PolyMAC: A staggered finite volume method on general meshes for incompressible Navier-Stokes equations

Pierre-Loïc Bacq, Antoine Gerschenfeld, Michael Ndjinga

PolyMAC: A Staggered Finite Volume Method on General Meshes for Incompressible Navier-Stokes equations

Pierre-Loïc Bacq[1], Antoine Gerschenfeld[1], and Michael Ndjinga[1]

[1]Université Paris-Saclay, CEA - DES/ISAS/DM2S/STMF, F91191 Gif-sur-Yvette, France

March 18, 2024

Abstract

We consider the numerical resolution of the incompressible Navier-Stokes equations. We present a new compatible Finite Volume discretisation that generalises the famous Marker-and-Cell (MAC) method to polyhedral meshes that we call PolyMAC. In the first part of the paper, we recall the principles of compatible schemes and detail the key operators of the discretisation. The convergence and robustness of PolyMAC is assessed numerically on a benchmark from the FVCA conferences. We consider a problem of industrial complexity that allows us to confirm the robustness of PolyMAC on complex problems. The second part of the article is dedicated to the efficient numerical resolution of the resulting linear system. Concretely, we use a PISO-like prediction-correction approach and develop efficient preconditioners for linear systems. In particular, we show that the saddle-point system arising from the correction step is very challenging for iterative methods on distorted meshes. In this work, we develop a robust preconditioner based on an algebraic transformation of the system. In particular, this new preconditioner shows impressive convergence on problems of industrial complexity.

1 Introduction

We consider the discretisation of the incompressible Navier-Stokes equations

$$\begin{cases} \partial_t \vec{u} + (\vec{u}\cdot\nabla)\vec{u} - \nu\Delta\vec{u} + \nabla p &= \vec{f}\,, \\ \nabla\cdot\vec{u} &= 0\,, \end{cases} \qquad \textbf{in } \Omega\,, \tag{1}$$

where Ω is a 2D or 3D domain. $\vec{u}$ is the velocity of the fluid, p the pressure and $\nu > 0$ the viscosity. The discretisation of Equation (1) is at the core of Computational Fluid Dynamics (CFD) and has been studied abundantly. Most common methods include Finite Differences (FD) [15, 40], Finite Elements (FE) [22, 36] and Finite Volumes (FV) [21, 29] schemes.

In this article, we present a *mimetic* or *compatible* FV method that is called PolyMAC, for reasons specified below. Mimetic methods aim at preserving continuous relations at the discrete level, such as conservation laws. In particular, the discrete representation of the unknowns is done in agreement with their physical natures. Mimetic methods have been

developed since the 1950s and comprehensive summary of their development can be found in [28] while a modern introduction is presented in [11, Chapter 2].

In general, FV methods are appreciated in industrial contexts for their conservative properties [17] which make it easy to ensure the conservation of key variables at the discrete level such as the mass or energy. It is especially crucial in the context of nuclear operation and safety: most thermal-hydraulics simulation codes (CATHARE, TRACE, RELAP, FLICA,...) rely on FV methods [9, 10]. Moreover, these codes favour staggered spatial discretisations similar to the MAC scheme [23] which are known to avoid perturbations caused by spurious modes.

However, MAC schemes are defined on Cartesian grids which strongly limit their application range. Industrial applications are more and more sophisticated and unstructured meshes with polyhedral elements are necessary to yield suitable discrete models. However, few efforts have been made to extend the MAC scheme to finite volume on general meshes - see however [19, 31]. More recently, Bonelle and Ern developed a mimetic FV discretisation of Stokes equations [12, 11], which was further extended to Navier-Stokes for face-based schemes [30]. Beltman *et al* also present a discretisation of the incompressible Navier-Stokes equations on a polytopal mesh in [7].

In this paper, we propose a new mimetic scheme which generalizes the MAC scheme to polyhedral meshes: PolyMAC. This scheme is available on the *open-source* platform TRUST[1] [14] maintained by the Commissariat à l'Energie Atomique et aux Energies Alternatives (CEA). The development of this numerical scheme was subject to the following requirements:

- Suitable for polyhedral and non-conforming meshes,
- Similar properties of convergence, stability and conservation as MAC,
- Equivalent to the MAC scheme on Cartesian grids.

We offer a detailed presentation of PolyMAC along a numerical analysis showing its robustness with respect to the mesh.

PolyMAC can be seen as a generalisation of the MAC-scheme to polyhedral meshes. As such, it reproduces the staggering of the unknowns of the original scheme: the pressure is located at the center of the cells while the velocity is discretised by its normal components at the faces. The accuracy and robustness of PolyMAC is evaluated against a benchmark inspired by the FVCA conferences [24, 18, 13] and include very fine and distorted meshes. We use those results to evaluate the convergence order of PolyMAC in both velocity and pressure. We extend this analysis to a problem of industrial complexity that represents an assembly of nuclear rods for a sodium-cooled fast reactor. We show that the robustness of PolyMAC still holds on this complex case.

