FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG
FACULTY OF SCIENCES

Opimization of Submarine Body

Final Presentation

Aju Abraham, Akhila B Cherian, Jishnu Jayaraj, Prabhu Vijayan
Department of Mathematics, Friedrich-Alexander University of Erlangen-Nuremberg
August 7, 2019

Introduction

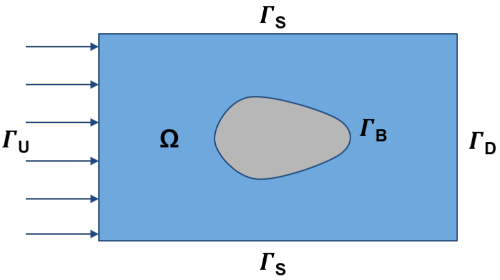
- Obtain an optimized submarine body shape with minimal drag force for laminar flow for
  - Radial symmetry
  - Constant Volume
- Obtain a family of shapes representing minimum drag force

Mathematical Modelling

- Objective Function :

$$\min_{\boldsymbol{X}} \qquad J = f(D)$$
$$\text{subject to} \quad V(\boldsymbol{X}) = V_0$$

- $\boldsymbol{X}$: Boundary coordinates
- D : Drag Force
- V_0 : Initial Volume



2D Navier-Stokes Equation

Mass Conservation

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y} = 0 \tag{1}$$

Momentum conservation

$$\frac{\partial(\rho u)}{\partial t} + \frac{\partial(\rho uu)}{\partial x} + \frac{\partial(\rho uv)}{\partial y} = -\frac{\partial p}{\partial x} + \mu\left[\frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{xy}}{\partial y}\right] \tag{2}$$

$$\frac{\partial(\rho v)}{\partial t} + \frac{\partial(\rho uv)}{\partial x} + \frac{\partial(\rho vv)}{\partial y} = -\frac{\partial p}{\partial y} + \mu\left[\frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \tau_{yy}}{\partial y}\right] \tag{3}$$

Navier-Stokes Equation (2D Incompressible & Steady-state flow)

Mass Conservation

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0 \tag{4}$$

Momentum conservation

$$\frac{\partial(uu)}{\partial x}+\frac{\partial(uv)}{\partial y}=-\frac{1}{\rho}\frac{\partial p}{\partial x}+\nu\left[\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right] \tag{5}$$

$$\frac{\partial(uv)}{\partial x}+\frac{\partial(vv)}{\partial y}=-\frac{1}{\rho}\frac{\partial p}{\partial y}+\nu\left[\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right] \tag{6}$$

Navier-Stokes Equation with Boundary Condtions

Navier-Stokes Equation

$$\mathbf{u} \cdot \nabla \mathbf{u} + \nabla p - \nabla \cdot \{\nu[\nabla \mathbf{u} + (\nabla \mathbf{u})^T]\} = \mathbf{0}$$

$$\nabla \cdot \mathbf{u} = \mathbf{0}$$

Boundary Conditions

$\mathbf{u} = (U, 0)$ on Γ_U X $(0, T)$

$\mathbf{t} = \{-p\mathbf{I} + \nu[\nabla \mathbf{u} + (\nabla \mathbf{u})^T]\} \cdot \mathbf{n} = \mathbf{0}$ on Γ_D X $(0, T)$

$t_1 = 0, u_2 = 0$ on Γ_S X $(0, T)$

$\mathbf{u} = \mathbf{0}$ on Γ_B X $(0, T)$

$\mathbf{u}(\mathbf{x}, 0) = \mathbf{u}_0$ in Ω_0

Drag Force Calculation

- Horizontal components of pressure and viscous force on the submarine boundary
- Pressure force,

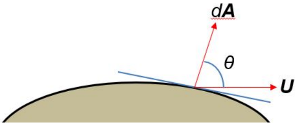
$$\mathbf{F}_p = -p\mathbf{dA} \tag{7}$$

- Viscous force,

$$\mathbf{F}_\tau = \tau\mathbf{dA} \tag{8}$$

- Drag Force

$$D = \int_{\Gamma_B} dF_x = \int_{\Gamma_B} (-pdA)\cos\theta + \int_{\Gamma_B} (\tau dA)\sin\theta \tag{9}$$



