Friedrich-Alexander Universität Erlangen-Nürnberg

Modelling, Simulation and Optimization
(Practical Course - Summer Term 2019)

Optimization of Submarine Body

Aju Abraham
Akhila Biju Cherian
Jishnu Jayaraj
Prabhu Vijayan

Supervised by
Prof. Dr. Florian Frank
Prof. Dr. Günther Grün

25. July 2019

Contents

List of Tables

List of Figures

1 Introduction

Both manned and unmanned submarines are widely used for research, equipment installation and maintenance, and as weapons. The problem of designing submarine body for low drag and better fuel efficiency has been the subject of many studies dating back to the beginning of last century.

In this project, the optimization set up is implemented in a 2D spatial domain for a steady state, incompressible flow. The optimal state is defined by the reduction of drag force subjected to the body. An iterative procedure is followed to obtain a shape which experiences a lesser drag force compared to the previous iterations. The procedure is followed until a required tolerance for the drag force reduction is reached. An objective function representing the drag force is formulated mathematically and solved to obtain the optimality conditions. The objective function should be minimized satisfying the state equation and constant volume condition. An adjoint method is utilized to combine the constraints and the objective function. A weighted gradient method is used to update the coordinates. The results are compared for different initial shapes and varying flow parameters. Finally a family of shapes can be concluded that exhibit minimum drag force.

2 Literature review

In order to solve the shape optimization problem, the minimum drag force of the body has to be determined which is located in an incompressible viscous fluid flow by the arbitrary Lagrangian Eulerian (ALE) finite element method and an optimal control theory in which the objective function is expressed by the drag and lift force. This method is described in [1]. This objective can be transformed into a minimization problem without constraint conditions by the Lagrange multiplier method/ adjoint method. And as a minimization technique, the weighted gradient method is applied.

The article [2] presents a new approach to a shape optimization problem of a body located in the unsteady incompressible viscous flow field based on an optimal control theory. The optimal state is defined by the reduction of drag and lift forces subjected to the body.

In [3] the shape optimization in the incompressible Navier-Stokes flow is performed by using automatic differentiation. The drag and lift forces can be reduced by changing the surface coordinates which are calculated by the automatic differentiation.

The article [4] presents a NSGA-2 (genetic algorithm) based multi-objective (drag and volume) optimization model integrated with computational fluid dynamics for shape design of submarine hull described by a simple 5-parameter formula.

3 Mathematical modeling

Based on the theory, that the optimal shape should minimize the squared sum of fluid forces acting on the submarine body, adjoint equation method is used to derive the optimization function. The gradient of the optimization function with respect to the submarine surface coordinates should be introduced to minimize the objective function. The weighted gradient method is also utilized to find the new coordinates.

Let $\Omega \subset \mathbb{R}^2$ denote the spatial domain with $\boldsymbol{X}$ representing the coordinate of the submarine boundary associated with Ω within the time $t \in (0,T)$. Let Γ denote the boundary of the flow domain Ω with the incompressible viscous fluid occupying it. The submarine body is represented by the space Ω_S. Suppose that the boundary of Ω is denoted by $\Gamma = \Gamma_U \cup \Gamma_D \cup \Gamma_S \cup \Gamma_B$. Where Γ_B is boundary of the submarine body, Γ_U is the upstream boundary of the domain, Γ_D is the downstream boundary of the domain and Γ_S is the top and bottom boundary of the domain.The physics of general fluid flows can be represented using the Navier-Stokes equations. The two major equations considered are the mass conservation (1) and the momentum conservation (2) equations. The general form in symbolic notation is given as below:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = 0 \qquad in \quad \Omega \times (0,T) \tag{1}$$

$$\frac{\partial (\rho \boldsymbol{u})}{\partial t} + \rho \boldsymbol{u} \cdot \nabla \boldsymbol{u} + \nabla p - \nabla \cdot \left\{ \mu [\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T] \right\} = \boldsymbol{0} \qquad in \quad \Omega \times (0,T) \tag{2}$$

where $\boldsymbol{u}$ and p are velocity and pressure, ρ is density, μ dynamic viscosity.

For an incompressible, steady-state flow, the density is constant and there is no change in velocity with respect to time. Thus the equations (1) and (2) reduce as:

$$\boldsymbol{u} \cdot \nabla \boldsymbol{u} + \nabla p - \nabla \cdot \{ \nu [\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T] \} = \boldsymbol{0} \qquad in \quad \Omega \times (0,T) \tag{3}$$

$$\nabla \cdot \boldsymbol{u} = 0 \qquad in \quad \Omega \times (0,T) \tag{4}$$

where Re is Reynolds number and ν is kinematic viscosity and is defined as

$$\nu = \frac{\mu}{\rho} = \frac{1}{Re}$$

Considering the flow domain shown in Fig. 1, in which a solid body with boundary Γ_B is laid in the external flow. The boundary condition and divergence-free initial condition for this problem are given as:

$\boldsymbol{u} = (U,0) \quad on \quad \Gamma_U \times (0,T)$

$\boldsymbol{t} = \{-p\mathbb{I} + \nu [\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T]\} \cdot \boldsymbol{n} = \boldsymbol{0} \quad on \quad \Gamma_D \times (0,T)$

$t_1 = 0, u_2 = 0 \quad on \quad \Gamma_S \times (0,T)$

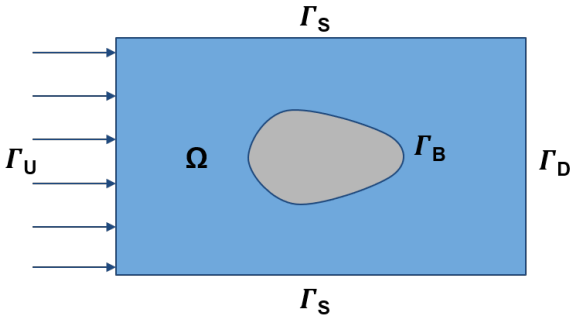


Fig. 1: Physical domain and boundaries enclosing water and the submarine body

$$\boldsymbol{u} = \boldsymbol{0} \quad on \quad \Gamma_B \times (0,T)$$

$$\boldsymbol{u}(\boldsymbol{x},0) = \boldsymbol{u}_0 \quad in \quad \Omega_0$$

where U - constant inlet velocity, $\boldsymbol{t}$ - traction vector, $\mathbb{I}$ - unit tensor, $\boldsymbol{n}$ - outward normal unit vector to Γ.

The forces exerted on the body are due to the pressure force and the viscous force. The horizontal component of force is known as the drag force D and the vertical component is the lift force $\boldsymbol{L}$. Considering an axisymmetric shape, lift force can be neglected. The pressure force $\boldsymbol{F}_p$ and the viscous force $\boldsymbol{F}_T$ on an infinitesimally small area element $\boldsymbol{dA}$ on the submarine boundary surface can be calculated as follows:

$$\boldsymbol{F}_p = -p\mathbb{I}\boldsymbol{dA} \tag{5}$$

$$\boldsymbol{F}_T = \mathbb{T}\boldsymbol{dA} \tag{6}$$

where $\boldsymbol{dA}$ - area normal vector, and $\mathbb{T}$ - shear stress on Γ_B. The drag force is obtained by integrating the traction $\boldsymbol{t}$ on Γ_B and considering the horizontal components of the pressure and viscous force as:

$$D = \int_{\Gamma_B} (-p\mathbb{I}\boldsymbol{dA}\cos\theta) + \int_{\Gamma_B} (\mathbb{T}\boldsymbol{dA}\sin\theta) \tag{7}$$

where θ is the angle between area normal vector $\boldsymbol{dA}$ and the direction of flow.

