Navier Stokes - Flow Past a Cylinder

Giorgio Daneri[1] and Elia Vaglietti[2]

Politecnico di Milano.

Contributing authors: giorgio.daneri@mail.polimi.it; elia.vaglietti@mail.polimi.it;

1 Introduction

The Navier-Stokes equations are a set of partial differential equations that describe the behavior of fluid flow. They were first derived by Claude-Louis Navier and Sir George Gabriel Stokes in the 19th century and are fundamental in fluid mechanics. The equations describe how the velocity field of a fluid evolves over time, taking into account factors such as viscosity, pressure gradients, and external forces. The Navier–Stokes equations serve as a mathematical representation of momentum balance within Newtonian fluids, incorporating the principles of mass conservation. They are derived by applying Isaac Newton's second law to fluid motion, under the assumption that the stress within the fluid comprises a diffusing viscous term (proportional to the velocity gradient) and a pressure term, thus characterizing viscous flow. In contrast to the closely related Euler equations, the Navier–Stokes equations incorporate viscosity, rendering them parabolic equations with improved analytical properties, at the cost of not being entirely integrable.

The Navier-Stokes equations find applications in various fields, including:
1. Computational fluid dynamics (CFD): numerically solving the Navier-Stokes equations to simulate fluid flow behavior in various scenarios.
2. Civil engineering: studying fluid flow in pipes, channels, and rivers for designing infrastructure such as bridges and dams.
3. Oceanography: modeling ocean currents and circulation patterns.
4. Weather forecasting: simulating atmospheric flow to predict weather patterns.
5. Mechanical engineering: designing pumps, turbines, and other fluid-handling equipment.
6. Biomedical engineering: studying blood flow in vessels and airflow in the respiratory system.
7. Aerospace engineering: understanding airflow around aircraft, missiles, and spacecraft.

2 The Problem

The general form of the non-stationary Navier-Stokes equations for an incompressible viscous fluid is:

$$\begin{cases} \frac{\partial \vec{u}}{\partial t} + (\vec{u}\cdot\nabla)\vec{u} - \nu\Delta\vec{u} + \nabla p = \vec{f} & in\ \Omega \\ \nabla\cdot\vec{u} = 0 & in\ \Omega \\ \vec{u} = \vec{g} & on\ \Gamma_D \subset \partial\Omega \\ \nu\nabla\vec{u}\vec{n} - p\vec{n} = \vec{h} & on\ \Gamma_N = \partial\Omega \setminus \Gamma_D \\ \vec{u}(t=0) = \vec{u}_{initial} & in\ \Omega \end{cases} \tag{1}$$

Where:
- $\vec{v}$ is the velocity vector field of the fluid.
- t is time.

- p is the pressure field.
- ν is the kinematic viscosity (ratio of viscosity to density).
- $\vec{f}$ represents external forces such as gravity or electromagnetic forces.

The first equation expresses the conservation of the momentum, while the second is called continuity equation and expresses the conservation of mass. Let us break down each term:
1. $\frac{\partial \mathbf{v}}{\partial t}$: this term represents the rate of change of velocity with respect to time. It accounts for how the velocity of the fluid changes over time.
2. $(\vec{v}\cdot\nabla)\vec{v}$: this term represents the convective acceleration, also known as the advective term. It describes how the velocity field advects or carries fluid particles along its path. It essentially accounts for the acceleration due to the fluid's motion itself.
3. $-\nabla p$: this term represents the pressure gradient force. It describes how pressure differences within the fluid cause it to accelerate. Pressure gradients drive fluid flow from regions of high pressure to regions of low pressure.
4. $\nu\nabla^2\vec{v}$: this term represents the viscous forces acting within the fluid. ν is the kinematic viscosity, and ∇^2 is the Laplacian operator. This term describes how the viscosity of the fluid resists the relative motion between adjacent layers of the fluid, leading to viscous dissipation and damping of fluid motion.
5. $\vec{u} = \vec{g}$: this term represents the non-homogeneous Dirichlet boundary conditions for the inlet surface, where the flow enters the mesh. When imposed on a PDE, it specifies the values that a solution needs to take along the boundary of the domain.
6. $\nu\nabla\vec{u}\vec{n} - p\vec{n} = \vec{h}$: this term represents the Neumann boundary condition for the outlet surface. It specifies the values of the derivative applied at the boundary of the domain.

2.1 The weak formulation

Let us define the spaces for velocity and pressure like follows:

- $V = \{v \in [H^1(\Omega)]^d \ s.t. \ v_{|\Gamma_D} = g\}$
- $V_0 = \{v \in [H^1(\Omega)]^d \ s.t. \ v_{|\Gamma_D} = 0\}$
- $Q = L^2(\Omega)$

Then, let us take the test function $v \in V_0$, multiply the first equation in (1) by it and integrate over Ω:

$$\int_\Omega \frac{\partial \vec{u}}{\partial t}\vec{v}d\vec{x} + \int_\Omega (\vec{u}\cdot\nabla)\vec{u}\vec{v}d\vec{x} + \int_\Omega \nu\Delta\vec{u}\vec{v}d\vec{x} + \int_\Omega \nabla p\vec{v}d\vec{x} = \int_\Omega \vec{f}\cdot\vec{v}d\vec{x} \tag{2}$$

Using Green's formulae we get:

$$\int_\Omega \frac{\partial \vec{u}}{\partial t}\vec{v}d\vec{x} + \int_\Omega [(\vec{u}\cdot\nabla)\vec{u}]\vec{v}d\vec{x} + \int_\Omega \nu\nabla\vec{u} : \nabla\vec{v}d\vec{x} - \int_\Omega p\nabla\cdot\vec{v}d\vec{x} = \int_\Omega \vec{f}\cdot\vec{v}d\vec{x} + \int_{\Gamma_N} \nu\nabla\vec{u}\vec{n}\cdot\vec{v}d\sigma - \int_{\Gamma_N} p\vec{n}\cdot\vec{v}d\sigma \tag{3}$$

The above formulation can be conveniently recast using bilinear and trilinear forms [1]. Therefore we define bilinear continuous forms $a(\cdot,\cdot) : (u,v) \in V\times V \to a(u,v) \in \mathbb{R}$ and $b(\cdot,\cdot) : (u,q) \in V\times Q \to b(u,q) \in \mathbb{R}$:

$$a(u,v) = \int_\Omega \nu\nabla\vec{u} : \nabla\vec{v}d\vec{x} = \int_\Omega \nu\sum_{i=1}^{d}\nabla\vec{u} : \nabla\vec{v}d\vec{x}$$
$$b(u,q) = \int_\Omega p\nabla\cdot\vec{v}d\vec{x}$$

