OF-1 Report :
Computational Simulations of a Lid-driven Cavity

Terry Murray (tnm3843), Nicole Olvera (no4342), Andres Suniaga (aas5778)

02/11/2025

Contents

---

1 Introduction

This report invesitages the canonical case of CFD, a lid-driven cavity. The *OpenFOAM* software is used to compute the velocity and pressure fields at a given Reynolds number, which is the sole non-dimensional property that dictates the flow for the cavity. Thus we consider the nondimensionalized form of the incompressible, steady Navier-Stokes equations. *Paraview* software is used for visualizations of the velocity and pressure fields of the cavity which are included in the report.

Experiments between different mesh refinments and Reynolds numbers are conducted to show changes in velocity fields, nondimensionalized shear stresses, nondimensionalized forces, and the execution or *wallclock* time per step of each refinement.

2 Nondimensional Navier-Stokes equations

The incompressible, constant density, ρ, and viscocity, μ, steady form of the Navier-Stokes equations govern the prescribed two-dimensional fluid flow problem. The continuity and momentum equations are non-dimensionalized according to the following scales:

- Length scale L $= \dfrac{x}{\tilde{x}} = \dfrac{y}{\tilde{y}}$
- Velocity scale U $= \dfrac{u}{\tilde{u}} = \dfrac{v}{\tilde{v}}$
- Pressure Scale $\tilde{P} = \dfrac{P}{\rho U^2}$
- Reynolds number Re $= \dfrac{\rho U L}{\mu} = \dfrac{UL}{\nu}$ where $\nu = \mu/\rho$ is the kinematic viscocity

Continuity

$$\frac{\partial \tilde{u}}{\partial \tilde{x}} + \frac{\partial \tilde{v}}{\partial \tilde{y}} = 0$$

X-Momentum

$$\tilde{u}\frac{\partial \tilde{u}}{\partial \tilde{x}} + \tilde{v}\frac{\partial \tilde{u}}{\partial \tilde{y}} = -\frac{\partial \tilde{p}}{\partial \tilde{x}} + \frac{1}{Re}\left(\frac{\partial^2 \tilde{u}}{\partial \tilde{x}^2} + \frac{\partial^2 \tilde{u}}{\partial \tilde{y}^2}\right)$$

Y-Momentum

$$\tilde{u}\frac{\partial \tilde{v}}{\partial \tilde{x}} + \tilde{v}\frac{\partial \tilde{v}}{\partial \tilde{y}} = -\frac{\partial \tilde{p}}{\partial \tilde{y}} + \frac{1}{Re}\left(\frac{\partial^2 \tilde{v}}{\partial \tilde{x}^2} + \frac{\partial^2 \tilde{v}}{\partial \tilde{y}^2}\right)$$

In the momentum equations, the only non-dimensional number that appears is the Reynolds number. When the Reynolds number becomes very large ($Re \to \infty$), the viscous terms on the RHS of the equations become negligible, implying inviscid flow, reducing the momentum equations to the Euler equations. When the Reynolds number becomes very small ($Re \to 0$), the viscous terms on the RHS of the equations become very large, implying a highly viscous flow, reducing the momentum equations to the Stokes equations. The Stokes and Euler equations are listed below.

Euler Equations:

Continuity as $Re \to \infty$:

$$\frac{\partial \tilde{u}}{\partial \tilde{x}} + \frac{\partial \tilde{v}}{\partial \tilde{y}} = 0$$

X-Momentum as $Re \to \infty$:

$$\tilde{u}\frac{\partial \tilde{u}}{\partial \tilde{x}} + \tilde{v}\frac{\partial \tilde{u}}{\partial \tilde{y}} = -\frac{\partial \tilde{p}}{\partial \tilde{x}}$$

Y-Momentum as $Re \to \infty$:

$$\tilde{u}\frac{\partial \tilde{v}}{\partial \tilde{x}} + \tilde{v}\frac{\partial \tilde{v}}{\partial \tilde{y}} = -\frac{\partial \tilde{p}}{\partial \tilde{y}}$$

Stokes Equations:

Continuity as $Re \to 0$:

$$\frac{\partial \tilde{u}}{\partial \tilde{x}} + \frac{\partial \tilde{v}}{\partial \tilde{y}} = 0$$

X-Momentum as $Re \to 0$:

