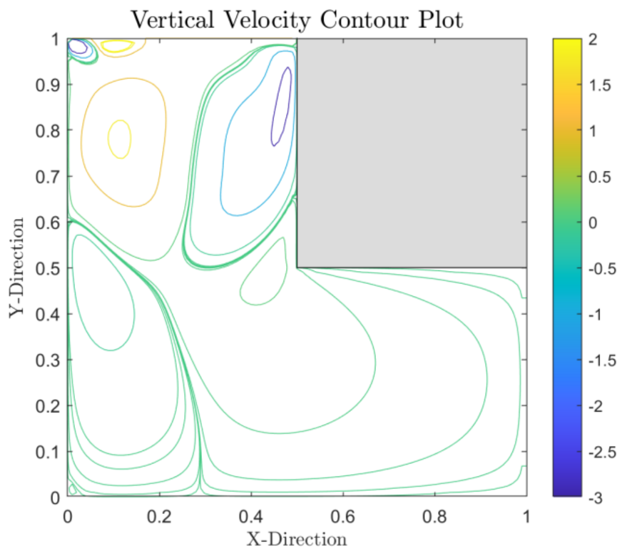

Figure 5. Vertical Velocity Field from MATLAB

As expected, the results of the velocity flow field agree with that of the stream function and vorticity. We can see a large area of positive flow on the left-hand side wall near the inlet, along with an area of largely negative flow on the right-hand side wall at the inlet. These contradicting directions of flow are what gives rise to the vortice that we see forming at the inlet. We also see that there is a small amount of vertical flow at the outlet, which is expected. We now look at the horizontal velocity flow field.

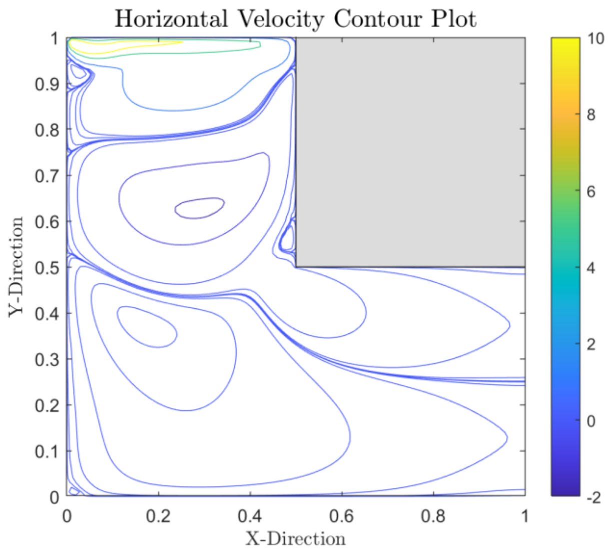

Figure 6. Horizontal Velocity Field from MATLAB

Again, we see results that agree with the previous plots. This plot, which displays the contours of the horizontal velocity, shows an area of large positive velocity at the inlet and large negative velocity below that. We also see much of the flow turning into horizontal flow towards the outlet, which is what is expected to happen for flow through a duct.

The results from this were quite surprising due to the turbulent-like nature of the flow at the inlet. In order to investigate this system further, we will be changing the inlet velocity. This is done in an attempt to observe whether a change in the properties of the system will change the overall characteristics of the flow, or if the characteristics are inherent to the system.

Results for $Re = 200, V = 2$

We want to see how the system will react when one of the parameters that drives the flow changes. This can be done through either changing the Reynolds or changing the initial inlet velocity. This formulation did not handle high Reynolds numbers well, so we will be changing the inlet velocity to twice of that in the previous model, $V = 2$. We begin by looking at the results for the stream function of this new system.

