A. Introduction

We have data for the velocity field of blood flow inside a human heart, and we want to use this data to determine the pressure field acting on the walls of the ventricle. Blood is an incompressible Newtonian fluid, so we can use the Navier-Stokes equations to find a relationship between velocity and pressure.

$$\frac{\partial \mathbf{v}}{\partial t} + \nabla \mathbf{v} \cdot \mathbf{v} = \nu \nabla^2 \mathbf{v} - \nabla p \tag{1}$$

$$\operatorname{div}(\mathbf{v}) = 0 \tag{2}$$

B. Method

Using integration by parts and the finite element method, we can convert the Navier-Stokes equations into a system of equations that can be used to solve for the pressure field.

First we define the function spaces:

$$S_v = \{\mathbf{v} \mid \mathbf{v} = \bar{\mathbf{v}} \; on \; \partial\Omega_v\}, \; V_v = \{\mathbf{w} \mid \mathbf{w} = \mathbf{0} \; on \; \partial\Omega_v\}$$

$$S_p = \{\mathrm{p} \mid \mathrm{p} = \bar{\mathrm{p}} \; on \; \partial\Omega_p\}, \; V_p = \{\mathrm{q} \mid \mathrm{q} = 0 \; on \; \partial\Omega_p\}$$

Then we define $\mathbf{b}$

$$\mathbf{b} = \nu \nabla^2 \mathbf{v} - \frac{\partial \mathbf{v}}{\partial t} - (\mathbf{v} \cdot \nabla)\mathbf{v}$$

Substituting $\mathbf{b}$ into (1), we get

$$\nabla \mathrm{p} - \mathbf{b} = 0$$

Taking the divergence, we get

$$\operatorname{div}(\nabla \mathrm{p} - \mathbf{b}) = 0 \tag{3}$$

We can now multiply this result by a weight function and use integration by parts to obtain the weak form

$$\int_\Omega \mathrm{q} \operatorname{div}(\nabla \mathrm{p} - \mathbf{b}) \, \mathrm{dV} = 0$$

Divergence is a linear operator

$$\int_\Omega \mathrm{q} \operatorname{div}(\nabla \mathrm{p}) - \operatorname{div}(\mathbf{b}) \, \mathrm{dV} = 0$$

Integration is a linear operator

$$\int_\Omega \mathrm{q} \operatorname{div}(\nabla \mathrm{p}) \, \mathrm{dV} = \int_\Omega \mathrm{q} \operatorname{div}(\mathbf{b}) \, \mathrm{dV}$$

$$\int_\Omega \mathrm{q}\,\nabla\cdot\nabla\mathrm{p}\,\mathrm{dV} = \int_\Omega \mathrm{q}\,\nabla\cdot\mathbf{b}\,\mathrm{dV}$$

Integration by parts: $\nabla\cdot f\,\mathbf{G} = \nabla f\cdot\mathbf{G} + f\,\nabla\cdot\mathbf{G}$

$$\int_{\partial\Omega} \mathrm{q}\nabla\mathrm{p}\cdot\mathbf{n}\,\mathrm{dA} - \int_\Omega \nabla\mathrm{q}\cdot\nabla\mathrm{p}\,\mathrm{dV} = \int_{\partial\Omega} \mathrm{q}\mathbf{b}\cdot\mathbf{n}\,\mathrm{dA} - \int_\Omega \nabla\mathrm{q}\cdot\mathbf{b}\,\mathrm{dV}$$

$$\int_\Omega \nabla\mathrm{q}\cdot\nabla\mathrm{p}\,\mathrm{dV} = \int_\Omega \nabla\mathrm{q}\cdot\mathbf{b}\,\mathrm{dV} + \int_{\partial\Omega} \mathrm{q}(\nabla\mathrm{p}-\mathbf{b})\cdot\mathbf{n}\,\mathrm{dA} \tag{4}$$