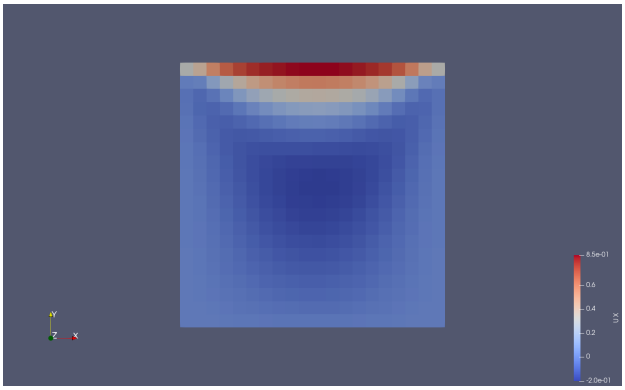
$$0 = -\frac{\partial \tilde{p}}{\partial \tilde{x}} + \frac{1}{Re}\left(\frac{\partial^2 \tilde{u}}{\partial \tilde{x}^2} + \frac{\partial^2 \tilde{u}}{\partial \tilde{y}^2}\right)$$

Y-Momentum as $Re \to 0$:

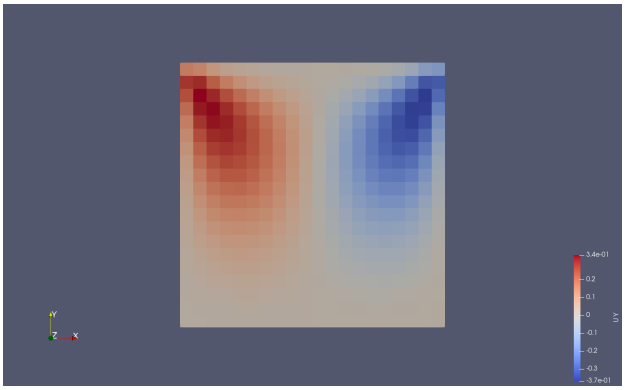
$$0 = -\frac{\partial \tilde{p}}{\partial \tilde{y}} + \frac{1}{Re}\left(\frac{\partial^2 \tilde{v}}{\partial \tilde{x}^2} + \frac{\partial^2 \tilde{v}}{\partial \tilde{y}^2}\right)$$