Finally, one of the major contributions of this work lies in the development of a robust iterative method to solve the linear system resulting from the discretisation. Indeed, Navier-Stokes equations lead to a *double* saddle-point system which is hard to solve with iterative solvers [1] and in practice, engineers often fall back on direct solvers. Here, we present an efficient method based on a PISO-like approach [25, 26] which allows to solve the resulting system with only iterative solvers. We highlight the complexity of the systems resulting from distorted meshes and show that classical block preconditioners are inefficient. Instead, we propose an innovative block preconditioner that presents an increased robustness and yields satisfactory results on all test cases.

[1]Note that there are currently three versions of PolyMAC available on TRUST. A brief description of each can be found in [3]. We focus here on the first PolyMAC version, called PolyMAC I in [3].

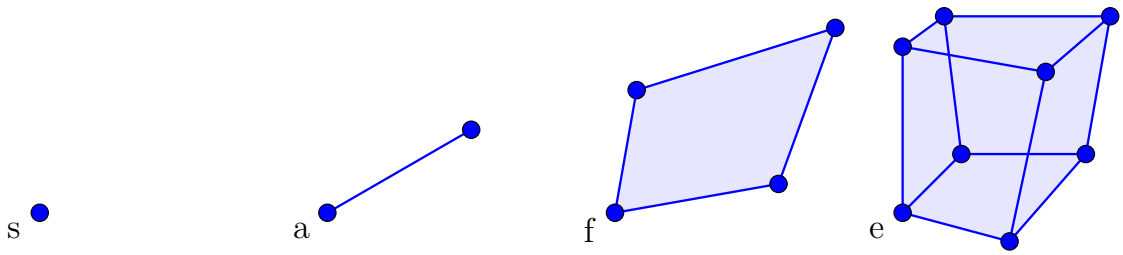


Figure 1: Control Volumes on the primal mesh for PolyMAC

The remaining of this paper is organised as follows. In Section 2, we describe briefly mimetic methods and introduce the PolyMAC discretisation. In Section 3, the benchmark is introduced and the convergence analysis is performed to demonstrate numerically the robustness of the approach. In Section 4, we introduce an iterative method to solve the linear systems resulting from the PolyMAC discretisation. Finally, conclusions are drawn in Section 5.

2 The PolyMAC scheme

PolyMAC belongs to the class of *mimetic* methods that have been developed since the 50s - see [28] for a historical review of such methods. The main characteristic of mimetic methods is to preserve some mathematical properties at the discrete level - for instance conservation laws or exact identities - even on polyhedral meshes. First, the experience shows an increase in robustness and accuracy. Second, a scheme that conserves physical quantities - such as the mass in the incompressible Navier-Stokes equations - is very satisfying for the end users. Among all variations of mimetic methods that have been developed through the years, PolyMAC is a Finite Volume method that lies in the framework of Compatible Discrete Operators (CDO) [11].

As for all FV methods, the unknowns and equations are integrated over control volumes built on the mesh. The precise choice of control volumes defines the method and in the case of CDO, they are chosen according to the physical nature of the fields [11]:

- a potential field will be discretised at vertices $s \in S$ (dim=0),
- a circulation along edges $a \in A$ (dim=1),
- a flux at the faces $f \in F$ (dim=2),
- a density on volumes $e \in E$ (dim=3).

These four types of control volumes are illustrated in Figure 1 and make up the *primal* mesh. We make the following hypotheses in PolyMAC:

- faces f are planar, with the center of gravity $\vec{x}_f$
- cells e are star-shaped with respect to their center of gravity $\vec{x}_e$, *i.e.* $\forall \vec{x} \in e, [\vec{x}_e, \vec{x}] \subset e$.

A key idea of CDO schemes is the explicit use of a dual mesh for the discretisation: the unknowns as well as the equations can be discretised on the primal and the dual mesh. Note that both meshes interact during the discretization process but the dual mesh does not need to be seen explicitly by an external user.