Objective Function

$$
\begin{aligned}
J = &\ \tfrac{1}{2}\int_0^T (q_1 D^2 + q_2 L^2)dt \\
- &\int_0^T \int_\Omega \mathbf{u}^* \left\{ \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} + \nabla p - \nabla\cdot\{\nu[\nabla\mathbf{u} + (\nabla\mathbf{u})^T]\} \right\} d\Omega dt \\
+ &\int_0^T \int_\Omega p^* \nabla\cdot\mathbf{u} d\Omega dt \quad + \quad \lambda\left[\textstyle\sum_{e=1}^m \{v_e(X_i)\}^{(l)} - V_0\right]
\end{aligned}
$$

Stationary Condition

$$\partial J \stackrel{!}{=} 0$$

Optimality Condition Equations

$$-\frac{\partial \mathbf{u}^*}{\partial t} + (\nabla \mathbf{u})^T \mathbf{u}^* - \mathbf{u} \cdot \nabla \mathbf{u}^* + \nabla p^* - \nabla \cdot \{\nu[\nabla \mathbf{u}^* + (\nabla \mathbf{u}^*)^T]\} = \mathbf{0}$$

$$\nabla \cdot \mathbf{u}^* = 0$$

Boundary Conditions

$\mathbf{u}^* = \mathbf{0}$ on Γ_U X $(0, T)$

$\mathbf{s} = \{\mathbf{u}\mathbf{u}^* - p^*\mathbf{I} + \nu[\nabla \mathbf{u}^* + (\nabla \mathbf{u}^*)^T]\} \cdot \mathbf{n} = \mathbf{0}$ on Γ_D X $(0, T)$

$s_1 = 0, u_2^* = 0$ on Γ_S X $(0, T)$

$\mathbf{u}^* = (q_1 D, 0)$ on Γ_B X $(0, T)$

$\mathbf{u}^*(x, T) = \mathbf{0}$ in Ω

Weighted Gradient Method

$$K^i = J^i + \frac{1}{2}(X^{(i+1)} - X^i)^T W (X^{(i+1)} - X^i)$$

$$\partial K^i = grad(J)^i \partial X^i + W(X^{(i+1)} - X^i)\partial X^i = 0$$

$$X^{(i+1)} = X^i - \frac{1}{W} grad(J)^i$$

where,

$$grad(J)^i = -\nabla \mathbf{u}^T \cdot \mathbf{s}$$

Numerical Scheme

Finite Volume Method

$$\frac{\partial u_i u_j}{\partial x_j} + \frac{1}{\rho}\frac{\partial p}{\partial x_i} - \nu \frac{\partial^2 u_i}{\partial x_j \partial x_j} = 0$$

$$\int_V \frac{\partial u_i u_j}{\partial x_j} dV - \int_V \nu \frac{\partial^2 u_i}{\partial x_j x_j} dV + \int_V \frac{1}{\rho}\frac{\partial p}{\partial x_i} dV = 0$$

Gauss Divergence Theorem

$$\int_V \frac{\partial a_j}{\partial x_j} dV = \int_S a_j n_j dS$$

Discretisation of Navier-Stokes Equation

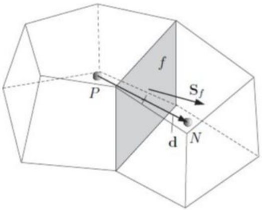
- **Convective Term**

$$\int_V \frac{\partial u_i u_j}{\partial x_j} dV = \int_S u_i u_j n_j dS \quad \approx \sum_f \mathbf{S}_f \cdot (u_i)_f (u_j)_f = \sum_f F(u_j)_f$$

where $F = \mathbf{S}_f \cdot (u_i)_f$

velocity evaluated by Upwind Differencing scheme,

$$u_f = \begin{cases} u_P, & \text{for } F \geq 0 \\ u_N, & \text{for } F < 0 \end{cases} \qquad (10)$$



- **Diffusive Term**

$$\int_V \nu \frac{\partial^2 u_i}{\partial x_j x_j} dV = \int_V \frac{\partial}{\partial x_j}\left(\nu \frac{\partial u_i}{\partial x_j}\right) dV = \int_S \nu \frac{\partial u_i}{\partial x_j} n_j dS \approx \sum_f \nu \mathbf{S}_f \cdot \left(\frac{\partial u_i}{\partial x_j}\right)_f$$

