MSc in Mathematical Engineering

Course: Numerical Analysis for Partial Differential Equations

Lecturer: Prof Paola F. Antonietti
Teaching Assistant: Dr. F. Regazzoni
Tutor: Dr. C.B. Leimer Saglio

Type B project

A.Y. 2023/2024

A flexible Semi-Explicit Object-Oriented 3-D Finite-Difference algorithm Chorin-Temam based solver for Brinkman Equation

---

Authors:

Davide Galbiati,
Alessandra Gotti

Tutors:

Prof. Luca Dede',
Dr. Michele Bucelli

Abstract

This project aims to develop an efficient and parallel multi-physics solver in C++, leveraging the capabilities of the Portable, Extensible Toolkit for Scientific Computation (PETSc). The ultimate goal is to apply this solver to the Brinkman Equation to simulate blood flow through the carotid artery. Our approach is based on the Chorin-Temam method, that through a projection method solves the evolutionary incompressible Navier-Stokes equations. By making slight modifications to the algorithm, it becomes possible to address more complex flows described by the Brinkman Equation.

The Finite Difference approach was chosen since parallelization efficiency is one of the main aims of this project and a Cartesian 3D staggered grid best fits such a request. This method allows for efficient modeling of arbitrary geometries, as with sufficient refinement, the Brinkman equation can handle complex shapes without relying on a triangulated mesh (as in Finite Element Methods).

Since this method involves multiple computational steps, the code has been developed in a modular manner, enabling it to handle additional problems, such as incompressible Euler equations, Stokes equations, parabolic problems, and stationary elliptic problems. To accurately simulate blood flow within the carotid artery, pre-processing was performed using the Vascular Modeling Toolkit (VMTK).

The results were satisfactory, as we successfully developed a semi-explicit Navier-Stokes algorithm that is stable and convergent under CFL condition. Future work could focus on gaining a deeper understanding of blood flow dynamics to impose more physiologically accurate initial and boundary conditions, which would better reflect realistic scenarios.

Keywords: Chorin-Temam; Semi-Explicit, Fully Dirichlet, Brinkman equation; PETSc, VMTK, parallel implementation; Navier-Stokes; Multiphysics Solver

Contents

1 Introduction

The Navier-Stokes equations are of paramount importance in cardiovascular simulations, particularly in modeling blood flow. In our project, we focus on simulating the flow within the carotid artery, though the methodology is applicable to any geometry. Despite using a classic solver algorithm, namely the Chorin-Temam method with finite differences, which is well known to suffer from the boundary layer errors related to the splitting approach and intrinsic lower accuracy of a cartesian grid, this way we developed a highly flexible and parallel solver capable of big-scaling. Moreover, another advantage of our approach lies in the fully explicit treatment of nonlinear term, which largely enhances the computational efficiency of the solver. Finally, while PETSc is a "low-level" library that requires a steep learning curve, it offers significant flexibility in terms of customizability.

The algorithm itself remains stable as long as an appropriate CFL condition is chosen. In our simulations, we explored Reynolds numbers (Re) in the range of 1 to 2000. Convergence tests were performed at Re = 1 to avoid an overly penalizing timestep, with more accurate simulations performed using a timestep of 1e-3 and 70 refinements per spatial direction, all within an adimensional framework. This is the range within which we validated our code, ensuring its robustness and reliability.

This study contributes to the broader field of computational fluid dynamics (CFD) by demonstrating how finite differences methods, especially explicit approaches, may still present advantages, offering great scalability especially for large-scale simulations in complex geometries.

The report is structured as follows: we begin with a theoretical introduction to the Navier-Stokes equations and the Chorin-Temam projection method. This is followed by a discussion on the theoretical stability and convergence of our approach. We then explain the process of generating the carotid artery domain using the Vascular Modeling Toolkit (VMTK). Finally, we present the numerical results of our simulations, demonstrating the effectiveness of our method.

2 Governing equations

2.1 The Navier-Stokes problem

The incompressible Navier-Stokes system is considered, whose governing equations are defined on a domain $\Omega \in \mathbf{R}^d$ (with $d = (2, 3)$ the domain dimension), in the interval $(0, T)$ as follows:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} - \nu \Delta \mathbf{u} + \nabla p = \mathbf{f} \quad \text{in } \Omega \times (0, T), \tag{1}$$

$$\nabla \cdot \mathbf{u} = 0 \quad \text{in } \Omega \times (0, T), \tag{2}$$

with the following boundary and initial conditions:

$$\mathbf{u} = \mathbf{g}(\mathbf{x}, t) \quad \text{on } \partial\Omega \times [0, T) \quad \text{(Dirichlet boundary condition)}, \tag{3}$$

$$\mathbf{u}(\mathbf{x}, 0) = \mathbf{h}(\mathbf{x}) \quad \text{in } \Omega \quad \text{(initial condition)}. \tag{4}$$

where: - $\mathbf{u}$ is the velocity field, - p is the pressure (divided by density), - ν is the kinematic viscosity, - $\mathbf{f} \in L^2(\mathbf{R}^+, [L^2(\Omega)^d]$ is the external force term per unit of mass, - $\mathbf{g}(\mathbf{x}, t)$ is the prescribed velocity on the boundary $\partial\Omega$, - $\mathbf{h}(\mathbf{x})$ is the initial velocity distribution, and

By assuming fully Dirichlet boundary conditions, the continuous problem is expected to be stable. Indeed, using Gronwall's Lemma [5], it is possible to derive a viscosity-independent stability estimate for all $t \in [0, T)$,

$$\|\mathbf{u}(t)\|^2_{L^2(\Omega)} + 2\nu \int_0^t \|\nabla \mathbf{u}(s)\|^2_{L^2(\Omega)}\, ds \leq \left(\|\mathbf{u}_0\|^2_{L^2(\Omega)} + \int_0^t \|\mathbf{f}(s)\|^2_{L^2(\Omega)}\, ds \right) e^t. \tag{5}$$

Thus, the stability constant grows exponentially in time but remains independent of the viscosity. Hence the problem, analytically, is stable.

2.2 The Brinkman penalization method

In this study, internal flow within a carotid artery is modeled. Suppose the domain Ω is a parallelepiped that represents the computational region encompassing the artery. The walls of the artery are embedded within Ω and are represented as solid boundaries defining the domain O_v, where the blood flow effectively flows. The **no-slip boundary condition**, i.e., no relative motion between flow velocity and ∂O_v on the channel wall, is expressed as:

$$\mathbf{u} = 0 \quad \text{on } \partial O_v,$$

To enforce this condition, a **penalization** term is introduced, and the Brinkmann problem for a generic channel flow reads (with the same notations as before):

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} - \nu \Delta \mathbf{u} + \nabla p + \frac{1}{\eta}\chi(\mathbf{x})(\mathbf{u}) = \mathbf{f} \quad \text{in } \Omega \times (0, T), \tag{6}$$

$$\nabla \cdot \mathbf{u} = 0 \quad \text{in } \Omega \times (0, T), \tag{7}$$

$$\mathbf{u} = \mathbf{g}(\mathbf{x}, t) \quad \text{on } \partial\Omega_\nu \times [0, T) \quad \mathbf{u} = 0 \quad \text{on } \partial\Omega \setminus \partial\Omega_\nu \times [0, T) \tag{8}$$

$$\mathbf{u} = \mathbf{h}(\mathbf{x}, 0) \quad \text{in } \Omega_\nu \quad \mathbf{u} = 0 \quad \text{in } \Omega \setminus \Omega_\nu \tag{9}$$

where:

$\eta > 0 \in \mathbb{R}$ is the penalization coefficient, set at $1e - 6$ for our simulations.

$\chi(\mathbf{x}, t)$ is a characteristic function identifying the regions occupied by O_v.

The characteristic function $\chi(\mathbf{x})$ is defined as a function $\Omega \to \mathbf{R}$ such that

$$\chi(\mathbf{x}, t) = \begin{cases} 0 & \text{if } \mathbf{x} \in O_v, \\ 1 & \text{otherwise.} \end{cases}$$

It is clear that such approach is suited when cartesian domain is chosen, simplifying the handling of complex geometries, such as curved walls, implicitly enforcing the required bc's.

3 Numerical scheme

The goal is to apply a finite difference scheme to numerically discretize the incompressible evolutionary Navier-Stokes equations. For this purpose, a structured Cartesian grid, composed of uniform cells will be employed. This discretization approach leverages a staggered grid configuration, which is widely used in computational fluid dynamics for its numerical stability and accuracy.

In the staggered grid setup, the velocity components u, v, and w, corresponding to the x-, y-, and z-directions, are defined at distinct locations within each grid cell. Specifically, the u-velocity component is located on the LEFT and RIGHT faces of each cell, while the v-velocity component is positioned on the TOP and BOTTOM faces. Similarly, the w-velocity component is assigned to the FRONT and BACK faces. This arrangement ensures that the velocity field is defined at the interfaces of adjacent cells. The pressure field p, on the other hand, is defined at the CENTERS of the cells. The general scheme can be appreciated in the next two figures.

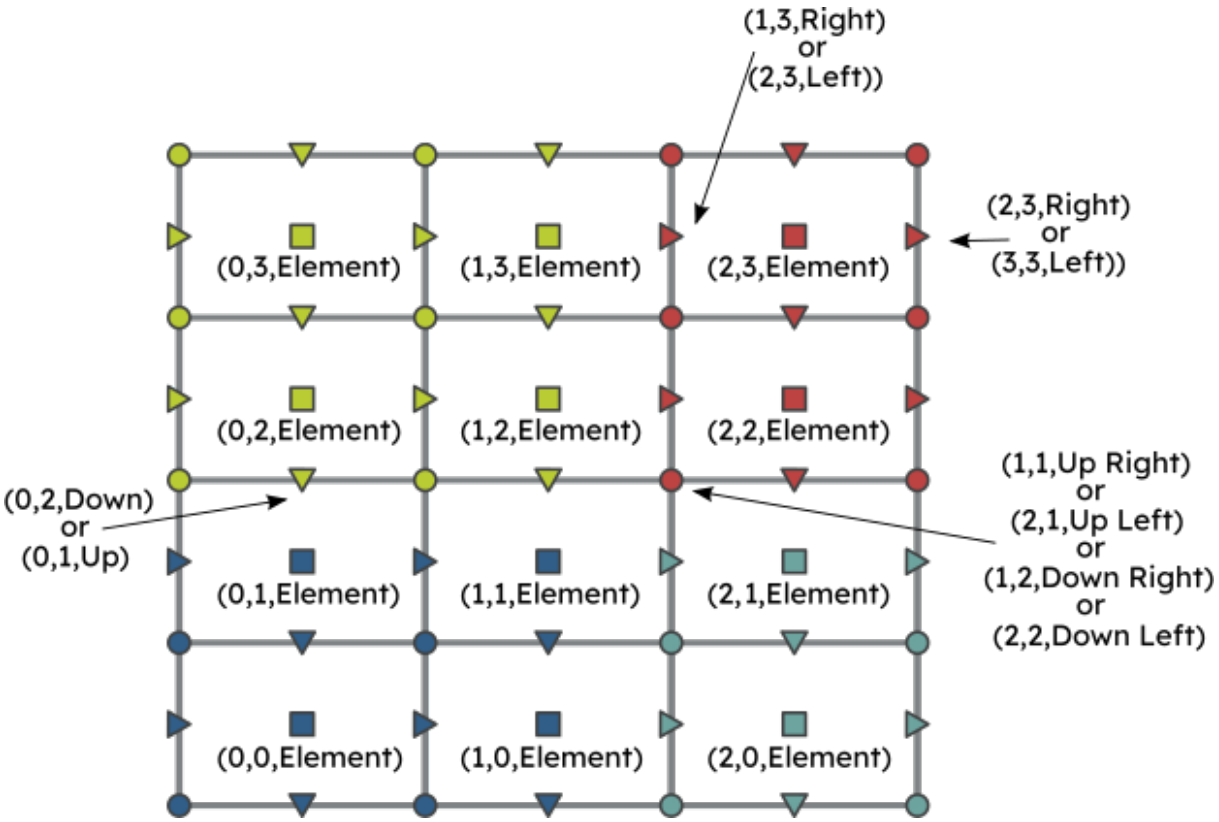

Fig. 1. PETSc DMStag object: xy-plane 2D-slice with rank division from petsc.org/main/manual/dmstag/

