XLVI CILAMCE
VITÓRIA - ES - BRAZIL - 2025
XLVI IBERO-LATIN AMERICAN CONGRESS ON COMPUTATIONAL METHODS IN ENGENEERING

A High-Performance ADI–Pressure Projection Method for Solving the Navier–Stokes Equations

Raphael N. Laroca[1], Iury Igreja[1,2]

[1]*Department of Computer Science, Federal University of Juiz de Fora*
Campus Universitário, Rua José Lourenço Kelmer, s/n - São Pedro, Juiz de Fora, 36036-900, Minas Gerais, Brazil
raphael.laroca@gmail.com
[2]*Graduate Program in Computational Modeling, Federal University of Juiz de Fora*
Campus Universitário, Rua José Lourenço Kelmer, s/n - São Pedro, Juiz de Fora, 36036-900, Minas Gerais, Brazil
iury.igreja@ufjf.br

Abstract.

The Navier–Stokes equations are fundamental for describing fluid motion, yet they remain among the most challenging problems in mathematical and computational fluid dynamics. Their strong nonlinearity, coupled with the necessity of simultaneously approximating velocity and pressure fields, leads to a highly complex and computationally demanding system. In particular, mixed formulations introduce compatibility requirements that add further difficulty to ensuring stable and accurate solutions. Moreover, the consistent enforcement of mass conservation remains a delicate aspect in numerical schemes. To address these challenges, this work proposes a simple and efficient numerical strategy that combines an Alternating Direction Implicit (ADI) method with a Pressure Projection technique. The ADI method efficiently handles the velocity computation, while the Pressure Projection ensures the enforcement of the divergence-free condition, preserving mass conservation naturally. For enhanced computational performance, the velocity field is solved in CPU using OpenMP parallelization in the ADI steps, while GPU acceleration using CUDA is employed for the solution of the pressure equation. The proposed methodology is validated through comparisons with benchmark solutions from the literature, applied to classical CFD test cases. The results demonstrate the accuracy, efficiency, and robustness of the approach, highlighting its potential for scalable and reliable simulations in computational fluid dynamics.

Keywords: CFD Simulation, ADI Method, High-Performance, Navier-Stokes, Pressure Projection.

1 Introduction

The Navier–Stokes equations are fundamental for describing fluid motion across a wide range of scientific and engineering fields, including aerodynamics, computer graphics, and biomedical applications [1–3]. Solving these equations accurately presents significant computational challenges, primarily due to their inherent nonlinearity and the difficulty in solving velocity and pressure fields while ensuring mass conservation [4, 5]. This complexity leads to a substantial computational cost, especially for practical, large-scale simulations. This work presents an efficient numerical approach designed to tackle these difficulties.

In this context, this work presents a numerical method for solving the transient incompressible Navier–Stokes equations that combines an Alternating Direction Implicit (ADI) scheme [6] with a pressure projection method [7] to enforce incompressibility. Conventional approaches for Navier-Stokes equations often employ ADI via stream-function–vorticity formulations [8–10]. However, such formulations face inherent limitations as these functions lack uniqueness in three-dimensional configurations and suitable boundary conditions are frequently unavailable [1]. Our method, conversely, directly solves the primitive variables (velocity and pressure), circumventing these challenges.

To achieve a stable computational implementation of the ADI method for the velocity-pressure Navier-Stokes problem, we employ a Marker and Cell (MAC) grid [11]. The adopted computational strategy involves solving the ADI discretization for each velocity component, resulting in tridiagonal linear systems. These systems are efficiently solved using the Thomas algorithm [12], which multiple instances of it are parallelized with OpenMP. Subsequently, the linear system resulting from the Poisson problem for the pressure field is solved using the BiCGSTAB iterative method [13], with acceleration achieved through a GPU implementation. This methodology is

validated through three-dimensional numerical experiments demonstrating accurated results compared to literature and achieving computational efficiency comparable to that reported by [14], a study employing similar numerical methodologies and parallelization strategies.

2 Mathematical and Computational Model

In this section we present the mathematical and computational approach to solve the Navier-Stokes equations.