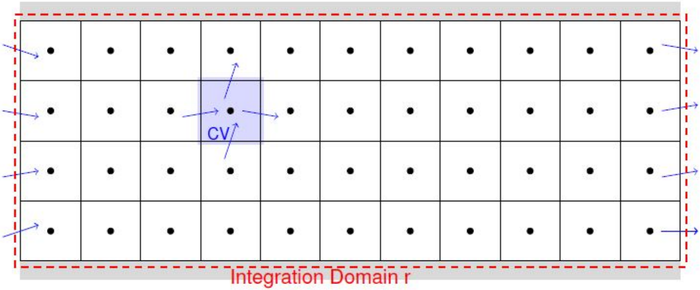
Implicit face gradient discretisation when the length vector **d** is orthogonal to the face plane, i.e. parallel to $\mathbf{S}_f$,

$$\mathbf{S}_f \cdot \left(\frac{\partial u_i}{\partial x_j}\right)_f = |S_f| \frac{u_N - u_P}{|d|}$$

d - length vector between cells P and N.

- **Pressure Term** is approximated by Gauss integration,

$$\int_V \frac{1}{\rho}\frac{\partial p}{\partial x_i}dV = \int_S \frac{1}{\rho} p n_j dS \approx \sum_f \frac{1}{\rho}\mathbf{S}_f p_f$$



Discretisation of Adjoint Equation

Finite Volume Method

$$-\frac{\partial u_i^*}{\partial t}+\frac{\partial}{\partial x_j}u_j^* u_i-\frac{\partial}{\partial x_j}u_j u_i^*+\frac{\partial p^*}{\partial x_i}-\nu\frac{\partial^2 u_i^*}{\partial x_j \partial x_j}=0$$

Temporal discretisation

Denoting all the spatial terms as $\mathscr{A}\phi$ where $\mathscr{A}$ is any spatial operator, then

$$\int_t^{t+\delta t}\left[-\frac{\partial}{\partial t}\int_V \mathbf{u}^* dV+\int_V \mathscr{A}\phi\, dV\right]dt=0$$

Using Euler implicit scheme

$$-\int_t^{t+\delta t}\left[\frac{\partial}{\partial t}\int_V \mathbf{u}^* dV\right]dt = -\int_t^{t+\delta t}\frac{(\mathbf{u}_P^* V)^{(n+1)} - (\mathbf{u}_P^* V)^n}{\delta t}dt$$

$$= -\frac{(\mathbf{u}_P^* V)^{(n+1)} - (\mathbf{u}_P^* V)^n}{\delta t}\delta t$$

$$\int_t^{t+\delta t}\left[\int_V \mathscr{A}\phi\, dV\right]dt = \int_t^{t+\delta t}\mathscr{A}^*\phi\, dt$$

where $\mathscr{A}^*$ - spatial discretisation of $\mathscr{A}$. Using implicit discretisation of spatial terms

$$\int_t^{t+\delta t}\mathscr{A}^*\phi\, dt = \mathscr{A}^*\phi^{n+1}\delta t$$

Boundary points in Discretisation

Dirichlet or Fixed value BC prescribes dependent variable on the boundary. When one face is boundary,

- ϕ_b fixed value is used. (eg. convective term)
- face gradient is calculated using ϕ_b and ϕ_P (eg. Laplacian term)

$$\mathbf{S}_f \cdot (\nabla\phi)_f = |S_f| \frac{\phi_b - \phi_P}{|\mathbf{d}|}$$

Boundary points in Discretisation

Neumann or Fixed gradient BC prescribes gradient of variable normal to the boundary.