3 Flow at Re = 10

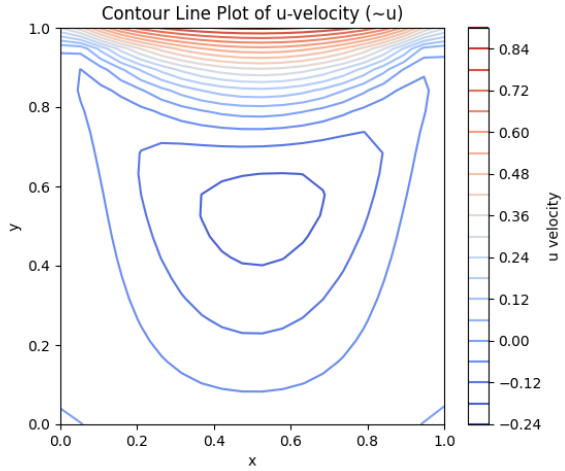
3.1 Plots of Velocity



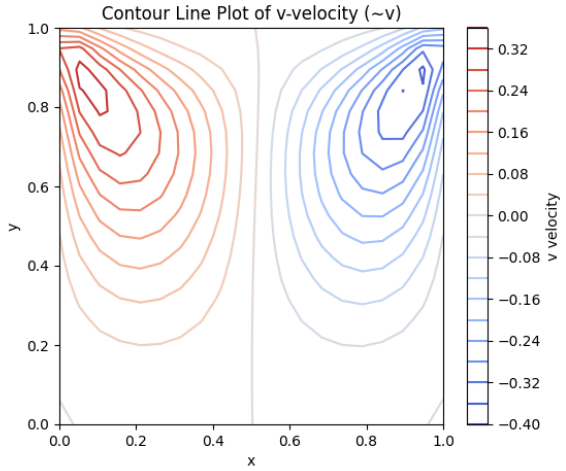
(a) Plot from Paraview of u



(b) Plot from Paraview of v



(a) Contour Plot of u/U



(b) Contour Plot of v/U

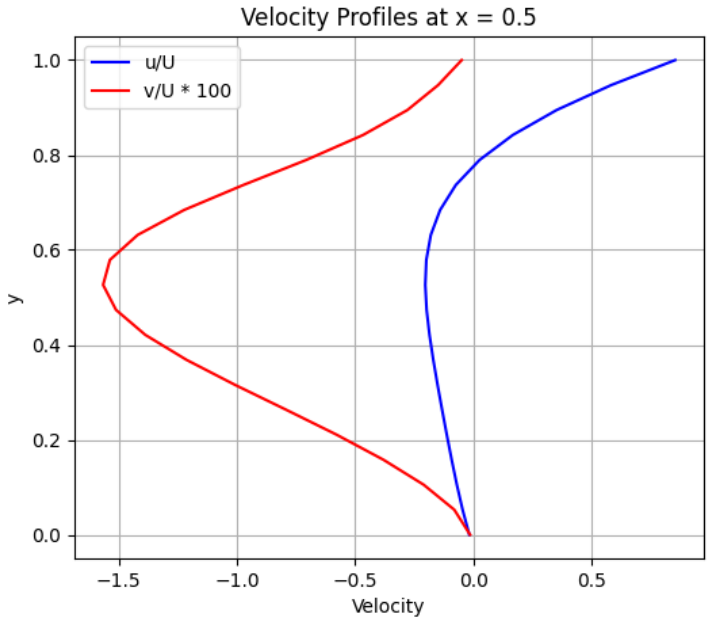
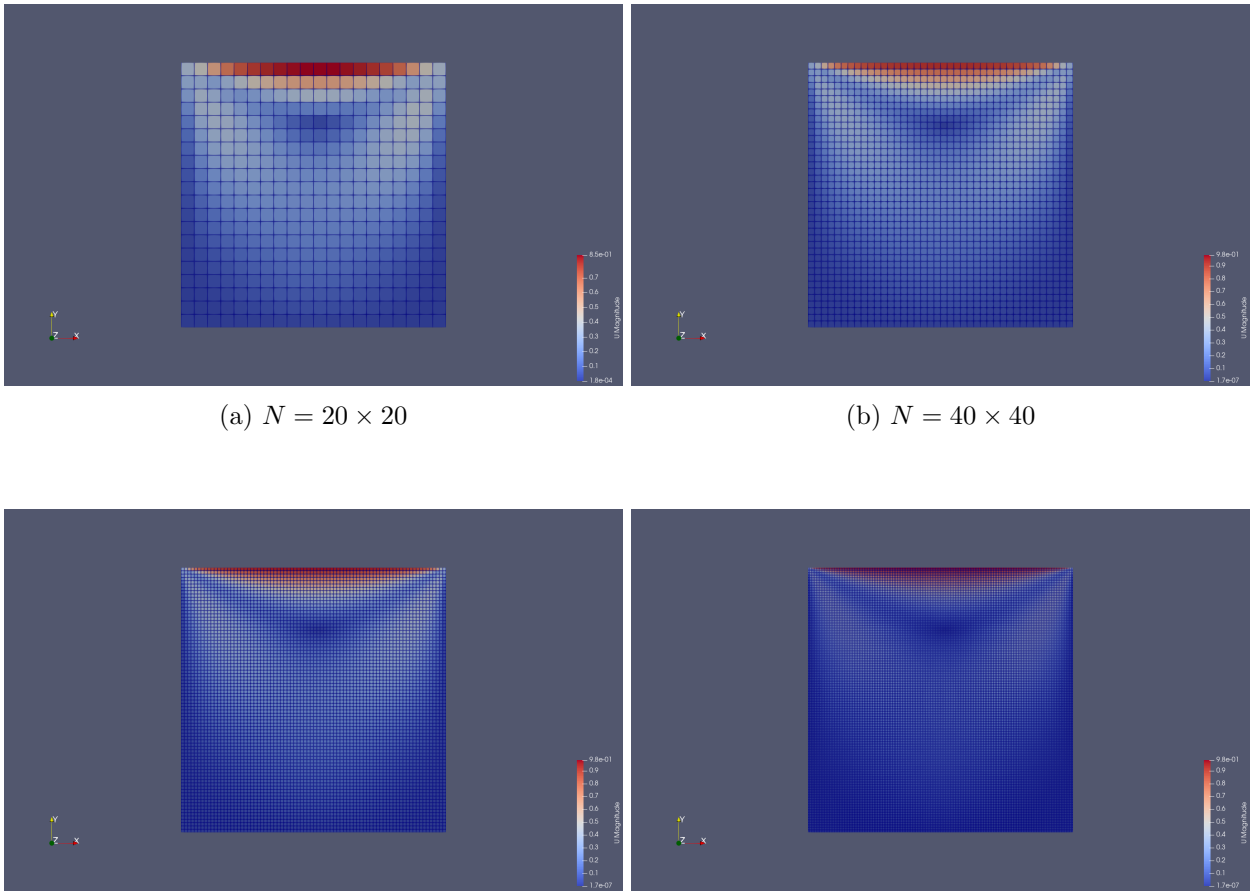