Simplifying, we get

$$\int_\Omega \nabla\mathrm{q}\cdot\nabla\mathrm{p}\,\mathrm{dV} = \int_\Omega \nabla\mathrm{q}\cdot\mathbf{b}\,\mathrm{dV} \tag{5}$$

Alternatively, we can notice that

$$\frac{\partial\mathbf{v}}{\partial t} = 0,\ \mathrm{div}(\nu\nabla^2\mathbf{v}) = \nu\nabla^2(\mathrm{div}(\mathbf{v})) = 0$$

We take the divergence of the Navier-Stokes equation

$$\mathrm{div}(\frac{\partial\mathbf{v}}{\partial\mathrm{t}} + \nabla\mathbf{v}\cdot\mathbf{v} = \nu\nabla^2\mathbf{v} - \nabla\mathrm{p})$$

Simplifying, we get

$$\mathrm{div}(\nabla\mathbf{v}\cdot\mathbf{v}) = \mathrm{div}(-\nabla\mathrm{p})$$

We can now multiply this result by a weight function and use integration by parts to obtain the weak form

$$\int_\Omega \mathrm{q}\,\nabla\cdot(\nabla\mathbf{v}\cdot\mathbf{v})\,\mathrm{dV} = -\int_\Omega \mathrm{q}\,\nabla\cdot(\nabla\mathrm{p})\,\mathrm{dV}$$

Integration by parts

$$\int_\Omega \mathrm{q}\,\mathrm{div}(\nabla\mathbf{v}\cdot\mathbf{v})\,\mathrm{dV} = -\int_{\partial\Omega} \mathrm{q}\nabla\mathrm{p}\cdot\mathbf{n}\,\mathrm{dA} + \int_\Omega \nabla\mathrm{q}\cdot\nabla\mathrm{p}\,\mathrm{dV}$$

Rearranging, we get

$$\int_\Omega \nabla\mathrm{q}\cdot\nabla\mathrm{p}\,\mathrm{dV} = \int_\Omega \mathrm{q}\,\mathrm{div}(\nabla\mathbf{v}\cdot\mathbf{v})\,\mathrm{dV} + \int_{\partial\Omega} \mathrm{q}\nabla\mathrm{p}\cdot\mathbf{n}\,\mathrm{dA} \tag{6}$$

We can use the weak form to solve for the pressure field using the finite element method.

1. Generate mesh on Ω_p.
2. Assign material properties and known velocity field.

3. Define function spaces for pressure and weight function.
4. Assign boundary conditions.
5. Define $\mathbf{b} = \nu\nabla^2\mathbf{v} - \frac{\partial \mathbf{v}}{\partial t} - (\mathbf{v}\cdot\nabla)\mathbf{v}$
6. Define $a = \nabla \mathrm{q} \cdot \nabla \mathrm{p}$ and $l = \nabla \mathrm{q} \cdot \mathbf{b}$ from (5).
7. Solve a = l for p with the given boundary conditions using FEniCS.

C. Results

Plotting the data with Paraview, we can look at the pressure field compared to the velocity field in the left ventricle of the heart.

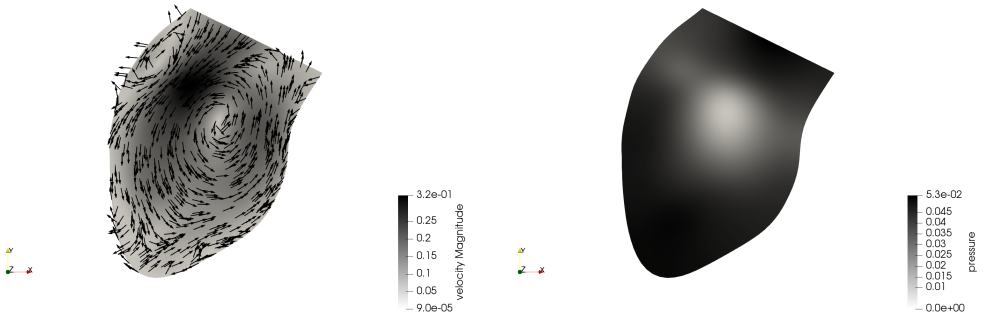

Figure 1: Velocity and pressure field at t = 0.