$$g_b = \left(\frac{\mathbf{S}}{|\mathbf{S}|} \cdot \nabla \phi \right)_f$$

When one face is boundary,

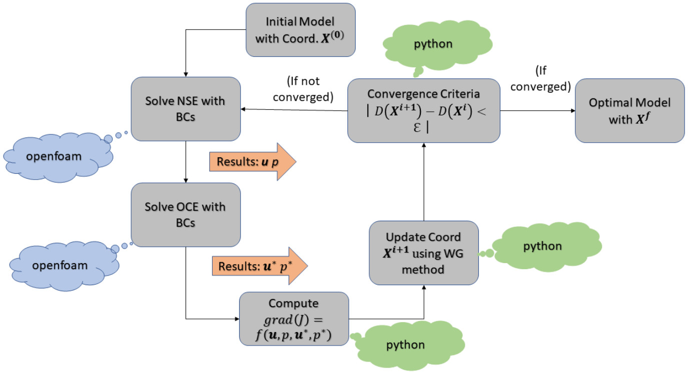
- ϕ_f is interpolated using ϕ_P and g_b

$$\phi_f = \phi_P + \mathbf{d} \cdot (\nabla \phi)_f = \phi_P + |\mathbf{d}| g_b$$

- for face gradient,

$$\mathbf{S}_f \cdot (\nabla \phi)_f = |\mathbf{S}| g_b$$

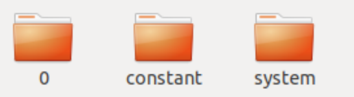
Implementation



OpenFoam

- Stands for 'Open-source Field Operation And Manipulation'
- An open source software for solving flow simulation
- Functionalities are defined as C++ libraries
- Advantages:
    - Easy customization of existing solver as per the requirement
    - Inputs are defined in text files known as dictionaries, enabling easy manipulation of simulations

Simulation Case structure



- *0* Folder: Initial and Boundary conditions
- *constant* Folder: Flow properties such as Reynold's No.
- *system* Folder: Geometry definition, Meshing set up, Simulation settings such as timestep, numerical schemes etc

Geometry and Meshing

```
vertices
(
        (-3 0 0)        //Vertex 0
        (-1 0 0)        //Vertex 1
);

blocks
(

    hex (0 1 2 3 19 20 21 22) (20 10 1) simpleGrading (1 1 1)   //0

);

edges
(

    spline 1 2 ((-1 0.2 0.1))

);
```

- Entities of the geometry, namely, vertices, edges and blocks are defined in a text file
- Simulation domain constructed using hexagonal blocks (using vertex points and edges)
- For each block, number of meshes can be defined in x,y and z directions

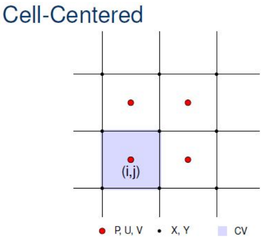
Running Simulation

- Simulation settings: Time step control, data write control, Discretization schemes
- Boundary Conditions and Initial Conditions
- Creating mesh with the utility *blockMesh*
- Checks mesh for errors with utility *checkMesh*
- Run solver *simpleFoam/ pisomosiFoam*

Solver Implementation

Grid Arrangement

- Collocated Grid - All variables $\boldsymbol{u}$, p, $\boldsymbol{u^*}$ and $p*$ are stored at cell centre



Predictor Corrector Approach

An approach to couple the momentum conservation and mass conservation equations

- Step 1: Prediction- Obtain predicted velocity neglecting pressure gradient
- Step 2: Poisson's equation- Obtained pressure using predicted velocity
- Step 3: Correction- Obtained corrected velocity from predicted velocity and predicted pressure
- Repeat step 1 to 3 until convergence

Solver Implementation - *simpleFoam*

- OpenFoam's existing solver *simpleFoam* used for solving NSE
- Added a functionality to calculate the Drag force around the submarine
- Code for NSE Equations (4,5,6) in *simpleFoam* solver:

```
  fvVectorMatrix UEqn
(
    fvm::div(phi,U)
    + turbulence->divDevReff(U)

);

solve(UEqn == -fvc::grad(p));
```

Solver Implementation - *pisomosiFoam*

- A new solver named *pisomosiFoam* is created
- Code for Optimality Condition Equations in *pisomosiFoam*:

```
fvVectorMatrix UAdjEqn
(
    -fvm::ddt(UAdj)  + (UAdj & fvc::grad(U)) -fvm::div(phi,UAdj)
    - turbulence->divDevReff(UAdj)

);

solve(UAdjEqn == -fvc::grad(pAdj));
```

- The parameters ***u***, *p*, ***u****, $p*$ on the submarine boundary printed out in text format

Additional Features

Preventing Failures

- Varying the Weight factor W
- Varying tolerance of $\boldsymbol{X}$ update

Faster Convergence

- Considering only the half plane due to symmetry
- Varying the Weight factor W
- Varying maximum number of iterations
- Varying tolerance for Drag Force convergence