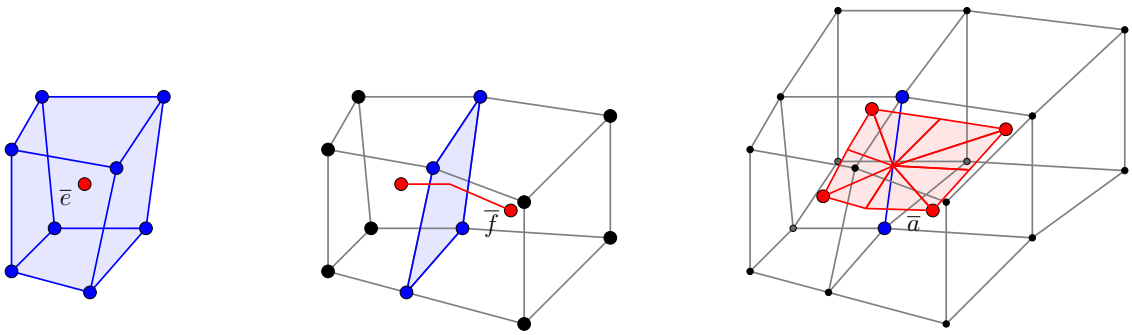


Figure 2: Control Volumes on the dual mesh for PolyMAC

There are several ways of defining a dual mesh; in this work, we consider a barycentric dual mesh. Three *dual* control volumes are defined, as illustrated in Figure 2:

- dual cells $\overline{e} \in \overline{E}$ (dim=0): center of gravity of cell e. Note that $\#\overline{E} = \#E$.
- dual faces $\overline{f} \in \overline{F}$ (dim=1): for the face $f \in F$, the union of the segments joining $\vec{x}_f$ with $\vec{x}_{am}$ and $\vec{x}_{av}$, the centers of gravity of the upstream and downstream cells respectively. Note that $\#\overline{F} = \#F$.
- dual edges $\overline{a} \in \overline{A}$ (dim=2): for the edge $a \in A$, the union of the surfaces defined by the dual faces $\overline{f}$ - of each face f surrounding a - joining $\vec{x}_a$, the middle of the edge a. Note that $\#\overline{A} = \#A$.

We denote by $|\cdot|$ the measure of a control volume, *e.g.* $|e|$ is the volume of cell e. Moreover, $\vec{n}_f$ is the normal vector to the face f and $\vec{t}_a$ is the tangential vector to the edge a.

In the FV framework, the discretization of a field F is realized by the integration of this field on some control volumes. According to the principles of CDO schemes, the following choices are made for PolyMAC:

- the velocity $\vec{v}$ is discretised by the average of its normal components at the faces:

$$[v(x,t)]_f \;=\; \frac{1}{|f|}\int_f (\vec{v}\cdot\vec{n}_f)\,\mathrm{d}S\,, \tag{2}$$

- the vorticity $\vec{\omega}$ by the average of its tangential components at the edges:

$$[\omega(x,t)]_a \;=\; \frac{1}{|a|}\int_a (\vec{\omega}\cdot\vec{t}_a)\mathrm{d}x\,, \tag{3}$$

- the pressure p by its value at the dual cell:

$$[p(x,t)]_{\overline{e}} \;=\; p(x_e,t)\,. \tag{4}$$

We define similarly the following quantities

- at the cells e:

$$[\cdot]_e \;=\; \frac{1}{|e|}\int_e \cdot\,\mathrm{d}V\,, \tag{5}$$

- at the dual faces $\overline{f}$:

$$[\cdot]_{\overline{f}} \;=\; \frac{1}{|\overline{f}|}\int_{\overline{f}} \vec{\cdot}.\vec{\tau}_{\overline{f}}\mathrm{d}x\,, \tag{6}$$

- at the dual edges $\overline{a}$:

$$[\cdot]_{\overline{a}} \;=\; \frac{1}{|\overline{a}|}\int_{\overline{a}} \vec{\cdot}.\vec{n}_{\overline{a}}\mathrm{d}S\,. \tag{7}$$

We denote by $[\cdot]_I$ the column vector containing all values $[\cdot]_i$ for $i \in I$.

This choice combined with appropriate control volumes for the discretization of the equations and with the classical integral theorems leads to *exact* discrete operators for the gradient, the divergence and the curl - in the sense that the CDO does not introduce any consistency error, as we will show in Section 2.1. Moreover, such a discretisation process helps to represent correctly the null space of these operators and to prevent the apparition of spurious modes in the numerical solutions [11].

Approximations will still be required for other operators which will introduce some error in the numerical scheme. Such operators will be called *approximated* in the following sections.

2.1 Exact operators

As said above, the definition of the control volumes in a CDO scheme is done so that some operators are exact. In other words, the discretisation on control volumes A of a field $O(F)$ resulting of the application of a linear operator O on a field F discretised on control volumes B can be written as

$$[O(F)]_A = O_{disc}\,[F]_B\ ,$$

where O_{disc} is the (exact) discrete representation of the operator O. It is a matrix of dimensions $N_A \times N_B$. There are four such exact operators in PolyMAC.