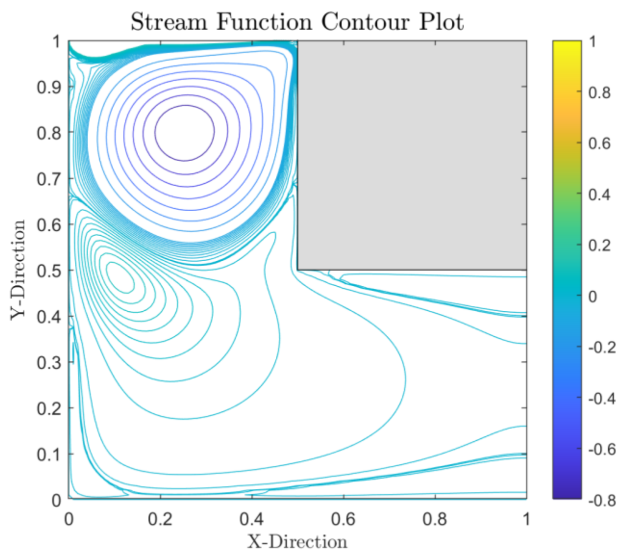

Figure 7. Stream Function from MATLAB

We can see that the results from this plot bears a similar appearance to the previous simulation, with two main areas of flow concentration. We again look at the vorticity solution to get a better idea of the characteristics of the flow.

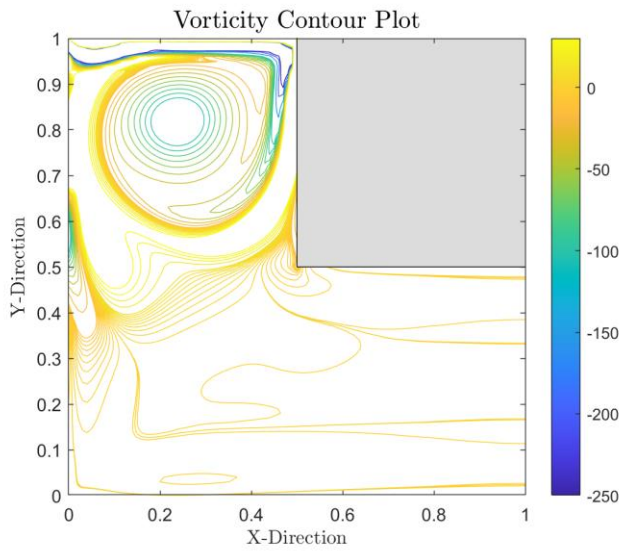

Figure 8. Vorticity Contours from MATLAB

This plot again shows that the characteristics behind the flow are not changing despite the change in the inlet velocity. We do see some higher magnitude vorticity occurring on the left-hand side wall, along with a more intense vorice at the inlet. We observe the vertical velocity next.

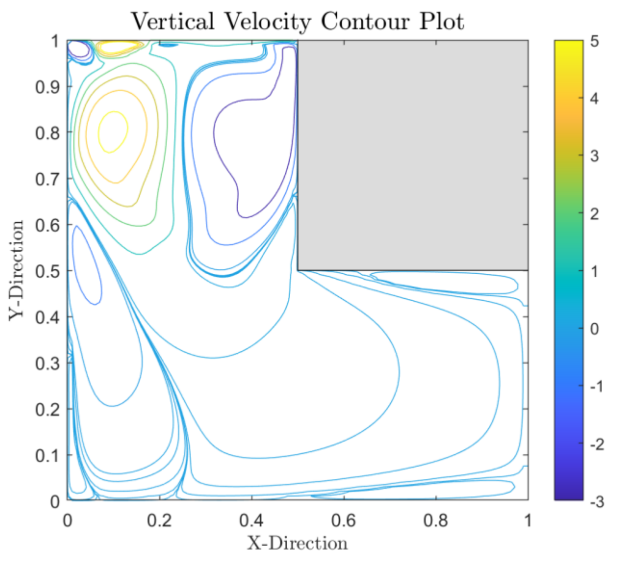

Figure 9. Vertical Velocity Field from MATLAB

As was seen in the above plots, a similar characteristic shape to the flow is formed as compared to the previous value for the inlet velocity. The two solutions just differ in magnitude. We observe a similar effect for the horizontal velocity.