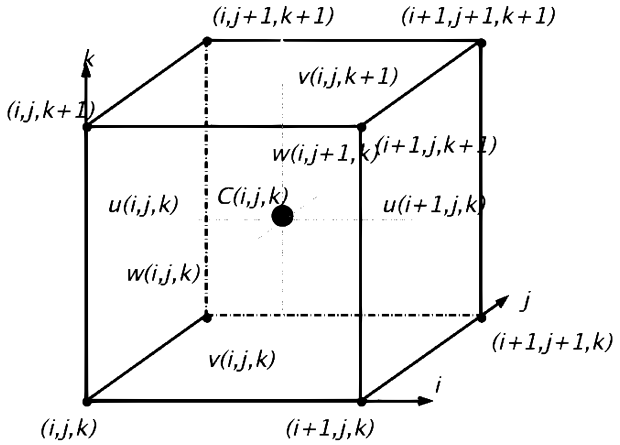

Fig. 2. PETSc DMStagobject: cell from [7]

By separating the locations of velocity and pressure variables, the staggered grid approach avoids co-locating multiple quantities at the same grid points. This design prevents numerical artifacts, such as spurious oscillations in the pressure field, which can arise in collocated grid arrangements as shown in [1]

3.1 Navier-Stokes splitting scheme: the Chorin-Teman algorithm for time discrete advancement

Given that computational efficiency is also a primary objective of this work, solving the system as a fully monolithic linear system has been discarded due to its poor scalability when large-scale problems are solved in parallel. Instead, a splitting scheme is employed to decouple the system into three separate steps, of which the most computational expensive is explicitly solved. This approach should significantly improve the computational performance while being stable, under CFL constraint (see Stability analysis).

In particular, we consider a splitting method, which effectively decouples the velocity and pressure fields. Consider a differential equation (that may be present for some differential problem) of the following type:

$$\frac{\partial w}{\partial t} + L_1 w + L_2 w = f,$$

where L_1 and L_2 represent two differential operators. A fractional step method is based on the idea of splitting the time advancement from t^n to t^{n+1}, with a dt time separataion, into two sub-problems, each handling one of the differential operators. Specifically, the system is solved in two stages as follows:

$$\frac{w_e^{n+1} - w^n}{\Delta t} + L_1 w_e^{n+1} = 0,$$

$$\frac{w^{n+1} - w_e^{n+1}}{\Delta t} + L_2 w^{n+1} = f,$$

where w_e^{n+1} is an intermediate auxiliary velocity and Δt a suitable time step.

Looking at the momentum equation (1), when diffusion-advection operator is split from incompressibility constraint, such splitting scheme is called projection method. Given a suitable time discretization of time step Δt that is $[t_0 = 0, ...t_{n-1}, t_n = T]$, the Navier-Stokes system is divided as follows (with the same notations taken before):

$$\frac{\mathbf{u}_e^{n+1} - \mathbf{u}^n}{\Delta t} - \nu \Delta \mathbf{u}_e^{n+1} + (\mathbf{u}^n \cdot \nabla)\mathbf{u}^n = 0 \quad \text{in } \Omega \times [t_1 ... T], \tag{10}$$

$$\mathbf{u}_e^{n+1} = \mathbf{g}(\mathbf{x}, t_{n+1}) \quad \text{on } \partial\Omega \times [t_0 ... T] \tag{11}$$

$$\mathbf{u}_e(\mathbf{x}, 0) = \mathbf{h}(\mathbf{x}) \quad \text{in } \Omega. \tag{12}$$

where $\mathbf{u}^n$ is the velocity field obtained from the previous time step, and

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}_e^{n+1}}{\Delta t} + \nabla p^{n+1} = \mathbf{f^n}, \quad \text{with} \quad \operatorname{div} \mathbf{u}^{n+1} = 0 \quad \text{in } \Omega \times [t_1 ... T], \tag{13}$$

$$\mathbf{u}^{n+1} = \mathbf{g}(\mathbf{x}, t_{n+1}) \cdot \mathbf{n} \quad \text{on } \partial\Omega \times [t_0 ... T] \tag{14}$$

These two sub-problems require appropriate boundary conditions. First step is an advection-diffusion problem for the intermediate velocity $\mathbf{u}_e^{n+1} \in [H^1(\Omega)]^d$, and it inherits the same boundary conditions as the original Navier-Stokes problem. In this case, fully Dirichlet bc's are considered. Second step involves projection and the solution is sought in the space $H_{\text{div}} = \{\mathbf{v} \in [L^2(\Omega)]^d \,|\, \text{div}\,\mathbf{v} = 0\}$. However, in this space, the trace of the velocity field over the boundary, $\mathbf{u}|_{\partial\Omega}$, cannot be defined. Only the trace of the normal component of the velocity, $\mathbf{u}\cdot\mathbf{n}|_{\partial\Omega}$, can be imposed on the boundary, as documented in [3]. This implies that step two can be well posed in H_{div} if $\mathbf{u}\cdot\mathbf{n}|_{\partial\Omega} = \mathbf{g}\cdot\mathbf{n}|_{\partial\Omega}$, where g is the boundary condition, is imposed. For this reason, this discrepancy is source of a splitting error inherent in the fractional step method.

By such result, the final solution can be updated as

$$\mathbf{u}^{n+1} = \mathbf{u}_e^{n+1} - \Delta t \nabla p^{n+1}, \tag{15}$$

As a matter of fact, it can be shown that by Helmholtz decomposition theorem $\mathbf{u}^{n+1}$ is the orthogonal projection of $\mathbf{u}_e^{n+1}$ in the subspace $H_{\text{div}} \subset L^2$.
The solution of second step can be efficiently formulated as an elliptic problem on pressure. Indeed, applying the divergence operator to the first equation gives:

$$\text{div}\left(\frac{\mathbf{u}^{n+1} - \mathbf{u}_e^{n+1}}{\Delta t} + \nabla p^{n+1}\right) = \text{div}(\mathbf{u}^{n+1}) - \text{div}(\mathbf{u}_e^{n+1}) + \text{div}(\nabla p^{n+1}) = 0. \tag{16}$$

This leads to an implicit and stationary Poisson problem for the pressure p^{n+1}:

$$\Delta p^{n+1} = \frac{1}{\Delta t}\text{div}(\mathbf{u}_e^{n+1}). \tag{17}$$

To derive the boundary condition for this equation, we consider the normal component of the first equation evaluated on the boundary that yelds:

$$\frac{1}{\Delta t}\left(\mathbf{u}^{n+1}\cdot\mathbf{n} - \mathbf{u}_e c^{n+1}\cdot\mathbf{n}\right) + \nabla p^{n+1}\cdot\mathbf{n} = 0 \quad \text{on } \partial\Omega. \tag{18}$$

Being both velocities equal to $\mathbf{g}\cdot\mathbf{n}|_{\partial\Omega}$, a (non-physical) homogeneous Neumann condition on the pressure arises:

$$\frac{\partial p^{n+1}}{\partial n} = 0. \tag{19}$$

In conclusion, the method that has been implemented reads:

Step 1: Update of the velocity field $\mathbf{u}^{e_1}$:

$$\frac{\mathbf{u}_{e_1} - \mathbf{u}^n}{\Delta t} + (\mathbf{u}^n\cdot\nabla)\mathbf{u}^n = 0, \quad \text{in } \Omega\times[t_1...T] \tag{20}$$

$$\mathbf{u}_{e_1} = \mathbf{g}(\mathbf{x}, t_{n+1}) \quad \text{on } \partial\Omega\times[t_0...T] \tag{21}$$

$$\mathbf{u}_e(\mathbf{x}, 0) = \mathbf{h}(\mathbf{x}) \quad \text{in } \Omega. \tag{22}$$

Step 2: Update of the velocity field $\mathbf{u}^{e_2}$:

$$\frac{\mathbf{u}_{e_2} - \mathbf{u}_{e_1}}{\Delta t} - \nu\Delta\mathbf{u}^{e_2} = \mathbf{f}, \quad \text{on } \partial\Omega\times[t_1...T], \tag{23}$$

$$\mathbf{u}_{e_2} = \mathbf{g}(\mathbf{x}, t_{n+1}) \quad \text{on } \partial\Omega \times [t_0...T] \tag{24}$$

$$\mathbf{u}_{e_2}(\mathbf{x}, 0) = \mathbf{u}_{e_1}(\mathbf{x}, 0) \quad \text{in } \Omega \tag{25}$$

Step 3: Calculation of the pressure field p^{n+1}:

$$\Delta p^{n+1} = \frac{1}{\Delta t}\text{div}\,\mathbf{u}_{e_2}, \quad \text{in } \Omega, \tag{26}$$

$$\frac{\partial p^{n+1}}{\partial \mathbf{n}} = 0, \quad \text{on } \partial\Omega, \tag{27}$$

The update finally reads:

$$\mathbf{u}^{n+1} = \mathbf{u}_{e_2} - \Delta t \nabla p^{n+1}. \tag{28}$$

For flexibility reasons, the first step has been sub-divided into two sub-steps, each accounting for the non-linear and diffusive terms. One should grasp that if this sub-division is respected in the code, also a pure convective and parabolic problem solver are naturally developed.

In order to ensure that the homogeneous Neumann problem is well-posed the *compatibility condition* must be satisfied ([3]). Consider the following Poisson problem:

$$\Delta p^{n+1} = \frac{1}{\Delta t}\text{div}\,\mathbf{u}_{e_2}, \quad \text{in } \Omega, \tag{29}$$

$$\frac{\partial p^{n+1}}{\partial \mathbf{n}} = 0, \quad \text{on } \partial\Omega, \tag{30}$$

The homogeneous Neumann boundary condition physically means no flux across the boundary.