Figure 3: u/U and v/U through the center of the cavity

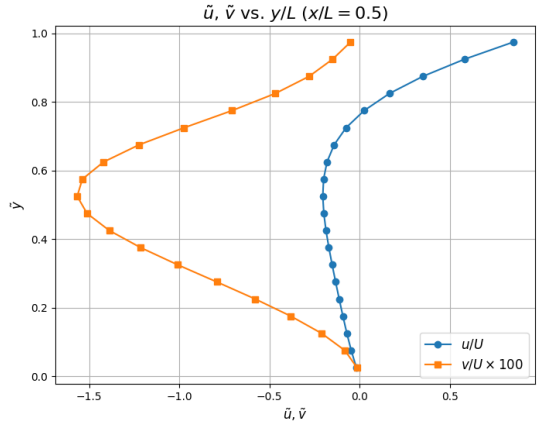
3.2 Solution Refinement



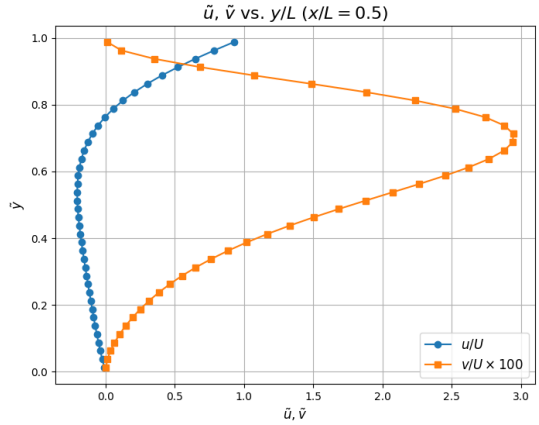
(a) $N = 20 \times 20$ (b) $N = 40 \times 40$

(c) $N = 80 \times 80$ (d) $N = 160 \times 160$

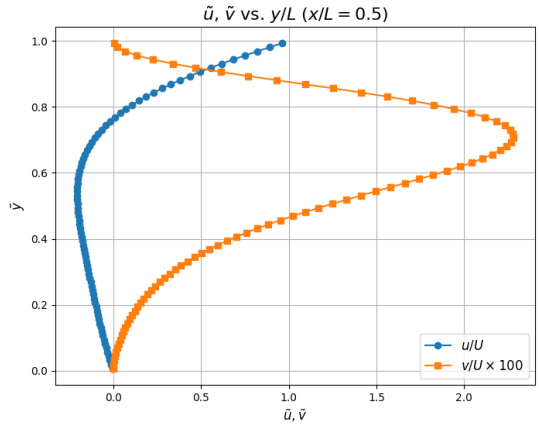
Figure 4: Visualizing effect of gridpoint refinement on U in *Paraview*



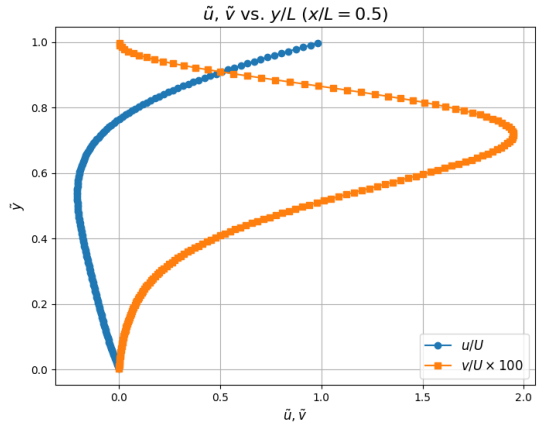
(a) Original



(b) First Refinement



(c) Third Refinement



(d) Fourth Refinement

Figure 5: Effect of increased gridsize and decreased time step size on $\tilde{u}$, $\tilde{v}$ vs. $\tilde{y}$

There is a noticable variation in the scaled up y-component of **U** as we refine the mesh further. The x-component of **U** seems to curve more and reach closer to $u/U = 1$ at $\tilde{y} = 1$ as the mesh is futher refined too.