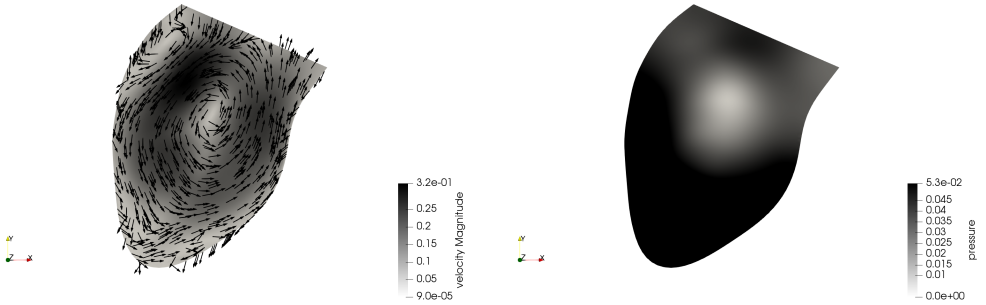

Figure 2: Velocity and pressure field at t = 10.

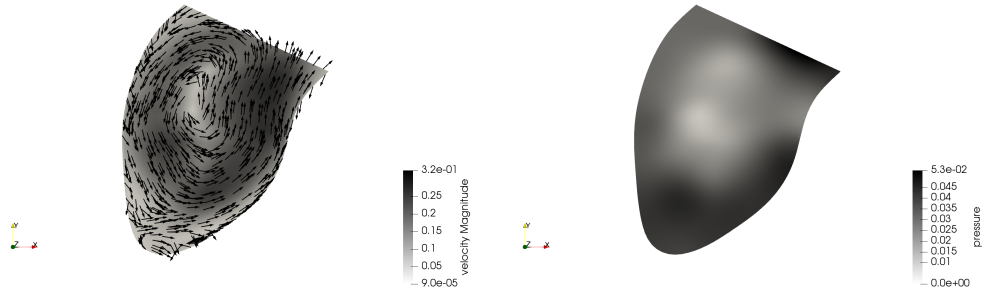

Figure 3: Velocity and pressure field at t = 20.

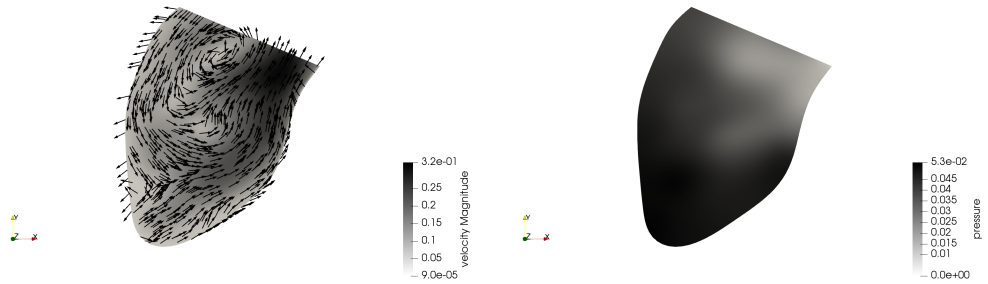

Figure 4: Velocity and pressure field at t = 30.