To guarantee the existence of a solution, one must enforce a compatibility condition, derived by integrating the equation over the domain Ω:

$$\int_\Omega \Delta p^{n+1}\, d\Omega = \int_\Omega \frac{1}{\Delta t}\text{div}\,\mathbf{u}_{e_2}\, d\Omega. \tag{31}$$

Under sufficient regularity assumptions, by applying the *divergence theorem*, the left-hand side can be rewritten as a boundary integral:

$$\int_{\partial\Omega} \frac{\partial p^{n+1}}{\partial \mathbf{n}}\, dS = \int_\Omega \frac{1}{\Delta t}\text{div}\,\mathbf{u}_{e_2}\, d\Omega. \tag{32}$$

Since the Neumann boundary condition imposes that $\frac{\partial p^{n+1}}{\partial n} = 0$ on $\partial\Omega$, the left-hand side of this equation vanishes:

$$0 = \int_\Omega \frac{1}{\Delta t}\text{div}\,\mathbf{u}_{e_2}\, d\Omega. \tag{33}$$

Thus, the compatibility condition becomes:

$$\int_\Omega \operatorname{div} \mathbf{u}_{e_2}\, d\Omega = \int_{\partial\Omega} g \cdot \mathbf{n}\, d\Omega = 0. \tag{34}$$

And it can be seen that, if the boundary condition is chosen to be solenoidal, such constraint is automatically respected. This conditions is important because, looking at the term written as a flux, it imposes that the total flux across the domain should be 0, meaning that no mass is created or destroyed, namely (under incompressible hypothesis) mass is conserved. Nevertheless, such problem still admits one solution up to a constant. To fix pressure, it is sufficient to set a grid pressure node to a specific value to finally constrain the problem. In particular we chose to set a near-boundary value node to 0. It is worth remarking that doing this we give up on seeking a desired pressure field (e.g. imposing some known value). Nevertheless, the problem is well-posed and it is worth remarking that in Navier-Stokes equation only the gradient of pressure appears in the formulation and not the value of pressure itself.

To find the Brinkman counterpart in the projection method framework the algorithm has to be slightly modified as follows (for simplicity reasons we assume to impose suitable boundary conditions and initial condition on Ω, accounting for the domain "fictitious" subdivision into "flowing" and "not flowing" parts ($\Omega_\nu \quad and \quad \Omega \setminus \Omega_\nu$)

Step 1: Update of the velocity field $\mathbf{u}^{e_1}$:

$$\frac{\mathbf{u}_{e_1} - \mathbf{u}^n}{\Delta t} + (\mathbf{u}^n \cdot \nabla)\mathbf{u}^n = 0, \quad \text{in } \Omega \times [t_1 ... T] \tag{35}$$

$$\mathbf{u}_{e_1} = \mathbf{g}(\mathbf{x}, t_{n+1}) \quad \text{on } \partial\Omega \times [t_0 ... T] \tag{36}$$

$$\mathbf{u}_{e_1}(\mathbf{x}, 0) = \mathbf{h}(\mathbf{x}) \quad \text{in } \Omega. \tag{37}$$

Step 2: Update of the velocity field $\mathbf{u}^{e_2}$:

$$\frac{\mathbf{u}_{e_2} - \mathbf{u}_{e_1}}{\Delta t} - \nu\Delta\mathbf{u}^{e_2} + \frac{1}{\eta}\chi(\mathbf{x})(\mathbf{u_{e_2}}) = \mathbf{f}, \quad \text{on } \partial\Omega \times [t_1 ... T], \tag{38}$$

$$\mathbf{u}_{e_2} = \mathbf{g}(\mathbf{x}, t_{n+1}) \quad \text{on } \partial\Omega \times [t_0 ... T] \tag{39}$$

$$\mathbf{u}_{e_2}(\mathbf{x}, 0) = \mathbf{u}_{e_1}(\mathbf{x}, 0) \quad \text{in } \Omega \tag{40}$$

Step 3: Calculation of the pressure field p^{n+1}:

$$\Delta p^{n+1} = \frac{1}{\Delta t}\operatorname{div} \mathbf{u}_{e_2}, \quad \text{in } \Omega, \tag{41}$$

$$\frac{\partial p^{n+1}}{\partial \mathbf{n}} = 0, \quad \text{on } \partial\Omega, \tag{42}$$

The update finally reads:

$$\mathbf{u}^{n+1} = \mathbf{u}_{e_2} - \Delta t \nabla p^{n+1}. \tag{43}$$

with the same notation as before η is the penalization parameter. It should be highlighted that such little modification does not affect the method itself, being added just an implicit linear term.

3.2 Space discretization

As for space dscretization, as said before the domain is uniformely discretized in cells of volume $[h_x \times h_y \times h_z]$. Spatial discretization is pretty straighforward but it is worth remarking the discretization of the non linear term. Supposing a solenoidal field the convective term can be discretised on the $k-th$ cell as done in [6]

$$\mathbf{u}_k^n \cdot \nabla u_k^n = \frac{(u^n)_{k+1}^2 - (u^n)_{k-1}^2}{h_x} + \frac{(u^n v^n)_{k+1} - (u^n v^n)_{k-1}}{h_y} + \frac{(u^n w^n)_{k+1} - (u^n w^n)_{k-1}}{h_z}, \quad (44)$$

$$\mathbf{u}_k^n \cdot \nabla v_k^n = \frac{(v^n)_{k+1} - (v^n)_{k-1}}{h_y} + \frac{(v^n u^n)_{k+1} - (v^n u^n)_{k-1}}{h_x} + \frac{(v^n w^n)_{k+1} - (v^n w^n)_{k-1}}{h_z}, \quad (45)$$

$$\mathbf{u}_k^n \cdot \nabla w_k^n = \frac{(w^n)_{k+1} - (w^n)_{k-1}}{h_z} + \frac{(w^n u^n)_{k+1} - (w^n u^n)_{k-1}}{h_x} + \frac{(w^n v^n)_{k+1} - (w^n v^n)_{k-1}}{h_y}, \quad (46)$$

The nonlinear term is discretized by considering operations on edges, faces, and cell centers. Calling two discretized velocity components ξ, η, where $\xi \neq \eta$, the above terms that mix velocity components (non-homogeneous terms) can be discretely derived, interpolating the two quantities on the edges taking an average. For near boundary interpolation, the operation in performed recurring to a boundary condition value at $h/2$ distance from the "physical domain". Following this, the product is differentiated within the cell using a central difference scheme as [1]

$$\frac{1}{4h}\left((\eta(\chi) + \eta(\chi + h))\left(\xi(\chi) + \xi(\chi + h)\right) - \left(\eta(\chi) + \eta(\chi - h)\right)\left(\xi(\chi) + \xi(\chi - h)\right)\right). \quad (47)$$

In the case where $\xi = \eta$, which accounts for the first terms in the nonlinear expansion, interpolation is performed at the cell centers, and the differentiation is carried out using a central difference scheme, with the results placed back on the faces. The stencil remains identical to the one described earlier. The only distinction arises when approximating the first derivative on faces orthogonal to the component direction. To maintain second-order accuracy, a ghost value beyond the boundary condition assignment is required. For this reason, a second-order stencil using only interior nodes has been implemented, expressed as:

$$\xi'(x) \approx \frac{1}{h}\left(-2\xi(\chi \pm h) + 3\xi(\chi \pm 2h) - \xi(\chi \pm 3h)\right) \quad (48)$$

For the diffusion implicit problem 3-point stencil second order formula is used and laplacian approximation reads

$$\Delta\Phi(x, y, z) \approx \frac{u(x-h, y, z) - 2u(x, y, z) + u(x+h, y, z)}{h^2} + \frac{u(x, y-h, z) - 2u(x, y, z) + u(x, y+h, z)}{h^2} + \frac{u(x, y, z-h) - 2u(x, y, z) + u(x, y, z+h)}{h^2}$$

It is important to note that the three-point stencil is applied to every face in the domain. This choice is motivated by the fact that Dirichlet boundary conditions are imposed at a distance of

[1] Here and in all the following proofs a uniformely and equally h-spaced grid is considered for ease of notation

$h/2$ from the physical domain faces, specifically on those faces orthogonal to the ones where the unknowns are defined. On the faces parallel to the ones where the solution is defined, no ghost nodes are required, as the solution directly coincides with the prescribed boundary conditions. For example, the x-directed velocity component is assigned to the physical nodes on the LEFT and RIGHT faces, while for the UP, DOWN, FRONT, and BACK faces, the boundary condition is applied at distance $h/2$ from the physical boundary. This approach is deliberate and does not introduce any inconsistency, as the staggered grid inherently lacks nodes on the physical domain faces that are orthogonal to the ones where the solution is defined. Moreover we have remained consistent in the whole code, without recurring to "different distances" depending on the interpolation scheme.

For the Neumann problem, a second-order 3-point stencil is also applied to the cell centers (where pressure is defined). However, when computing values at the centers of cells near the boundaries, a correction is necessary to account for the Neumann homogeneous condition. In the following example, the pressure correction for the second derivative in the x-direction on the left-side cells is given by:

$$p''(x,y,z) \approx \frac{p(x-h,y,z) - 2p(x,y,z) + p(x+h,y,z)}{h^2} \tag{49}$$

$$\frac{p(x,y,z) - p(x-h,y,z)}{h} = 0 \tag{50}$$

thus providing the following correction:

$$p''(x,y,z) = \frac{-p(x,y,z) + p(x+h,y,z)}{h^2}$$

4 Stability of the Scheme

The stability of the Chorin-Temam algorithm can be analyzed by considering two distinct subproblems: the fully explicit convective problem and the implicit parabolic one. It is evident that the stability constraint is primarily dictated by the explicit component; however, for completeness, we analyze the stability constraints associated with each step separately.
For the convective equation, the main constraint, as in the case of transport problems, is given by the CFL (Courant-Friedrichs-Levy) condition, on the time step, as claimed in [5], which is expressed as follows:

$$\Delta t \leq C \frac{h}{\|\mathbf{u}\|_\infty}, \tag{51}$$

where h is the spatial grid size and $\|\mathbf{u}\|_\infty$ represents the maximum velocity of the flow within the computational domain and C a positive constant. The CFL condition ensures that the numerical solution does not propagate too rapidly, which could lead to numerical instabilities such as oscillatory instability or the uncontrolled growth of numerical errors. Furthermore, it should be noted that the explicit nature of this step demands careful attention when dealing with high-velocity flows or fine spatial grids.
For the parabolic problem (Brinkman version), the linearity properties of the governing equations allows us to employ von Neumann analysis to establish stability for the chosen grid. While simpler proofs could be constructed for this specific problem, von Neumann analysis is particularly suited to finite difference frameworks. It highlights the fact that vector fields are defined at grid

nodes and provides a rigorous method for assessing numerical scheme stability. This approach evaluates the amplification of Fourier modes in the error, ensuring that no spurious error growth occurs over successive time steps.