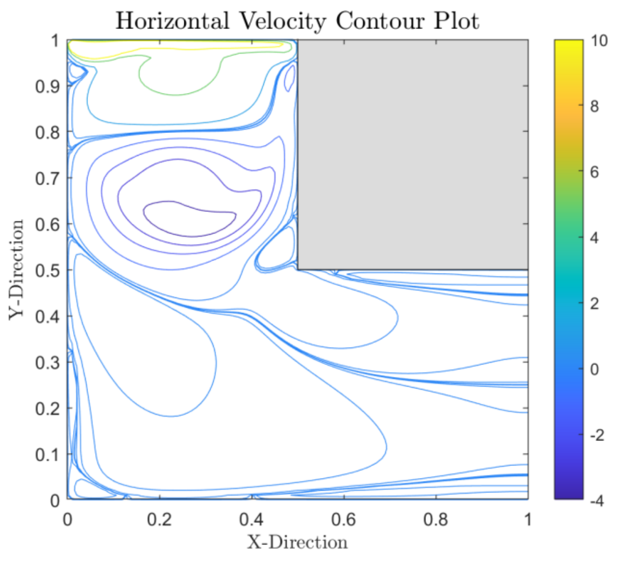

Figure 10. Horizontal Velocity Field from MATLAB

We will now attempt to verify the results found from this MATLAB code by using the commercial software FLUENT. The setup and results from FLUENT are outlined below.

Verification (Fluent)

In this section, we will be going over how this was modelled in FLUENT, what technique was chosen to solve the problem, and what the results from the simulation came out to. Since within FLUENT the properties of the fluid have to be well defined, the results from this and MATLAB will differ in magnitude, since we only needed to define the Reynolds within the MATLAB code. However, the characteristics of the flow should remain the same, so that is what we will be focusing on for the comparison between the two solutions to the flow.

Flow Domain Setup

The geometry of the problem was set up in FLUENT using their built-in CAD software, Space Claim. The mesh spacing was chosen such that there were twice as many divisions on the left-hand side wall and bottom wall than at the inlet, right-hand side wall, upper wall, and outlet, since these areas are half the size. The mesh was chosen to be 100 divisions per unit length. Each of the walls, the inlet, and the outlet were all defined in order to establish boundary conditions later.

In the FLUENT software, we want to make sure that the correct modelling technique is being used under the viscous tab. For this model, a k-omega formulation was used in order to solve the system. Initially a laminar formulation was chosen. However, this was seen to be incorrect. This will be discussed in the conclusion section of the report. The fluid properties must also be adjusted within the materials tab.

The boundary conditions were established such that the inlet velocity was a constant across the gap, which for our purposes was simply set to $U = 1$. The outlet boundary was set using a pressure difference such that the Gauge pressure was zero. A no slip condition was used at the walls. The initial condition of the system was set using a standard initialization, with each component set to zero.

Each simulation was run using 500 iterations, since beyond this many iterations the error within the system was effectively not changing. The convergence conditions were set to be 10^{-6}. The results of the simulation yield contour plots of the stream function, vorticity magnitude, vertical velocity field, and horizontal velocity field.

Results

The first result from running the FLUENT simulation is taken from the solution for the stream function.

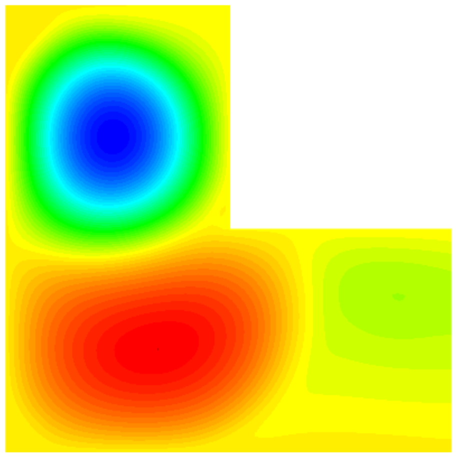

Figure 11. Stream Function from FLUENT

This result has a very similar characteristic to the solution found using MATLAB, with an area near the inlet forming what should be a vortice. We also see a similar area forming beneath this, with the outlet also following a similar characteristic appearance to that found by MATLAB. Looking at the plot for the vorticity yields a similar result in agreeing with the MATLAB code.