Moreover we introduce a trilinear form $c(\cdot;\cdot,\cdot) : (u,v,w) \in V\times V\times V \to c(u,v,w) \in \mathbb{R}$ associated to the non-linear convective term:

$$c(w;u,v) = \int_\Omega [(\vec{w}\cdot\nabla)\vec{u}]\vec{v}d\vec{x} = \sum_{i,j=1}^{d}\int_\Omega \vec{w}_j\frac{\partial\vec{u}_i}{\partial\vec{x}_j}\vec{v}_i d\vec{x}$$

For the above trilinear form we have the following results:
PROPOSITION 1 : The trilinear form $c(\cdot;\cdot,\cdot)$ is continuous on $[\mathrm{H}^1(\Omega)]^d$. For any given $u \in V$, the map $v \to c(u,u,v)$ is linear and continuous on V.
PROPOSITION 2 : let u,v,w in $\mathrm{H}^1(\Omega)$ and assume $\nabla \vec{w} = 0$ and $\vec{w} \cdot \vec{n}_{|\partial\Omega} = 0$. Then the trilinear form $c : [\mathrm{H}^1(\Omega)]^d \to \mathbb{R}$ satisfies the following properties, which are equivalent:

$$c(w;u,v) = 0$$
$$c(w;u,v) + c(w;v,u) = 0$$

We can rewrite the problem as follows:

$$\int_\Omega \frac{\partial \vec{u}}{\partial t} \vec{v} d\vec{x} + a(\vec{u},\vec{v}) + c(\vec{u};\vec{u},\vec{v}) + b(\vec{v},p) = \underbrace{\int_\Omega \vec{f} \cdot \vec{v} d\vec{x} + \int_{\Gamma_N} \vec{h} \cdot \vec{v} d\vec{x}}_{F(\vec{v})} \tag{4}$$

$$\int_\Omega \frac{\partial \vec{u}}{\partial t} \vec{v} d\vec{x} + a(\vec{u},\vec{v}) + c(\vec{u};\vec{u},\vec{v}) + b(\vec{v},p) = F(\vec{v}) \tag{5}$$

Analogously, we multiply the second equation in (1) by another test function for pressure $q \in Q$, we integrate again over Ω and obtain:

$$\int_\Omega q \nabla \cdot \vec{u} d\vec{x} = 0 \tag{6}$$

which can be rewritten as:

$$b(\vec{u},q) = 0 \tag{7}$$

Since non-homogeneous Dirichlet boundary conditions are given for the inlet, we write $\vec{u} = \vec{u_0} + \vec{R}(\vec{g})$, where $\vec{u_0} \in V_0$ and $\vec{R}(\vec{g}) \in V$ is an arbitrary lifting function such that $\vec{R}(\vec{g}) = \vec{g}$ *on* Γ_D and $\nabla \cdot \vec{R}(\vec{g}) = 0$. We apply this reasoning to the conservation of momentum equation and obtain:

$$\int_\Omega \frac{\partial \vec{u}}{\partial t} \vec{v} d\vec{x} + a(\vec{u_0},\vec{v}) + c(\vec{u_0};\vec{u_0},\vec{v}) + b(\vec{v},p) = F(\vec{v}) - a(\vec{R}(\vec{g}),\vec{v}) - c(\vec{R}(\vec{g});\vec{R}(\vec{g}),\vec{v}) \tag{8}$$

Let's now define the residual that will be used in the weak formulation:

$$R(\vec{u_0};p,\vec{v}) = \int_\Omega \frac{\partial \vec{u}}{\partial t} \vec{v} d\vec{x} + a(\vec{u_0},\vec{v}) + c(\vec{u_0};\vec{u_0},\vec{v}) + b(\vec{v},p) - F(\vec{v}) + a(\vec{R}(\vec{g}),\vec{v}) + c(\vec{R}(\vec{g});\vec{R}(\vec{g}),\vec{v}) \tag{9}$$

We now do the same for the continuity equation:

$$\underbrace{\int_\Omega q \nabla \cdot \vec{u_0} d\vec{x}}_{b(u_0,q)} = 0 \tag{10}$$

Therefore, the weak formulation reads: $\forall t$ find $\vec{u_0} \in V_0$ and $p \in Q$, such that, for each $\vec{v} \in V_0$ and $p \in Q$, there holds:

$$\begin{cases} R(\vec{u_0};p,\vec{v}) = 0 \\ b(\vec{u_0},q) = 0 \\ \vec{u}(t=0) = \vec{u}_{initial} \end{cases} \tag{11}$$

2.1.1 Linearization of the convective term

Notice that the problem now requires to find the zero of a function. We are going to use the Newton method to accomplish this. A first approach requires the computation of the derivative of $R(\vec{u_0},p,\vec{v})$. By supposing that R admits a Frechét derivative, we can compute directional derivatives (Gâteaux derivatives) instead:

$$(DR)_u(\vec{\delta},\gamma) = \lim_{\epsilon\to 0} \frac{R(\vec{u}+\epsilon\vec{\delta}; p+\epsilon\gamma, \vec{v}) - R(\vec{u},p,\vec{v})}{\epsilon} \tag{12}$$

$$= \frac{\partial}{\partial\epsilon}\bigg|_{\epsilon=0} \left(\int_\Omega \frac{\partial(\vec{u}+\epsilon\vec{\delta})}{\partial t}\vec{v}d\vec{x} + \int_\Omega \{[(\vec{u}+\epsilon\vec{\delta})\cdot\nabla](\vec{u}+\epsilon\delta)\}\vec{v}d\vec{x} + \int_\Omega \nu\nabla(\vec{u}+\epsilon\vec{\delta}) : \nabla\vec{v}d\vec{x} - \int_\Omega (p+\epsilon\gamma)\nabla\cdot\vec{v}d\vec{x} \right)$$

$$= \int_\Omega \frac{\partial}{\partial\epsilon}\bigg|_{\epsilon=0} \frac{\partial(\vec{u}+\epsilon\vec{\delta})}{\partial t}\vec{v}d\vec{x} + \int_\Omega \frac{\partial}{\partial\epsilon}\bigg|_{\epsilon=0} \{[(\vec{u}+\epsilon\vec{\delta})\cdot\nabla](\vec{u}+\epsilon\delta)\}\vec{v}d\vec{x} + \int_\Omega \frac{\partial}{\partial\epsilon}\bigg|_{\epsilon=0} \nu\nabla(\vec{u}+\epsilon\vec{\delta}) : \nabla\vec{v}d\vec{x}$$
$$- \int_\Omega \frac{\partial}{\partial\epsilon}\bigg|_{\epsilon=0} (p+\epsilon\gamma)\nabla\cdot\vec{v}d\vec{x}$$