2.1 Governing Equations

Considering a domain $\Omega \subset \mathbb{R}^3$, with Lipschitz boundary $\Gamma = \partial\Omega$ and time interval $(0, T]$, the incompressible and Newtonian Navier-Stokes equations can be presented as: *Given* ν, $\mathbf{F}$, $\mathbf{u}_0$ *and* $\mathbf{g}$, *find* $[\mathbf{u}, p]$, *such as*

$$\frac{\partial \mathbf{u}}{\partial t} - \nu\nabla^2\mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u} + \nabla p = \mathbf{F}, \qquad \text{in } \Omega \times (0, T] \tag{1}$$

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

$$\mathbf{u} = \mathbf{g} \qquad \text{on } \Gamma \times (0, T], \tag{3}$$

$$\mathbf{u}(\mathbf{x}, 0) = \mathbf{u}_0 \qquad \text{in } \Omega. \tag{4}$$

In system (1)-(4), $\mathbf{u} = (u, v, w)$ denotes the velocity field, ν is the viscosity, p represents the pressure field, $\mathbf{g}$ specifies the Dirichlet boundary condition for the velocity, and $\mathbf{u}_0$ is the initial velocity field.

One of the numerical difficulties in solving the Navier-Stokes equations in their mixed form is the coupling of velocity and pressure fields, which often results in significant computational costs and challenges [4]. Although methods for the simultaneous solution of both fields exist [15], these are generally computationally expensive. To overcome this difficulty, we adopted a simpler approach that separates the equations into two distinct problems: one for velocity and the other for pressure.

Thus, rearranging the Navier-Stokes equations, decoupling velocity from pressure, we can write (1)-(2) as follows.

$$\frac{\partial \mathbf{u}}{\partial t} - \nu\nabla^2\mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u} = 0, \tag{5}$$

$$\frac{\partial \mathbf{u}}{\partial t} = -\nabla p \quad \text{such that} \quad \nabla \cdot \mathbf{u} = 0. \tag{6}$$

2.2 ADI Method and Pressure Projection

To solve the problem defined by equations (5) and (6), we utilize a numerical scheme implemented on a MAC (Marker-and-Cell) mesh [11]. The velocity components are determined using the Peaceman and Rachford ADI method [6] derived from equation (5). Subsequently, the pressure field in (6) is found using a central difference approximation. Finally, to enforce the incompressibility constraint, a correction is applied to the velocity field through Chorin's Projection Method [7].

Therefore, applying the ADI discretization on the Eq.(5), we obtain the following system

$$\begin{cases} \left(\dfrac{3}{\Delta t} - \nu\delta_{xx} + u^n_{i,j,k}\delta_x\right)\mathbf{u}^{n+1/3}_{i,j,k} = \left(\dfrac{3}{\Delta t} + \nu\delta_{yy} + \nu\delta_{zz} - v^n_{i,j,k}\delta_y - w^n_{i,j,k}\delta_z\right)\mathbf{u}^n_{i,j,k} \\ \left(\dfrac{3}{\Delta t} - \nu\delta_{yy} + v^n_{i,j,k}\delta_y\right)\mathbf{u}^{n+2/3}_{i,j,k} = \left(\dfrac{3}{\Delta t} + \nu\delta_{xx} + \nu\delta_{zz} - u^n_{i,j,k}\delta_x - w^n_{i,j,k}\delta_z\right)\mathbf{u}^{n+1/3}_{i,j,k} \\ \left(\dfrac{3}{\Delta t} - \nu\delta_{zz} + w^n_{i,j,k}\delta_z\right)\mathbf{u}^{n+1}_{i,j,k} = \left(\dfrac{3}{\Delta t} + \nu\delta_{xx} + \nu\delta_{yy} - u^n_{i,j,k}\delta_x - v^n_{i,j,k}\delta_y\right)\mathbf{u}^{n+2/3}_{i,j,k} \end{cases} \tag{7}$$

In this context, δ represents a central finite difference approximation of the partial derivative in the specified direction, discretized on a MAC grid. For example, the second-order and first-order approximations along the x-direction are, respectively

$$\delta_{xx}\mathbf{u}_{i,j,k} = \frac{\mathbf{u}_{i-1/2,j,k} - 2\mathbf{u}_{i,j,k} + \mathbf{u}_{i+1/2,j,k}}{\Delta h^2} \qquad \text{and} \qquad \delta_x\mathbf{u}_{i,j,k} = \frac{\mathbf{u}_{+1/2,j,k} - \mathbf{u}_{i-1/2,j,k}}{\Delta h}. \tag{8}$$

This approach extends similarly to approximations in other directions (e.g., y and z). The choice of central differences for the first-order derivative maintains a quadratic convergence rate for the velocity, as demonstrated by

the results presented in Section 3. Furthermore, the non-linearity of the convective term is addressed by lagging the velocity field to the previous timestep. This approach preserves the independent nature of each component, allowing them to be solved in an uncoupled fashion, which yields significant scalability and performance gains.