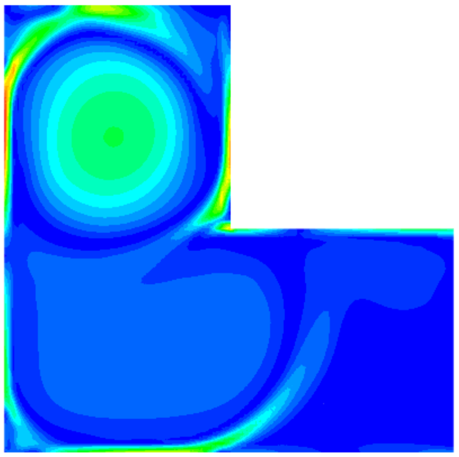

Figure 12. Vorticity Contours from FLUENT

Comparing Figure 12 to Figures 4 and 8, we can clearly see that the solution from FLUENT yields the same solution. We can continue to verify these results by comparing the vertical and horizontal velocity fields. We begin by comparing the vertical velocity field.

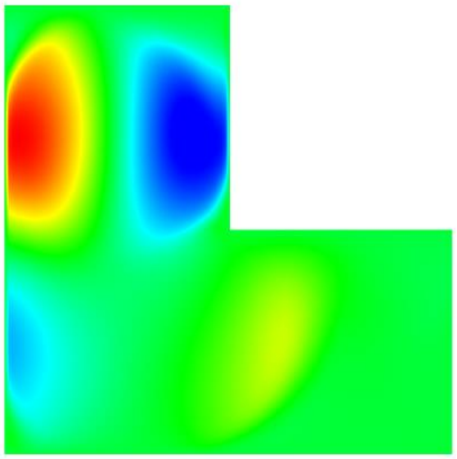

Figure 13. Vertical Velocity Field from FLUENT

Comparing this plot to that of Figures 5 and 9, we see that there are concentration of flow forming in similar areas. On the left-hand side wall, we see a high concentration of positive flow on the upper side, and a concentration of negative flow on the lower side. The right-hand side wall also resembles the solution found from MATLAB, with a high concentration of negative flow along the wall. These concentrated areas of flow are what give rise to the vortice forming at the inlet. We continue to compare the results from FLUENT and MATLAB with the horizontal velocity field.

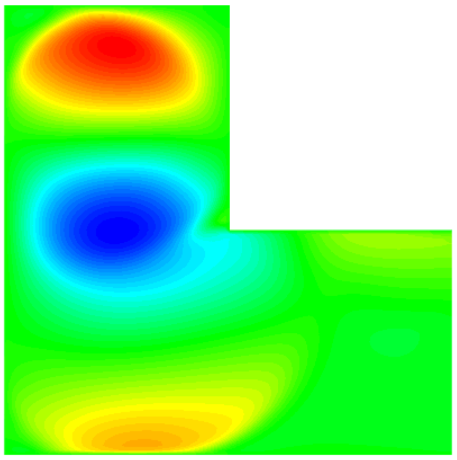

Figure 14. Horizontal Velocity Field from FLUENT

Again, a similar result is found as compared to Figures 6 and 10. A large concentration of positive flow is found at the inlet, with a large concentration of negative flow below that. These two areas of flow are again due to the vortice that forms at the inlet. The area of positive flow in the above figure is more pronounced than that found in Figures 6 and 10. However, that area of concentrated flow is still present in those figures.

Ultimately, the solution found from FLUENT agrees with the solution found from MATLAB. Although the magnitudes were not able to be compared, the general character of the flow was in agreeance between the two different models.

Conclusion

We started this paper with wondering what the flow through a square 90° duct would look like. The approach taken to solve this problem was done using the psi-omega formulation, which was coded in MATLAB. These results were then compared to the commercial software FLUENT using a k-omega formulation. Although the magnitudes of the two styles were not the same due to the way in which the system was defined within the code, the characteristics of the flow were the same.