From which we obtain:

$$(DR)_u(\vec{\delta},\gamma) = \int_\Omega \frac{\partial\vec{\delta}}{\partial t}\vec{v}d\vec{x} + \int_\Omega [(\vec{\delta}\cdot\nabla)\vec{u}]\vec{v}d\vec{x} + \int_\Omega [(\vec{u}\cdot\nabla)\vec{\delta}]\vec{v}d\vec{x} + \int_\Omega \nu\nabla\vec{\delta} : \nabla\vec{v}d\vec{x} - \int_\Omega \gamma\nabla\cdot\vec{v}d\vec{x} \tag{13}$$

with $\vec{\delta} \in V_0, \gamma \in Q, \vec{v} \in V_0$. The Newton's method for this problem reads: given an initial guess $\vec{u}^{(0)}, p^{(0)}$, iterate for $k = 0, 1, 2, ...$ and until convergence:

1. Find initial guess $\vec{u}_0$ and p_0, set tolerance τ
2. Compute $\vec{\delta}^{(k)}, \gamma^{(k)}$ by solving the linear problem: $(DR)_u(\vec{\delta}^{(k)}, \gamma^{(k)}) = -R(\vec{u}_0, p, \vec{v})\ \ \forall \vec{v} \in V, p \in Q$
3. Update the approximation $u^{(k+1)} = u^{(k)} + \delta^{(k)}, p^{(k+1)} = p^{(k)} + \gamma^{(k)}$
4. Check the residual norm: $E_{k+1} = ||F(\vec{u}^{k+1}, p^{k+1})||$. If $E^{k+1} \leq \tau$ stop, otherwise go back to step 2.

Notice that the problem at step 2 is a linear differential problem in weak form, which can be solved by using the finite element method.

2.1.2 Finding a suitable initial guess

As mentioned earlier, the initial guess for the Newton method needs to be close enough to the solution of the system in order to obtain convergence. In such a case, the convergence order should be quadratic. When the viscosity is large, e.g. $\nu \geq \frac{1}{400}$, a suitable initial guess can be obtained by solving the Stokes equation instead. Problems arise if the viscosity is very small, corresponding to a large Reynolds number, since the nonlinear convective term $(\vec{u}\cdot\nabla)\cdot\vec{u}$ becomes dominant. The Stokes equation neglects this term, introducing an approximation which is too coarse in such cases. Though we won't work with high Reynolds numbers, meaning that the flow is laminar enough to allow the above approximation. We need to solve the system:

$$\begin{aligned} -\nu\nabla\vec{u} + \nabla p &= \vec{f} \\ -\nabla\cdot\vec{u} &= 0 \end{aligned}$$

in order to get an initial guess for (11).

2.1.3 Boundedness of the terms

Let us discuss the boundedness of the integrals at play. We employ the notation $\mathbf{H}^k(\Omega) = [\mathrm{H}^k(\Omega)]^d$ and $\mathbf{L}^p(\Omega) = [\mathrm{L}^p(\Omega)]^d$. For every function $\vec{v} \in \mathbf{H}^1(\Omega)$, we denote its norm by: [2]

$$||\vec{v}||_{\mathbf{H}^1(\Omega)} = (\sum_{k=1}^{d} ||v_k||^2_{\mathbf{H}^1(\Omega)})^{1/2} \tag{14}$$

and its seminorm by:

$$|\vec{v}|_{\mathbf{H}^1(\Omega)} = (\sum_{k=1}^{d} |v_k|^2_{\mathbf{H}^1(\Omega)})^{1/2} \tag{15}$$

The notation $||\vec{v}||_{\mathbf{L}^p(\Omega)}$, $1 \leq p \leq \infty$, has analogous meaning. By invoking Poincaré inequality, we can state that $|\vec{v}|_{\mathbf{H}^1(\Omega)}$ is equivalent to $||\vec{v}||_{\mathbf{H}^1(\Omega)}$ for all function in V, provided that the Dirichlet boundary condition has a positive measure.
All integrals involving bilinear forms are finite.

$$\left|\nu \int_\Omega \nabla\vec{u}\cdot\nabla\vec{v}d\vec{x}\right| \leq \nu|\vec{u}|_{\mathbf{H}^1(\Omega)}|\vec{v}|_{\mathbf{H}^1(\Omega)}$$
$$\left|\int_\Omega p\nabla\cdot\vec{v}d\vec{x}\right| \leq ||p||_{L^2(\Omega)}|\vec{v}|_{\mathbf{H}^1(\Omega)}$$
$$\left|\int_\Omega q\nabla\vec{u}d\vec{x}\right| \leq ||q||_{L^2(\Omega)}|\vec{u}|_{\mathbf{H}^1(\Omega)}$$

The same reasoning can be applied to the trilinear convective term, which is finite as well. We use the following result: $\forall\vec{v}\in\mathbf{H}^1(\Omega)$, then $\vec{v}\in\mathbf{L}^4(\Omega)$ and $\exists C>0$ s.t. $||\vec{v}||_{\mathbf{L}^4(\Omega)} \leq C\ ||\vec{v}||_{\mathbf{H}^1(\Omega)}$
We use the three-term Holder inequality

$$\left|\int_\Omega fgh\ d\vec{x}\right| \leq ||f||_{L^p(\Omega)}||g||_{L^q(\Omega)}||h||_{L^r(\Omega)} \tag{16}$$

which is valid $\forall\ p,q,r>1$ such that $p^{-1}+q^{-1}+r^{-1}=1$, we conclude that

$$\left|\int_\Omega[(\vec{u}\cdot\nabla)\vec{u}]\cdot\vec{v}\ d\vec{x}\right| \leq ||\nabla\vec{u}||_{\mathbf{L}^2(\Omega)}||\vec{u}||_{\mathbf{L}^4(\Omega)}||\vec{v}||_{\mathbf{L}^4(\Omega)} \leq C^2||\vec{u}||_{\mathbf{H}^1(\Omega)}||\vec{v}||_{\mathbf{H}^1(\Omega)} \tag{17}$$

Now we shall discuss the uniqueness of the solution. If $\Gamma_D = \partial\Omega$, meaning that only Dirichlet boundary conditions are present, the pressure variable appears only in terms if its gradient. For this reason, if we call $(\vec{u},p)$ a solution of (11) for any possible constant c the couple $(\vec{u},p+c)$ is a solution too, since $\nabla(p+c)=\nabla(p)$. In order to make sure that the solution is unique, we require the pressure to have null average, hence $\int_\Omega p\ d\Omega = 0$. We will then consider the following pressure space:

$$\mathrm{Q} = \mathrm{L}_0^2(\Omega) = \{p\in\mathrm{L}^2(\Omega) : \int_\Omega p\ d\Omega = 0\} \tag{18}$$

Though, in the case our problem it is sufficient to define Q as L^2, since Neumann boundary conditions are also present, i.e. $\Gamma_N \neq \emptyset$.