By solving Eq. (7), an intermediate non-mass-conserving velocity field, denoted by $\hat{\mathbf{u}}^n$, is obtained. This field requires correction to enforce mass conservation, a process enabled by the splitting approach using Equation (6). To utilize Eq. (6) for this purpose, we first apply a simple time discretization to the velocity term, then take the divergence on both sides, yielding

$$\nabla \cdot (\mathbf{u}^{n+1} - \hat{\mathbf{u}}^n) = \nabla \cdot (-\Delta t \nabla p). \tag{9}$$

Following [7], by imposing the incompressibility condition on $\mathbf{u}^{n+1}$ (i.e., $\nabla \cdot \mathbf{u}^{n+1} = 0$), Eq. (9) simplifies to

$$\nabla^2 p = \frac{1}{\Delta t} \nabla \cdot (\hat{\mathbf{u}}^n). \tag{10}$$

This equation provides a direct expression for the pressure solution that is approximated in the following form

$$(\delta_{xx} + \delta_{yy} + \delta_{zz})\, p^{n+1}_{i,j,k} = \frac{1}{\Delta t} \left(\delta_x \hat{u}^n_{i,j,k} + \delta_y \hat{v}^n_{i,j,k} + \delta_z \hat{w}^n_{i,j,k} \right). \tag{11}$$

Following Chorin's Projection Method [7], upon solving Eq. (11), the pressure p^{n+1} is obtained and applied to correct the intermediate velocity field $\hat{\mathbf{u}}^n$ through the application of the discretized form of Eq.(6), enforcing the system's incompressibility condition, as follows

$$\left(u^{n+1}_{i,j,k}, v^{n+1}_{i,j,k}, w^{n+1}_{i,j,k} \right) = \left(\hat{u}^n_{i,j,k}, \hat{v}^n_{i,j,k}, \hat{w}^n_{i,j,k} \right) - \Delta t \left(\delta_x, \delta_y, \delta_z \right) p^{n+1}_{i,j,k}. \tag{12}$$

Once the mass-conserving velocity field is obtained, we then advance the simulation to the next timestep by repeating the entire resolution process, whose computational implementation details are described in the next Section.

2.3 Computational Implementation

The overall implementation was developed in `C++` to meet critical performance requirements. OpenMP is employed for multi-threading during the assembly and solution of the linear systems produced by ADI method (7), with each velocity component allocated a dedicated thread. The assembly and solution of corresponding systems are further parallelized by slave threads managing individual mesh lines, an approach that provides significant scalability and notable performance gains compared to the serial (single-threaded) counterpart, as discussed in Section 3.

However, unlike the ADI method, which produces easily solvable tridiagonal linear systems, the approach presented in Eq.(11) for pressure yields a large and sparse matrix that is more computationally demanding. This discretization uses a 7-Point Stencil in 3D, implying, at worst, 7 non-zero entries per line. The matrix values are precomputed once and stored in CSR sparse matrix compressed format, given that the number of elements grows quadratically with the total discretization points in each direction. As the pressure resolution represents the highest computational cost in our methodology, the computation is copied to the GPU and we utilize iterative algorithms in combination with GPU parallelization. Specifically, the linear system, derived from Eq. (11), undergoes a single initialization. Since this system remains static throughout all experiments, for each iteration, only the right-hand side vector requires updating. The implementation for this specific component was developed using `AMGX` [16], employing `BiCGSTAB` [13] as the chosen solver due to its stability and efficiency for the intended purposes. Although the possibility of inverting the matrix was considered, given that the system remains constant for any given iteration in the experiments, the quadratic growth of system size relative to input size renders memory availability problematic.

The experiments were conducted on a single machine running Ubuntu 24.04. It features a Ryzen 3 3100 3.9 GHz processor with 4 physical cores, 32 GB of DDR4 3200MHz RAM, and a RTX3060 12GB graphics card with 3584 CUDA cores. The results were processed using both ParaView [17] and custom-developed plotters utilizing `Matplotlib` [18].