A vortice at the inlet was observed in both solutions, along with a flow density below that. The outlet had what appeared to be laminar flow developing, which is expected since it was a straight channel. An explanation for the vortice at the inlet could potentially come from the fact that the inlet velocity was constant across the gap, a situation that could arise from some sort of pump. This constant velocity meant that the flow was undeveloped at the inlet, yet was quickly met with a sharp 90° turn. This led to the turbulent-like flow that we see at the inlet.

If this had been through just a typical channel, the solution would have been able to take on a laminar form. However, due to the geometry of the system this was not the case. This problem was made apparent when trying to solve the system using a laminar model within FLUENT. The result obtained from FLUENT with a laminar model was drastically different from the solution that was developed within MATLAB. An attempt was made to try and replicate these results from FLUENT by establishing constant streamlines along the walls and lowering the Reynolds. However, when the system was solved within FLUENT using the k-omega formulation, the results came out to have the same characteristics as the results from MATLAB. What was quickly noticed was that a laminar model would not work in this system. This implied that establishing the streamline values along the wall would not make sense, since streamlines do not exist in turbulent flow.

This does not solve the problem of why a vortice was observed to have formed at the inlet. Intuitively, it would make more sense for an eddy to form at the bottom left hand corner. What was observed in both the MATLAB and FLUENT solutions was quite different however. One possible explanation for this would be that the square corner created an area of low pressure due to escaped fluid. However, since the system is not symmetric, the same low-pressure area was not observed on the other side of the flow. Since the velocity at the inlet was uniform, this allowed for flow to tend towards the right-hand side where the corner is, filling the area of low-pressure. This then created an area of lower pressure on the left-hand side since the fluid was tending towards the right-hand side. This then perpetually created the vortice seen.

If a non-uniform, fully-developed flow had been established at the inlet, this phenomenon would possibly not have been noticed. This is something to look into more in a future project in an attempt to alleviate this turbulence at the inlet. Another interesting project to continue this system would be to investigate the effect of changing the geometry to a T-shape junction. The symmetry in that system could possibly alleviate the turbulence at the inlet.

This report was done in order to verify the understanding of the material taught in this course. This report utilized both a self-written code written in MATLAB, along with the commercial software FLUENT which was used to verify the results obtained from using MATLAB. The solution found from both techniques yielded the same general result, confirming the code that was written in MATLAB. This demonstrated a functional knowledge of both writing CFD code, along with using commercial software that is readily available for use.

References

[1] Pirozzoli, S., Modesti, D., Orlandi, P., & Grasso, F. (2018). Turbulence and secondary motions in square duct flow. *Journal of Fluid Mechanics, 840,* 631-655. doi:10.1017/jfm.2018.66

[2] Dey, P., Das, A.K.R. A numerical study on effect of corner radius and Reynolds number on fluid flow over a square cylinder. *Sādhanā* **42,** 1155–1165 (2017). https://doi.org/10.1007/s12046-017-0680-2

[3] Matyka, Maciej. “Solution to two-dimensional Incompressible Navier-Stokes Equations with SIMPLE, SIMPLER and Vorticity-Stream Function Approaches. Driven-Lid Cavity Problem: Solution and Visualization.” (2004).

Appendix

This is the start to the Appendix section of the report. This is all of the supplementary material that was not necessary to include in the main body of the report, but is still important to show in order to verify the model was correctly derived. There are two parts to this Appendix; Appendix A deals with the derivation behind the problem formulation, and Appendix B gives the MATLAB code used to solve the problem.

Appendix A: Derivations

This section outlines the required derivations for this paper: Governing Equations and Boundary Conditions.