Introducing a finite dimensional space $V_{0,h}\subset V_0$ for velocity, $Q_h\subset Q$ for pressure, the semi-discrete Galerkin formulation reads: for all $t\in(0,T]$ find $u_{0,h}\in V_{0,h}$ and $q_h\in Q_h$ such that:

$$\begin{cases}\int_\Omega \frac{\partial\vec{u}_{0h}}{\partial t}\vec{v_h}d\vec{x} + a(\vec{u_{0h}},\vec{v_h}) + c(\vec{u}_{0h};\vec{u}_{0h},\vec{v}_h) + b(\vec{v_h},p) = F(\vec{v_h}) - a(\vec{R}(\vec{g}),\vec{v_h}) \\ b(\vec{u_{0h}},q_h) = 0 \\ \vec{u}_h(t=0) = \vec{u}_{initial,h}\end{cases} \tag{19}$$

where $\vec{u}_{inital,h}$ is the projection of the initial condition to $V_{0,h}$. Such problem is called semi-discretization of (11), as the temporal variable has not yet been discretized.

2.1.4 Stability of Galerkin Formulation

Let us introduce the LBB condition, a sufficient condition for a saddle point problem to have a unique solution that depends continuously on the input data. Given the bilinear continuous form $b(\cdot,\cdot):(u,q)\in V\times Q$, we have the following stability condition:

$$\forall q_h\in Q_h\ \exists\beta_h>0, \exists\vec{v}_h\in V_h : b(\vec{v}_h,q_h) \leq \beta_h||\vec{v_h}||_V||q_h||_Q$$

This amounts to choosing the two spaces V_h and Q_h in a compatible way, meaning that the FE velocity space is "rich enough" with respect to the FE pressure space. This ensures well-posedness and full approximation order for the FE linear problem. The LBB condition can be reformulated as a compatibility condition as follows:

$$\exists \beta_h > 0 : \inf_{q_h \in Q_h} \sup_{\vec{v_h} \in V_h} \frac{b(\vec{v_h}, q_h)}{||\vec{v_h}||_V ||q_h||_Q} \geq p_h$$

2.2 Finite Element Formulation - Semi-discrete formulation

Let us introduce a partition over Ω, and let $V_{0,h} = V_0 \cap [x_h^{r_u}(\Omega)]^d$ and $Q_h = Q \cap X_h^{r_p}(\Omega)$. Then, the semi-discrete weak formulation reads: for all $t \in (0, T)$ find $\vec{u}_{0,h} \in V_{0,h}$ and $p_h \in Q_h$ such that, for all $\vec{v}_h \in V_{0,h}$ and $q_h \in Q_h$, there holds

$$\begin{cases} \int_\Omega \frac{\partial \vec{u}_{0h}}{\partial t} \vec{v_h} d\vec{x} + a(\vec{u_{0h}}, \vec{v_h}) + c(\vec{u}_{0h}; \vec{u}_{0h}, \vec{v}_h) + b(\vec{v_h}, p) = F(\vec{v_h}) - a(\vec{R}(\vec{g}), \vec{v_h}) \\ b(\vec{u_{0h}}, q_h) = 0 \\ \vec{u}_h(t=0) = \vec{u}_{initial,h} \end{cases} \tag{20}$$

Introducing the basis function $\{\varphi_i\}_{i=1}^{N_h^u}$ of the space V_h and $\{\phi_i\}_{i=1}^{N_h^p}$ for the space Q_h, the semi-discrete formulation can be rewritten as the following system of ODEs:

$$\begin{cases} M\frac{d\vec{u}}{dt} + A\vec{u}(t) + C\vec{u}(t) + Bp(t) = \vec{F}(t) \quad \text{for } t \in (0, T] \\ B\vec{u}(t) = 0 \\ \vec{u}(0) = \vec{u}_{initial} \end{cases} \tag{21}$$

where

- $\vec{u}(t) = (U_1(t), U_2(t), ..., U_{N_h}(t))^T$ is the vector of time-dependent control variables;
- M is the mass matrix, whose entries are $M_{ij} = \int_\Omega \varphi_i \varphi_j d\vec{x}$;
- A is the stiffness matrix associated with the bilinear form $a(\vec{u}, \vec{v})$, whose entries are $A_{ij} = \int_\Omega \nu \nabla \vec{\varphi}_i : \nabla \vec{\varphi}_j d\vec{x}$;
- C is the matrix associated with the trilinear form $c(\vec{u}; \vec{u}, \vec{v})$, whose entries are $C_{ij} = \int_\Omega (\vec{\varphi}_i \cdot \nabla)\vec{\varphi}_i \vec{\varphi}_j d\vec{x}$;
- B is the matrix associated to the bilinear form $b(\vec{v}, p)$, whose entries are $B = -\int_\Omega \phi_i \nabla \cdot \vec{\varphi}_i d\vec{x}$;
- $\vec{F}(t)$ is the vector associated to the forcing term $\vec{f}$, whose entries are $\vec{F}_i(t) = \int_\Omega \vec{f} \cdot \vec{\varphi}_i d\vec{x} + \int_{\Gamma_N} \vec{h} \cdot \vec{\varphi}_i d\vec{x}$.

2.3 Theta method - fully discrete formulation

Let us introduce a partition of the time interval into N_T sub-intervals $[t^n, t^{n+1}]$ of width Δt, with $n = 0, 1, 2, ..., N_T - 1$ and $t^0 = 0, t^{N_T} = T$. We denote with a superscript n the approximation of the solution at the discrete time t^n, i.e. $\vec{u}^n = \vec{u}(t^n)$. Let $\theta \in [0, 1]$ be the parameter of the theta method. Then, the fully discrete formulation is the following:

$$\begin{cases} M\frac{\vec{\delta}}{\Delta t} + \theta A\vec{u}^{n+1} + (1-\theta)A\vec{u}^n + \theta Bp^{n+1} + (1-\theta)Bp^n = \theta\vec{F}^{n+1} + (1-\theta)\vec{F}^n \quad \text{for } n = 0, ..., N_T - 1 \\ \theta B\vec{u}^{n+1} + (1-\theta)B\vec{u}^n = 0 \\ \vec{u}^0 = \vec{u}_{initial} \end{cases} \tag{22}$$

Supposing the forward Euler method is used, which corresponds to $\theta = 0$. Substituting $\theta = 0$, the formulation reduces to

$$\begin{cases} M\,\dfrac{\vec{u}^{n+1} - \vec{u}^n}{\Delta t} + A\,\vec{u}^n + B\,p^n = \vec{F}^n, \quad n = 0, \ldots, N_T - 1, \\ B\,\vec{u}^n = 0, \\ \vec{u}^0 = \vec{u}_{\text{initial}}. \end{cases} \tag{23}$$

Our implementation features only the forward Euler method, which is an explicit discretization method to approximate the derivative of a function. It computes an approximation of the derivative using the values at the previous time step. It is fast to compute, but suffers from conditional stability. It is first order accurate in time, i.e. the local truncation error is $\mathcal{O}(\Delta t^2)$ while the global error is $\mathcal{O}(\Delta t)$. For the convective part, a CFL condition of the form

$$\Delta t \leq C_{\text{conv}} \frac{h}{\|\vec{u}\|_\infty}$$

must be satisfied, where:

- h is the characteristic spatial discretization size,
- $\|\vec{u}\|_\infty$ is a suitable norm (often the maximum value) of the velocity field,
- C_{conv} is a constant determined by the discretization (typically $C_{\text{conv}} \leq 1$).