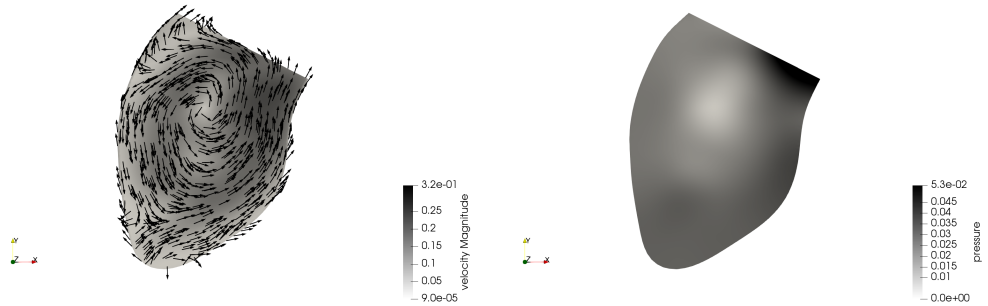

Figure 5: Velocity and pressure field at t = 40.

D. Discussion

From the plots, we see that most of the pressure is along the walls of the heart. There is almost no pressure at the center of the heart. We know that the velocity data spans one cardiac cycle, and we see that the pressure increases, then decreases to 0, and finally increases again. This tells us that the pressure on the walls of the heart changes throughout the cardiac cycle with every heartbeat. Additionally, by comparing the velocity plots to the pressure plots, we see that the periods of high pressure happen during periods of low velocity. This agrees with the relationships from the equations. We can tell that this heart is diseased because it remains slightly open during the contraction.

Python Code 1: Finite Element Analysis

```
# Poisson Pressure Correction Applied to a Measured Velocity Field
# Updated Solution Template File for Dolfin Version 2018.01

# Import libraries
import numpy as np
import dolfin as dlf

# Material properties
nu = 3.77e-6 # m^2/s

# Assumes 57 data files at 1 sec time steps in a sub-folder called Data
first_filenum          = 1
last_filenum           = 57
mesh_filename_root     = './Data/lv_mesh.'
velocity_filename_root = './Data/lv_vel.'
dt                     = 1. / 57.    # Assuming T = 1s

# Load mesh and velocity from the last tstep to allow estimation of v_dot at
# first tstep.
mesh_filename = mesh_filename_root + str(last_filenum) + '.h5'
mesh     = dlf.Mesh()
mesh_in = dlf.HDF5File(dlf.MPI.comm_world, mesh_filename, 'r')
mesh_in.read(mesh, 'mesh', False)
mesh_in.close()

V = dlf.VectorFunctionSpace(mesh, 'CG', 1)

velocity_filename = velocity_filename_root + str(last_filenum) + '.h5'
velocity = dlf.Function(V)
vel_in   = dlf.HDF5File(dlf.MPI.comm_world, velocity_filename, 'r')
vel_in.read(velocity, 'velocity')
vel_in.close()

# Set up output files (result will go in a folder called Results)
fout = dlf.File('Results/pressure.pvd')
```

```
for i in range(first_filenum, last_filenum+1):
    # Load new mesh
    mesh_filename = mesh_filename_root + str(i) + '.h5'
    mesh          = dlf.Mesh()
    mesh_in       = dlf.HDF5File(dlf.MPI.comm_world, mesh_filename, 'r')
    mesh_in.read(mesh, 'mesh', False)
    mesh_in.close()

    # Define new function spaces Q,V, and T for scalara, vectors, and tensors,
    # respectively
    V = dlf.VectorFunctionSpace(mesh, "CG", 1)
    Q = dlf.FunctionSpace(      mesh, "CG", 1)
    T = dlf.TensorFunctionSpace(mesh, "CG", 1)

    # Project old velocity field to new mesh. Need to allow extrapolation since
    # domain is changing.
    velocity.set_allow_extrapolation(True)
    velocity_prev = dlf.project(velocity, V)

    # Load new velocity field
    velocity_filename = velocity_filename_root + str(i) + '.h5'
    velocity          = dlf.Function(V)
    vel_in            = dlf.HDF5File(dlf.MPI.comm_world, velocity_filename, 'r')
    vel_in.read(velocity, 'velocity')
    vel_in.close()

    # Project gradient of velocity to CG1 space so that div(grad_velocity) can
    # be used to approximate laplacian of velocity
    grad_velocity = dlf.project(dlf.grad(velocity), T)

    # Define test, trial, and solution functions called p_test, p_trial, and p,
    # respectively
    p_test = dlf.TestFunction(Q)
    p_trial = dlf.TrialFunction(Q)
    p = dlf.Function(Q)

    # Define Dirichlet BC to pin the node with the largest x coordinate
    coords      = mesh.coordinates()
    x_dirichlet = -1e8
    y_dirichlet = -1e8
    for j in range(coords.shape[0]):
        if coords[j, 0] > x_dirichlet:
            x_dirichlet = coords[j, 0]
            y_dirichlet = coords[j, 1]

    def dirichlet_node(x, on_boundary):
        return dlf.near(x[0], x_dirichlet) and dlf.near(x[1], y_dirichlet)

    bc = dlf.DirichletBC(Q, dlf.Constant(0.), dirichlet_node, method='pointwise')
```

```
# Define b vector
b = (nu*dlf.div(grad_velocity))-((velocity-velocity_prev)/dt) \ -
    (dlf.dot(dlf.grad(velocity), velocity))

# Set up a( , ) and l( ) forms and solve, placing solution in p
a = dlf.dot(dlf.grad(p_test), dlf.grad(p_trial)) * dlf.dx
L = dlf.dot(dlf.grad(p_test), b) * dlf.dx
dlf.solve(a == L, p, bcs = bc)

# Needed so that fieldnames are consistent in vtu files
p.rename('pressure', 'pressure')

# Shift pressure such that all values are >= 0. This is done because PPE
# only gives relative pressures anyway.
p.vector().set_local(p.vector().get_local() - p.vector().get_local().min())

# Save
fout << p
```