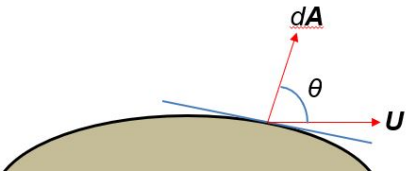


Fig. 2: Representation of an area element with flow direction and area normal vector

3.1 Objective function

The objective function J is defined by the temporal integration of squared sum of drag force from time $t = 0$ to $t = T$ to find the surface coordinates of the target body with minimal drag:

$$J = \frac{1}{2}\int_0^T qD^2 dt \tag{8}$$

where q is the weight corresponding to drag force. The optimality condition can be derived considering the minimization of (8) as:

$$\begin{aligned} \min_{\boldsymbol{X}} \quad & J = f(D(\boldsymbol{X})) \\ \text{subject to} \quad & V(\boldsymbol{X}) = V_0 \end{aligned}$$

where
$\boldsymbol{X}$: Boundary coordinates of Body, defined by surface Γ_B
D : Drag force
V_0 : Initial volume

3.2 Extended objective function - adjoint method

The constraints are considered along with the objective function in adjoint equation method. The Navier-Stokes equations (3) and (4) along with the volume constraint equation (**??**) are multiplied with the adjoint velocity ($\boldsymbol{u}^*$), adjoint pressure (p^*) and lagrange multiplier (λ) respectively and added up to the objective fucntion (8) to form the extended objective function (J^*) as follows:

$$\begin{aligned} J^* = \quad & \frac{1}{2}\int_0^T qD^2 dt \quad - \quad \int_0^T \int_\Omega \boldsymbol{u}^* \{ \frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot\nabla\boldsymbol{u} + \nabla p - \nabla\cdot\{\nu[\nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^T]\}\} d\Omega dt \\ + \quad & \int_0^T \int_\Omega p^* \nabla\cdot\boldsymbol{u} d\Omega dt \quad + \quad \lambda\left[\left\{\int_{\Omega_S} V(\boldsymbol{X}) d\Omega_S\right\}^{(l)} - V_0\right] \end{aligned} \tag{9}$$

where all the flow field terms ($\boldsymbol{u}$, p, $\boldsymbol{u}^*$ and p^*), force term (D) are dependent on the coordinate vector $\boldsymbol{X}$. If the Navier-Stokes equations (3) and (4) are satisfied, then the second and third term in J^* vanishes and with constant volume condition the fourth term vanishes. To minimize J^*, the gradient of J^* with respect to geometrical surface coordinates ($\boldsymbol{X}_i$) should be introduced. The stationary condition can be derived from the first variation of the extended objective function.

For convenience, the extended objective function is divided in three terms as force term (I_1), momentum conservation term (I_2) and mass conservation term (I_3).

Variation of force term:

$$I_1 = \frac{1}{2}\int_0^T qD^2 dt$$

$$\delta I_1 = \int_0^T qD\delta D dt \tag{10}$$

The fluid force ($\boldsymbol{F}$) acting on the body is obtained by integrating the traction $\boldsymbol{t}$ on the boundary Γ_B as:

$$F_i = -\int_{\Gamma_B} t_i d\Gamma \tag{11}$$

Using the force definition (11), the equation (10) can be rewritten using variation of force as,

$$\delta I_1 = \int_0^T qD\Big(-\int_{\Gamma_B} \delta t_1 d\Gamma\Big) dt$$

$$\delta I_1 = -\int_0^T \int_{\Gamma_B} qD\delta t_1 d\Gamma dt \tag{12}$$

Variation of momentum conservation term:

$$I_2 = -\int_0^T \int_\Omega \boldsymbol{u}^* \{\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot\nabla\boldsymbol{u} + \nabla p - \nabla\cdot\{\nu[\nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^T]\}\} d\Omega dt$$

$$I_2 = -\int_0^T \int_\Omega u_i^* (\dot{u}_i + u_j u_{i,j} + p_{,i} - \nu[u_{i,j} + u_{j,i}]_{,j}) d\Omega dt$$

$$\begin{aligned}\delta I_2 = &-\int_0^T \int_\Omega \delta u_i^* (\dot{u}_i + u_j u_{i,j} + p_{,i} - \nu[u_{i,j} + u_{j,i}]_{,j}) d\Omega dt \\ &-\int_0^T \int_\Omega u_i^* (\delta\dot{u}_i + \delta(u_j u_{i,j}) + \delta p_{,i} - \nu\delta[u_{i,j} + u_{j,i}]_{,j}) d\Omega dt\end{aligned}$$

For convenience, the first term in δI_2 is represented as B_1 further in derivation

$$B_1 = -\int_0^T \int_\Omega \delta u_i^* (\dot{u}_i + u_j u_{i,j} + p_{,i} - \nu[u_{i,j} + u_{j,i}]_{,j}) d\Omega dt \tag{13}$$

and the second term in δI_2 is subdivided in four terms as variation of time derivative of velocity (A_1), variation of convection term (A_2), variation of pressure term (A_3) and variation of diffusion term (A_4).

Variation of time derivative of velocity:

$$A_1 = -\int_0^T \int_\Omega u_i^* \delta\dot{u}_i d\Omega dt = -\int_\Omega \int_0^T u_i^* \delta\dot{u}_i dt d\Omega$$

By partial integration,

$$A_1 = -\int_\Omega \Big[[u_i^* \delta u_i]_0^T - \int_0^T \dot{u_i}^* \delta u_i dt\Big] d\Omega$$

$$A_1 = -\int_\Omega u_i^*(T)\delta u_i(T) d\Omega + \int_\Omega u_i^*(0)\delta u_i(0) d\Omega + \int_0^T \int_\Omega \dot{u_i}^* \delta u_i d\Omega dt \tag{14}$$

Variation of convection term:

$$A_2 = -\int_0^T \int_\Omega u_i^* \delta(u_j u_{i,j}) d\Omega dt = -\int_0^T \int_\Omega u_i^* u_j \delta u_{i,j} d\Omega dt$$

By partial integration,

$$A_2 = -\int_0^T \left[\int_\Gamma u_j u_i^* \delta u_i \boldsymbol{n} d\Gamma - \int_\Omega (u_j u_i^*)_{,j} \delta u_i d\Omega \right] dt \tag{15}$$

Variation of pressure term:

$$A_3 = -\int_0^T \int_\Omega u_i^* \delta p_{,i} d\Omega dt$$

By partial integration,

$$A_3 = -\int_0^T \left[\int_\Gamma u_i^* \delta p \boldsymbol{n} d\Gamma - \int_\Omega u_{i,i}^* \delta p d\Omega \right] dt \tag{16}$$

Variation of diffusion term:

$$A_4 = \int_0^T \int_\Omega \nu u_i^* \delta[(u_{i,j} + u_{j,i})_{,j}] d\Omega dt$$

$$A_4 = \int_0^T \int_\Omega \nu u_i^* [\delta(u_{i,j})_{,j} + \delta(u_{j,i})_{,j}] d\Omega dt$$

Letting,

$$A_{41} = \int_0^T \int_\Omega \nu u_i^* \delta(u_{i,j})_{,j} d\Omega dt$$

By partial integration,

$$A_{41} = \int_0^T \left[\int_\Gamma \nu u_i^* \delta u_{i,j} \boldsymbol{n} d\Gamma - \int_\Omega \nu u_{i,j}^* \delta u_{i,j} d\Omega \right] dt$$

By partial integration,

$$A_{41} = \int_0^T \left[\int_\Gamma \nu u_i^* \delta u_{i,j} \boldsymbol{n} d\Gamma - \int_\Gamma \nu u_{i,j}^* \delta u_i \boldsymbol{n} d\Gamma + \int_\Omega \nu (u_{i,j}^*)_{,j} \delta u_i d\Omega \right] dt \tag{17}$$

Letting,

$$A_{42} = \int_0^T \int_\Omega \nu u_i^* \delta(u_{j,i})_{,j} d\Omega dt$$

By partial integration,

$$A_{42} = \int_0^T \left[\int_\Gamma \nu u_i^* \delta u_{j,i} \boldsymbol{n} d\Gamma - \int_\Omega \nu u_{i,j}^* \delta u_{j,i} d\Omega \right] dt$$

By partial integration,

$$A_{42} = \int_0^T \left[\int_\Gamma \nu u_i^* \delta u_{j,i} \boldsymbol{n} d\Gamma - \int_\Gamma \nu u_{i,j}^* \delta u_j \boldsymbol{n} d\Gamma + \int_\Omega \nu (u_{i,j}^*)_{,i} \delta u_j d\Omega \right] dt$$

$$A_{42} = \int_0^T \left[\int_\Gamma \nu u_j^* \delta u_{i,j} \boldsymbol{n} d\Gamma - \int_\Gamma \nu u_{j,i}^* \delta u_i \boldsymbol{n} d\Gamma + \int_\Omega \nu (u_{j,i}^*)_{,j} \delta u_i d\Omega \right] dt \tag{18}$$

Variation of mass conservation term:

$$I_3 = \int_0^T \int_\Omega p^* \nabla \cdot \boldsymbol{u} d\Omega dt = \int_0^T \int_\Omega p^* u_{i,i} d\Omega dt$$

$$\delta I_3 = \int_0^T \int_\Omega \delta p^* u_{i,i} d\Omega dt + \int_0^T \int_\Omega p^* \delta u_{i,i} d\Omega dt$$

By partial integration,

$$\delta I_3 = \int_0^T \int_\Omega \delta p^* u_{i,i} d\Omega dt + \int_0^T \left[\int_\Gamma p^* \delta u_i \boldsymbol{n} d\Gamma - \int_\Omega p^*_{,i} \delta_{ij} \delta u_i d\Omega \right] dt \tag{19}$$