Simulation Results

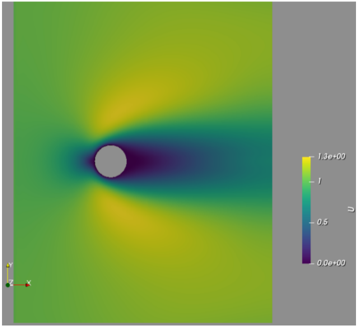
Different Scenarios

- Constant volume, varying Re (Scenario 1, 2, 3)
- Varying volume, constant Re (Scenario 1, 4)
- Constant volume, Re and varying shape (Scenario 1, 5)

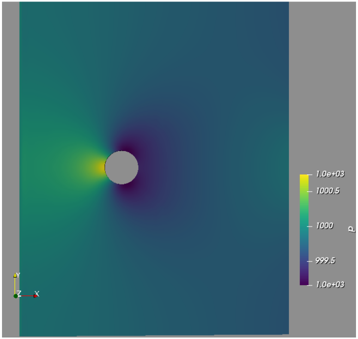
Scenarios	Re	Initial shape	Volume
1	100	circle - 0.5 radius	V_1
2	500	circle - 0.5 radius	V_1
3	1000	circle - 0.5 radius	V_1
4	100	circle - 0.75 radius	$V_1 + x$
5	100	hexagon	V1

Scenario 1

Re 100; Initial shape - 0.5 Radius Circle; Volume V_1



(a)

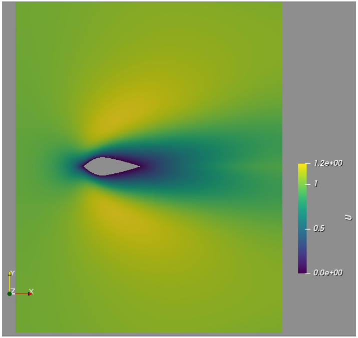


(b)

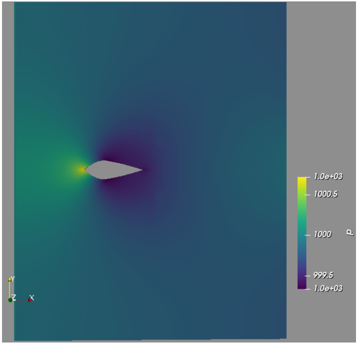
Figure: (a) Velocity contour initial (b) Pressure contour initial

Scenario 1

Re 100; Initial shape - 0.5 Radius circle; Volume V_1



(a)

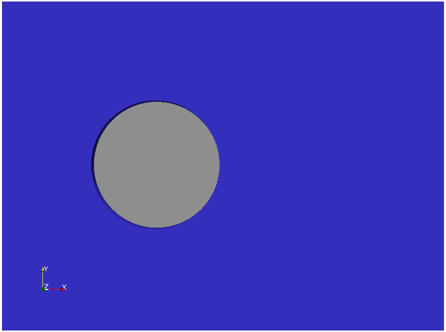


(b)

Figure: (a) Velocity contour final (b) Pressure contour final

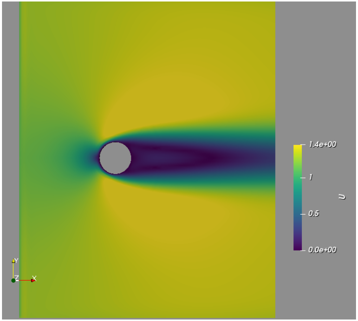
Scenario 1

Re 100; Initial shape - 0.5 Radius circle; Volume V_1

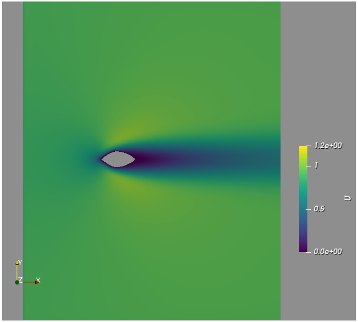


Scenario 2

Re 500; Initial shape - 0.5 Radius Circle; Volume V_1



(a)