The analysis begins by examining the difference between the numerical solution and its discretized analytical counterpart, leading to an equation that governs the error. Let $e_k = u_k - u_{ex,k}$ represent the error at grid point k. Substituting this definition into the governing equation results in an expression for the error evolution:

$$\hat{e_k} - \tilde{e_k} = \frac{1}{\nu} \frac{\hat{e_{k+1}} - 2\hat{e}_k + \hat{e_{k-1}}}{h^2} - \frac{\chi}{\eta}\hat{e}. \tag{52}$$

This equation forms the foundation for analyzing the stability of the numerical scheme. Here, $\hat{e_k}$ and $\tilde{e_k}$ represent the error values at different time levels.

Assuming the error function is integrable in the space-frequency space, its Inverse-Fourier-Transform reads:

$$e(x,t) = \int_{-\infty}^{\infty} E_k(t) e^{ikx}\, dk \tag{53}$$

,
where $E_k(t)$ is an unknown function that determines the evolution of the error. Substituting and using the linearity of the problem, for the integrands the following equation must be valid:

$$\frac{\hat{e_k}}{\tilde{e_k}} = \frac{1}{1 + 4\frac{h^2}{\nu} sin^2(\frac{kh}{2}) + \frac{1}{\eta}\chi} \tag{54}$$

Define the amplification factor

$$G \equiv \frac{\hat{e_k}}{\tilde{e_k}} \tag{55}$$

The condition for the error to remain bounded is that

$$|G| \leq 1. \tag{56}$$

that is clearly always satisfied for all k. This simple demonstration proves the unconditional stability of the implicit parabolic problem in the present FD-framework.

5 Convergence of the Scheme

Assuming that (u, p), the solution to the projection method, is sufficiently regular, the following convergence estimates in time are satisfied (as shown in [4] and [5])

$$\|u^n - u_{\text{ex}}\|_{L^2(\Omega)^d} \leq C(u, p, T)\Delta t,$$
$$\|p^n - p_{\text{ex}}\|_{L^2(\Omega)^d} \leq C(u, p, T)\Delta t^{1/2}.$$

The introduction of the artificial Neumann boundary condition on pressure leads to a numerical boundary layer that restricts the scheme from achieving full first-order accuracy for pressure in the L^2 norm. In this sense, the scheme exhibits an irreducible splitting error of order $O(\Delta t^{1/2})$.

In the dedicated section, we are going to numerically retrieve such results.

As for spatial convergence, what follows is a complete spatial convergence analysis to prove second-order convergence in space of the method. Unlike the time convergence, which is well-documented, we could not find any existing proof for the spatial convergence on a 3D-staggered grid. Therefore, we have derived the full proof for this case.

First, the focus is on non-linear term. Next proof is suited to take into account any non-linear contribute that appears in advection equation. As a matter of fact, with the exam same notation the following (interpolated)-quantities may be defined over cell-centres, for homogeneous non-linear terms, or over cell-edges, for non-homogeneous non-linear terms. For two different velocity components, say (η, ξ), interpolation products on some cell reads:

$$\left(\frac{\eta(\chi)+\eta(\chi+h)}{2}\right)\cdot\left(\frac{\xi(\chi)+\xi(\chi+h)}{2}\right)=\frac{1}{4}\left(\eta(\chi)+\eta(\chi+h)\right)\left(\xi(\chi)+\xi(\chi+h)\right)$$

And with the same notation also

$$\left(\frac{\eta(\chi)+\eta(\chi-h)}{2}\right)\cdot\left(\frac{\xi(\chi)+\xi(\chi-h)}{2}\right)=\frac{1}{4}\left(\eta(\chi)+\eta(\chi-h)\right)\left(\xi(\chi)+\xi(\chi-h)\right)$$

is computed.

Then, χ-directed derivative derivative is computed with a central difference approach:

$$\frac{1}{4h}\left(\left(\eta(\chi)+\eta(\chi+h)\right)\left(\xi(\chi)+\xi(\chi+h)\right)-\left(\eta(\chi)+\eta(\chi-h)\right)\left(\xi(\chi)+\xi(\chi-h)\right)\right)$$

Now, Taylor-expanding the above result up to second order (for ease of notation $O(n)$ has been neglected), it follows that

$$\begin{aligned}\frac{1}{4h}\Bigg(&\left(\eta(\chi)+\left(\eta(\chi)+h\eta'(\chi)+\frac{h^2}{2}\eta''(\chi)\right)\right)\cdot\left(\xi(\chi)+\left(\xi(\chi)+h\xi'(\chi)+\frac{h^2}{2}\xi''(\chi)\right)\right)\\&-\left(\eta(\chi)+\left(\eta(\chi)-h\eta'(\chi)+\frac{h^2}{2}\eta''(\chi)\right)\right)\cdot\left(\xi(\chi)+\left(\xi(\chi)-h\xi'(\chi)+\frac{h^2}{2}\xi''(\chi)\right)\right)\Bigg)\end{aligned}$$

This result of computation reads

$$(\eta\xi'+\eta'\xi)+\frac{1}{4h}\cdot h^3(\eta'\xi''+\eta''\xi')+O(h^3)$$

Taking the difference between such result and its analytical (discretized) couterpart obviously we get that

$$e(\chi)=((\eta\xi)'-\eta\xi)'_{ex})=-\frac{1}{4h}\cdot h^3(\eta'\xi''+\eta''\xi')+O(h^3) \tag{57}$$

that is also

$$|e(\chi)| \leq \frac{1}{4} \cdot h^2(|\eta'\xi'' + \eta''\xi'|)$$

This proves second order convergence for the non-linear staggered stencils. As said before, for homogeneous terms ($\eta = \xi$) the same proof works.
The only remaining stencil to guarantee is the 3-point stencil, whose structure is well known to be second order. Here a full proof of the fact for a generic quantity ξ in direction χ, may it be defined on cell-faces or cell-centers:

$$\xi''(\chi) = \frac{\xi(\chi - h) - 2\xi(\chi) + \xi(\chi + h)}{h^2} \tag{58}$$

Let us consider the Taylor expansion of $\xi(\chi + h)$ (neglecting $O(n)$:

$$\xi(\chi + h) = \xi(\chi) + h\xi'(\chi) + \frac{h^2}{2}\xi''(\chi) + \frac{h^3}{6}\xi'''(\chi) + \frac{h^4}{24}\xi^{(4)}(\chi),$$

and $\xi(\chi - h)$:

$$\xi(\chi - h) = \xi(\chi) - h\xi'(\chi) + \frac{h^2}{2}\xi''(\chi) - \frac{h^3}{6}\xi'''(\chi) + \frac{h^4}{24}\xi^{(4)}(\chi).$$

Substituting these two expansions into 3-point stencil we get

$$\frac{1}{h^2}(h^2\xi''(\chi) + \frac{h^4}{12}\left(\xi^{(4)}(\chi) + \xi^{(4)}(\chi)\right) + O(h^3).$$

The error is given by:

$$e(\chi) = \xi''(\chi) - \frac{1}{h^2}(h^2\xi''(\chi) + \frac{h^4}{12}\left(\xi^{(4)}(\chi) + \xi^{(4)}(\chi)\right) + O(h^3)$$

Thus,

$$|e(\chi)| \leq \frac{h^2}{6}|\xi^{(4)}(\chi)|.$$

6 Solving the algebraic systems and preconditioning

Before introducing the structure of the matrices used in solving the implicit problems, it is important to discuss the "different" construction method employed by PETSc. In fact, the way in which PETSc allocates the matrix entries is relatively complex, but it is also highly efficient and user-friendly. Each element is set using the following implementation:

```
...
col[idx].i = ex;
col[idx].j = ey;
col[idx].k = ez;
col[idx].loc = CELL_POSITION;
col[0].c = 0;
valA[0] = //stencil value
...
```

In this way, the structure of the grid is naturally embedded into the matrix, and the library automatically handles the inherent structure of the matrix based on any given data structure (DMStag object). This approach significantly simplifies the implementation of complex grid operations and ensures that the computational efficiency is maintained.
For the parabolic problem a grid in a single direction with n points and grid spacing h, the 1D discrete parabolic problem matrix is given by:

$$L = \frac{1}{h^2}\begin{bmatrix} val & 1 & 0 & \cdots & 0 \\ 1 & val & 1 & \cdots & 0 \\ 0 & 1 & val & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1(\text{boundary node}) \\ 0 & 0 & 0 & \cdots & val \end{bmatrix}$$

This is a tridiagonal matrix with $val = -2 - \frac{h^2 Re}{dt} - \frac{h^2 Re\chi}{\eta}$ on the main diagonal and 1 on the two adjacent diagonals. The matrix is scaled by $\frac{1}{h^2}$ to account for the grid spacing h. It is important to highlight the structure of such matrix. The first row handles the stencil modification near the boundary of the domain, where a ghost node would typically be required. Instead, the boundary condition is directly applied at the right-hand side of the matrix. Similarly, the second-to-last row incorporates the boundary condition and stores an identity row with a 1 on the diagonal. This approach, that enlightens the structure of a staggered grid, ensures that the boundary condition is incorporated into the solution. For instance, in the case of the x-component, the solution will directly include the boundary condition values at the LEFT and RIGHT positions.
For the 3D discrete Laplacian with different spacings h_x, h_y, and h_z in the three directions, the Laplacian can be constructed using the Kronecker product. Specifically, the total 3D Laplacian is the sum of the Laplacians in the three directions, each of which should be computed separately and then combined.
For each direction, the 1D matrix is:

In the x-direction: The matrix L_x of size $n_x \times n_x$, scaled by $\frac{1}{h_x^2}$,

In the y-direction: The matrix L_y of size $n_y \times n_y$, scaled by $\frac{1}{h_y^2}$,

In the z-direction: The matrix L_z of size $n_z \times n_z$, scaled by $\frac{1}{h_z^2}$.