A-1. Governing Equations

Poisson Equation – SOR Approach

We start with the psi-omega Poisson equation.

$$\nabla^2\psi = \frac{\partial^2\psi}{\partial x^2} + \frac{\partial^2\psi}{\partial y^2} = -\omega$$

We use a central space approach to discretize.

$$\frac{\psi^n_{i+1,j} - 2\psi^n_{i,j} + \psi^n_{i-1,j}}{\Delta x^2} + \frac{\psi^n_{i,j+1} - 2\psi^n_{i,j} + \psi^n_{i,j-1}}{\Delta y^2} = -\omega^n_{i,j}$$

We will assume equal spacing in the x and y-directions, defining $h = \Delta x = \Delta y$. Solving for $\psi^n_{i,j}$:

$$\psi^n_{i,j} = \frac{1}{4}\left(\psi^n_{i+1,j} + \psi^n_{i-1,j} + \psi^n_{i,j+1} + \psi^n_{i,j-1} + h^2\omega^n_{i,j}\right)$$

Applying a Gauss – Seidel method:

$$\psi^{n+1}_{i,j} = \frac{1}{4}\left(\psi^{n+1}_{i+1,j} + \psi^n_{i-1,j} + \psi^{n+1}_{i,j+1} + \psi^n_{i,j-1} + h^2\omega^{n+1}_{i,j}\right)$$

Extending this to SOR – Method:

$$\psi^{n+1}_{i,j} = \frac{\alpha}{4}\left(\psi^{n+1}_{i+1,j} + \psi^n_{i-1,j} + \psi^{n+1}_{i,j+1} + \psi^n_{i,j-1} + h^2\omega^{n+1}_{i,j}\right) + (1-\alpha)\psi^n_{i,j}$$

Where $\alpha \in (1,2)$ is the relaxation factor, and has an optimal value for discretized Poisson solutions.

$$\alpha_{opt} = \frac{2}{1+\sqrt{1-p^2}}, \qquad p = \frac{\cos(\pi/n_x) + (\Delta x/\Delta y)^2\cos\left(\pi/n_y\right)}{1+(\Delta x/\Delta y)^2}$$

Transport Equation

We have the nondimensionalized transport equation for ω:

$$\frac{D\omega}{Dt} = \frac{\partial\omega}{\partial t} + \vec{v}\cdot\nabla\omega = \frac{1}{Re}\nabla^2\omega$$

Using a FTCS scheme:

$$\frac{\omega_{i,j}^{n+1}-\omega_{i,j}^{n}}{\Delta t} = -u_{i,j}^{n}\left(\frac{\omega_{i+1,j}^{n}-\omega_{i-1,j}^{n}}{2\Delta x}\right) - v_{i,j}^{n}\left(\frac{\omega_{i,j+1}^{n}-\omega_{i,j-1}^{n}}{2\Delta x}\right) + \frac{1}{Re}\left[\frac{\omega_{i+1,j}^{n}-2\omega_{i,j}^{n}+\omega_{i-1,j}^{n}}{\Delta x^2} + \frac{\omega_{i,j+1}^{n}-2\omega_{i,j}^{n}+\omega_{i,j-1}^{n}}{\Delta y^2}\right]$$

Solving for $\omega_{i,j}^{n+1}$:

$$\omega_{i,j}^{n+1} = \omega_{i,j}^{n} - u_{i,j}^{n}\Delta t\left(\frac{\omega_{i+1,j}^{n}-\omega_{i-1,j}^{n}}{2\Delta x}\right) - v_{i,j}^{n}\Delta t\left(\frac{\omega_{i,j+1}^{n}-\omega_{i,j-1}^{n}}{2\Delta x}\right) + \frac{\Delta t}{Re}\left[\frac{\omega_{i+1,j}^{n}-2\omega_{i,j}^{n}+\omega_{i-1,j}^{n}}{\Delta x^2} + \frac{\omega_{i,j+1}^{n}-2\omega_{i,j}^{n}+\omega_{i,j-1}^{n}}{\Delta y^2}\right]$$

A-2. Boundary Conditions

Psi Inlet Formulation

We begin with the given boundary condition of

$$v_{i,ny} = V$$

Due to no horizontal velocity at inlet, we have that $\frac{\partial\psi}{\partial y} = 0$. Therefore, we can say the following.

$$\psi(x) = f(x) = \int_0^x v(s)ds$$

Integrating over x, we have the final result.

$$\psi_{i,ny} = Vx_i$$

Appendix B: MATLAB Code

```
% Title: Vorticity-Stream Function at Square Corner
% Author: Jonathan Blisko
% Date: 05/17/20

clear all
close all
clc
```