Discretisation of the gradient: The discretisation at the dual faces $\overline{f} \in \overline{F}$ of the gradient of a field defined at the dual cells $\overline{e} \in \overline{E}$ - such as the pressure in the case of PolyMAC - is such an operator. Indeed, the gradient theorem for line integrals yields:

$$\int_{\overline{f}} \nabla p \cdot \vec{\tau}_{\overline{f}} \;=\; [p]_{\overline{e}_{av}} - [p]_{\overline{e}_{am}}\ .$$

In other words,

$$[\nabla p]_{\overline{f}} \;=\; \frac{[p]_{\overline{e}_{av}} - [p]_{\overline{e}_{am}}}{|\overline{f}|} \;=\; \frac{1}{|f||\overline{f}|}(G\,[p]_{\overline{E}})_{\overline{f}}\,, \tag{8}$$

where $[p]_{\overline{e}_{av}}$ is downstream with respect to the face f and $[p]_{\overline{e}_{am}}$ upstream. The matrix G is of dimension $\#\overline{F} \times \#\overline{E} = \#F \times \#E$.

Discretisation of the divergence: The discretisation at the cells $e \in E$ of a field discretised by its normal components to the faces $f \in F$ - as is the velocity $\vec{v}$ in PolyMAC - can be deduced from Ostrogradsky theorem [5]:

$$\int_e \nabla \cdot \vec{v} \;=\; \int_{\partial e} \vec{v} \cdot \vec{n}_{\partial e}\,.$$

In terms of the PolyMAC variables, we have:

$$[\nabla \cdot \vec{v}]_e \;=\; \frac{1}{|e|}\sum_{f \mathrm{V} e} |f|\,[v]_f \;=\; \frac{1}{|e|}(D\,[v]_F)_e\,, \tag{9}$$

where $f\mathrm{V}e$ indicates the sum over all faces f of the cell e and the normal components of the velocity are pointing outward of the cell e. Moreover, the matrix D is of dimension $\#E \times \#F = \#\overline{E} \times \#\overline{F}$ and the discretisation choices made in Equations (8) and (9) ensure

$$G \ = -D^T \tag{10}$$

where $\cdot^T$ indicates the transpose of the matrix.

Discretisation of the curl: The curl operator will appear twice in the continuous form of the incompressible Navier-Stokes equations as will be written later in Equation (23): in the momentum equation and in the definition of the vorticity. As before, we will use the integral theorems of vector calculus to determine an exact discretisation of these operators - in this case, the Green-Stokes theorem [5]:

$$\int_S \nabla \times \vec{v} \cdot \vec{n}_S \mathrm{d}S \ = \ \int_{\partial S} \vec{v} \cdot \vec{t}_{\partial S}$$

where ∂S is positively oriented with respect to S. In the primal and dual meshes, we have two kinds of surfaces: the primal faces f and the dual edges $\overline{a}$.

First, on the primal faces, we get the following result:

$$[\nabla \times \vec{\omega}]_f \ = \ \frac{1}{|f|} \sum_{a \mathrm{V} f} |a| \, [\omega]_a \ = \ \frac{1}{|f||\overline{f}|} (R^F \, [\omega]_A)_f \tag{11}$$

where we have implicitly considered the curl of the vorticity which makes sense given we discretize the vorticity by its tangential component at the edges. As before, $a\mathrm{V}f$ indicates that the sum is taken over all the edges a surrounding the face f oriented anti-clockwise with respect to the normal $\vec{n}_f$. The matrix R^F is then of dimensions $\#F \times \#A = \#\overline{F} \times \#\overline{A}$.

On the dual edges, the Green-Stokes theorem yields:

$$[\nabla \times \vec{v}]_{\overline{a}} \ = \ \frac{1}{|a|} \sum_{\overline{f} \mathrm{V} \overline{a}} |\overline{f}| \, [v]_{\overline{f}} \ = \ \frac{1}{|a||\overline{a}|} (R^A \, [v]_{\overline{F}})_{\overline{a}} \tag{12}$$

where we have implicitly considered the curl of the velocity. $\overline{f}\mathrm{V}\overline{a}$ indicates that the sum is taken over all dual faces $\overline{f}$ surrounding the dual edge $\overline{a}$ oriented anti-clockwise with respect to the tangent $\vec{t}_a$ of the edge a.The matrix R^A is then of dimensions $\#\overline{A} \times \#\overline{F} = \#A \times \#F$. Moreover, the normalisation in Equations (11) and (12) yields

$$R^A \ = \ (R^F)^T \, . \tag{13}$$

2.2 Approximated operators

It is necessary to define relations between the different control volumes. For instance, a field that is discretised at the faces may be needed at the dual faces to discretize another term of the equations. This is done by the so-called *Hodge* operator at the faces. However, those operators are not exact and introduce some error in the discretisation process, hence the name *approximated* operators. Four of them are needed in PolyMAC.