For minimizing the extended objective function, the first variation of the objective function with respect to the coordinate vector ($\boldsymbol{X}$) should vanish.

$$\delta J^* \stackrel{!}{=} 0$$

Combining all the terms of equations (12) to (19):

$$\delta J^* = \delta I_1 + \delta I_2 + \delta I_3 \quad = \delta I_1 + B_1 + A_1 + A_2 + A_3 + A_{41} + A_{42} + \delta I_3$$

$$\begin{aligned}
\delta J^* = & - \int_0^T \int_{\Gamma_B} qD\delta t_1 d\Gamma dt \\
& - \int_0^T \int_\Omega \delta u_i^* (\dot{u}_i + u_j u_{i,j} + p_{,i} - \nu [u_{i,j} + u_{j,i}]_{,j}) d\Omega dt \\
& - \int_\Omega u_i^*(T) \delta u_i(T) d\Omega + \int_\Omega u_i^*(0) \delta u_i(0) d\Omega + \int_0^T \int_\Omega \dot{u_i}^* \delta u_i d\Omega dt \\
& - \int_0^T \int_\Gamma u_j u_i^* \delta u_i \boldsymbol{n} d\Gamma dt + \int_0^T \int_\Omega (u_j u_i^*)_{,j} \delta u_i d\Omega dt \\
& - \int_0^T \int_\Gamma u_i^* \delta p \delta_{i,j} \boldsymbol{n} d\Gamma dt + \int_0^T \int_\Omega u_{i,i}^* \delta p d\Omega dt \\
& + \int_0^T \int_\Gamma \nu u_i^* \delta u_{i,j} \boldsymbol{n} d\Gamma dt - \int_0^T \int_\Gamma \nu u_{i,j}^* \delta u_i \boldsymbol{n} d\Gamma dt + \int_0^T \int_\Omega \nu (u_{i,j}^*)_{,j} \delta u_i d\Omega dt \\
& + \int_0^T \int_\Gamma \nu u_j^* \delta u_{j,i} \boldsymbol{n} d\Gamma dt - \int_0^T \int_\Gamma \nu u_{j,i}^* \delta u_i \boldsymbol{n} d\Gamma dt + \int_0^T \int_\Omega \nu (u_{j,i}^*)_{,j} \delta u_i d\Omega dt \\
& + \int_0^T \int_\Omega \delta p^* u_{i,i} d\Omega dt + \int_0^T \int_\Gamma p^* \delta u_i \boldsymbol{n} d\Gamma dt - \int_0^T \int_\Omega p_{,i}^* \delta ij \delta u_i d\Omega dt
\end{aligned}$$

$$
\begin{aligned}
\delta J^* = & -\int_0^T \int_\Omega \delta u_i^* (\dot{u}_i + u_j u_{i,j} + p_{,i} - \nu[u_{i,j} + u_{j,i}]_{,j}) d\Omega dt \\
& + \int_0^T \int_\Omega \delta p^* u_{i,i} d\Omega dt \\
& - \int_0^T \int_\Omega \Big\{ -\dot{u_i}^* - (u_j u_i^*)_{,j} + p_{,i}^* - \nu(u_{i,j}^* + u_{j,i}^*)_{,j} \Big\} \delta u_i d\Omega dt \\
& + \int_0^T \int_\Omega u_{i,i}^* \delta p d\Omega dt \\
& - \int_0^T \int_{\Gamma_B} qD \delta t_1 d\Gamma dt \\
& - \int_\Omega u_i^*(T) \delta u_i(T) d\Omega + \int_\Omega u_i^*(0) \delta u_i(0) d\Omega \\
& + \int_0^T \left[\int_{\Gamma_U} + \int_{\Gamma_D} + \int_{\Gamma_B} + \int_{\Gamma_S} u_i^* \delta t_i \right] d\Gamma dt \\
& - \int_0^T \left[\int_{\Gamma_U} + \int_{\Gamma_D} + \int_{\Gamma_B} + \int_{\Gamma_S} s_i \delta u_i \right] d\Gamma dt
\end{aligned}
$$

where $t_i = \{-p\delta_{ij} + \nu(u_{i,j} + u_{j,i})\} n_j$

and $s_i = \{u_j u_i^* - p^* \delta_{ij} + \nu(u_{i,j}^* + u_{j,i}^*)\} n_j$

Applying boundary conditions of Navier-Stokes equation, few terms in the above equation vanishes, resulting in the following form:

$$
\begin{aligned}
\delta J^* = & -\int_0^T \int_\Omega \Big\{ -\dot{u_i}^* - (u_j u_i^*)_{,j} + p_{,i}^* - \nu(u_{i,j}^* + u_{j,i}^*)_{,j} \Big\} \delta u_i d\Omega dt \\
& + \int_0^T \int_\Omega u_{i,i}^* \delta p d\Omega dt \\
& + \int_0^T \int_{\Gamma_U} u_i^* \delta t_i d\Gamma dt \\
& - \int_0^T \int_{\Gamma_D} s_i \delta u_i d\Gamma dt \\
& - \int_0^T \int_{\Gamma_S} s_1 \delta u_1 d\Gamma dt + \int_0^T \int_{\Gamma_S} u_2^* \delta t_2 d\Gamma dt \\
& + \int_0^T \int_{\Gamma_B} (u_1^* - qD) \delta t_1 d\Gamma dt + \int_0^T \int_{\Gamma_B} u_2^* \delta t_1 d\Gamma dt \\
& - \int_\Omega u_i^*(T) \delta u_i(T) d\Omega + \int_\Omega u_i^*(0) \delta u_i(0) d\Omega \\
& - \int_0^T \int_{\Gamma_B} s_i \delta u_i d\Gamma dt
\end{aligned}
\tag{20}
$$

Setting each term equals to zero, to satisfy the optimality condition, the adjoint equation and the boundary conditions are obtained.

3.3 Optimality condition equations (OCE)

The derivation of optimality condition above leads to the following set of equations and boundary conditions, which can be referred as the optimality condition equations.

$$\nabla \cdot \boldsymbol{u}^* = 0 \tag{21}$$

$$-\frac{\partial \boldsymbol{u}^*}{\partial t} + (\nabla \boldsymbol{u})^T \boldsymbol{u}^* - \boldsymbol{u} \cdot \nabla \boldsymbol{u}^* + \nabla p^* - \nabla \cdot \{\nu[\nabla \boldsymbol{u}^* + (\nabla \boldsymbol{u}^*)^T]\} = \boldsymbol{0} \tag{22}$$

Boundary Conditions :

$\boldsymbol{u}^* = \boldsymbol{0} \qquad on \qquad \Gamma_U \times (0,T)$

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

$s_1 = 0, u_2^* = 0 \qquad on \qquad \Gamma_S \times (0,T)$

$\boldsymbol{u}^* = (q_1 D, 0) \qquad on \qquad \Gamma_B \times (0,T)$

$\boldsymbol{u}^*(\boldsymbol{x},T) = \boldsymbol{0} \qquad in \qquad \Omega$

With all other terms in (20) zero, the equation results in:

$$\delta J^* = -\int_0^T \int_{\Gamma_B} s_i \delta u_i d\Gamma dt$$

The variation δu_i is defined as:

$$\delta u_i = \frac{\partial u_i}{\partial \boldsymbol{X}_j} \delta X_j = u_{i,j} \delta \boldsymbol{X}_j$$

where $\boldsymbol{X}_j$ is the coordinate vector that defines shape of the body. Using the definition of variation of u_i in δJ^*,

$$\delta J^* = -\int_0^T \int_{\Gamma_B} s_i u_{i,j} \delta \boldsymbol{X}_j d\Gamma dt$$

Since $\delta \boldsymbol{X}_j$ is an arbitrary variation, the gradient to adjust the shape of target body is given as:

$$grad(J^*) = -s_j u_{j,i} \tag{23}$$

3.4 Weighted gradient method

The weighted gradient method is scarcely dependent on the initial value, thus this method is adopted as the minimization technique in this work. Here, a modified objective function, $K^{(i)}$ is obtained by adding a penalty term to the extended objective function as shown below:

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

where i is the iteration number of minimization and W is the weighting diagonal matrix. If the modified objective function converges to minimum, the penalty term can also be zero, *i.e.*, the minimization pro-

blem of the modified and extended objective function are the same. Thus applying stationary condition to the modified objective function K and assuming $\boldsymbol{X}^i$ is the optimal solution,

$$\delta K^i = grad(J^*)^i \delta \boldsymbol{X}^i + W(\boldsymbol{X}^{(i+1)} - \boldsymbol{X}^i)\delta \boldsymbol{X}^i \stackrel{!}{=} 0$$

The relationship between the new and the current coordinate vector can thus be established as:

$$\boldsymbol{X}^{(i+1)} = \boldsymbol{X}^i - \frac{1}{W} grad(J^*)^i \tag{24}$$

where $grad(J^*)^i = -\nabla \boldsymbol{u}^T \cdot \boldsymbol{s}$ as given in (23)

4 Numerical schemes

The Navier-Stokes equations and the optimality condition equations to be solved are PDEs that involve derivatives of tensors with respect to time and space. In this chapter it will be explained, how the derivatives are discretized in the complete domain to form a set of algebraic equations. Different schemes can be used to solve PDEs. Finite volume method is used in this work for representing and evaluating PDEs in the form of solvable algebraic equations.