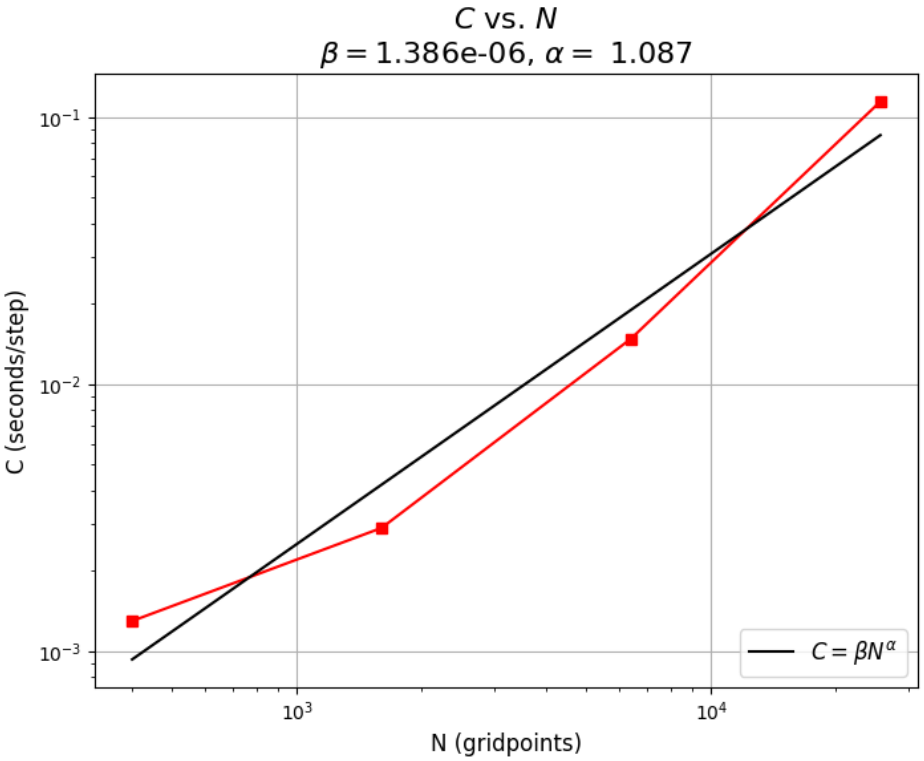


Figure 6: Execution time per step C increases with higher gridpoints N

Q: What can you conclude about the increase in wallclock time as you refine the grid?

A: Higher refinement will take longer per iteration than at lower refinement. It is clear in Fig. 6 with our estimate of $\alpha = 1.087$ in the fit $C = \beta N^{\alpha}$ that with an order of magnitude increase in gridpoints there is an order of magnitude increase in the execution time per step.

4 Force on the Lid

Now we want to investigate the dependance of the force on Reynolds number. We define a few nondimensionalized terms, I love you andres

- $\tilde{F} = \dfrac{F}{\mu U} = \displaystyle\int_0^1 \tilde{\tau}(\tilde{x})\, d\tilde{x}$
- $\tilde{\tau} = \dfrac{\tau L}{\mu U} = \dfrac{\partial \tilde{u}}{\partial \tilde{y}}$

With these terms and our solution for Re = 10 (using the 80x80 grid), we generate a plot $\tilde{\tau}$ vs. $\tilde{x}$:

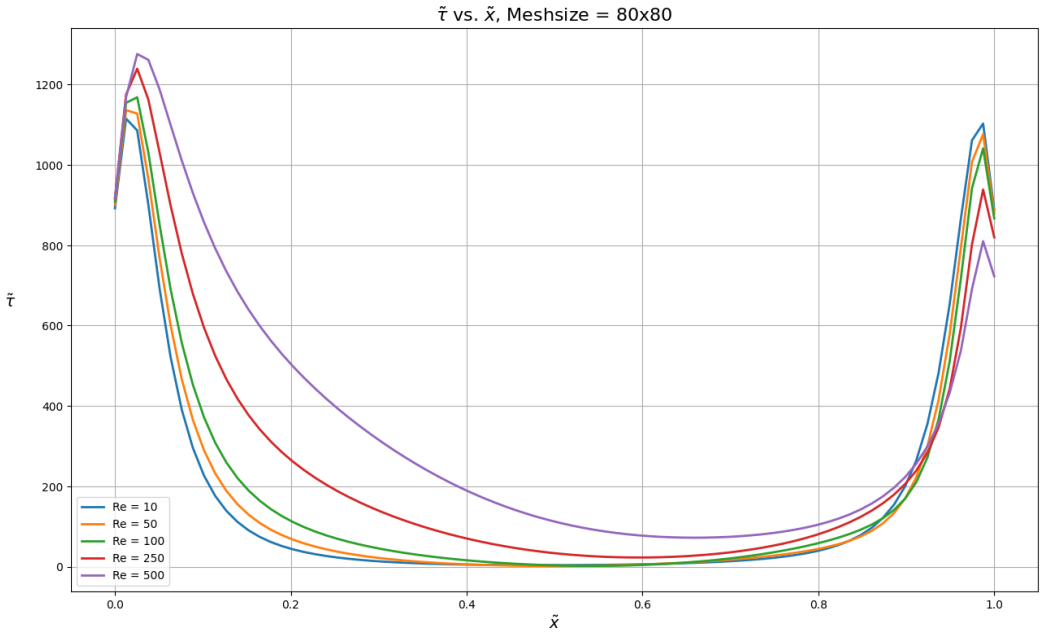


Figure 7: Shear stress has a major decrease over the center of the cavity

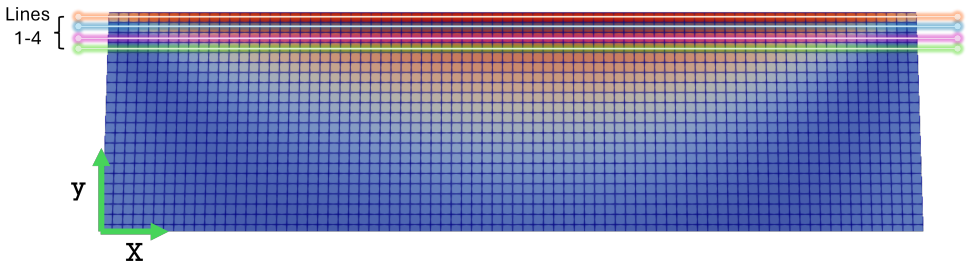


Figure 8: Extracting shear stress through polynomial fitting on top 4 rows 80x80 Mesh

To obtain the nondimensionalized shear stress curves as shown in Fig. 7, we fit a second-order polynomial as a function of y through the first four rows of the 80x80 grid as seen in Fig. 8. That is,

$$u_i(y) = a_i y^2 + b_i y + c_i$$

where i is the gridpoint 1 to 80 along the line. Recall $\tilde{\tau} = \dfrac{\partial \tilde{u}}{\partial \tilde{y}}$. We obtain $\tilde{\tau}$ on the lid by differentiating $u_i(y)$ which turns out to be $u_i'(y) = 2a_i y + b_i$ and evaluate it at $y = 1$ which is the position of the lid of the cavity.

NOTE: At higher Reynolds, the end time of the simulation had to be increased for convergence.

Integrating the $\tilde{\tau}$ curves via trapezoidal method we are able to get a relationship between the nondimensionalized force versus the Reynolds number, $\tilde{F}$ vs. Re:

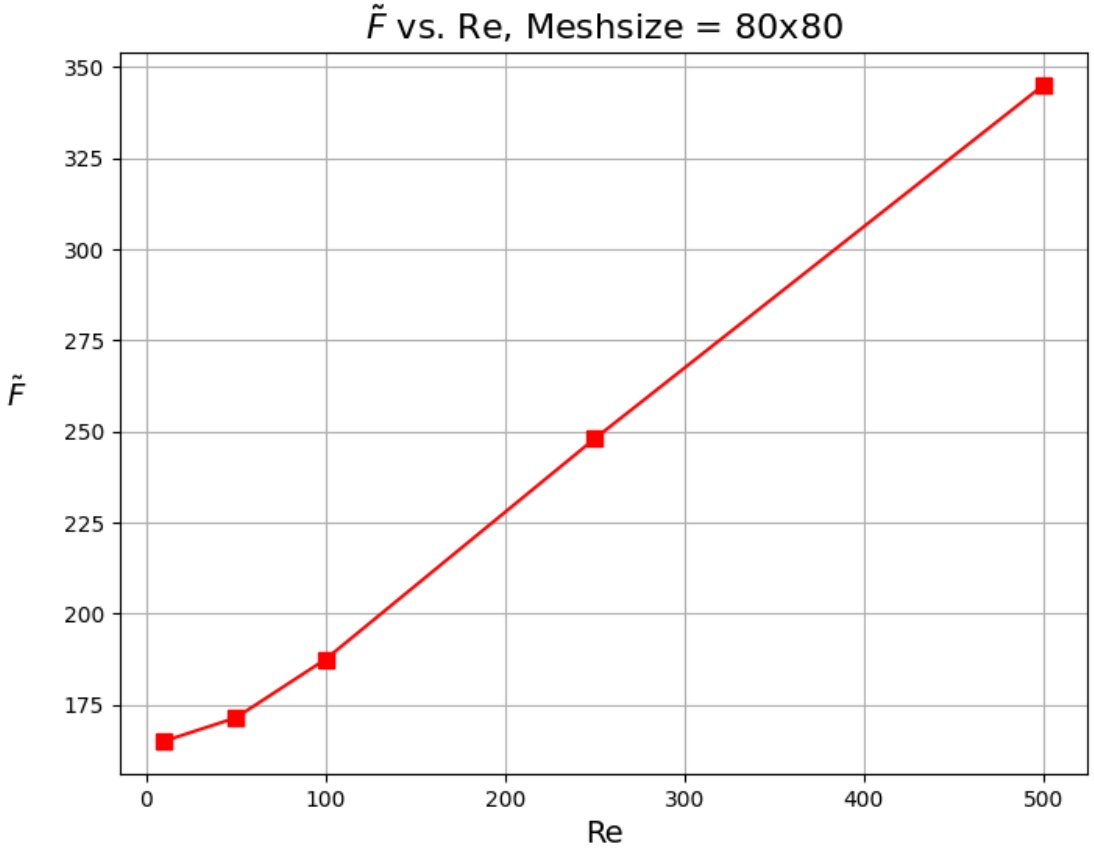


Figure 9: Force increases with higher Reynolds flow

The trapezoidal method of integration,

$$\int_0^1 \tilde{\tau}(\tilde{x})\, d\tilde{x} = \sum_{i=0}^{80} \frac{\tilde{\tau}_i + \tilde{\tau}_{i+1}}{2} \Delta\tilde{x}$$

where $\Delta\tilde{x} = \frac{1}{80}$.

Appendix

A Code

PDF of code starts on next page.

Contour Plot

```
import numpy as np
import matplotlib.pyplot as plt

U = r"OF1\data\cavity.original\0.5\U"

with open(U, 'r') as file:
    lines = file.readlines()

velocity_data = []
reading = False

for line in lines:
    line_stripped = line.strip()
    if reading:
        if line_stripped.startswith(")"):
            break
        if line_stripped.startswith("(") and line_stripped.endswith(")"):
            line_no_paren = line_stripped[1:-1]

            parts = line_no_paren.split()
            if len(parts) >= 2:
                u_val = float(parts[0])
                v_val = float(parts[1])
                velocity_data.append((u_val, v_val))
    if line_stripped.isdigit():
        reading = True

velocity_array = np.array(velocity_data)
print("Number of velocity points:", velocity_array.shape[0])

u_values = velocity_array[:, 0]
v_values = velocity_array[:, 1]

grid_size = (20, 20)
if velocity_array.shape[0] != grid_size[0] * grid_size[1]:
    raise ValueError("The number of data points does not match a 20x20
grid.")

u_grid = u_values.reshape(grid_size)
```

```
v_grid = v_values.reshape(grid_size)

x = np.linspace(0, 1, grid_size[0])
y = np.linspace(0, 1, grid_size[1])
X, Y = np.meshgrid(x, y)

plt.figure(figsize=(6, 5))
contour_u = plt.contour(X, Y, u_grid, levels=20, cmap="coolwarm")
plt.xlabel("x")
plt.ylabel("y")
plt.title("Contour Line Plot of u-velocity (~u)")
plt.colorbar(contour_u, label="u velocity")
plt.show()

# Contour line plot for v-velocity (vertical component)
plt.figure(figsize=(6, 5))
contour_v = plt.contour(X, Y, v_grid, levels=20, cmap="coolwarm")
plt.xlabel("x")
plt.ylabel("y")
plt.title("Contour Line Plot of v-velocity (~v)")
plt.colorbar(contour_v, label="v velocity")
plt.show()

center_index = grid_size[0] // 2
u_profile = u_grid[:, center_index]  # u at x = 0.5 for all y
v_profile = v_grid[:, center_index]  # v at x = 0.5 for all y
plt.figure(figsize=(6, 5))
plt.plot(u_profile, y, label="u/U", color="blue")
plt.plot(v_profile*100, y, label="v/U * 100", color="red")
plt.xlabel("Velocity")
plt.ylabel("y")
plt.title("Velocity Profiles at x = 0.5")
plt.legend()
plt.grid()
plt.show()
```