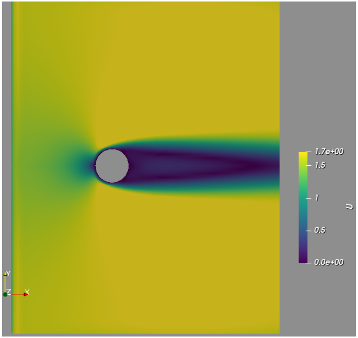


(b)

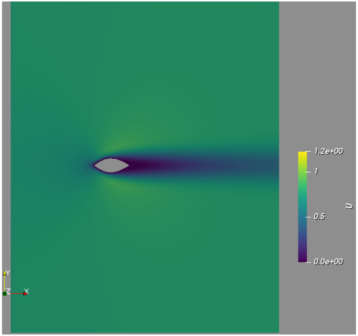
Figure: (a) Velocity contour initial (b) Velocity contour final

Scenario 3

Re 1000; Initial shape - 0.5 Radius Circle; Volume V_1



(a)

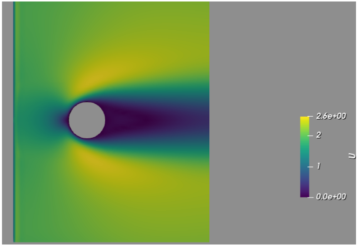


(b)

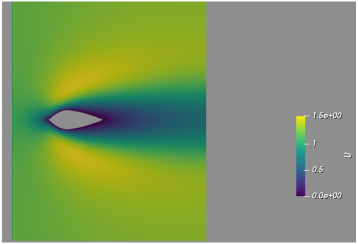
Figure: (a) Velocity contour initial (b) Velocity contour final

Scenario 4

Re 100; Initial shape - 0.75 Radius Circle; Volume V_1+x



(a)

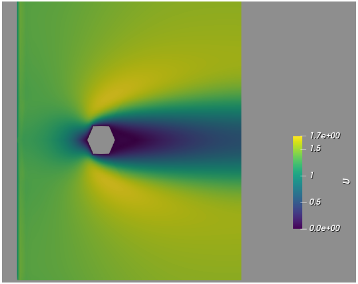


(b)

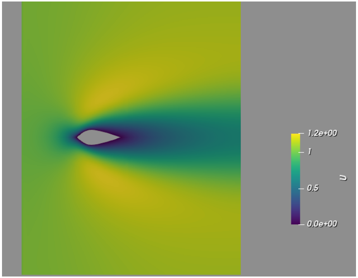
Figure: (a) Velocity contour initial (b) Velocity contour final

Scenario 5

Re 100; Initial shape - Hexagon; Volume V_1



(a)

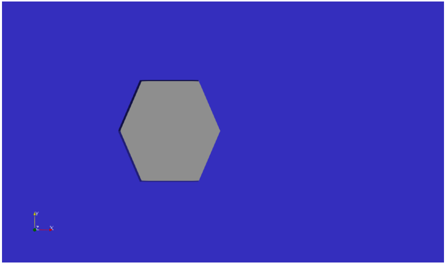


(b)

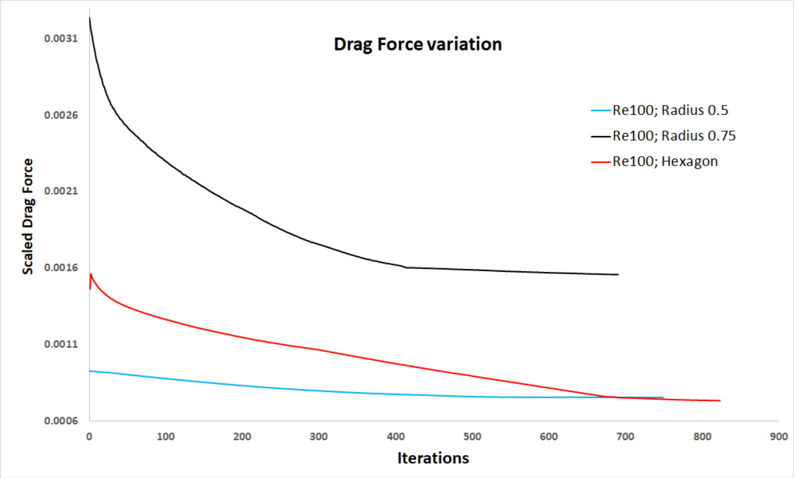
Figure: (a) Velocity contour initial (b) Velocity contour final