- The discretisation of a vector field at the cells $[\vec{v}]_E$ from its discretisation at the faces $[v]_F$

$$[\vec{v}]_e \ \approx \ \frac{1}{|e|} \left(\Pi^F \, [v]_F \right)_e \, . \tag{14}$$

This relation is established in [35] and is based on the integral indentity

$$\int_e \vec{v}\mathrm{d}V + \int_e \vec{x}(\nabla\cdot\vec{v})\mathrm{d}V \;=\; \sum_{f\mathrm{V}e}\int_f v_f\vec{x}\mathrm{d}S\,. \tag{15}$$

The second term can be neglected at first order and we get

$$[\vec{v}]_e \;\approx\; \frac{1}{|e|}\sum_{f\mathrm{V}e}[v]_f\,|f|(\vec{x}_e-\vec{x}_f)\,, \tag{16}$$

which can be rewritten as Equation (14).

- The discretisation of a vector field at the cells $[\omega]_E$ from its discretisation at the edges $[\omega]_A$

$$[\vec{\omega}]_e \;\approx\; \frac{1}{|e|}\left(\Pi^A\,[\omega]_A\right)_e\,. \tag{17}$$

This relation is established in a similar fashion as above [35].

- The discretisation of a vector field at the dual faces $[v]_{\overline{F}}$ from its discretisation at the faces $[v]_F$ (Hodge operator at the faces)

$$[v]_{\overline{f}} \;\approx\; \frac{1}{|f||\overline{f}|}\left(M^{(f)}\,[v]_F\right)_{\overline{f}}\,. \tag{18}$$

This operator is not unique but satisfy local properties of symmetry and consistency [11]. The discretisation of the velocity at the dual faces is expressed as

$$|\overline{f}|\,[v]_{\overline{f}} \;=\; \int_{\vec{x}_{e_{am}}}^{\vec{x}_f}\vec{v}\cdot\mathrm{d}\vec{x} + \int_{\vec{x}_f}^{\vec{x}_{e_{av}}}\vec{v}\cdot\mathrm{d}\vec{x} \;=\; |\overline{f}_{e_{am}}|\,[v]_{\overline{f},e_{am}} + |\overline{f}_{e_{av}}|\,[v]_{\overline{f},e_{av}}\;, \tag{19}$$

where e_{am} is the upstream cell and e_{av} the downstream cell. Each term can be expressed at first order

$$|\overline{f}_e|\,[v]_{\overline{f},e} \;=\; \int_{\vec{x}_e}^{\vec{x}_f}\vec{v}\cdot\mathrm{d}\vec{x} \;\approx\; [\vec{v}]_e\cdot(\vec{x}_f-\vec{x}_e) = \frac{1}{|f||\overline{f}_e|}\sum_{f'\mathrm{V}e}((\Pi_e^F)^T\Pi_e^F)_{ff'}\,[v]_{f'}\;. \tag{20}$$

The matrix defined in the right hand side of Equation (20) is not positive definite because it is not of maximal rank. Mimetic methods usually define the local Hodge operator by the sum [7]

$$M_e^{(f)} \;=\; (\Pi_e^F)^T\Pi_e^F + (N_e^F)^T P_e N_e^F\,, \tag{21}$$

where N_e^F is a matrix whose columns are vectors of an orthonormal basis of Π_e^F and P_e is a symmetric positive definite matrix. It typical to choose $P_e=\gamma I$, where I is the identity. $\gamma=|e|/2$ makes $M_e^{(2)}$ a diagonal matrix on a Cartesian mesh. The matrices $M_e^{(f)}$ can then be assembled into the global matrix $M^{(f)}$ from Equation (18).

- The discretisation of a vector field at the dual edges $[\omega]_{\overline{A}}$ from its discretisation at the edges $[\omega]_A$ (Hodge operator at the edges)

$$[\omega]_{\overline{a}} \;\approx\; \frac{1}{|a||\overline{a}|}\left(M^{(a)}\,[\omega]_A\right)_{\overline{a}}\,. \tag{22}$$

can be obtained in a similar fashion.

As opposed to the exact operators described in Section 2.1, all these operators introduce an error in the discretisation. Note also that they are not uniquely defined. In the case of PolyMAC, they are of order 1. A more detailed analysis of the discrete Hodge operators can be found in [11, 7].