The total 3D Laplacian is then obtained by combining the parabolic matrices in the three directions using the Kronecker product:

$$L_{3D} = L_x \otimes I_y \otimes I_z + I_x \otimes L_y \otimes I_z + I_x \otimes I_y \otimes L_z$$

where I_x, I_y, and I_z are identity matrices of dimensions n_x, n_y, and n_z, respectively.
The final result is a matrix with a strong diagonal dominance. Finally, even if such matrix preserves a kind of "regularity" being diagonally dominated, it is far to be symmetric as the rows [...0...1...] introduced to account for boundary conditions break such property.
For a grid along a single direction with n points and spacing h, the 1D Laplacian matrix with Neumann boundary conditions is given by:

$$L = \frac{1}{h^2}\begin{bmatrix} -1 & 1 & 0 & 0 & \cdots & 0 \\ 1 & -2 & 1 & 0 & \cdots & 0 \\ 0 & 1 & -2 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & 0 & \cdots & 1 & -1 \end{bmatrix}_{n\times n}$$

This matrix represents a discretization of the 1D Laplacian based on the stencil $1, -2, 1$ with Neumann boundary conditions. The first and last rows of the matrix are modified to enforce the boundary conditions of zero flux at the edges. Specifically, the first row contains -1 and 1 (for the left Neumann boundary), and the last row contains 1 and -1 (for the right Neumann boundary).
To construct the 3D Laplacian structure, one considers the 1D Laplacians for each direction as before, obtaining:

$$L_{3D} = L_x \otimes I_y \otimes I_z + I_x \otimes L_y \otimes I_z + I_x \otimes I_y \otimes L_z$$

Where I_x, I_y, and I_z are identity matrices for each direction. The final result is a matrix that exhibits strong diagonal dominance. While this matrix could, in principle, be symmetric, a modification was made to constrain the problem and ensure that the pressure is defined up to a constant. To achieve this, one pressure node was fixed, thereby breaking the symmetry of the matrix.
Regarding the solution of both linear systems, we have chosen to employ iterative methods due to several advantages they offer. Iterative methods, particularly when implemented with libraries such as PETSc, are well-suited for large, sparse systems as they allow for efficient memory usage and can be parallelized, leading to significant performance improvements in high-dimensional problems. Given that the matrix does not possess symmetry, we opt for the GMRES (Generalized Minimal Residual) method. GMRES is Krilov-based robust iterative solver well-suited for non-Hermitian systems, providing an efficient approach for problems of this nature.
To understand the foundations of Krylov methods for the iterative solution of linear systems, it is essential to revisit some fundamental results from linear algebra. These results form the theoretical basis for iterative approaches ([5]).
Let $A \in \mathbb{R}^{N\times N}$ be a square matrix. Its characteristic polynomial is defined as:

$$p_N(\lambda) = \det(A - \lambda I) = \sum_{i=0}^{N} c_i \lambda^i,$$

where $c_0 = \det(A)$ represents the determinant of A.
A key result, the *Cayley-Hamilton theorem*, states that any square matrix satisfies its own characteristic polynomial:

$$p_N(A) = 0, \quad \forall A \in \mathbb{R}^{N\times N}.$$

If A is invertible (as is our problems), its inverse A^{-1} can be expressed as a polynomial of degree $N-1$ in A. This follows directly from the Cayley-Hamilton theorem. Specifically, we have:

$$p_N(A) = 0 \implies A^{-1}p_N(A) = A^{-1}\det(A) + \sum_{i=1}^{N} c_i A^{i-1} = 0,$$

which leads to the expression:

$$A^{-1} = -\frac{1}{\det(A)} \sum_{i=1}^{N-1} c_{i+1} A^i = p_{N-1}(A).$$

Now, consider the linear system $AU = F$ with an initial guess solution U_0. Let the initial residual be defined as $R_0 = F - AU_0$ and the initial error as $E_0 = U - U_0$. Then, the error can be written as:

$$E_0 = A^{-1}R_0 = -\frac{1}{\det(A)} \sum_{i=0}^{N-1} c_{i+1}A^i R_0.$$

The *Krylov space* generated by the matrix A starting from the initial vector R_0 is defined as:

$$K_N(A, R_0) = \text{span}\{R_0, AR_0, \ldots, A^{N-1}R_0\}.$$

It is important to note that $E_0 \in K_N(A, R_0)$, which implies that the initial error lies within the Krylov space.
Krylov methods, such as GMRES, construct a sequence of approximate solutions U_k such that, for each iteration $k = 1, 2, \ldots$:

$$U_k - U_0 \in K_k(A, R_0),$$

or equivalently:

$$R_k - R_0 = A(U_k - U_0) \in \text{span}\{AR_0, A^2R_0, \ldots, A^kR_0\}.$$

The residual R_k can be explicitly written as:

$$R_k = R_0 + \sum_{i=1}^{k} c_i A^i R_0 = p_k(A)R_0,$$

where $p_k(A)$ is a polynomial of degree at most k.
Minimizing the residual R_k in the Euclidean norm $\|\cdot\|$, corresponds to solving the following minimization problem:

$$\|R_k\| = \min_{p_k \in \mathscr{P}_k} \|p_k(A)R_0\|,$$

where $\mathscr{P}_k$ denotes the space of polynomials of degree at most k.
The GMRES algorithm, designed to minimize the norm of the residual R_k at each iteration, leverages this property. The idea is that as iterations increase, the Krilov space is richer and richer, hence a more constrained and smaller residual is found. A critical theoretical result is that, since $p_N(A) = 0$ (by the Cayley-Hamilton theorem), GMRES is guaranteed to find the exact solution in at most N steps when implemented in exact arithmetic. However, in practice, GMRES often converges with a desired tolerance in significantly fewer iterations.
It can be shown ([5]) that

$$\|R_k\| \leq K(A) \min_{p_k \in P_k} \max_i |p_k(\lambda_i)| \, \|R_0\|,$$

$K(A)$ is the condition number of matrix A, p_k is the polynomial of degree k used for the approximation and λ_i are the eigenvalues of the operator.
Thus, the convergence is closely related to the behavior of the polynomial approximation over the spectrum of the operator and to the condition number. For example, if the matrix operator is ill-conditioned and its eigenvalues are very spread a higher condition number will arise and a higher degree polynomial will be needed to lower $minmax|...|$ under a desired tolerance. Convergence in this case will not be fast. This issue could be solved by using proper preconditioning to solve a more regular system.
Among several tries, we found out that for the parabolic problem the diagonal preconditioner

performed the best, while better performances where obtained for the Neumann problem with the block-diagonal preconditioner.

For the block-diagonal preconditioner the matrix A is typically partitioned into blocks:

$$A = \begin{pmatrix} A_{11} & A_{12} & \dots & A_{1k} \\ A_{21} & A_{22} & \dots & A_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ A_{k1} & A_{k2} & \dots & A_{kk} \end{pmatrix}$$

The preconditioner M is then constructed as a block diagonal matrix, consisting of the diagonal blocks of A:

$$M = \text{diag}(A_{11}, A_{22}, \dots, A_{kk}).$$

The preconditioner is applied to the system by multiplying both sides of the original equation by M^{-1}, resulting in a transformed system:

$$M^{-1}AU = M^{-1}F.$$

This transformed system is then solved iteratively by GMRES. As said before the application of the preconditioner improves the spectral properties of the matrix making it less ill-conditioned.

7 Preprocessing of the domain

To prepare the carotid artery geometry for Navier-Stokes simulations, we performed a preprocessing workflow aimed at enhancing the simulation domain and ensuring accurate computational results. This involved multiple steps, leveraging the Vascular Modelling Toolkit (VMTK) to manipulate and improve the surface mesh.

First, the STL file representing the carotid artery was converted into the VTP format to use the functions provided by VMTK. Once in VTP format, we computed the centerlines of the geometry using the ‘vmtkcenterlines‘ function with the ‘openprofiles‘ seed selector. This step was critical to define the inlet and outlet sections of the geometry. The centerlines provide a clear reference for the orientation and alignment of the model. The command for centerlines is:

```
vmtkcenterlines -seedselector openprofiles -ifile carotid.vtp -ofile
centerlines.vtp
```

The second major step involved the addition of flow extensions to both the inlet and outlet of the geometry. Using the ‘vmtkflowextensions‘ pipeline, we generated extensions with an adaptive length, an extension ratio of 5 (meaning every extension will be five times the average artery diameter) and a normal estimation ratio of 1 (meaning the normal at inlet and outlet is computed taking into account exclusively the very initial and final sections). It is assumed these parameters ensured that the extensions were sufficiently long to allow the flow profile to fully develop before interacting with the domain of interest. However, we had to account for the available computational power, as even longer extensions, while generally recommended for improved accuracy, would have significantly increased the computational cost. The command to add flow extension is:

```
vmtksurfacereader -ifile carotid.vtp -pipe vmtkflowextensions -centerlinesfile
centerlines.vtp -adaptivelength 1 -extensionratio 5 -normalestimationratio 1
    -interactive 0 -pipe vmtksurfacewriter -ofile carotid-extended.vtp
```

Finally, the geometry was remeshed to improve uniformity of the surface. This step was performed using the 'vmtksurfaceremeshing' tool with specific parameters, including an edge length of 0.5 (empirically tuned). This uniformity is essential to assure the ray-casting algorithm for the Brinkman algorithm would work properly. The command for re-meshing is:

```
vmtksurfaceremeshing -ifile carotid-extended.vtp -preserveboundary 1
-elementsizemode edgelength -edgelength 0.5 -entityidsarray CellEntityIds -ofile
carotid-extended-remeshed.vtp
```

In summary, the preprocessing workflow transformed the raw carotid artery model into a simulation-ready geometry by adding flow extensions, ensuring proper alignment through the use of centerlines, and remeshing the surface to achieve uniform quality. 23 and 20 illustrate the geometry before and after preprocessing.

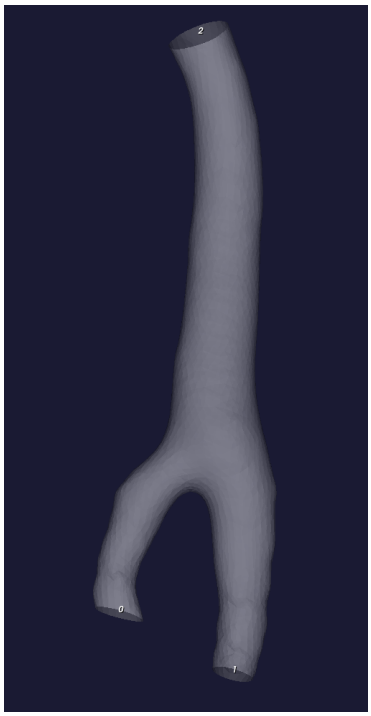

Fig. 3. Original artery

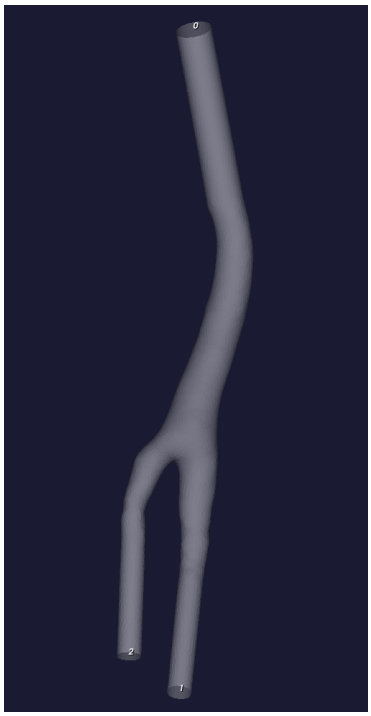

Fig. 4. Flow-extended artery

8 Numerical results

To validate the convergence of the developed numerical method, a fully three-dimensional analytical solution to the Navier-Stokes equations has been employed ([2]). The boundary conditions imposed are of Dirichlet type on all faces of the domain. The particular form of the chosen solution has not been selected arbitrarily; rather, it has been deliberately designed to include components that depend on all three Cartesian coordinates. This careful selection ensures that any coding errors, inconsistencies, or deficiencies in the numerical method are likely to manifest as substantial errors due to non-linearity mixing up velocity components ant every time step, particularly when conducting convergence studies.
The problem is, therefore, well-suited for evaluating the space and time convergence properties of the algorithm.