4.1 Finite Volume Method (FVM)

The unknown variables are calculated at discrete points on the meshed geometry by approximating the integral form of the equations rather than the usual derivative form. The domain is discretized into a number of control volumes (CV) surrounding each node point on mesh. All the conservation equations in its integral form has to be satisfied in the CVs and thus in the entire flow domain.

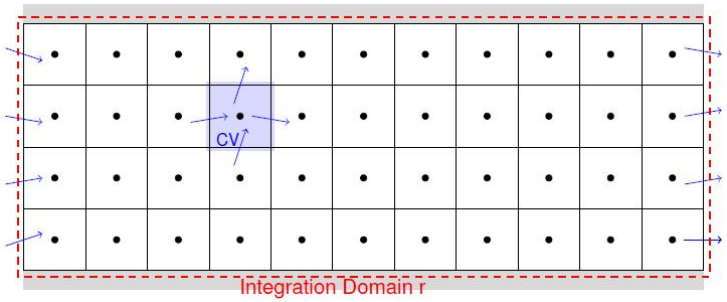


Fig. 3: Control volume in integration domain

4.2 Discretization of Navier-Stokes equation

The Navier-Stokes equation can be expressed in integral form for FVM over a considered small control volume V and cell Surface $\boldsymbol{S}$ as:

$$\int_V \frac{\partial u_i u_j}{\partial x_j} dV - \int_V \nu \frac{\partial^2 u_i}{\partial x_j x_j} dV + \int_V \frac{1}{\rho} \frac{\partial p}{\partial x_i} dV = 0 \tag{25}$$

The volume integrals in a PDE that contain a divergence term can be converted to surface integrals using Gauss-Divergence theorem. Then these terms are evaluated as fluxes entering in and leaving out of the surfaces of each CV. Thus satisfying all conservation equations, since fluxes entering a CV is same as that leaving the adjacent CV. For any vector $\boldsymbol{a}$, the divergence theorem is as below:

$$\int_V \frac{\partial a_j}{\partial x_j} d\Omega = \int_S a_j n_j dS \tag{26}$$

4.2.1 Convection term

Applying equation (26) to the convective term in (25), it is approximated as:

$$\int_V \frac{\partial u_i u_j}{\partial x_j} dV = \int_S u_i u_j n_j dS \quad \approx \quad \sum_f \boldsymbol{S}_f \cdot (u_i)_f (u_j)_f = \sum_f F (u_j)_f \tag{27}$$

where $(u_i)_f$, $(u_j)_f$ are velocities in x, y direction on face f, S_f is surface area on f, and $F = S_f \cdot (u_i)_f$. An Upwind Differencing Scheme (UDS) is used for the approximation of velocity on the face considered using the velocity on the cell centre. The upwind differencing scheme chooses the velocity of the centre node of the cell which is upstream to the surface. Mathematically,

$$\boldsymbol{u}_f = \begin{cases} \boldsymbol{u}_P, & \text{for } F \geq 0 \\ \boldsymbol{u}_N, & \text{for } F < 0 \end{cases} \tag{28}$$

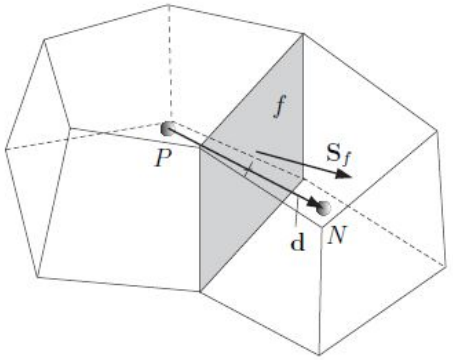


Fig. 4: Parameters in FVM

4.2.2 Diffusion term

Similar to the convection term, the diffusion term in (25) is discretized by applying equation (26). The second order derivative can be transformed to equation (29):

$$\int_V \nu \frac{\partial^2 u_i}{\partial x_j x_j} dV = \int_V \frac{\partial}{\partial x_j} \left(\nu \frac{\partial u_i}{\partial x_j} \right) dV = \int_S \nu \frac{\partial u_i}{\partial x_j} n_j dS \quad \approx \quad \sum_f \nu \boldsymbol{S}_f \cdot \left(\frac{\partial u_i}{\partial x_j} \right)_f \tag{29}$$

An implicit face gradient discretization is used when the length vector $\boldsymbol{d}$ between the centre of the cell of interest P and centre of neighbouring cell N is orthogonal to the face plane, i.e. parallel to S_f as:

$$\boldsymbol{S}_f \cdot \left(\frac{\partial u_i}{\partial x_j}\right)_f = |\boldsymbol{S}_f| \frac{(u_i)_N - (u_i)_P}{|\boldsymbol{d}|}$$

4.2.3 Pressure term

The pressure term is discretized to equation (30) by applying equation (26) and subsequently Gauss integration.

$$\int_V \frac{1}{\rho} \frac{\partial p}{\partial x_i} dV = \int_S \frac{1}{\rho} p n_j dS \approx \sum_f \frac{1}{\rho} \boldsymbol{S}_f p_f \tag{30}$$

where p_f is the pressure on the face f

4.3 Discretization of optimality condition equation

The equation for optimality condition is expressed in the integral form (31) for discretization using FVM. Since the equation involves both time and spatial derivatives, a volume integral and a time integral are used. For a small time interval δt,

$$\int_t^{t+\delta t} \left[\int_V -\frac{\partial u_i^*}{\partial t} dV + \int_V \frac{\partial (u_j^* u_i)}{\partial x_j} dV - \int_V \frac{\partial (u_j u_i^*)}{\partial x_j} dV - \int_V \nu \frac{\partial^2 u_i^*}{\partial x_j x_j} dV + \int_V \frac{\partial p^*}{\partial x_i} dV \right] dt = 0 \tag{31}$$

4.3.1 Temporal discretization

Denoting all the spatial terms as $\mathscr{A}\phi$ where $\mathscr{A}$ is any spatial operator and ϕ is the face field, the equation (31) can be rewritten as:

$$\int_t^{t+\delta t} \left[-\frac{\partial}{\partial t} \int_V \boldsymbol{u}^* dV + \int_V \mathscr{A}\phi dV \right] dt = 0 \tag{32}$$

For the temporal discretization, an Euler implicit scheme is used, that relates the values of previous and current time step. Discretizing the time derivative term in equation (32),

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

And the time discretization of the spatial term of equation (32) is expressed as:

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

where $\mathscr{A}^*$ is the spatial discretization of $\mathscr{A}$.
Using implicit discretization, the spatial terms are expressed at the current time step $(n+1)$ as

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

Combining both discretized terms, the time discretized equation can be written as

$$-\frac{(\boldsymbol{u}_P^* V)^{n+1} - (\boldsymbol{u}_P^* V)^n}{\delta t} + \mathscr{A}^* \phi^{n+1} = 0 \tag{33}$$

It is first order accurate in time, guarantees boundedness and is unconditionally stable.

4.3.2 Spatial discretization

The spatial discretization of optimality condition equations is similar to that of the Navier-Stokes equation. The two convection terms in equation (31) are discretized as in equation (27). The diffusion term in equation (31) is discretized as shown in equation (29) and the pressure term in equation (31) is discretized as in equation (30).

4.4 Discretization of boundary points

There are different schemes for the discretization at the boundary points of the domain, which has to be considered when one of the faces is a boundary face.

4.4.1 Dirichlet or fixed value boundary condition

Dirichlet boundary condition prescribes dependent variable on the boundary.

- At the boundary, the dependant variable can take up the boundary value ϕ_b (eg. in convection term).
- For discretizing a gradient of the dependant variable (eg. in laplacian term), the boundary value can be used as:

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

  where, ϕ_b is the boundary value, ϕ_p is the value of the dependant variable on the cell centre of boundary cell, and $|\boldsymbol{d}|$ is half the cell distance.