3 Lift and drag coefficients

3.1 Theoretical definitions

A fluid flowing around an object exerts a force on it based on its shape and material properties. **Lift** is the component of this force that is perpendicular to the oncoming flow direction. It conventionally acts in an upward direction in order to counter the force of gravity (e.g. for the wings of an aircraft), but it is defined to act perpendicular to the flow and therefore can act in any direction. It does not matter if the fluid is stationary and the object is moving through it or vice versa.
Drag, sometimes referred to as fluid resistance, is a force acting opposite to the relative motion of any object, moving with respect to a surrounding fluid. This can exist between two fluid layers, two solid surfaces, or between a fluid and solid surface. Drag forces tend to decrease fluid velocity relative to the solid object in the fluid's path. Note that drag force depends on the relative velocity of the flow with respect to the object. This is because drag force is proportional to the velocity of low-speed flow, and the squared velocity for high-speed flow. This distinction between low and high-speed flow is measured by the Reynolds number.
The **lift coefficient** is a dimensionless quantity that relates the lift generated by a lifting body to the fluid density around the body, the fluid velocity and an associated reference area. A lifting body is a foil or a complete foil-bearing body such as a fixed-wing aircraft.
The **drag coefficient** is a dimensionless quantity that is used to quantify the drag or resistance of an object in a fluid environment, such as air or water. It is used in the drag equation in which a lower drag coefficient indicates the object will have less aerodynamic or hydrodynamic drag. The drag coefficient is always associated with a particular surface area.

3.2 Mathematical definitions

The proposed paper [3] presents the following mathematical definitions for the lift and drag forces acting on a cylindrical obstacle:

$$F_L = -\int_S (\rho\nu \frac{\partial v_t}{\partial n} n_x + P n_y)\, dS \tag{24}$$

$$F_D = \int_S (\rho\nu \frac{\partial v_t}{\partial n} n_y - P n_x)\, dS \tag{25}$$

where S is the surface of the cylinder, ρ is the the fluid density, ν is the fluid viscosity, v_t is the tangential velocity on the surface S, n is the normal vector of S and P is the pressure.
Definitions for the lift and drag coefficient are also provided:

$$C_L = \frac{2F_a}{\rho \bar{U}^2 D} \tag{26}$$

$$C_D = \frac{2F_w}{\rho \bar{U}^2 D} \tag{27}$$

where $\bar{U}$ is the average inlet velocity, D is the diameter of the cylinder. These will be calculated after the corresponding forces. Though the definitions of F_a and F_w are not provided, thus it is necessary to find alternative definitions of both lift and drag forces as well as the relative coefficients. We then referred to [4] for the definitions of the drag force and drag coefficient, then derived the definitions for the lift force and coefficient, since the former is the tangential component, while the latter is the perpendicular component. The *stress tensor* is defined as $T(\vec{u}, p) = \nu(\nabla\vec{u} + \nabla\vec{u}^T) - pI$. The drag force is obtained by integrating the following scalar product over the surface S:

$$F_D = \int_S \rho(T(\vec{u}, p)\vec{n}) \cdot \hat{x}\, dS \tag{28}$$

This formula yields the following definitions, which are the ones used in the actual implementation:

$$F_L = \int_S \rho((\nu(\nabla\vec{u} + \nabla\vec{u}^T) - pI)\vec{n}) \cdot \hat{y}\, dS \tag{29}$$

$$F_D = \int_S \rho((\nu(\nabla\vec{u} + \nabla\vec{u}^T) - pI)\vec{n}) \cdot \hat{x}\, dS \tag{30}$$

where $\hat{x}$ and $\hat{y}$ are the unit vectors associated with the spatial axes, which allow us to extract the two components of the force acting on the cylinder.
The coefficients are then computed as follows:

$$C_L = \frac{2F_L}{\rho\bar{U}^2 D} \tag{31}$$

$$C_D = \frac{2F_D}{\rho\bar{U}^2 D} \tag{32}$$

4 aSIMPLE Preconditioner

In order to solve the Navier Stokes equations for a system with a relatively high number of degrees of freedom (e.g. > 50000), we need the GMRES solver to be efficient and moderately cheap. For this purpose have implemented the approximated Semi-Implicit Method for Pressure Linked Equations preconditioner, aSIMPLE for short. It is an approximation of the SIMPLE preconditioner, an iterative method that solves first the momentum equation and then updates both the pressure and velocity fields to conserve the mass by the means of the continuity equation. The SIMPLE preconditioner is defined as:

$$P_{\text{SIMPLE}} = \begin{pmatrix} F & 0 \\ B & -\tilde{S} \end{pmatrix} \begin{pmatrix} I & D^{-1}B^T \\ 0 & \alpha I \end{pmatrix}$$

where D is the diagonal of F, the velocity stiffness matrix, comprising both A and C matrices defined earlier, $\alpha \in (0, 1]$ is a parameter that damps the pressure update, $\tilde{S} = BD^{-1}B^T$ is an approximation of the Schur complement. SIMPLE is efficient especially when the matrix is diagonally dominant, for which we need the discretization step Δt to be small enough.

5 Implementation Problems

During the development of the Navier-Stokes solver we found some remarkable issues:

- We developed the *aSIMPLE* preconditioner as in [5], referenced in the project description. As specified there, this preconditioner could potentially quicken the computation by leveraging the parallelism of the HPC. A preconditioner is good when it can lower the number of iterations of the solver, while not increasing the time due to its calculation. Unfortunately this is not the case in out project. Employing our implementation of the aSIMPLE, also reducing the tolerance to compute it, results in a slower convergence to a solution compared to the other preconditioners (*BlockTriangular*, *BlockDiagonal*) that we offer. Despite the intensive investigating, we still have not found the problem.
- In the Newton iteration, when the viscosity coefficient ν is small enough, the velocity stiffness matrix is composed of the viscous and convective components. This matrix, which has to be used to compute the solution of the newton iteration, is ill-conditioned, especially when the Reynolds number is high. This makes the problem hard to solve without a proper preconditioner. As explained above we tried to develop one, but unsuccessfully.
  We tried to overcome this problem by employing the Augmented Lagrangian term, with poor results. Thus, we run our simulations increasing the non-linear term gently, starting with a complete Stokes

problem for the first iteration. Doing so, the current solution of the problem was close enough to the successive one, leading the Newton method not to diverge.
Still, we have failed in reaching sufficiently high Reynolds numbers, and we have not witnessed any turbulence whatsoever. The Newton method proved to be very computationally demanding, especially the assembly of the linearized term by the means of the Fréchet derivative. But even with a cluster at our disposal, we could not overcome the computational cost. This is probably a symptom of a partially wrong mathematical formulation of the problem, which renders the implementation ineffective in reaching the desired results.