Scenario 5

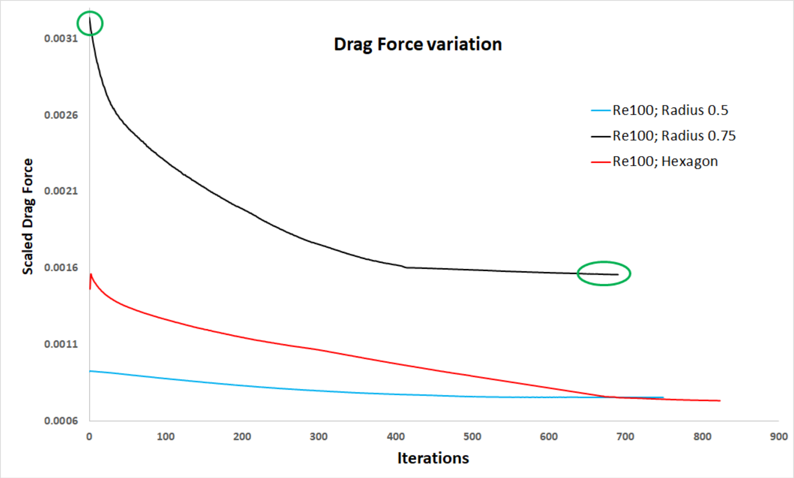
Re 100; Initial shape - Hexagon; Volume V_1