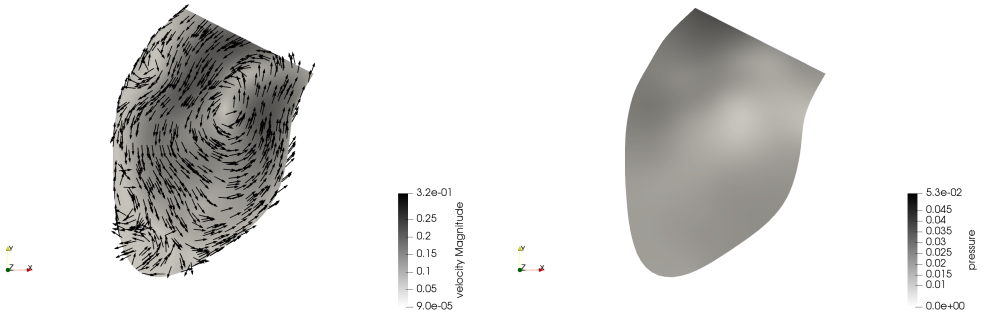

Figure 6: Velocity and pressure field at t = 50.

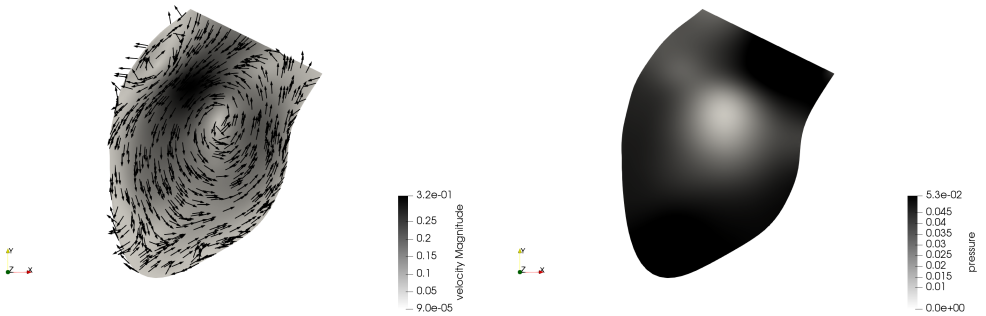

Figure 7: Velocity and pressure field at t = 56.