3 Numerical Results

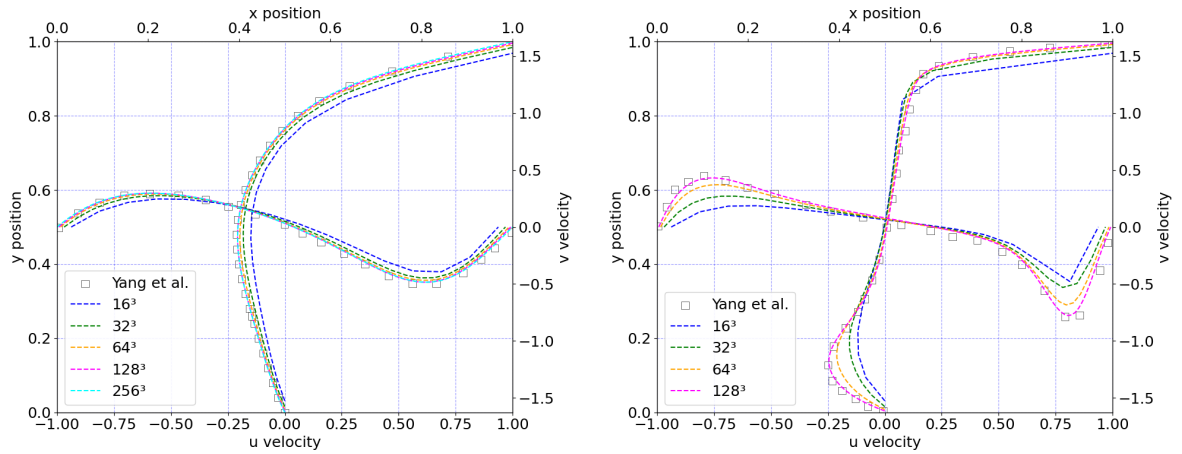
This section presents a study of the lid-driven cavity flow problem, with results compared to literature across various Reynolds numbers. Finally, an analysis of computational performance details the benefits and scalability achieved through the adopted parallelization strategies.

3.1 Lid-Driven Cavity Flow

To further validate our model, we used the lid-driven cavity flow problem, a well-established benchmark in CFD. This setup involves a closed square or cubic domain with a constant tangential velocity on the upper boundary and no-slip conditions elsewhere, generating a primary recirculating vortex and secondary corner vortices at higher Reynolds numbers. Given its geometric simplicity and complex flow physics, the lid-driven cavity offers comprehensive data for model validation, including extensive 2D [19] and 3D analyses [20] across various Reynolds numbers.

In all simulations, Δt is determined by the convective Courant-Friedrichs-Lewy (CFL) condition, expressed as $CFL = (U_{max}\Delta t)/\Delta h$. Given the strongly convective nature of the problems and to ensure numerical stability, the CFL number must be less than unity. As the value of U_{max}(the maximum absolute magnitude of the velocity field) is known at the beginning of the simulation and remains fixed throughout all experiments, Δt is constantly determined, satisfying the condition $\Delta t < \Delta h/U_{max}$.

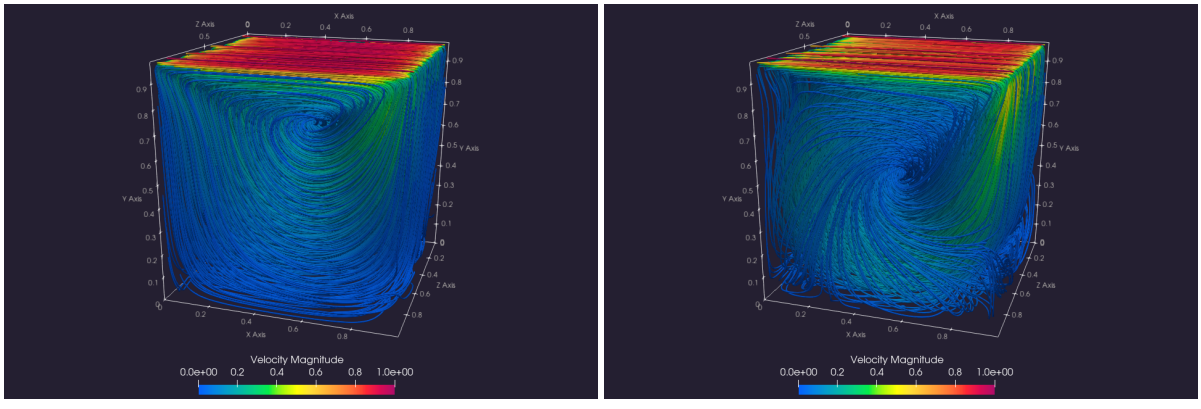
In this context, we conducted a 3D experiment testing Reynolds numbers $Re = 100$ and 1000. We compared the u and v velocity component contour distributions against experimental data from [21]. Through progressive mesh refinement, we ensured the reliability and accuracy of our results against this benchmark. The results can be seen in the Fig. 1.