4.4.2 Neumann or fixed gradient boundary condition

Neumann or fixed gradient boundary condition g_b prescribes the gradient of a dependant variable normal to the boundary. The term g_b is a specification on inner product of the gradient and unit normal to the boundary.

$$g_b = \left(\frac{\boldsymbol{S}}{|\boldsymbol{S}|} \cdot \nabla \phi \right)_f \tag{34}$$

- At the boundary, the dependant variable can be approximated (ϕ_f) using g_b (eg. in convection term) as below:

$$\phi_f = \phi_P + \boldsymbol{d} \cdot (\nabla\phi)_f = \phi_P + |\boldsymbol{d}| g_b \tag{35}$$

- For discretizing a gradient of the dependant variable (eg. in Laplacian term), the boundary value can be used as:

$$\boldsymbol{S}_f \cdot (\nabla\phi)_f = |\boldsymbol{S}| g_b \tag{36}$$

5 Implementation

The sequence of the complete algorithm for the shape optimisation task is represented in Fig. 5. The algorithm is implemented in *Python*. In each iteration two sets of equations are solved in order to find an improved submarine shape, ensuring reduction in drag force with respect to the previous iteration. Every iteration consists of a solution of a Navier-Stokes equation and consecutively a solution of an optimality condition for the minimization of the drag force on the submarine. These two set of equations are solved in solvers developed inside the open-source software, *OpenFoam*.

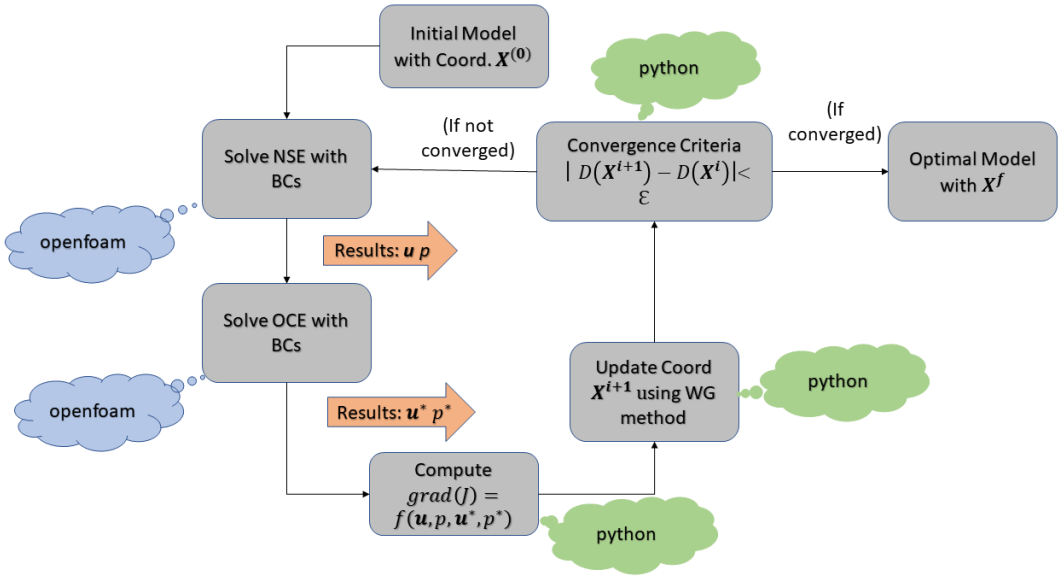


Fig. 5: Schematic representation of the algorithm

5.1 Algorithm

The algorithm starts with an assumed initial geometry of the submarine which is modelled and meshed using *OpenFoam*. *OpenFoam* is a very flexible and powerful tool for solving flow simulations. The Navier-Stokes equations are solved in order to obtain the unknown velocity $\boldsymbol{u}$ and the pressure p. These results are then used to solve the equations of the optimality condition to obtain the adjoint velocity $\boldsymbol{u}^*$ and adjoint pressure p^*. Further, the weighted gradient method (24) is used to update the coordinates of

the submarine boundary. The results of $\boldsymbol{u}$, p, $\boldsymbol{u}^*$ and p^* are used for the calculation of the new coordinates. Convergence of drag force on the submarine body is checked at this point to ensure the attainment of optimal drag shape for minimum drag force. Relative tolerance of 1e$-$5 for the drag force is the standard stopping criteria chosen for the algorithm. Additionally, an upper limit is fixed to the maximum number of iterations to ensure that the algorithm does not run for a very large number of iterations. If the convergence criteria is not met, the iteration continues further with the new shape until convergence.

5.2 Solver implementation

Solving the two sets of equations, namely the Navier-Stokes equations and the equations of optimality condition, are the integral part of the work. *OpenFoam* offers well developed solvers for solving Navier-Stokes Equation, which required minor corrections to suit the exact requirement of the task. An existing solver, *simpleFoam* is used to solve the steady state Navier-Stokes Equations. And a new solver was developed in *OpenFoam* for solving the optimality condition equations. Using the same software for the solvers helps reduce the data transfer time, which would have been affecting the simulation time immensely if two softwares were used.

5.2.1 Predictor-corrector method

The solution algorithm for solving Navier-Stokes equation and optimality condition equation is not straight forward, since the mass conservation equation does not involve the pressure term. So an additional equation needs to be formed and solved to ensure that the pressure and velocity are well coupled and satisfy both the momentum and mass conservation. An additional equation named Poisson's Equation is formed to solve the pressure separately. The predictor-corrector algorithm is followed in *OpenFoam* to couple the pressure and velocity for each time step. The time discretized momentum equation of Navier-Stokes is represented as in equation. (37) needs to be solved to obtain u^{n+1} and p^{n+1} in each time step.

$$\frac{(u_i)^{n+1} - (u_i)^n}{\Delta t} = \left[\frac{\partial (u_i u_j)}{\partial x_j}\right]^n + \nu \left[\frac{\partial^2 u_i}{\partial x_j x_j}\right]^n - \frac{1}{\rho}\left[\frac{\partial p}{\partial x_i}\right]^{n+1} \tag{37}$$

A predicted velocity equation (38) is formed disregarding the pressure term in (37)

$$\frac{(u_i)^{'} - (u_i)^n}{\Delta t} = \left[\frac{\partial (u_i u_j)}{\partial x_j}\right]^n + \nu \left[\frac{\partial^2 u_i}{\partial x_j x_j}\right]^n \tag{38}$$

Subtracting equation (38) from (37)

$$\frac{(u_i)^{n+1} - (u_i)^{'}}{\Delta t} = -\frac{1}{\rho}\left[\frac{\partial p}{\partial x_i}\right]^{n+1} \tag{39}$$

Taking divergence on equation (39)

$$\frac{\partial}{\partial x_i}\left[\frac{(u_i)^{n+1} - (u_i)^{'}}{\Delta t}\right] = -\frac{1}{\rho}\frac{\partial}{\partial x_i}\left[\left[\frac{\partial p}{\partial x_i}\right]^{n+1}\right] \tag{40}$$

Since u^{n+1} needs to be a divergent free velocity, the divergence of this term in (40) is zero and can be rewritten in the form

$$\frac{\partial}{\partial x_i}\left[\left[\frac{\partial p}{\partial x_i}\right]^{n+1}\right] = \frac{\rho}{\Delta t}\frac{\partial (u_i)'}{\partial x_i} \tag{41}$$

The following steps are followed for each time step for solving u^{n+1} and p^{n+1} iteratively :
Step 1: The momentum equation is solved as per the equation (38) to obtain a predicted velocity
Step2: The Poisson equation (41) is solved using the predicted velocity in step 1 to obtain the pressure
Step 3: The equation (39) is solved using the predicted velocity and the pressure obtained in step 2 to obtain a velocity which is more divergent free than the previously predicted velocity
Step 4: The steps 1 to 3 are repeated until a divergent free velocity within a tolerance limit is obtained

5.2.2 Solver for Navier Stokes equations

In *OpenFoam*, an existing solver could be easily customized and compiled as per the user's requirement. The existing solver *simpleFoam* is already well suited for the task at hand and has been used directly for solving the Navier-Stokes equation. The equivalent representative code in *OpenFoam* for the equations (3) and (4) combined with the predictor-corrector approach, are as below:

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

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

The predictor-corrector approach implemented for the Navier-Stokes equation could be represented as follows

```
  while (simple.loop(runTime))
    {
        {
            #include "UEqn.H"
            #include "pEqn.H"
        }
    }
```

Apart from solving the unknown velocity **u** and pressure p, it is required to obtain the drag force experienced by the submarine in each iterations. The drag force obtained is used for the solution of the optimality condition equations, and is updated as one of the boundary conditions of those equations. Drag force on a submerged moving body could be calculated using the horizontal components of the force due to pressure and the force due to viscosity. The library function is utilised to find the integral sum of the horizontal components of these two forces around the boundary of the submarine as per the equation (7).