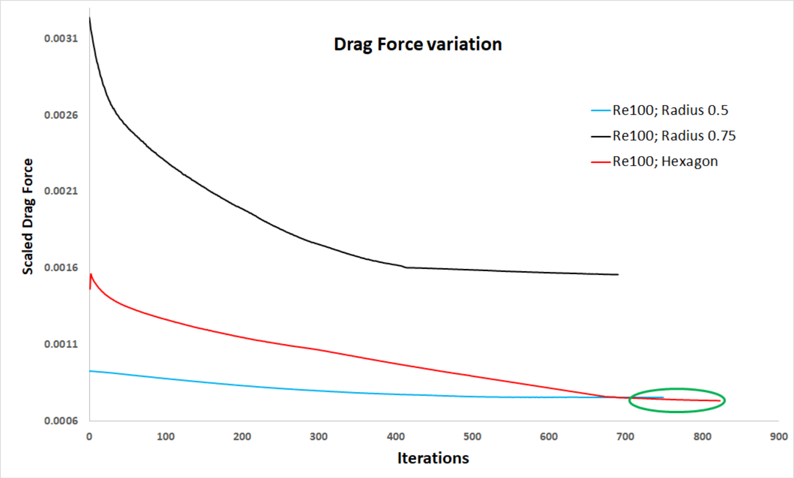
Shape and Volume comparison



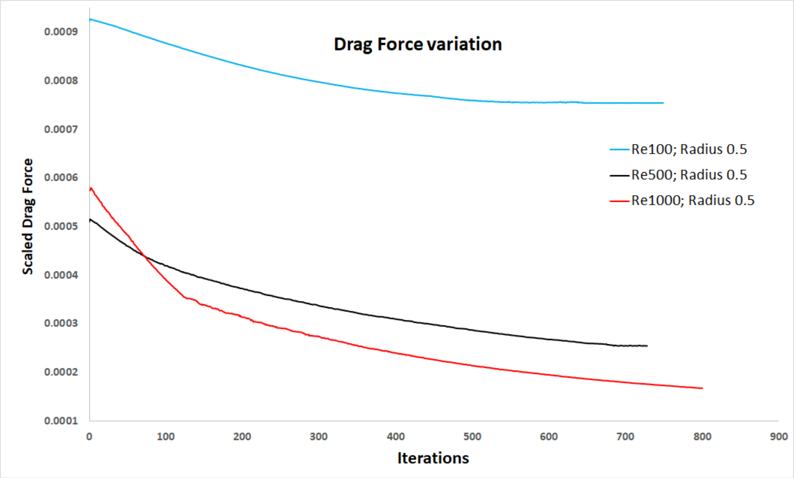
Shape and Volume comparison



Shape and Volume comparison



Re comparison



Project Tracking Tools

github

Software development version control using Git

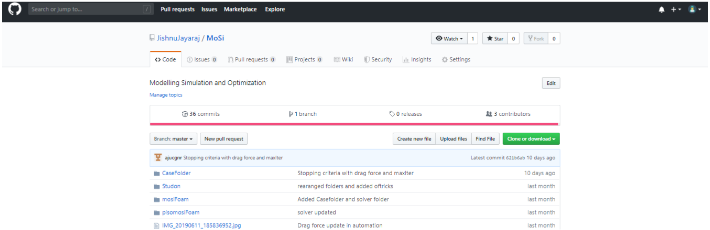


Figure: Version control

Mindmeister

To visualize, share and present thoughts

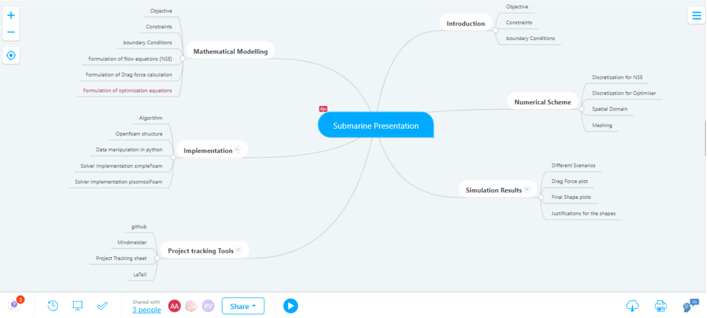
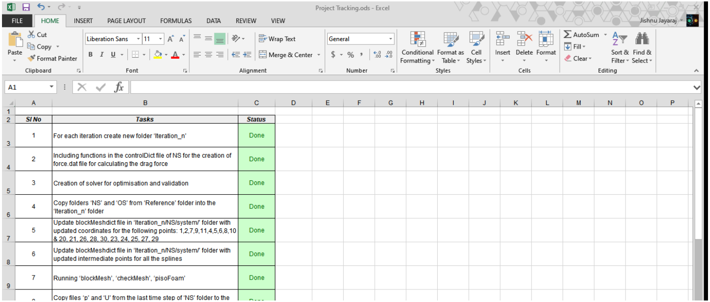


Figure: Mind meister

Project Tracking sheet

Assign work to individual members



Sl No	Tasks	Status
1	For each iteration create new folder 'Iteration_n'	Done
2	Including functions in the controlDict file of NS for the creation of force.dat file for calculating the drag force	Done
3	Creation of solver for optimisation and validation	Done
4	Copy folders 'NS' and 'OS' from 'Reference' folder into the 'Iteration_n' folder	Done
5	Update blockMeshdict file in 'Iteration_n/NS/system/' folder with updated coordinates for the following points: 1,2,7,9,11,4,5,6,8,10 & 20, 21, 26, 28, 30, 23, 24, 25, 27, 29	Done
6	Update blockMeshdict file in 'Iteration_n/NS/system/' folder with updated intermediate points for all the splines	Done
7	Running 'blockMesh', 'checkMesh', 'pisoFoam'	Done
8	Copy files 'p' and 'U' from the last time step of 'NS' folder to the	Done

Figure: Excel sheet

LaTeX

Online LaTeX editor, with real-time collaboration, version control

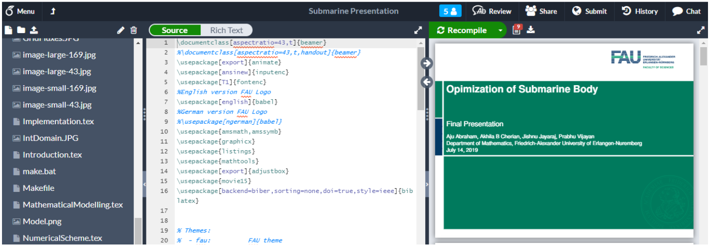


Figure: Overleaf

Thanks for listening.

Any questions?

References

References I

1. Shape optimization of body located in incompressible Navier-Stokes flow based on optimal control theory by H. Okumura, M.Kawahara

2. Shape optimization of an oscillating body in fluid flow by adjoint equation and ALE Finite Element Methods by Hiroki Yoshida and Mutsuto Kawahara