Refining the Solution

```
# Script for OF1 : Refining the Solution part


#-------------------------------#

import numpy as np
import matplotlib.pyplot as plt

#-------------------------------#



gridsize = [20,40,80,160]

i = 0

L = 0.1 #m
nu = 0.01 #m^2/s
U = 1 #m/s
Re = L*U/nu

for k in range(len(gridsize)):
    Nx = Ny = gridsize[k]

    file_path  = f'OF1\\code\\U\\verticalmidline_U_{Nx}.xy'

    y = []
    u = []
    v = []

    with open(file_path, 'r') as f:
        for l in f:
            k = l.split(" ")
            y.append(k[1])
            u.append(k[3])
            v.append(k[4])

    y = np.array(y,dtype=float)
    u = np.array(u,dtype=float)
    v = np.array(v,dtype=float)

    figk = plt.figure(i,figsize=(8,6))
```

```
    figk.canvas.manager.set_window_title(f"Velocity Component Plot for gridsize
{Nx}x{Ny}")

    c=100

    plt.plot(u,y/L,'-o',label=r"$u/U$")
    plt.plot(v*c,y/L,'-s',label=rf"$v/U \times {c}$")
    plt.title(r"$\tilde{u}$, $\tilde{v}$ vs. $y/L$ ($x/L = 0.5$)", fontsize=16)
    plt.ylabel(r"$\tilde{y}$",fontsize=12)
    plt.xlabel(r"$\tilde{u}, \tilde{v}$",fontsize=12)
    plt.grid()
    plt.legend(loc='lower right',fontsize = 12)

    i+=1

# times in seconds
EndTime = 0.5
ExecutionTimes = [0.13,0.58,5.93,91.66]
dt = [0.005,0.0025,0.00125,0.000625]
iterations = []
C = [] # execution time per step
N = []
for i in range(len(dt)):
    iterations.append(EndTime/dt[i])
    C.append(ExecutionTimes[i]/iterations[i])
    N.append(gridsize[i]**2)


# fit
a, b = np.polyfit(np.log10(N),np.log10(C),deg=1) # linear log fitting
b = 10**b

figNC = plt.figure(i+1,figsize=(8,6))
figNC.canvas.manager.set_window_title("Refinement Measure")

plt.loglog(N,C,'r-s')
plt.plot(N,b*N**a,'k',linewidth = 1.5, label = r"$C = \beta N^{\alpha} $")
plt.title(r"$C$ vs. $N$" '\n' r"$\beta = $"  f"{b:.3e}, " r"$\alpha = $" f"
{round(a,3)}", fontsize=16)
plt.ylabel("C (seconds/step)",fontsize=12)
plt.xlabel("N (gridpoints)",fontsize=12)
plt.grid()
plt.legend(loc='lower right',fontsize = 12)
plt.show()
```

Force on the Lid

```
# Script for OF1 : Refining the Solution part


#-------------------------------#

import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import trapezoid

#-------------------------------#

Re = [10,50,100,250,500]
endtime = [0.5, 1.5, 2.5, 3, 4]
F = []

for r in range(len(Re)):
    Ufile = rf"OF1\data\DeltaRe\cavityRe{Re[r]}\{endtime[r]}\U"
    print(f"Reading {Ufile}")

    Nx = Ny = 80 # gridsize

    L = 0.1 # 0.1m
    U = 1 #m/s

    u = []

    with open(Ufile, 'r') as f:
        lines = f.readlines()

    for i in lines:
        if i.startswith("("):
            k = i.split()[0][1:]
            u.append(k)

    # all x-component velocities from y=0 to y=1
    u = u[1:]
    u = np.array(u,dtype=float)

    Ux = np.zeros([Nx,Ny])


    # Mean x-component velocity in each row starting from row y=1 to y=0
```

```
    for i in range(Nx):
        for j in range(Ny):
            Ux[i,j] = u[(Nx*Ny-80*(i+1))+j]

    x = np.linspace(0,L,Ny)/L
    dx=dy=1/Nx

    tau = np.zeros(Nx)

    n = 4 #rows to fit over
    y=1 #evaluate derivative at this point

    #polynomial fitting
    for i in range(Nx):
        a,b,c = np.polyfit(x[0:n],Ux[0:n,i],2)
        tau[i] = 2*a*(y) + b


    figRe = plt.figure(0,figsize=(8,6))
    figRe.canvas.manager.set_window_title("Nondimensional Stress vs. x")

    plt.plot(x,tau,linewidth=2,label=f"Re = {Re[r]}")
    plt.title("$\\tilde{\\tau}$ vs. $\\tilde{x}$, Meshsize = " f"{Nx}x{Ny}",
fontsize = 16)
    plt.ylabel(r"$\tilde{\tau}$         ",fontsize=14,rotation=0)
    plt.xlabel(r"$\tilde{x}$",fontsize=14)
    plt.grid()
    plt.legend(loc="lower left")

    F.append(trapezoid(tau,x,dx=dx))


figRe = plt.figure(1,figsize=(8,6))
figRe.canvas.manager.set_window_title("Nondimensional Force vs. Re")
plt.plot(Re,F,'r-s',markersize=7)
plt.title("$\\tilde{F}$ vs. Re, Meshsize = " f"{Nx}x{Ny}", fontsize = 16)
plt.ylabel(r"$\tilde{F}$         ",fontsize=14,rotation=0)
plt.xlabel(r"Re",fontsize=14)
plt.grid()

plt.show()
```