(a) $Re = 100$ (b) $Re = 1000$

Figure 1. Comparison of dimensionless velocity components u and v with [21] for various mesh refinements at different Reynolds numbers.

The final iteration of each experiment can be seen in a 3D streamline plot made with ParaView in Figure 2.



(a) $Re = 100$ (b) $Re = 1000$

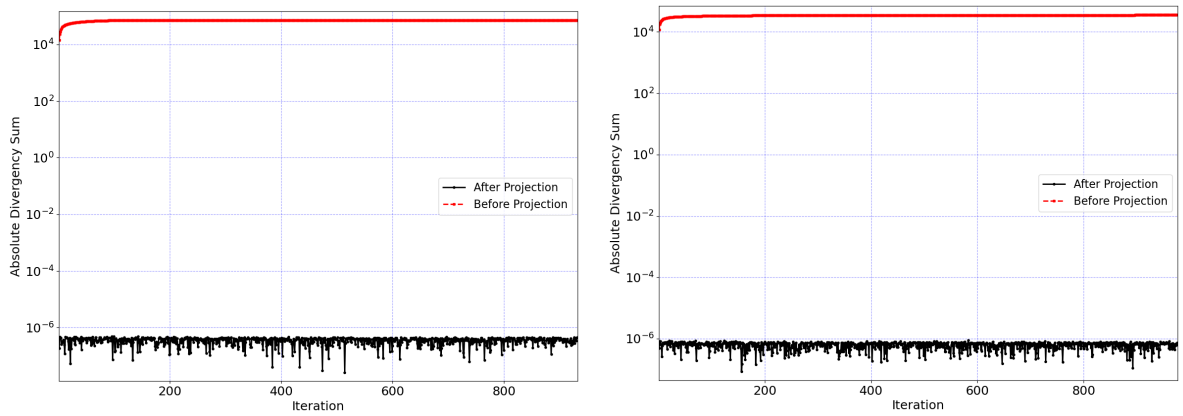
Figure 2. Final iteration of the experiments for Reynolds 100 (left) and 1000 (right).

The computational results demonstrate behavior consistent with theoretical expectations. Progressive mesh refinement yields solutions that exhibit monotonic convergence toward the benchmark reference data, with finer discretizations producing increasingly accurate approximations of the velocity field characteristics.

To assess the enforcement of the incompressibility constraint, the total divergence magnitude was computed and analyzed across all computational cells. The total divergence is quantified as the summation of the absolute values of individual cell divergences throughout the computational domain. This metric serves as a direct measure of mass conservation violation and provides validation into the effectiveness of the pressure-velocity coupling algorithm. Mathematically, we define this value as

$$\nabla \cdot \mathbf{u} = \sum_{i,j,k}^{N_y,N_x,N_z} \left| \delta_x \hat{u}^n_{i,j,k} + \delta_y \hat{v}^n_{i,j,k} + \delta_z \hat{w}^n_{i,j,k} \right| . \tag{13}$$

We have taken results from the 128^3 grid refinement arbitrarily and plotted it, as the pattern maintains the same for any choice. The results for the first thousand iterations are shown in Fig. 3.



(a) $Re = 100$ (b) $Re = 1000$

Figure 3. Values of divergency sum before and after projection.

Finally, we conducted a performance investigation on the model. By measuring the average computational wall time for each individual iteration in both the Transport equation that was solved using ADI in the CPU and the Pressure Equation, solved using iterative algorithms in GPU. The results can be visualized in Fig. 4.