6 Results

The NS solver we developed allows the user to choose between many different running options. This feature allows the users operating the solver and us to make the data analysis to find the best combination possible.

The options mentioned below are set as flags in the command line after calling the executable file of the application.

Short cmd	Long cmd	Values	Description
`-M`	`--read-mesh-from-file`		Read mesh from file instead or generate it inside the program
`-m`	`--mesh-size`	X,Y	Set mesh size (two integers separated by a comma)
`-r`	`--reynolds`	N	Set Reynolds number (floating point value)
`-s`	`--solver`	N	Select solver (valid values: 0: GMRES, 1: FGMRES, 2: Bicgstab)
`-t`	`--tolerance`	D	Set tolerance (floating point value)
`-p`	`--preconditioner`	N	Select preconditioner (valid values: 0: blockDiagonal, 1: blockTriangular, 2: aSIMPLE)
`-h`	`--help`		Display this help message

Simulation on Unilu cluster.

Due to the limited resources that are available on the MOX cluster, we decided to opt for a bigger one, to which we have access through the University of Luxembourg. We avoided installing and compiling the dealii and trilinos libraries by leveraging Singularity, a container platform specifically designed for HPC clusters. We used the latest version of the MK modules in order to create the container. The following command allows to build a container from a given URI: `singularity pull docker://pcafrica/mk:latest`. Then we launch the container with the command `singularity run mk_{version}.sif`, which allows us to compile the files as usual. Then we execute the Singularity Image Format file by `singularity -s exec mk_version.sif /bin/bash -c 'source /u/sw/etc/profile && module load gcc-glibc dealii && mpiexec -n 128 exec_path [options]'`, which spawns 128 MPI processes to solve the system in parallel thanks to the Trilinos wrappers for MPI.

Flow past a cylinder problem

The code has been tested only on the *flow past a cylinder* problem. In 2D it consists in a rectangular mesh ($2.2m \times 0.41m$) with a circle inside (diameter of $0.1m$). The Reynold number for this type of problem is formulated as follows:

$$Re = \frac{\overline{U}D}{\nu}$$

Where $\overline{U}$ is the mean velocity, D the diameter of the circle and ν the viscosity coefficient.

6.1 Strong scalability

We decided to conduct some performance analysis tests by choosing different solver and preconditioners to solve the system. The strong scalability is a crucial metric in the HPC domain. It allows us to evaluate how well the parallel version of the code scales in terms of time over processes, executed on the same problem (same size for every run). In the first plot, we also visualize the standard deviation of the execution times, since several runs were performed for each combination of parameters. For the simulations using the *BlockDiagonal* preconditioner, we performed only a single run due to timing constraints. As cited in section 5 we encountered some issues in running the *aSimple* preconditioner. For this reason, we haven't included it in our strong scalability test. Thus, we mainly focused on the analysis of the *Block Triangular* and *Block Diagonal* preconditioners. We developed two different plots for the scalabililty using the *BlockTriangular* and the *BlockDiagonal* because we're interested in the time execution of the program, which is drastically affected by the choiche of the preconditioner. This effect can be clearly seen in Figure 1 and Figure 2.

The tests were carried on using a tolerance of 10^{-10} and a target Reynolds of 100. Notably, a problem size of $(100, 70)$ using the *BlockDiagonal* preconditioner remains stuck after one or few more iterations independently to the solver used. For this reason, the execution time data with that preconditioner are calculated over an autogenerated mesh of size $(80, 50)$, while the tests using the *BlockTriangular* are calculated over an autogenerated mesh of size $(100, 70)$.

In conclusion we can state the the best configuration to run our simulation is to use the *FGMRES* along with the *BlockTriangular* preconditioner.