2.3 The Navier-Stokes equations

In this section, we will now present how the incompressible Navier-Stokes equations (1) are discretised in PolyMAC. We use a similar approach as [7] and [12]: the diffusion is expressed in terms of the vorticity $\vec{\omega}$ thanks to the incompressibility constraint. Here, the convection term is also written as the divergence of the tensor product of the velocity with itself $\nabla \cdot (\vec{v} \otimes \vec{v})$. The equations become

$$\begin{aligned} \partial_t \vec{v} + \nabla \cdot (\vec{v} \otimes \vec{v}) - \nu \nabla \times \vec{\omega} + \nabla p &= \vec{f}, \\ \nabla \times \vec{v} - \vec{\omega} &= 0, \\ \nabla \cdot \vec{v} &= 0, \end{aligned} \tag{23}$$

where the first equation represents the conservation of momentum, the second equation is the definition of the vorticity and the last one the continuity equation. Along those three equations, there are now three unknowns: the velocity $\vec{v}$, the vorticity $\vec{\omega}$ and the pressure p.

In PolyMAC:

- the velocity $\vec{v}$ is discretised by the average of its normal component at the faces $[v]_f$
- the vorticity $\vec{\omega}$ is discretised by the average of its tangential component at the edges $[\omega]_a$
- the pressure p is discretised by its value at the dual cells $[p]_{\overline{e}}$, *i.e.* by its value at the center of gravity of the cell e,

whereas the following choice was made for the equations:

- the momentum equation is discretised on the dual faces $\overline{f} \in \overline{F}$
- the definition of the vorticity is discretised on the dual edges $\overline{a} \in \overline{A}$
- the continuity equation is discretised at the cells $e \in E$.

For example, this means that the momentum equation is integrated along each dual face and the various terms appearing in this equation need to be expressed on these control volumes. This is explicited in the next paragraphs where we consider each equation in turn.

Momentum equation: As stated above, the momentum equation is discretised at the dual faces and each term must be expressed on those control volumes.

- Time derivative $\partial_t \vec{v}$. Since the velocity is discretised at the (primal) faces, we need to use the Hodge operators $M^{(f)}$ to deduce the expression of this term on the dual faces:

$$|f| \left|\overline{f}\right| [\partial_t \vec{v}]_{\overline{f}} = (M^{(f)} \partial_t [v]_F)_{\overline{f}}. \tag{24}$$

- Convective term $\nabla \cdot (\vec{v} \otimes \vec{v})$. We consider the following approach:

  1. Build the velocity at the cells $[\vec{v}]_e$ from the velocity at the faces by the relation (14): $[\vec{v}]_e = \frac{1}{|e|}(\Pi^F [v]_F)_e$.
  2. Convect the velocity at the cells $[\vec{v}]_e$ by a convection scheme[2] which yields $[\nabla \cdot (\vec{v} \otimes \vec{v})]_e$.

[2] PolyMAC offers the choice between an upwind or a centered scheme

3. Integrate the tangential component of the convection term at the elements on the dual faces which ultimately yields

$$|f|\left|\overline{f}\right|\left[\nabla\cdot(\vec{v}\otimes\vec{v})\right]_{\overline{f}}=(C^F([v]_F)\,[v]_F)_{\overline{f}}. \tag{25}$$

This approach was chosen to avoid the apparition of spurious modes in the velocity.

- Diffusion term $\nu\nabla\times\vec{\omega}$. Since $\vec{\omega}$ is discretised at the edges, the exact operator R^F gives the discretisation of $\nabla\times\vec{\omega}$ at the faces $[\nabla\times\vec{\omega}]_F=R^F\,[\omega]_A$. To get the discretisation at the dual faces, we just need to apply the Hodge operator $M^{(f)}$:

$$|f|\left|\overline{f}\right|\left[\nu\nabla\times\vec{\omega}\right]_{\overline{f}}=\nu(M^{(f)}R^F\,[\omega]_A)_{\overline{f}}. \tag{26}$$

- Pressure gradient ∇p. Since the pressure is discretised at the dual faces, we have directly

$$|f|\left|\overline{f}\right|\left[\nabla p\right]_{\overline{f}}\;=\;(G\,[p]_{\overline{E}})_{\overline{f}}. \tag{27}$$

Vorticity equation: A similar reasoning is applied to both terms of the equation. This time the control volumes are the dual edges.

- Curl of the velocity $\nabla\times\vec{v}$. The operator R^A allows to compute an exact representation of the curl operator from a field discretised on the dual faces. Since the velocity is discretised at the faces, we need first to apply the Hodge operator $M^{(f)}$. Ultimately, we get

$$|a||\overline{a}|\left[\nabla\times\vec{v}\right]_{\overline{a}}\;=\;(R^AM^{(f)}\,[v]_F)_{\overline{a}}. \tag{28}$$

- Diagonal term $\vec{\omega}$. The vorticity $\vec{\omega}$ is discretised at the edges and we simply need to apply the Hodge operator $M^{(a)}$ to get the discretisation at the dual edges:

$$|a||\overline{a}|\left[\omega\right]_{\overline{a}}\;=\;(M^{(a)}\,[\omega]_A)_{\overline{a}}. \tag{29}$$

Continuity equation: The last equation is discretised at the cells.