The solution is defined on the cube $[-0.5, 0.5]^3$.

$$
\begin{aligned}
u &= -a\left[e^{ax}\sin(ay+dz)+e^{az}\cos(ax+dy)\right]e^{-d^2t},\\
v &= -a\left[e^{ay}\sin(az+dx)+e^{ax}\cos(ay+dz)\right]e^{-d^2t},\\
w &= -a\left[e^{az}\sin(ax+dy)+e^{ay}\cos(az+dx)\right]e^{-d^2t},\\
p = -\frac{a^2}{2}\Big[&e^{2ax}+e^{2ay}+e^{2az}+2\sin(ax+dy)\cos(az+dx)e^{a(y+z)}\\
&+2\sin(ay+dz)\cos(ax+dy)e^{a(z+x)}\\
&+2\sin(az+dx)\cos(ay+dz)e^{a(x+y)}\Big]e^{-2d^2t}.
\end{aligned}
$$

Where parameters are set as $a=\frac{\pi}{4}$, $d=\frac{3\pi}{2}$, to appreciate a significantly variable solution with intersecting vortexes. $Re=1$ is set, meaning that the effects of advection and diffusion have the same importance in the physics of the flow. [2]
As for time convergence, first order time converge is expected to be appreciated. To this end, a suitable starting time-step $dt=0.00625$ has been fixed and, running over 16 iterations yelds a decay in the solution of around 50% on last iteration. Halving dt each time, exact first order convergence is observed as predicted by theoretical results. The same results have been achieved for (v,w) components. It is important to remark that a suitably high refinement space has been chosen, in order not for space discretization error to cover time convergence trend, as can be seen in 5

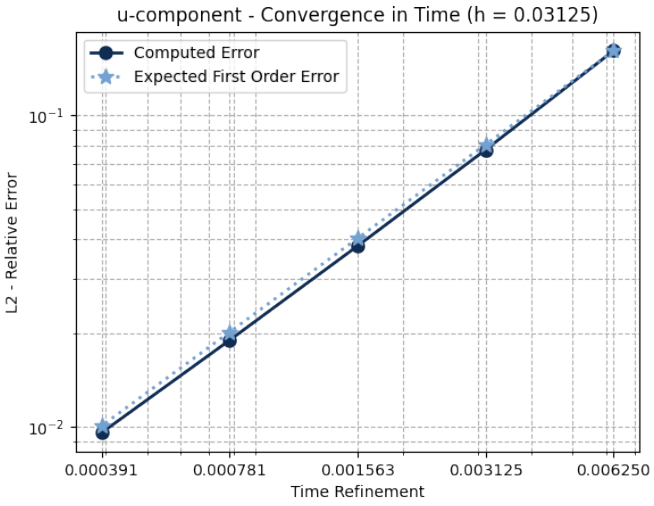

Fig. 5. Re = 1, time convergence

Regarding the convergence of pressure in time, the same simulation setup was used. As predicted by theoretical results, a convergence error rate of 0.5 was observed. Indeed, the numerical results confirm this trend. Finally we remark that pressure error appears significantly higher than velocity. This is due to the fact that convergence is checked between pressures (analytical and numerical) with different constants. The analytical solution fixes it to 0, while calculated one fixes a constant such that a pressure node is set to 0. Nevertheless, convergence trend is confirmed as shown in 6

[2] In the report we only report results for x-velocity component and pressure. COnvergence was tested for all single steps and functions obtained the predicted results. We report what seemed to be the most important.

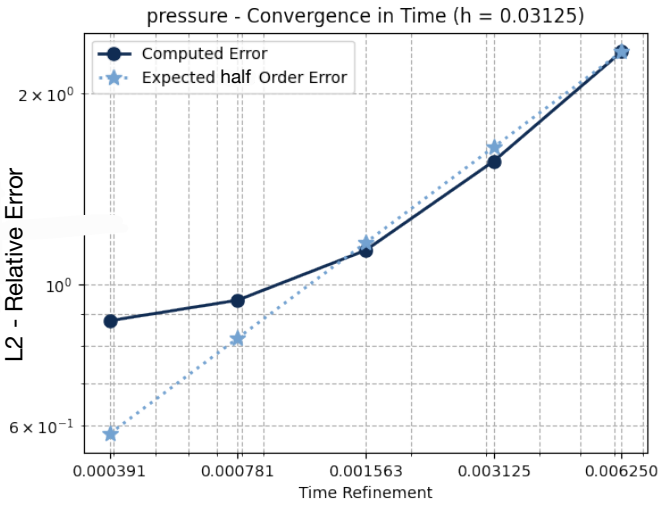

Fig. 6. Re = 1, time convergence

As for space convergence, a reasonably very small time step is chosen ($dt = 1.95313e - 4$) (also to make sure CFL condition is always respected) and subsequent iteration starting from 4 refinements is chosen. Indeed, correct second order of convergence is initially observed. Unfortunately, time discretization error is high enough to partially cover the true spatial convergence trend, therefore leading to a stagnation region as refinement increases, as in 7

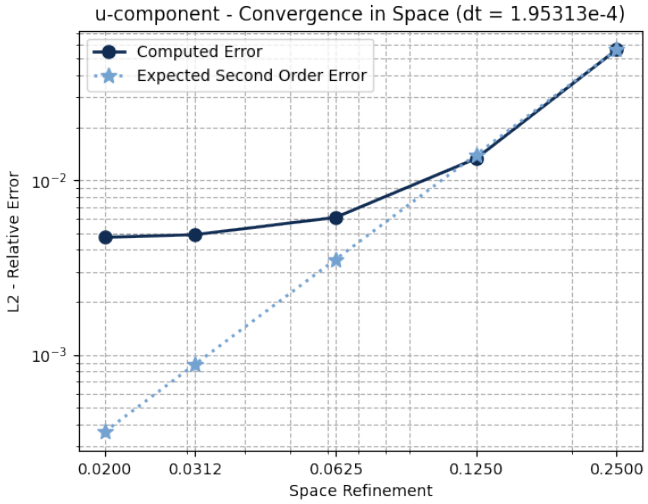

Fig. 7. Re = 1, space convergence

To ensure that no errors are present in the code, we conducted an additional simulation, reducing the time step by half.

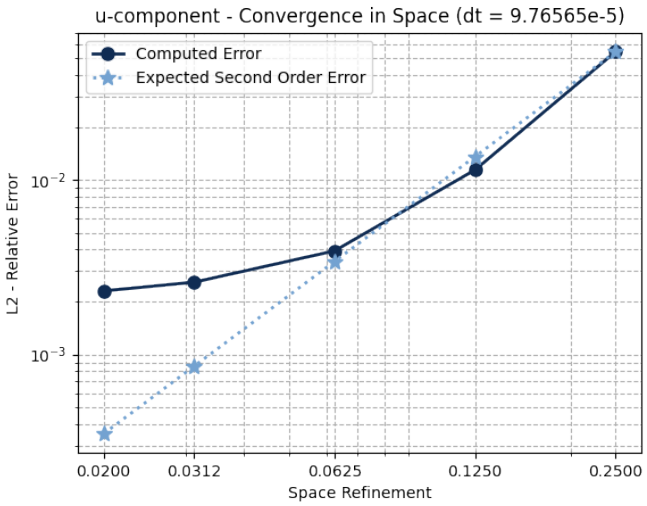

Fig. 8. Re = 1, space convergence

As it can be observed in 8, relative errors decreases sticking more to the expected trend. Stagnation is still present, but such result can be assumed to be a good indicator of the reliability of the implementation.
To complete the analysis of the implemented algorithm, we performed additional, computationally more expensive simulations, focusing on specific flow scenarios. In particular, we analyzed the Navier-Stokes equations for the previously discussed problem with Reynolds numbers $Re = 1$ and $Re = 2000$. The results, presented in the attached figures, highlight a significant difference in the flow characteristics. At $Re = 1$, the velocity magnitude and the z-component of the velocity exhibit smooth and stable profiles, due to the higher dissipative contribute. Conversely, at $Re = 2000$, the flow becomes more complex, with numerous vortices appearing across the domain, as evidenced by the higher variability in both the velocity magnitude and the z-component. These results are consistent with the expected transition from a more laminar to more turbulent-like behavior at higher Reynolds numbers.