5.2.3 Solver for optimality condition equations

OpenFoam has its own C++ library for developing and customising existing solvers in it. These libraries and its functions were used to develop a solver named *pisomosiFoam* which can solve the optimality condition equations. Since the equations are quite comparable to that of the Navier Stokes equation, one of the solver used for Navier-Stokes equation is used to develop the *pisomosiFoam*. The equivalent representative code in *OpenFoam* for the equations (22) and (21) are as below:

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

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

The predictor-corrector algorithm is used also to solve the equation for the adjoint variables $\boldsymbol{u}^*$ and p^* iteratively to couple the equations (22) and (21). The non-linear term is solved using the previous iteration's value of $\boldsymbol{u}^*$ for one of the unknowns in the non-linear term. The below code snippet represents the predictor-corrector algorithm implemented in *pisomosiFoam*:

```
while (runTime.loop()) //the simulation time loop
    {
        {
            #include "UAdjEqn.H"

            while (piso.correct())
            {
                #include "pAdjEqn.H"
            }
        }
    }
```

An additional feature implemented in the solver is the extraction of the velocity $\boldsymbol{u}$, pressure p, adjoint velocity $\boldsymbol{u}^*$, adjoint pressure p^*, the gradient of adjoint velocity $grad(\boldsymbol{u}^*)$ and the gradient of velocity $grad(\boldsymbol{u})$ from the mesh cells forming the surface boundary of the submarine. This information is further used in equation (24) to update the coordinates of the new boundary surface.

5.3 OpenFoam coupling

OpenFoam uses text files to read and write out data, which makes it convenient for automation with a programming language like *Python*. The simulation settings, including geometry creation and meshing are achieved through the inputs mentioned through the corresponding text files known as dictionaries. The main aspect of geometry modelling using *OpenFoam* is through such a dictionary known as

blockMeshDict file. The flow domain is split into three dimensional blocks each consisting of six faces (hexahedron), which are defined using vertices defined using cartesian coordinate system. The blocks can have edges which are straight lined or curved, as used in this work. A curved edge can be defined using multiple control points. Such a curved edge is used to define the submarine boundary and the control points on this curved edge are updated in each iteration to vary the shape of the submarine boundary. The block structure of the flow domain used in the task is represented in the Fig. 6, where the blue lines represent the blocks, black line represents the flow boundary.

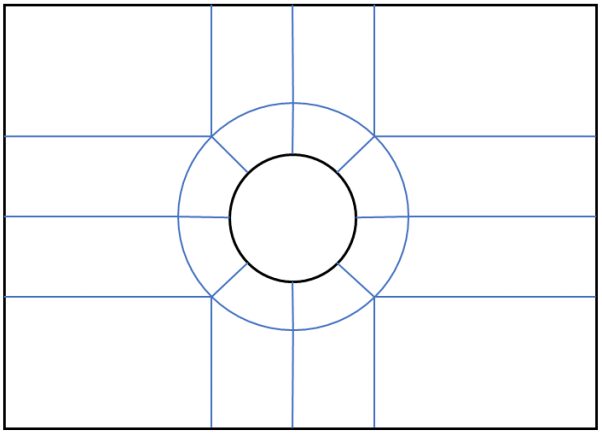


Fig. 6: Flow domain with blocks

The parameters extracted by the *pisomosiFoam* solver is written to a text file. The file contains the parameters required to calculate the updated coordinates as per the equation (24). The parameters are required only from the boundary representing the submarine boundary surface.

5.4 Additional features

There are different aspects which were addressed apart from the main algorithm, that includes fool-proofing the simulation by preventing failures and including aspects to improve the performance of the algorithm.

5.4.1 Convergence rate

The algorithm is not only expected to solve the problem but also to solve the problem efficiently in few iterations and within shortest time possible. Many parts of the algorithm were checked for improvements in speeding up the simulations. The following parameters in the simulation played a major role in deciding the number of iterations required for the simulation to converge:

1) The tolerance limit for the drag force

2) The weight used in the equation (24)

Maintaining a lower tolerance limit for the drag force would increase the simulation convergence time but result in a better converged solution. Similarly, a very low weight affects the rate of coordinate update, resulting in a slow convergence of the solution. It is important to note that a larger weight would also cause greater fluctuation in the result of consecutive iterations, which leads to simulation failure.

5.4.2 Failure prevention

There are different aspects of the simulation due to which the simulation might fail. One of the major factor for failure is disruption in the coordinate update, leading to wrong meshing of the fluid domain. If the coordinate update is not controlled smoothly, there are chances that the shape of the bodies between two iterations vary a lot and could cause irregular mesh cells. This mostly affect the Courant Number of the flow and causes instability in the simulation. The following probable failures are controlled and managed in the implementation:

1) Collapsing of neighbouring control points on the submarine boundary is prevented by maintaining a certain tolerance limit for the distance between any two control points.

2) The change in coordinate vector between two iterations are always maintained between a defined tolerance limit to avoid sudden change of shape

6 Results and discussion

Streamlined body reduces the so called wake region (the region of turbulence, downstream of the body) by reducing the pressure difference between the front and back of the body. This minimizes total drag by reducing the pressure drag at the cost of increasing the viscous drag due to an increase in the tail region. The end result depends on which effect dominates. The general optimized shape is shown in Fig. 7. The variation in total drag force with change in surface dimension is shown in Fig. 8.

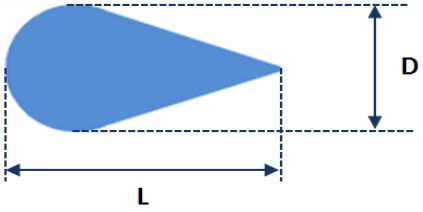


Fig. 7: Submarine with dimensions

A body which is more blunt (circle/ hexagon) in the front is considered as an initial design. As the minimization process continues, the body becomes more streamlined and appears to be contoured and sleek. The consistency of converged shape and drag force are validated under three following scenarios.

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

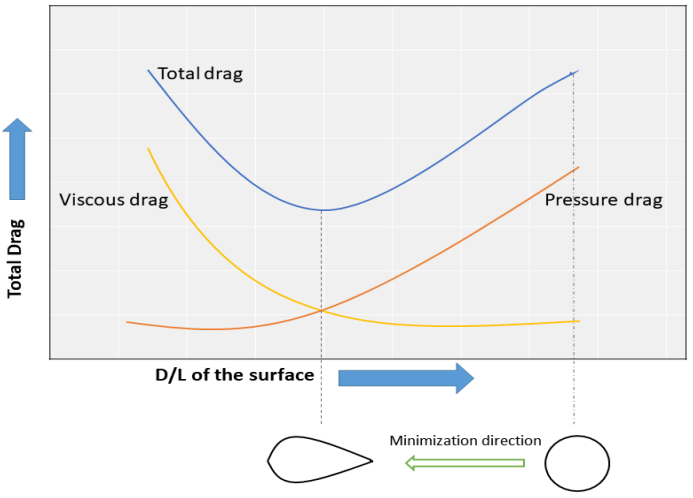


Fig. 8: Components of drag force and its variations with submarine shape

The scenarios are summarized as following five simulation cases for volume V_1 and small increase in volume ΔV_1 due to the increase in radius.

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	$V_1 + \Delta V_1$
5	100	hexagon	V1

Tab. 1: Different simulation scenarios studied

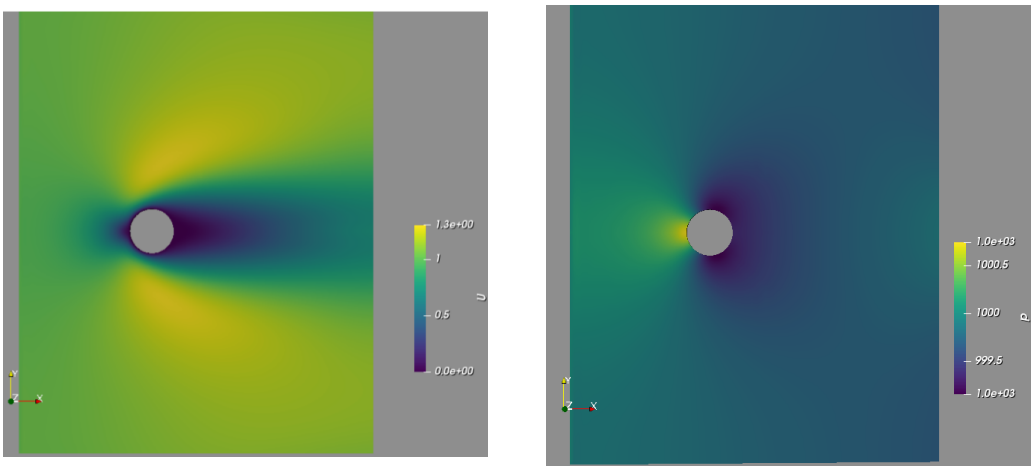
6.1 Scenario 1

The initial body is a circle of radius 0.5 with a fluid inlet velocity of 1 ms^{-1} and a Reynolds no. (*Re*) of 100. The velocity and pressure contour after first iteration is as shown in Fig. 9.