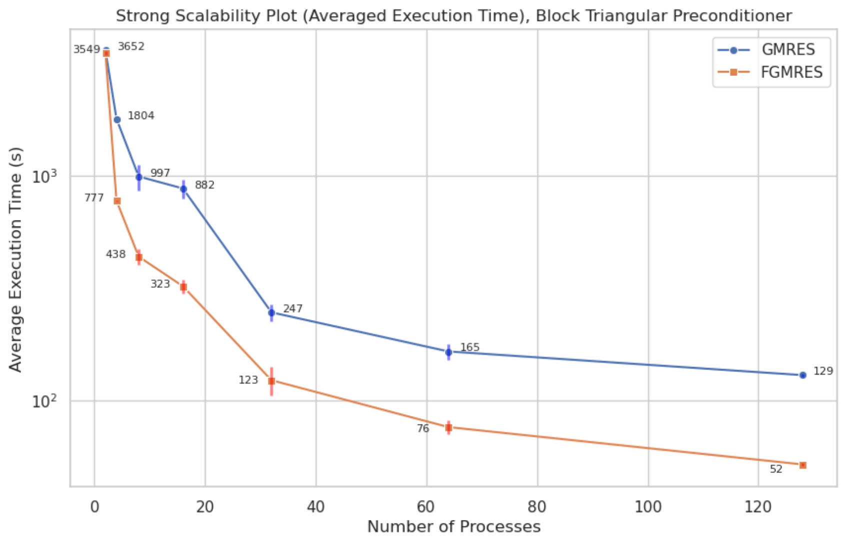


Fig. 1 Strong scalability using the Block Triangular preconditioner

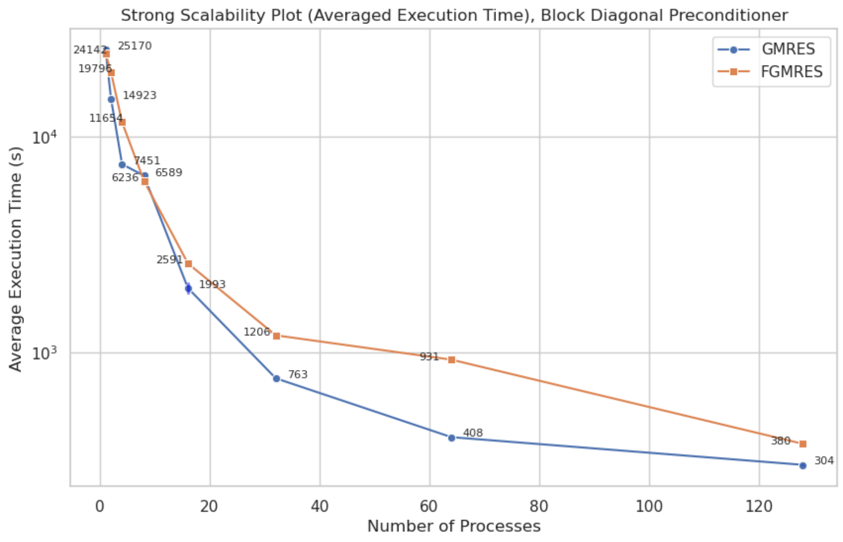


Fig. 2 Strong scalability using the Block Diagonal preconditioner

6.2 Speed up

Then, we analyzed the speed-up metrics between the *GMRES*, *FGMRES* solvers and *BlockDiagonal* and *BlockTriangular* preconditioners. This metric allows us to compare different combinations among them in a fair way, since every configuration is compared against its worst run. The results can be seen in Figure 3. The combination that scales better is unexpectedly the *GMRES, BlockDiagonal*. The shade around the lines reflects the standard deviation of the execution times.

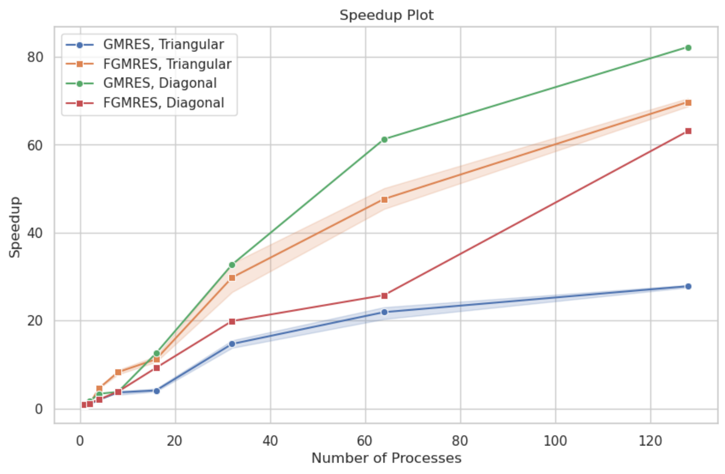


Fig. 3 Speedup of the four combination tested

6.3 Weak scalability

Another important metric is the weak scalability. A good weak scaling is achieved when the program takes the same time to execute a problem doubled in overall total size with twice the processes. The optimal weak scalability should be a straight horizontal line in a time over processes plot. Follows our weak scalability formulation:

# of Processes	Size per dim	Total Size
1	16 x 10	160
2	20 x 16	320
4	32 x 20	640
8	40 x 32	1280
16	64 x 40	2560
32	80 x 64	5120
64	128 x 80	10240
128	160 x 128	20480

Table 1 Weak scalability configuration

Again, Table 1 shows that we double the total size, as well as the processes.

As we can see in Figure 4, the weak scalability is acceptable. The increasing execution time with more cores (> 32) is probably due to the overhead in orchestrating the MPI processes. The test was carried out using the *FGMRES* solver and the *BlockTriangular* preconditioner, with a tolerance of 10^{-10}.

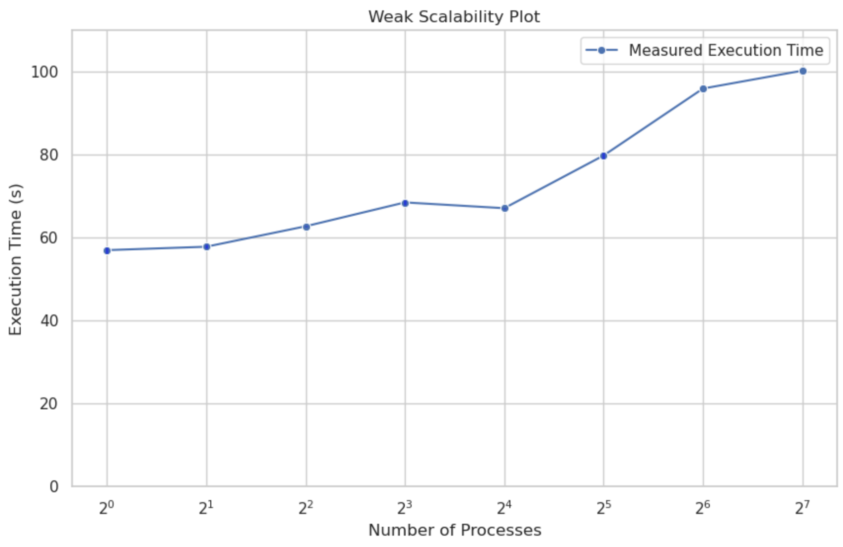


Fig. 4 Weak Scalability, *FGMRES*, *Block Triangular*, Tolerance 10^{-10}

6.4 Time Independent Navier-Stokes Solver

Finally, the visual result of our TINS solver. As mentioned in section 5, we had significant issues during the development of the application. For this reason, we simulated the "flow past a cylinder" only for a Reynold number that reaches ~ 10. After this limit, either the solution diverges or the solver does not converge. In Figure 5 and Figure 6 we can see the flow passing the cylinder both for the velocity and the pressure.

Hereafter we show only one simulation because it's the best we obtained from our runs. Other simulations with a lower Reynold's number would result in a very similar visual result or directly the solution of the Stokes problem.