Initialize Variables

```
% X-Direction
L=1;
deltax=0.01;
nx=L/deltax+1;
x=0:deltax:L;

% Y-Direction
H=L;
```

```
deltay=deltax;
ny=H/deltay+1;
y=0:deltay:H;

% Constants
U=2;
p=(cos(pi/(nx-1))+(deltax/deltay)^2*cos(pi/(ny-1)))/(1+(deltax/deltay)^2);
w_opt=2/(1+sqrt(1-p^2));
hf=floor((nx-1)/2);
Re=200;
deltat=deltax^2/(4*U);
c=deltat/(2*deltax);
r=deltat/deltax^2;
SORConvergence=1e-8;
SteadyConvergence=1e-6;
k=1;
max_it=50000;
diff=1;
```

Initial Conditions

```
w=zeros(nx,ny);
psi=zeros(nx,ny);
u=zeros(nx,ny);
v=zeros(nx,ny);

% Inlet and Oulet Stream Function
v(2:hf,ny)=U;
psi(2:hf+1,ny)=v(2:hf+1,ny).*linspace(L/2,0,hf)';
%psi(2:hf+1,ny)=deltax*v(2:hf+1,ny)+psi(2:hf+1,ny-1);

% Exclude Upper Right Quadrant
w(hf+2:nx,hf+2:ny)=NaN;
psi(hf+2:nx,hf+2:ny)=NaN;
u(hf+2:nx,hf+2:ny)=NaN;
v(hf+2:nx,hf+2:ny)=NaN;
```