As the optimization proceeds, the body becomes more streamlined and the wetting surface (the tail part of body) elongates, which helps in reducing the pressure drag significantly. But this elongation results in increase of viscous drag of the body. Thus the optimum drag is achieved in approximately 600 iterations. The final shape and the corresponding velocity and pressure contours are as shown in Fig. 10.

6.2 Scenario 2

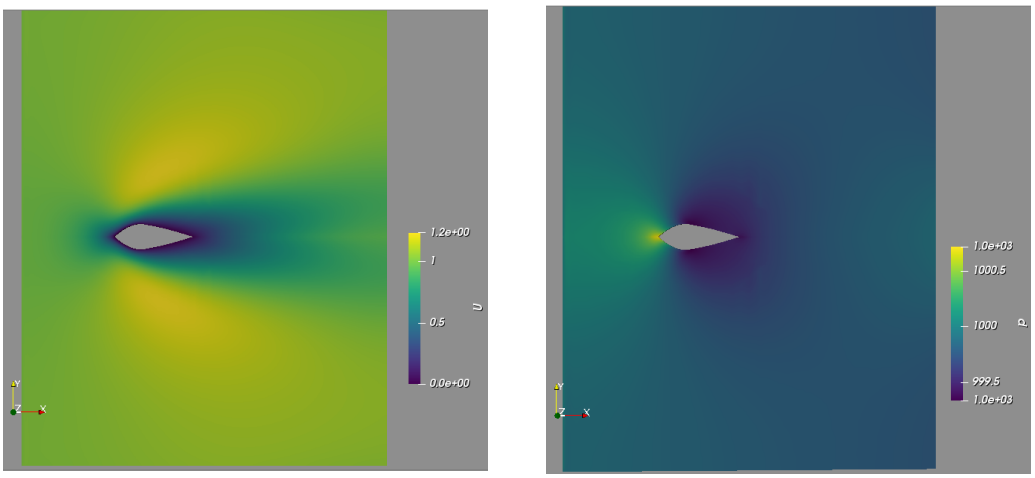
The flow parameters and initial body parameters are maintained the same as in scenario 1, except the *Re*. An *Re* of 500 is chosen here. The overall results remain similar to scenario 1 but the tail length of the converged body is shorter than that of scenario 1. The velocity contour of initial and final shapes are as shown in Fig. 11.



(a) Initial velocity contour

(b) Initial pressure contour

Fig. 9: Scenario 1: Initial velocity and pressure contours



(a) Final velocity contour

(b) Final pressure contour

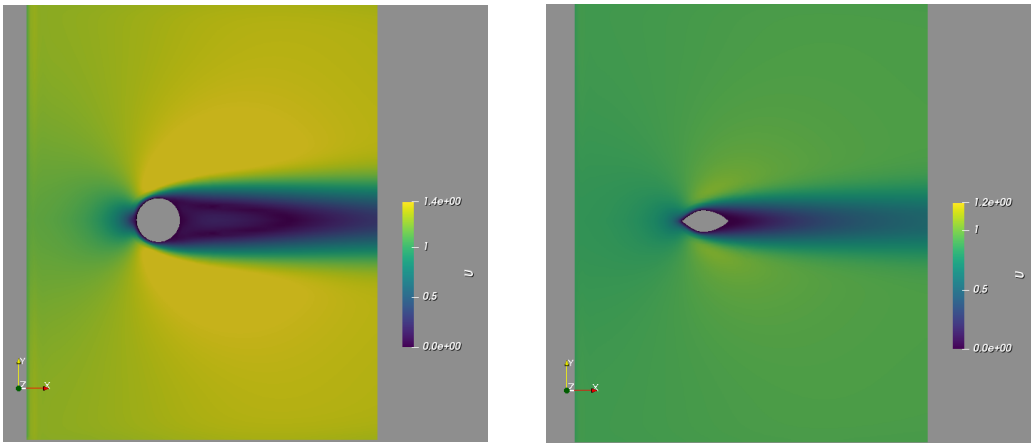
Fig. 10: Scenario 1: Final velocity and pressure contours

6.3 Scenario 3

Maintaining the same set up as the previous scenarios and increasing the *Re* further to a value of 1000, a much shorter tail length of the body is observed. The comparison figures of the initial and final shapes are as shown in Fig. 12

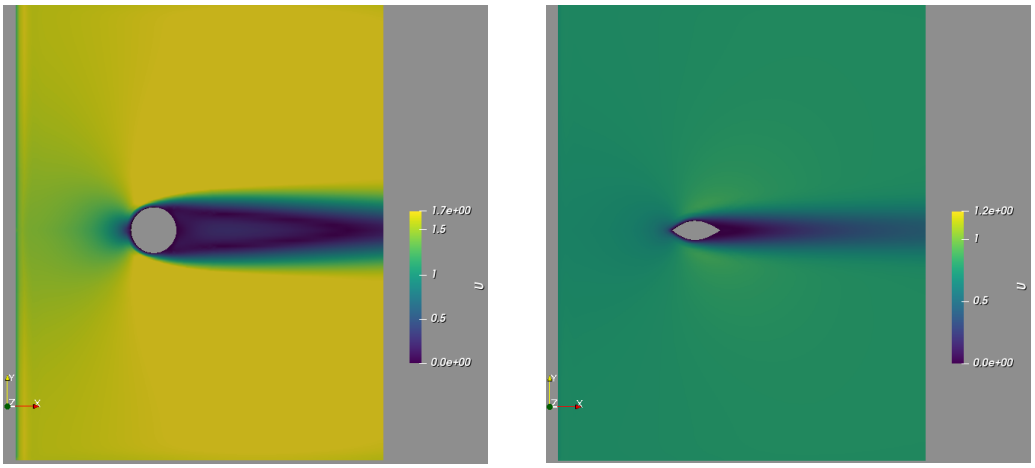
6.4 Scenario 4

Increasing the initial volume of the body further than the previous scenarios by increasing the radius of the circle to 0.75 and maintaining the inlet velocity at $1ms^{-1}$ and *Re* at 100, the drag forces of initial and converged shape are larger than the standard scenario 1. Since the viscous drag is proportional to the area exposed to the flow, the results are agreeable. The figures are shown in Fig. 13



(a) Initial velocity contour (b) Final velocity contour

Fig. 11: Scenario 2: Initial and final velocity contours



(a) Initial velocity contour (b) Final velocity contour

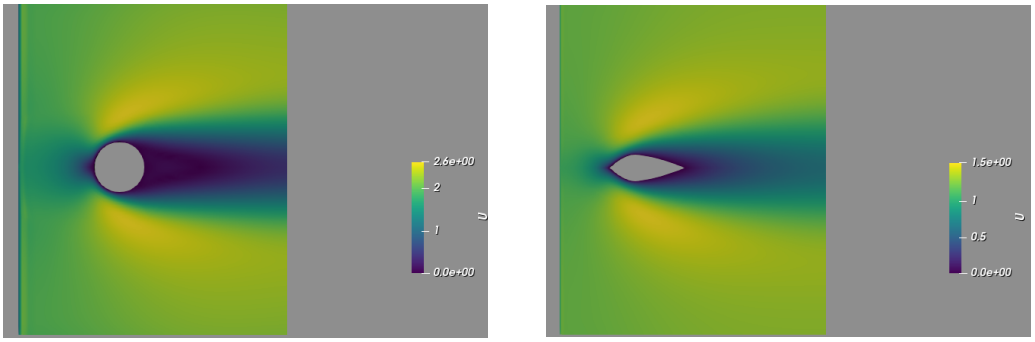
Fig. 12: Scenario 3

6.5 Scenario 5

With the initial body shape as hexagon, but maintaining the volume same as that of a circle of radius 0.5 as in scenarios 1,2 and 3 and maintaing the *Re* at 100, the converged shape resembles that of the scenario with an intial shape of circle with radius 0.5, under identical flow conditions. The velocity contours are shown in Fig. 14