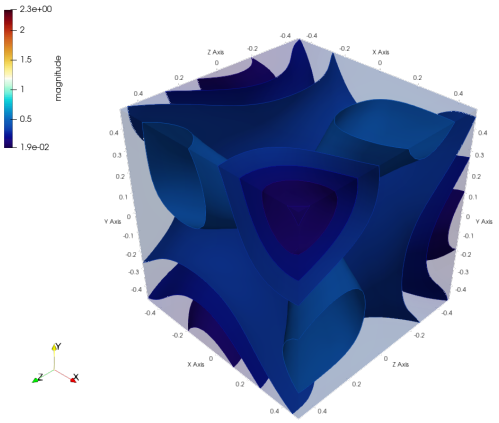

Fig. 9. Magnitude at Re = 1, T = 0.015, h = 1/32

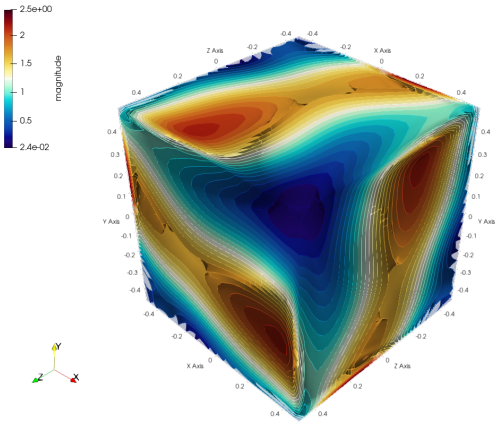

Fig. 10. Magnitude at Re = 2000, T = 0.015, h = 1/32

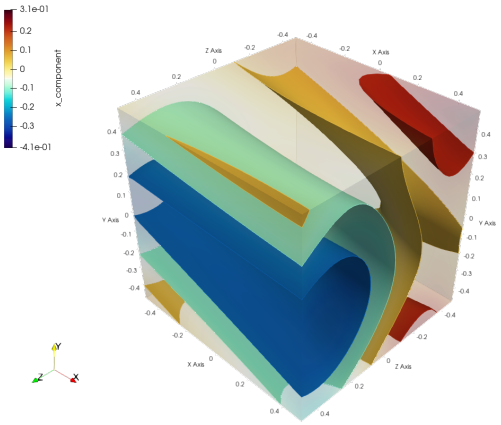

Fig. 11. x-component at Re = 1, T = 0.015, h = 1/32

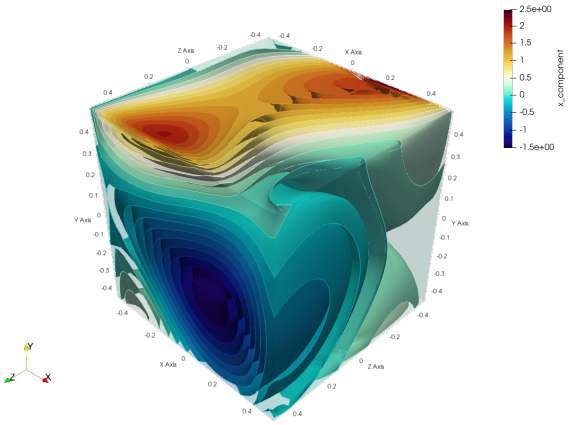

Fig. 12. x-component at Re = 2000, T = 0.015, h = 1/32

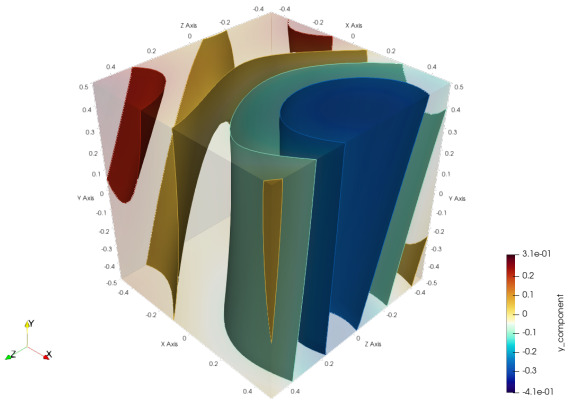

Fig. 13. y-component at Re = 1, T = 0.015, h = 1/32

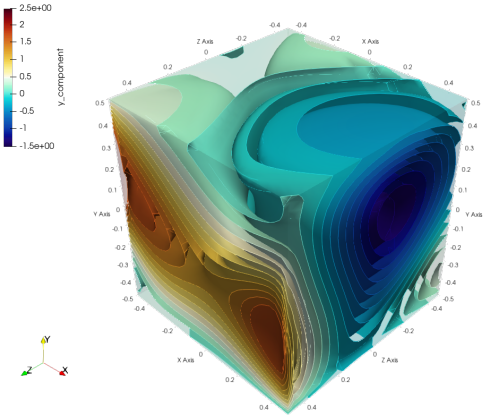

Fig. 14. y-component at Re = 2000, T = 0.015, h = 1/32

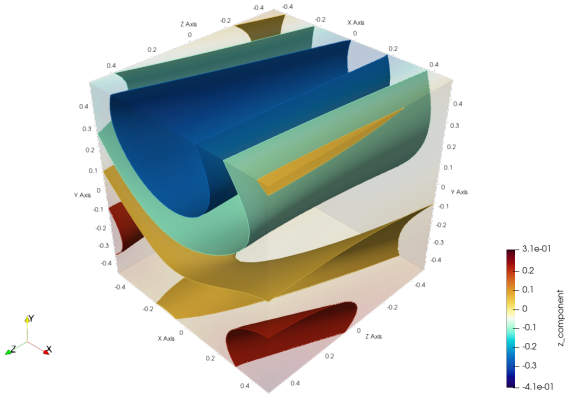

Fig. 15. w-component at Re = 1, T = 0.015, h = 1/32

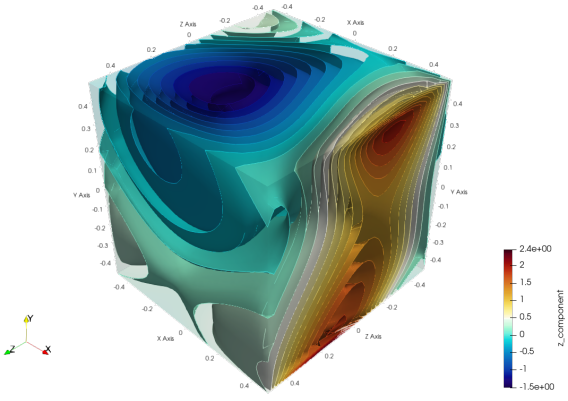

Fig. 16. w-component at Re = 2000, T = 0.015, h = 1/32

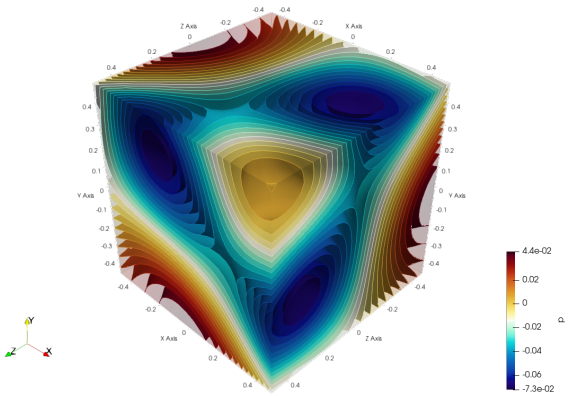

Fig. 17. pressure field at Re = 1, T = 0.015, h = 1/32

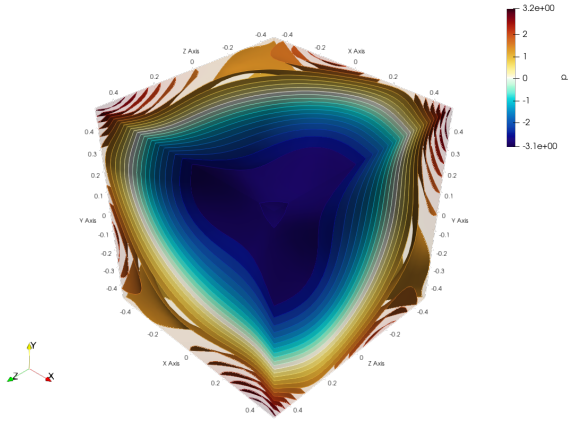

Fig. 18. pressure field at Re = 2000, T = 0.015, h = 1/32

8.1 Brinkman Flow

In this simulation, since the only way to apply boundary conditions in our code is by imposing Dirichlet conditions on the velocity, it is necessary to apply inlet and outlet conditions that respect the zero divergence constraint.

We start with the incompressibility condition for the velocity field, given by the differential form of the continuity equation:

$$\nabla \cdot \mathbf{v} = 0 \tag{59}$$

This equation implies that there is no net accumulation or depletion of mass in the control volume. To convert this to the integral form, we apply the divergence theorem and we obtain:

$$\int_V (\nabla \cdot \mathbf{v})\, dV = \int_S \mathbf{v} \cdot \mathbf{n}\, dA = 0 \tag{60}$$

Where: - V is the control volume, - S is the surface that bounds the control volume, - $\mathbf{n}$ is the unit normal vector pointing outward the surface S.
Next, we decompose the surface S into three parts: - S_{in}, the inlet surface, - S_{out} and S_{out2}, the outlet surfaces, - S_{lateral}, the lateral surface, where the velocity is assumed to be zero.
Thus, the total surface integral becomes:

$$\int_{S_{\text{in}}} \mathbf{v} \cdot \mathbf{n}\, dA + \int_{S_{\text{out}}} \mathbf{v} \cdot \mathbf{n}\, dA + \int_{S_{\text{out2}}} \mathbf{v} \cdot \mathbf{n}\, dA + \int_{S_{\text{lateral}}} \mathbf{v} \cdot \mathbf{n}\, dA = 0 \tag{61}$$

Since the velocity on the lateral surface S_{lateral} is assumed to be zero, the contribution from this surface is null, and we obtain:

$$\int_{S_{\text{in}}} \mathbf{v} \cdot \mathbf{n}\, dA + \int_{S_{\text{out}}} \mathbf{v} \cdot \mathbf{n}\, dA + \int_{S_{\text{out2}}} \mathbf{v} \cdot \mathbf{n}\, dA = 0 \tag{62}$$

This condition ensures that the flow entering the system is equal to the flow exiting, thereby maintaining mass conservation. In our case, we have a adimensional section of area of 25.6 (corresponding to a physical surface of $2.56e1 \quad mm^2$) as inlet (S_{in}) and two adimensional sections of area 10.7 ($1.07e1 \quad mm^2$) and 10.8 ($1.08e1 \quad mm^2$) as outflow (S_{out_1} and S_{out_2}). An adimensional velocity field of (0, -1, 0) is set as inlet and an adimensional velocity field of (0, -1.20, 0) and (0, -1.18, 0) is respectively imposed on S_{out_1} and S_{out_2}. Computing characteristic velocity from Re, D (characteristic artery diameter), $nu = 3.8e-6 \quad m^2/s$ (average blood kinematic viscosity) we obtain an inlet value of $(0, 1.33e-1, 0) \quad m/s$. By such result, and taking into account we iterated over 1000 steps with a discretization of $1e-3$ we retrieve a dimensional time of $4.39e-2s$. In our adimensional framework, we verified that such parameters allowed the flow to be fully developed at roughly 500 iterations. Regarding the boundary conditions, a non-zero Dirichlet condition for the velocity is applied only at the nodes corresponding to the inlet and outflow surfaces. At all other nodes, as well as on the faces of the parallelepiped, where no inflow/outflow is present, velocity is null. The Reynolds number is set according to the typical order of magnitude for a carotid artery, which is 200. Initial velocity inside the Brinkman domain is set to be $(0, 0, 0)$ in the whole domain. It is important to remark that such initial solution, and, most important, the time-dependent boundary conditions are set divergence-free. What follows is a collection of slices at different y-adimensional levels taken at the last time step (fully developed). Moreover also a plot of 20 contour lines is provided.