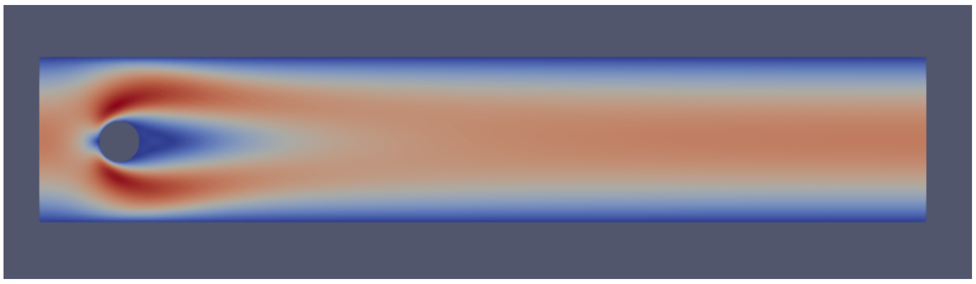


Fig. 5 Steady simulation - Velocity, Reynolds = 7

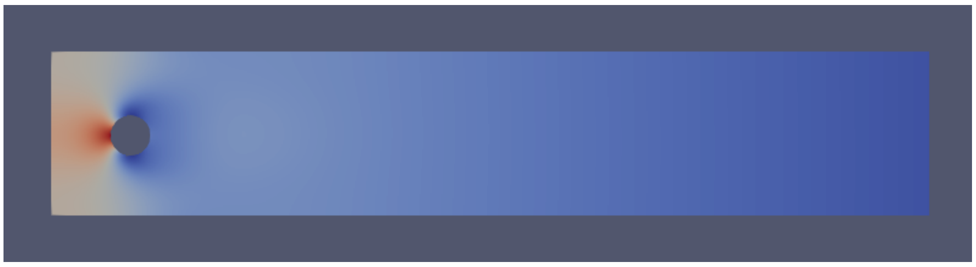
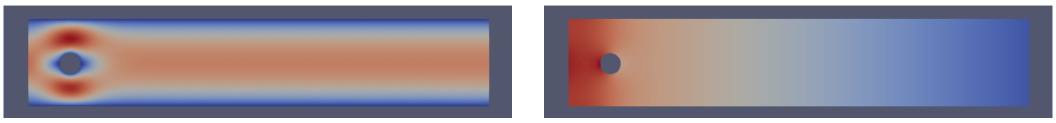


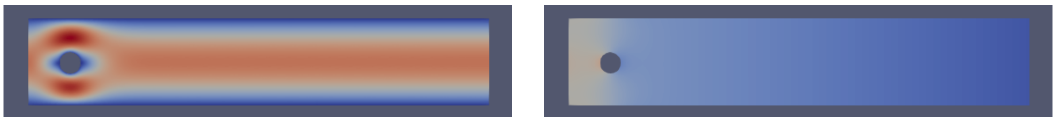
Fig. 6 Steady simulation - Pressure, Reynolds = 7

6.5 Time Dependent Navier-Stokes Solver

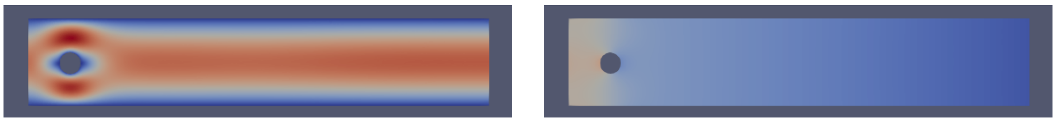
The same reasoning as above applies for the TDNS solver. As explained in subsection 2.3 we developed the forward Euler method (θ-method with $\theta = 0$). The following simulation uses a timestep of 0.001 and a Reynolds of 0.1.



$t = 1$



$t = 5$



$t = 20$

6.6 Drag and Lift results

We computed the drag and lift coefficients using a mesh with 11825 elements, 2nd order piece-wise polynomials for velocity and 1st order piece-wise polynomials for pressure, amounting to a total of 54550 degrees of freedom. The Reynolds number is 2. Unfortunately, after some initial time steps, the solver could not find a solution to the system with sufficient accuracy, thus resulting in the solution starting to diverge.

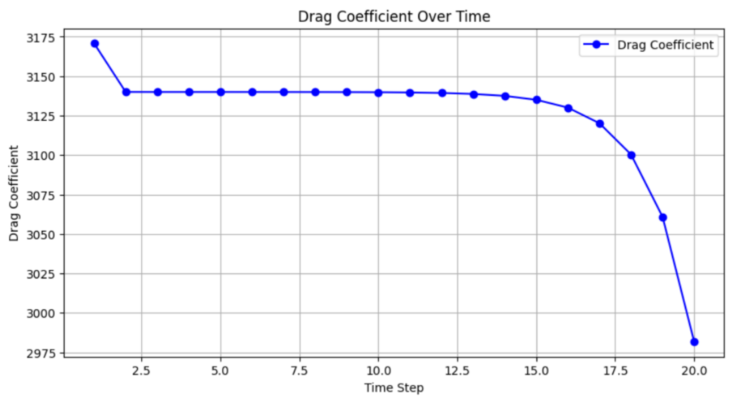


Fig. 7 Drag Coefficient Over Time

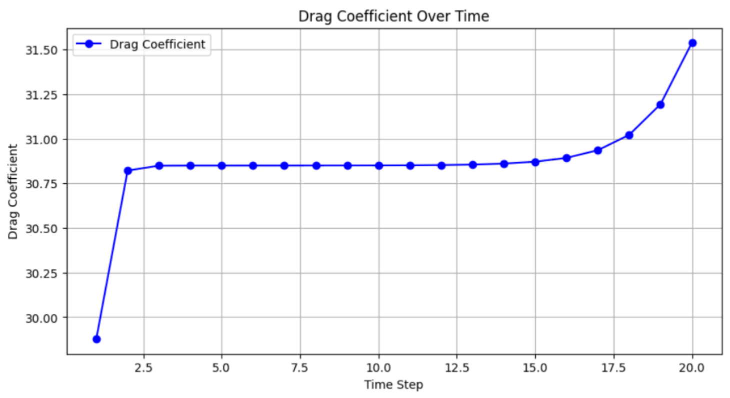


Fig. 8 Lift Coefficient Over Time

7 Conclusions

We developed a parallel Navier-Stokes simulator that solves the Time Dependent and Time Independent problems, after defining their rigorous mathematical formulations. Our implementation, that takes advantage of the linearization of the non-linear term through the Newton method, can be easily exploited using different linear algebra solvers (*GMRES*, *FGMRES*) and preconditioners (*BlockDiagonal*, *BlockTriangular*).

Unfortunately the results we achieved didn't match our expectation. As now the solver can only handle problems where the non-linear term is not dominant. Further efforts can be spent in finding a better formulation in order to solve turbulent flows.

Thanks to the HPC, we conducted a complete analysis of the simulator we built. Thus, the best configuration to use it is to employ the *FGMRES* solver coupled with the *BlockTriangular* preconditioner.

References

[1] Frey, P.: Theoretical and Numerical Issues of Incompressible Fluid Flows. https://www.ljll.fr/frey/cours/UPMC/NM491/P.%20Frey,%20chap3.pdf Accessed April 1, 2024

[2] Quarteroni, A., Quarteroni, S.: Numerical Models for Differential Problems vol. 2. Springer, Cham (2009)

[3] Schäfer, M., Turek, S., Durst, F., Krause, E., Rannacher, R.: Benchmark Computations of Laminar Flow Around a Cylinder. Springer, Wiesbaden (1996)

[4] Dede, L.: Optimal flow control for navier–stokes equations: drag minimization. International Journal for Numerical Methods in Fluids **55**(4), 347–366 (2007)

[5] Deparis, S., Grandperrin, G., Quarteroni, A.: Parallel preconditioners for the unsteady navier–stokes equations and applications to hemodynamics simulations. Computers & Fluids **92**, 253–273 (2014)