- Divergence $\nabla\cdot\vec{v}$. The definition of the exact operator D yields

$$|e|\left[\nabla\cdot\vec{v}\right]_e\;=\;(D\,[v]_F)_e. \tag{30}$$

Linear system: By combining all terms and equations into a single system, we get

$$\begin{pmatrix}\frac{M^{(f)}}{\Delta t}+C^F([v]_F^t) & M^{(f)}R^F & G\\ R^AM^{(f)} & -\frac{1}{\nu}M^{(a)} & 0\\ -D & 0 & 0\end{pmatrix}\begin{pmatrix}[v]_F^{t+\Delta t}\\ \nu\,[\omega]_A^{t+\Delta t}\\ [p]_{\overline{E}}^{t+\Delta t}\end{pmatrix}\;=\;\begin{pmatrix}\frac{M^{(f)}}{\Delta t}\,[v]_F^t\\ 0\\ 0\end{pmatrix}. \tag{31}$$

The time derivative is treated by an implicit scheme, except in the convective term $C^F([v]_F^t)$ where it is explicited. This can be seen as a Picard linearisation applied to the Navier-Stokes equations. This system is presents a double saddle-point structure and is challenging to solve as a whole. In this work, we study a splitting approach similar to PISO [25] - see Section 4.

Boundary conditions: To complete the discretisation of system (1) we need to take into account the boundary conditions, which may be imposed on the pressure and the velocity.

Concerning the pressure, the situation is quite simple. We usually encounter a condition directly on the pressure in the form $[p]_f = p_{boundary}$. To include it into the discretisation, we just need to update the term $G\,[p]_{\overline{E}}$.

However, the situation is more complex for the velocity. Indeed, boundary conditions on the velocity can come in different forms and affect different terms in system (31).

- If the normal component of the velocity is imposed on a face f, the equation related to $\overline{f}$ must be updated by $[v]_f = v_{boundary}$.
- If the tangential component of the velocity v_{tan} is imposed on a face f, the equations where the edges of f appear must be modified to take into account v_{tan} in the rotational $R^A M^{(f)}$. In order to do this, the integral on the boundary of the involved dual edges must be closed, as described on Figure 3. This modification is necessary as the boundary of the domain are not included in the dual edges.
- If a no-slip condition is imposed on a face f, the equations of the vorticity at the edges a of f can be replaced by $[\omega]_a = \omega_{boundary} = 0$.

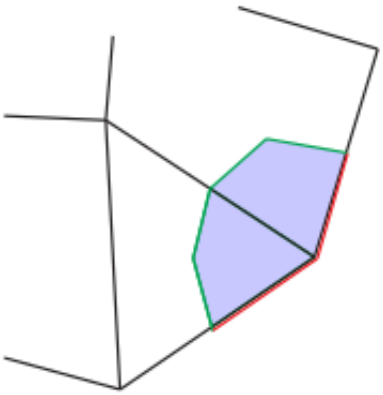


Figure 3: Closing of the integration path to take into account the boundary conditions imposing the tangential velocity. The dual edge is in blue, the integration path on the inside of the domain is in green and the additional contributions from the boundary conditions are in red.

3 Numerical study of the convergence

The convergence of PolyMAC is investigated here numerically on some test problems. We consider a typical Navier-Stokes problem that we discretise on a series of meshes chosen to illustrate the robustness of our method. These meshes were defined in benchmarks established and presented during previous cycles of the FVCA conferences [24, 18, 13] and they represent a discrete unit square in 2D and a discrete unit cube in 3D. They are listed in Table 1 and some of them are illustrated on Figure 4. Note that their characteristics are diverse: there are various geometric shapes (triangles, polygons, ...), some are 2D while other are 3D, some unstructured and some non-conforming that present hanging nodes. While this list is not exhaustive, the variety of meshes that it covers gives some reassurance about the robustness of the approach. To characterize the convergence of PolyMAC, each mesh

is refined successively. The number of sizes considered for each mesh is indicated in the column *Refinement*, while the largest number of cells - *i.e.* the number of cells on the finest mesh - is given in the column *Cells*. In the column *Matrix Size*, we indicate the size of the matrix solved during the Correction step, see Section 4.1. The dimension of the problem is indicated by the column *Dimension*. A small star * next to the name of a mesh indicates the presence of hanging nodes in this mesh.