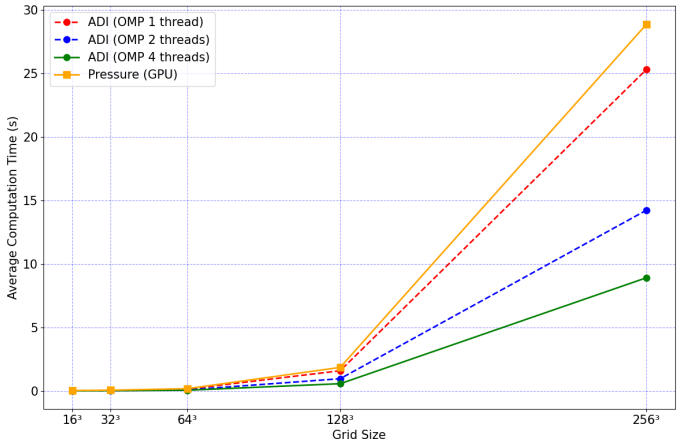


Figure 4. Time comparison for different Thread counts.

The pressure solution constitutes the computationally most expensive component of the present implementation, as also observed in [14]. This behavior is anticipated given that system size scales with the square of grid dimensions. The iterative algorithm currently employs a tolerance value of 10^{-12}, which directly affects the

residual divergence present in the resultant velocity field. Increased tolerance values produce inadequate pressure projection, consequently yielding velocity fields with elevated divergence magnitudes.

By varying the amount of threads to different values, we produce the Table 1 of total wall times for the complete solution of one ADI iteration to analyze the speedup and efficiency of the application. The timings for different Reynolds values are similar, as the number of operations per iteration is the same. The difference arises in the total experiment times, since higher Reynold Numbers take longer to reach the steady-state.

Table 1. ADI performance, speedup, and efficiency with varying thread counts for Reynolds 100.

Grid Size	ADI 1T [s]	ADI 2T [s]	ADI 4T [s]	Speedup 2T	Speedup 4T	Eff. 2T	Eff. 4T
16^3	0.001087	0.000870	0.000461	1.25	2.36	0.63	0.59
32^3	0.012170	0.009683	0.005947	1.26	2.05	0.63	0.51
64^3	0.126430	0.067937	0.042522	1.86	2.97	0.93	0.74
128^3	1.578865	0.950724	0.557389	1.66	2.83	0.83	0.71
256^3	25.31253	14.22495	8.901657	1.78	2.84	0.89	0.71

4 Conclusions and Future Works

This paper outlines the mathematical foundations underlying the proposed model and describes a subset of the experimental evaluations performed to assess its implementation. The obtained results demonstrate that the model achieves a high degree of accuracy and exhibits strong agreement with established theoretical and empirical findings in low Reynolds number regimes. Furthermore, the model consistently enforces the Incompressibility Condition, maintaining deviations within predefined tolerance thresholds. It was observed that lower tolerances for the Pressure solver made the amount of leftover divergency smaller. For the performance, the model had a good speedup ration with the increasing amount of threads and an acceptable efficiency of around 80% across more computationally intensive problems. It was observed that in hybrid CPU/GPU approaches, the most expensive step is the solution of the Pressure on the GPU. This was also documented on [14] when using hybrid CPU/GPU implementations and it is a point in we can explore on future works with more advanced pressure solvers and schemes.

The model exhibits considerable potential for deployment in performance-sensitive applications, particularly within the domain of computer graphics. In such contexts, where visual plausibility often supersedes strict adherence to physical accuracy, the model remains viable even when deviations from scientific precision occur. The model can be further optimized in a variety of ways, such as further parallelization of the grid in blocks, either by using OpenMP or MPI for greater scalability across multiple computer nodes, or by relaxing the restriction on the solution of the incompressibility condition for even faster solution at the cost of accuracy.

Additionally, the model may be adapted for integration with hybrid particle-based methodologies, including the FLIP approach [22]. In such implementations, the model could be employed during intermediate stages of particle-grid simulations to introduce a physically consistent and mathematically rigorous diffusion mechanism which FLIP lacks.

Acknowledgements. This work was supported by CNPq (grant number 305353/2022-5) and FAPEMIG (PIBIC/ FAPEMIG / UFJF – 2025).

Authorship statement. The authors hereby confirm that they are the sole liable persons responsible for the authorship of this work, and that all material that has been herein included as part of the present paper is either the property (and authorship) of the authors, or has the permission of the owners to be included here.

References

[1] R. Bridson. *Fluid simulation for computer graphics*. AK Peters/CRC Press, 2015.

[2] P. Bouillot, O. Brina, R. Ouared, K.-O. Lovblad, M. Farhat, and V. M. Pereira. Particle imaging velocimetry evaluation of intracranial stents in sidewall aneurysm: hemodynamic transition related to the stent design. *PLoS One*, vol. 9, n. 12, pp. e113762, 2014.