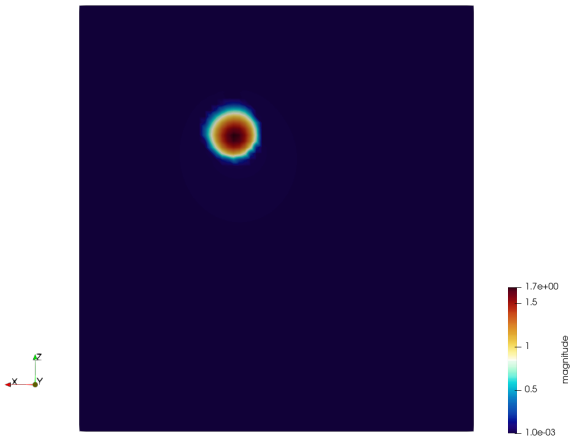

Fig. 19. Magnitude at Re = 200, T = 1, 70 refs, y = 295.11

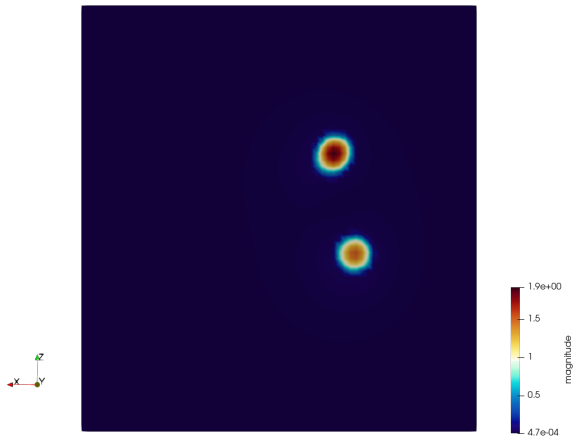

Fig. 20. Magnitude at Re = 200, T = 1, 70 refs, y = 266.44

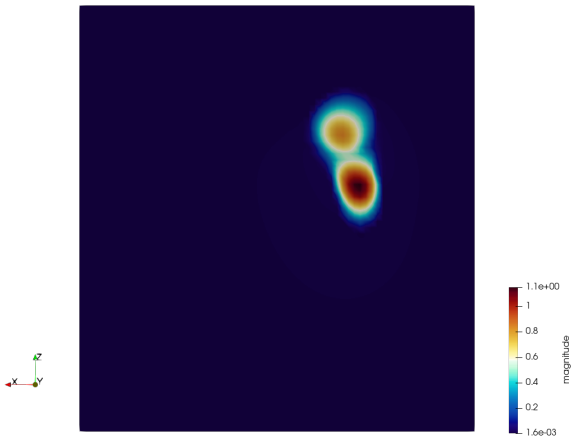

Fig. 21. Magnitude at Re = 200, T = 1, 70 refs, y = 260.55

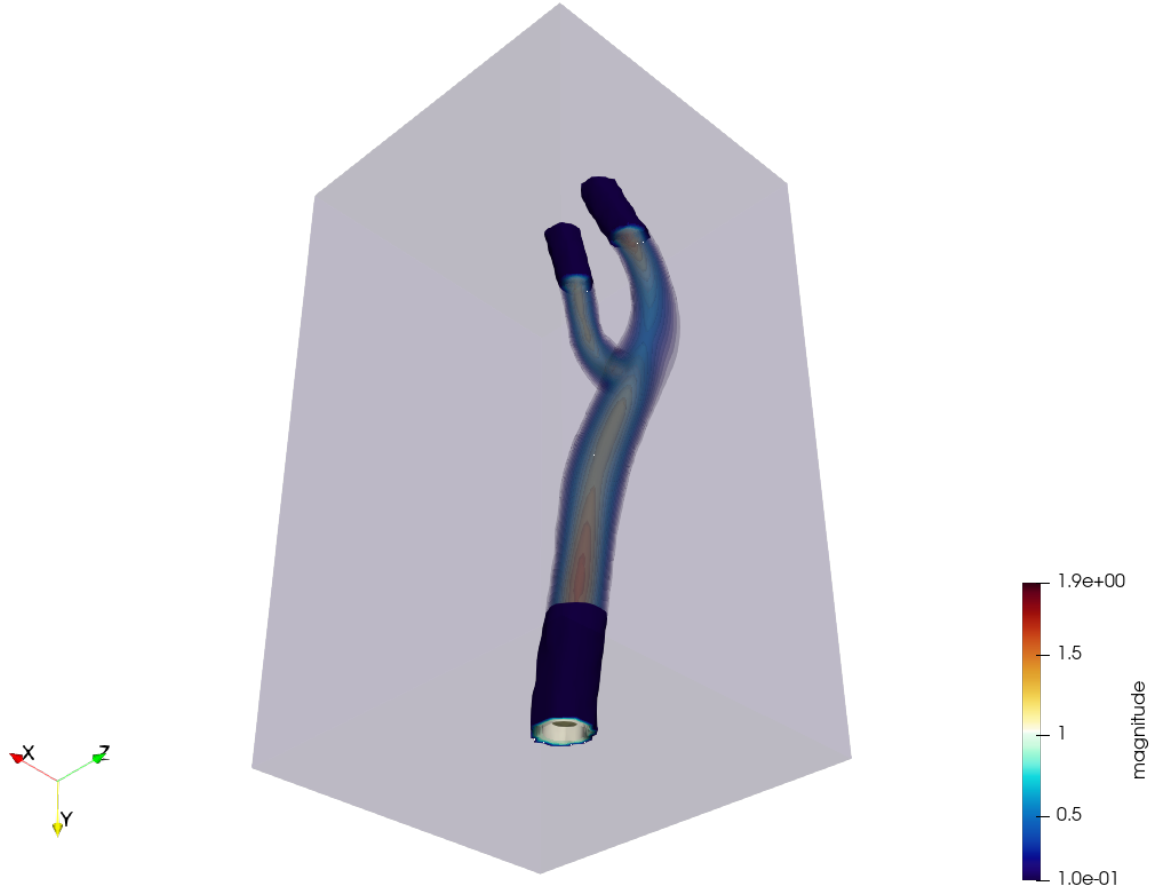

Fig. 22. Magnitude at Re = 200, T = 1, 70 refs, contour lines. Inlet and outlet have been darkes as should not be taken into considerations for CFD purposes.

To further verify stability, we also performed a final simulation with the same parameters as before except for Reynolds set to 900. Also in this case, stability was assured, leading to a slightly more turbolent flow. Such (little) difference is visible at the level of the bifurcation. A snapshot of the magnitude is given below. Unfortunately, some (very minimal) instabilities outside artery domain was present at higher Reynolds. Such little issue can be overcome with more iterations or better tuning penality constant. For more results, it is advisable to look at github.com/galbiatidavide/Finite-Differences-Navier-Stokes.

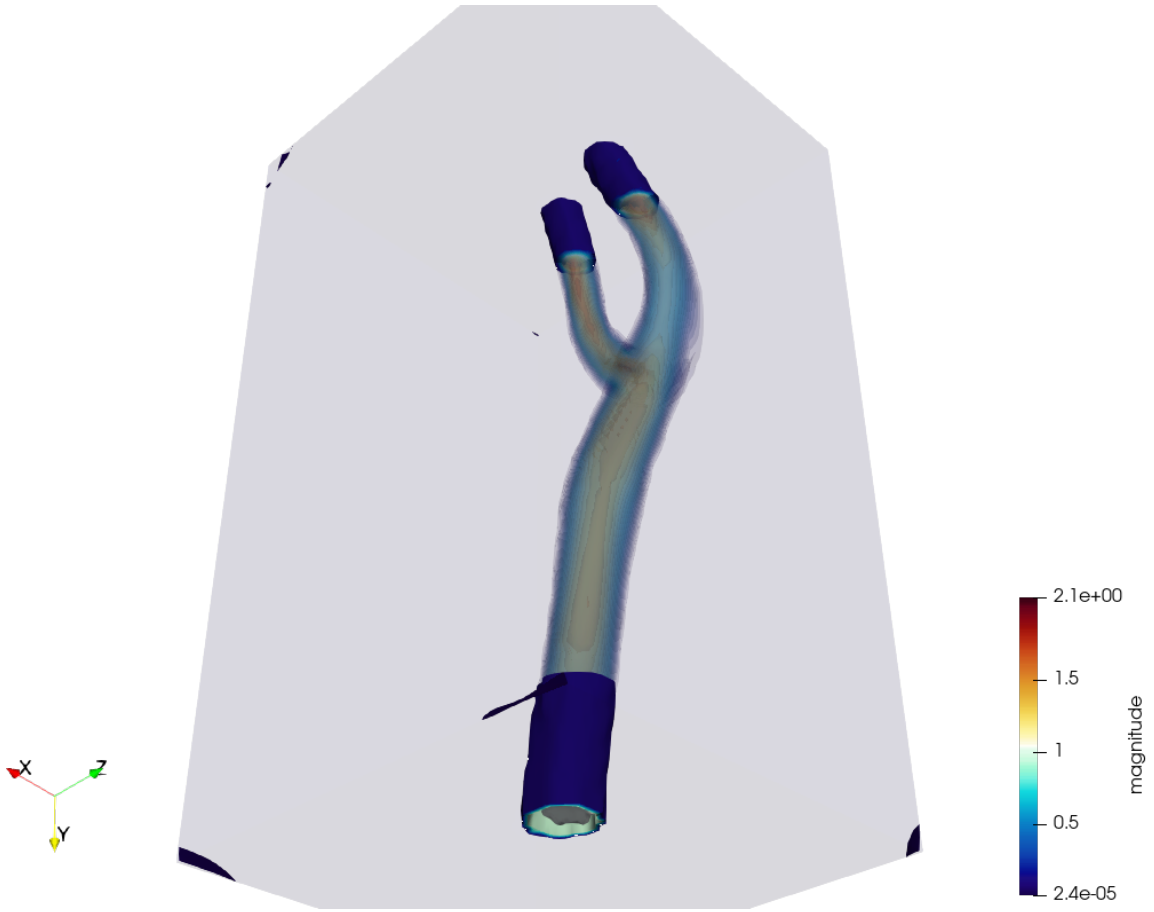

Fig. 23. Magnitude at Re = 900, T = 1, 70 refs, contour lines. Inlet and outlet have been darkes as should not be taken into considerations for CFD purposes.

9 Conclusion

The development of an efficient, parallel multi-physics solver in C++ using PETSc has proven successful in addressing the Brinkman Equation for simulating blood flow within the carotid artery. Leveraging the Chorin-Temam projection method, with its fully explicit treatment of the nonlinear term, allowed us to achieve significant computational efficiency while maintaining stability under a properly chosen CFL condition. The use of a Cartesian 3D staggered grid and the Finite Difference approach enabled efficient parallelization and ensured the scalability of our solver.

The validation process, conducted in an adimensional framework with Reynolds numbers up to 2000, demonstrated the robustness and accuracy of the method. The numerical results confirmed the convergence and stability of the semi-explicit Navier-Stokes algorithm, making it a viable tool for large-scale simulations of complex flows.

Future work could focus on addressing boundary inaccuracies by refining the treatment of boundary conditions. For instance, implementing homogeneous Neumann conditions at the outlet could enhance the physiological relevance of the simulations, particularly in cardiovascular applications. Furthermore, a deeper exploration of blood flow dynamics, including the incorporation of patient-specific data, would enable the imposition of more realistic initial and boundary conditions, thereby extending the applicability of the solver to clinical scenarios.

References

[1] A. W. Date. Solution of navier-stokes equations on non-staggered grid. *International Journal of Heat and Mass Transfer*, 36(7):1913–1922, 1993.

[2] C. R. Ethier and D. A. Steinman. Exact fully 3d navier-stokes solutions for benchmarking. *International Journal for Numerical Methods in Fluids*, 19:369–375, 1994.

[3] A. Ferrero, F. Gazzola, and M. Zanotti. *Elements of Advanced Mathematical Analysis for Physics and Engineering*. Ed. Esculapio, 2013.

[4] J. L. Guermond, P. Minev, and J. Shen. An overview of projection methods for incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 195:6011–6045, 2006. Received 10 June 2004; revised 22 August 2005; accepted 21 October 2005.

[5] N. Parolini. Computational fluid dynamics course notes, 2023. Politecnico di Milano, September 2023.

[6] B. Seibold. A compact and fast matlab code solving the incompressible navier-stokes equations on rectangular domains. Technical report, Massachusetts Institute of Technology, 2008.

[7] M. Valera, M. P. Thomas, M. Garcia, and J. E. Castillo. Parallel implementation of a petsc-based framework for the general curvilinear coastal ocean model. *Journal of Marine Science and Engineering*, 7(6):178, 2019.