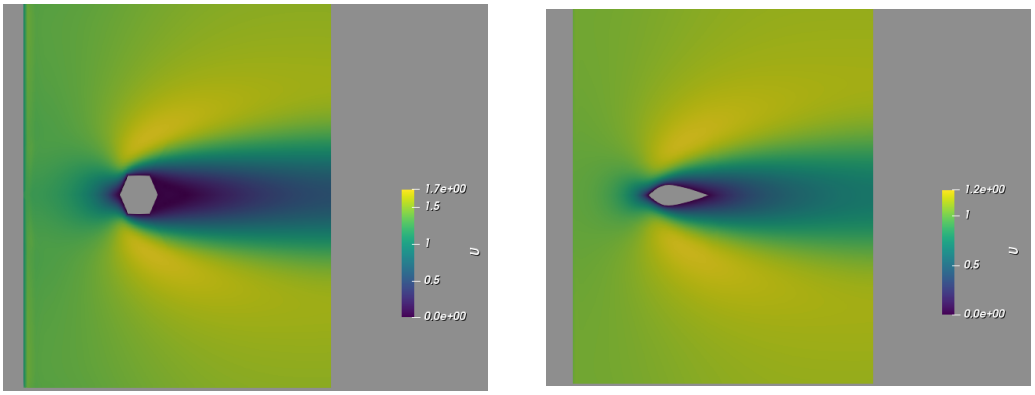
6.6 Creeping flow

The solver is also used to simulate creeping flow (Stokes flow) with $Re = 1$. Here the viscous force dominates over the pressure force. Hence, the shape needed to be optimized only to minimize viscous force. So the converged shape has an appearance with the upstream portion more rounded and with a shorter tail part as shown in Fig. 15.



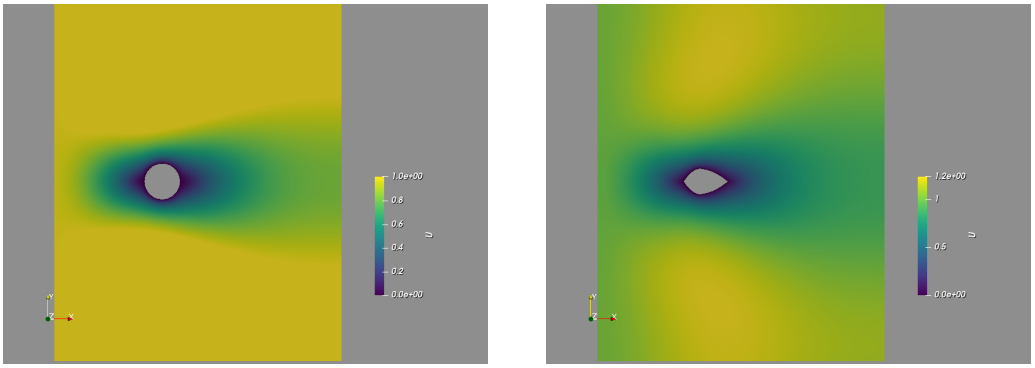
(a) Initial velocity contour (b) Final velocity contour

Fig. 13: Scenario 4: Initial and final velocity contours



(a) Initial velocity contour (b) Final velocity contour

Fig. 14: Scenario 5: Initial and final velocity contours



(a) Initial velocity contour (b) Final velocity contour

Fig. 15: Creeping flow: Initial and final velocity contours

6.7 Discussion

Comparing the variation in drag force over the iterations for different shapes and volume, the convergence is achieved between 600 and 800 iterations in all the cases as shown in Fig. 16. For circle of 0.75 radius, with larger initial volume, the drag force of initial and converged shape are also higher than the

standard case of circle of 0.5 radius. It can been seen that the initial volume affects the final convergence but not the rate of convergence.

Considering same volume for hexagon and for the circle of 0.5 radius, the converged drag is almost the same, although the convergence takes longer, as shown in Fig. 16, since the curved surface area of hexagon is more than that of circle. This proves that the initial body shape has no effect on the final convergence but only on the rate of convergence.

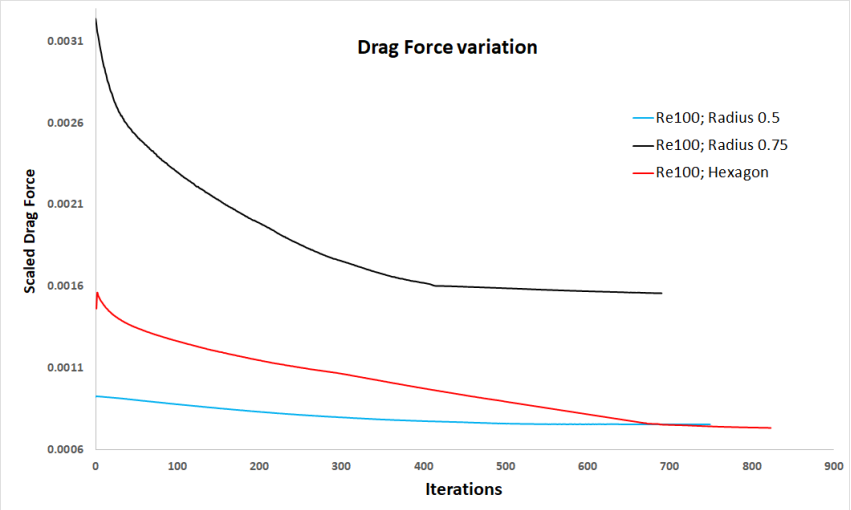


Fig. 16: Drag force variation for different initial shapes

The drag force variations for different *Re* is shown in Fig. 17. It can be seen that the drag force decreases with increasing *Re*. This is explained by the reduction in viscosity of the fluid. For higher *Re*, the viscosity reduces, so the viscous drag, and effectively reducing the overall drag force.

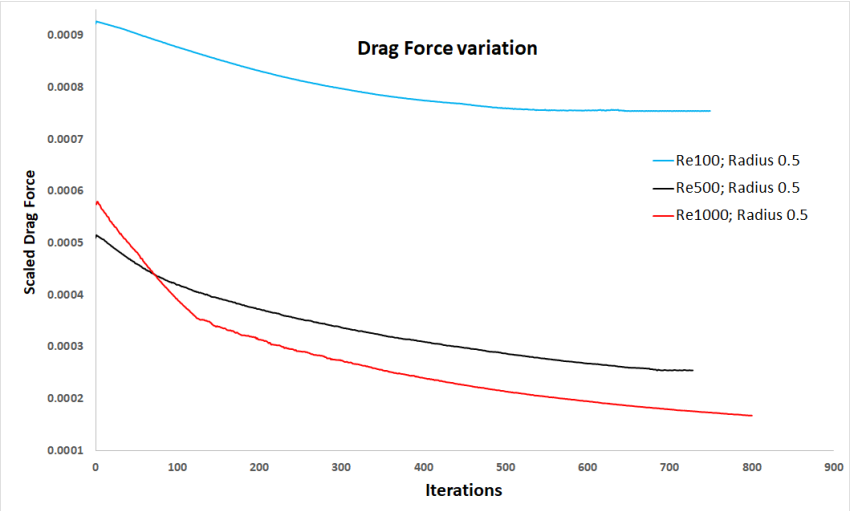


Fig. 17: Drag force variation for different Reynolds number

7 Conclusion

A formulation of shape optimization of a submarine in a two dimensional spatial domain using an incompressible Navier-Stokes equation by minimizing the drag force is presented in this paper. The objective function is formulated using adjoint method and the coordinates of new shape is calculated using weighted gradient method. Simulations are done for various scenarios. The simulation results show that a reduction in drag force of roughly 50% is achieved for circle of radius 0.75, with *Re* of 100.

The shape optimization study on different scenarios show that the final shape is dependent only on the initial volume and the Reynold's No. *Re*. All of the scenarios result in a quite similar streamlined shape and thus forms the class of shape which exhibits a minimum drag force.

References

[1] Hiroki Yoshida and Mutsuto Kawahara *Shape Optimization of an Oscillating Body in Fluid Flow by Adjoint Equation and ALE Finite Element Methods*

[2] H. Okumura, M.Kawahara *Shape Optimization of Body Located in Incompressible Navier-Stokes Flow based on Optimal Control Theory*

[3] Masato Sakamoto *Shape Optimization of A Body Located in Incompressible Flow using Automatic Differentiation*

[4] K.L Vasudev, R.sharma, and S.K Bhattacharyya *Multi-Objective shape optimization of submarine hull using genetic algorithm integrated with computational fluid dynamic*