[3] J. Winslow, H. Otsuka, B. Govindarajan, and I. Chopra. Basic understanding of airfoil characteristics at low reynolds numbers (10 4–10 5). *Journal of aircraft*, vol. 55, n. 3, pp. 1050–1061, 2018.
[4] J. H. Ferziger and M. Perić. *Computational methods for fluid dynamics*. Springer, 2002.
[5] V. John. *Finite element methods for incompressible flow problems*, volume 51. Springer, 2016.
[6] D. W. Peaceman and H. H. Jr. Rachford. The numerical solution of parabolic and elliptic differential equations. *Journal of the Society for industrial and Applied Mathematics*, vol. 3, n. 1, pp. 28–41, 1955.
[7] A. J. Chorin. The numerical solution of the navier-stokes equations for an incompressible fluid. *Bulletin of the American Mathematical Society*, vol. 73, n. 6, pp. 928–931, 1967.
[8] E. Erturk, T. C. Corke, and C. Gökçöl. Numerical solutions of 2-d steady incompressible driven cavity flow at high reynolds numbers. *International journal for Numerical Methods in fluids*, vol. 48, n. 7, pp. 747–774, 2005.
[9] E. Erturk. Numerical solutions of 2-d steady incompressible flow over a backward-facing step, part i: High reynolds number solutions. *Computers & Fluids*, vol. 37, n. 6, pp. 633–655, 2008.
[10] E. Erturk. Discussions on driven cavity flow. *International journal for numerical methods in fluids*, vol. 60, n. 3, pp. 275–294, 2009.
[11] F. H. Harlow, J. E. Welch, and others. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Physics of fluids*, vol. 8, n. 12, pp. 2182, 1965.
[12] L. H. Thomas. Elliptic problems in linear difference equations over a network. *Watson Sci. Comput. Lab. Rept., Columbia University, New York*, vol. 1, pp. 71, 1949.
[13] Van der H. A. Vorst. Bi-cgstab: A fast and smoothly converging variant of bi-cg for the solution of non-symmetric linear systems. *SIAM Journal on scientific and Statistical Computing*, vol. 13, n. 2, pp. 631–644, 1992.
[14] Y. Wang, M. Baboulin, K. Rupp, O. Le Maitre, and Y. Fraigneau. Solving 3d incompressible navier-stokes equations on hybrid cpu/gpu systems. In *High Performance Computing Symposium (HPC'14)*. Association for Computing Machinery, 2014.
[15] R. J. Labeur and G. N. Wells. Energy stable and momentum conserving hybrid finite element method for the incompressible navier–stokes equations. *SIAM Journal on Scientific Computing*, vol. 34, n. 2, pp. A889–A913, 2012.
[16] M. Naumov, M. Arsaev, P. Castonguay, J. Cohen, J. Demouth, J. Eaton, S. Layton, N. Markovskiy, I. Reguly, N. Sakharnykh, and others. Amgx: A library for gpu accelerated algebraic multigrid and preconditioned iterative methods. *SIAM Journal on Scientific Computing*, vol. 37, n. 5, pp. S602–S626, 2015.
[17] J. Ahrens, B. Geveci, C. Law, C. Hansen, C. Johnson, and others. 36-paraview: An end-user tool for large-data visualization. *The visualization handbook*, vol. 717, pp. 50038–1, 2005.
[18] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, vol. 9, n. 03, pp. 90–95, 2007.
[19] U. Ghia, K. N. Ghia, and C. Shin. High-re solutions for incompressible flow using the navier-stokes equations and a multigrid method. *Journal of computational physics*, vol. 48, n. 3, pp. 387–411, 1982.
[20] B.-N. Jiang, T. Lin, and L. A. Povinelli. Large-scale computation of incompressible viscous flow by least-squares finite element method. *Computer Methods in Applied Mechanics and Engineering*, vol. 114, n. 3-4, pp. 213–231, 1994.
[21] J.-Y. Yang, S.-C. Yang, Y.-N. Chen, and C.-A. Hsu. Implicit weighted eno schemes for the three-dimensional incompressible navier–stokes equations. *Journal of Computational Physics*, vol. 146, n. 1, pp. 464–487, 1998.
[22] J. U. Brackbill, D. B. Kothe, and H. M. Ruppel. Flip: a low-dissipation, particle-in-cell method for fluid flow. *Computer Physics Communications*, vol. 48, n. 1, pp. 25–38, 1988.