Solving Equations

```
while (diff>=SteadyConvergence && k<max_it)
    w_old=w;

    % Transport Equation: Left Column
    for ii=2:hf
        for j=2:ny-1
            w(ii,j)=w(ii,j)-c*u(ii,j)*(w(ii+1,j)-w(ii-1,j))-c*v(ii,j)*(w(ii,j+1)-w(ii,j-
1))+(r/Re)*(w(ii+1,j)+w(ii-1,j)+w(ii,j+1)+w(ii,j-1)-4*w(ii,j));
        end
    end

    % Transport Equation: Right Row
    for ii=hf+1:nx-1
```

```
        for j=2:hf
            w(ii,j)=w(ii,j)-c*u(ii,j)*(w(ii+1,j)-w(ii-1,j))-c*v(ii,j)*(w(ii,j+1)-w(ii,j-
1))+(r/Re)*(w(ii+1,j)+w(ii-1,j)+w(ii,j+1)+w(ii,j-1)-4*w(ii,j));
        end
    end

    SOR_error=1;
    while (SOR_error>=SORConvergence)
        psiold=psi;
        % SOR Method: Left Column
        for j=ny-1:-1:2
            for ii=2:hf
                psi(ii,j)=(w_opt/4)*(psi(ii+1,j)+psi(ii-1,j)+psi(ii,j+1)+psi(ii,j-
1)+deltax^2*w(ii,j))+(1-w_opt)*psi(ii,j);
            end
        end

        % SOR Method: Right Row
        for ii=hf+1:nx-1
            for j=2:hf
                psi(ii,j)=(w_opt/4)*(psi(ii+1,j)+psi(ii-1,j)+psi(ii,j+1)+psi(ii,j-
1)+deltax^2*w(ii,j))+(1-w_opt)*psi(ii,j);
            end
        end

        psi(nx,2:ny-1)=psi(nx-1,2:ny-1);

        SOR_error=max(max(abs(psi-psiold)));
    end

    % Velocity Update: Left Column
    for ii=2:hf
        for j=2:ny-1
            u(ii,j)=(psi(ii,j+1)-psi(ii,j-1))/(2*deltax);
            v(ii,j)=-(psi(ii+1,j)-psi(ii-1,j))/(2*deltax);
        end
    end

    % Velocity Update: Right Row
    for ii=hf+1:nx-1
        for j=2:hf
            u(ii,j)=(psi(ii,j+1)-psi(ii,j-1))/(2*deltax);
            v(ii,j)=-(psi(ii+1,j)-psi(ii-1,j))/(2*deltax);
        end
    end

    u(nx,2:ny-1)=u(nx-1,2:ny-1);    % Outlet Velocity
    v(nx,2:ny-1)=v(nx-1,2:ny-1);

    % Vorticity BC's: Left Column
    for ii=2:hf
        w(ii,1)=2*(psi(ii,1)-psi(ii,2))/(deltax^2);    % Bottom
        w(ii,ny)=2*(psi(ii,ny)-psi(ii,ny-1))/(deltax^2);    % Inlet
        w(1,ii)=2*(psi(1,ii)-psi(2,ii))/(deltax^2);    % L.H.S.
    end

    % Vorticity BC's: Right Row
    for ii=hf+1:nx-1
        w(ii,1)=2*(psi(ii,1)-psi(ii,2))/(deltax^2);    %Bottom
        w(ii,hf+1)=2*(psi(ii,hf+1)-psi(ii,hf))/(deltax^2);    % Top R.H.S.
        w(1,ii)=2*(psi(1,ii)-psi(2,ii))/(deltax^2);    % L.H.S.
        w(hf+1,ii)=2*(psi(hf+1)-psi(hf,ii))/(deltax^2);    % R.H.S.
    end
    w(nx,:)=w(nx-1,:);  % Outlet

    diff=max(max(abs(w-w_old)));

    k=k+1;
end

figure
[X,Y]=meshgrid(x,y);
vv=[-1e-5,-1e-4,-1e-3,-1e-2,-1e-1,-1,0,1,5e-6,1e-4,1e-3,1e-2,1e-1,-.1:.01:.1,-0.8:0.1:-0.2];
%vv=[0:0.05:0.5];
contour(Y,X,psi,vv);
axis equal
colorbar
colormap('parula')
hold on
t = (1/8:1/4:1)'*2*pi;
x = cos(t)+sqrt(2)/2+hf*deltax;
y = sin(t)+sqrt(2)/2+hf*deltax;
fill(y,x,[220 220 220]/255)
xlim([0 1])
ylim([0,1])
xlabel('X-Direction','Interpreter','Latex')
ylabel('Y-Direction','Interpreter','Latex')
title('Stream Function Contour Plot','Interpreter','Latex','Fontsize',14)
set(gcf,'RendererMode','manual')

figure
ww=[-250:50:-100,-10:1:10,10:5:30,-1e-2,1e-2,1e-4,-100:10:-20];
contour(Y,X,w,ww)
axis equal
colorbar
colormap('parula')
hold on
fill(y,x,[220 220 220]/255)
xlim([0 1])
ylim([0,1])
xlabel('X-Direction','Interpreter','Latex')
ylabel('Y-Direction','Interpreter','Latex')
title('Vorticity Contour Plot','Interpreter','Latex','Fontsize',14)
set(gcf,'RendererMode','manual')

figure
zz=[1e-5,1e-4,1e-3,1e-2,1e-1,1,-3,-1e-5,-1e-4,-1e-3,-1e-2,-1e-1,-1,-5e-3];
contour(Y,X,v,zz)
```

```
axis equal
colorbar
colormap('parula')
hold on
fill(y,x,[220 220 220]/255)
xlim([0 1])
ylim([0,1])
xlabel('X-Direction','Interpreter','Latex')
ylabel('Y-Direction','Interpreter','Latex')
title('Vertical Velocity Contour Plot','Interpreter','Latex','Fontsize',14)
set(gcf,'RendererMode','manual')

figure
kk=[1e-5,1e-4,1e-3,1e-2,1e-1,1,5,10,-1e-5,-1e-4,-1e-3,-1e-2,-1e-1,-1];
contour(Y,X,u,kk)
axis equal
colorbar
colormap('parula')
hold on
fill(y,x,[220 220 220]/255)
xlim([0 1])
ylim([0,1])
xlabel('X-Direction','Interpreter','Latex')
ylabel('Y-Direction','Interpreter','Latex')
title('Horizontal Velocity Contour Plot','Interpreter','Latex','Fontsize',14)
set(gcf,'RendererMode','manual')
```