Name	Refinement	Cells ($\times 10^4$)	Matrix Size ($\times 10^5$)	Dimension
`Cartesian`	7	6.5	1.9	2D
`Triangles`	5	10.4	2.6	2D
`Quadrangles`	7	6.5	1.9	2D
`Polygons`	6	5.3	2.1	2D
`Locally Refined 2D`*	5	16.4	4.9	2D
`Kershaw 2D`	6	1.0	0.31	2D
`Hexahedras`	5	3.3	1.3	3D
`Locally Refined 3D`*	4	9.0	3.7	3D
`Kershaw 3D`	4	26.2	16.1	3D
`Checkerboard`*	5	14.7	7.4	3D
`Voronoi`	5	0.03	0.03	3D
`Tetrahedras`	5	6.1	1.8	3D
`Random`	4	3.2	2.4	3D
`Prism`	4	12.8	4.5	3D
`PrismHexa`	4	6.7	/	3D

Table 1: Characteristics of the different meshes used in the benchmark. The star * indicates the presence of hanging nodes.

The test problem is a standard rotating Navier-Stokes problem described in [13] with a viscosity $\nu = 10^{-2}$ (hence, a Reynolds number of 10^2). The discretisation is run by the TRUST [14] software on a Intel(R) Xeon(R) Gold 5222 CPU @ 3.80GHz. Formally, we try to find $\vec{u}$ and p such that:

$$\begin{cases} -\nu\Delta\vec{u} + (\vec{u}\cdot\nabla)\vec{u} + \nabla p &= \vec{f} \quad \text{in } \Omega\,, \\ \nabla\cdot\vec{u} &= 0 \quad \text{in } \Omega\,, \end{cases} \tag{32}$$

where $\vec{u}$ is the velocity and p the pressure and $\nu > 0$ is the viscosity. As stated above, the domain $\Omega \subset \mathbb{R}^d$ ($d = 2$ or 3) is the unit square in 2D and the unit cube in 3D.

More precisely, in 2D, the analytical solution is given by

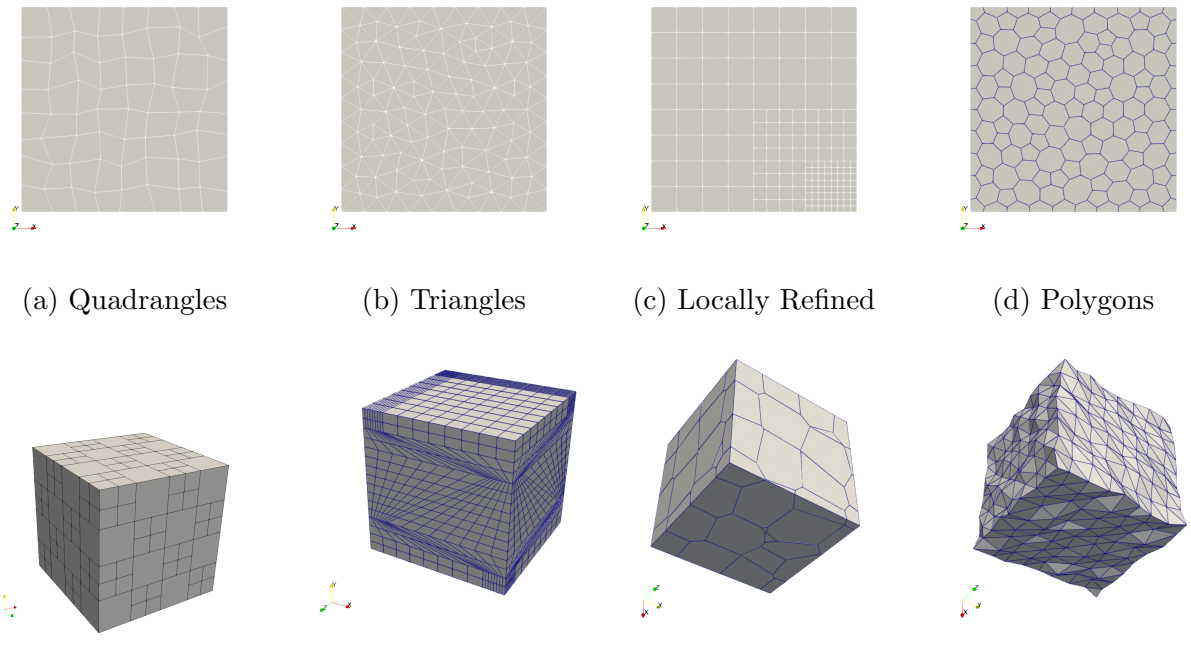
$$\begin{aligned} \vec{u} &= \begin{pmatrix} y \\ -x \end{pmatrix} \\ p &= (x^2+y^2)/2 - \tfrac{1}{3} \end{aligned}\,, \tag{33}$$

while in 3D, the exact solution is given by

$$\begin{aligned} \vec{u} &= \begin{pmatrix} y-z \\ z-x \\ x-y \end{pmatrix}\,, \\ p &= (x^2+y^2+z^2) - xy - xz - yz - \tfrac{1}{4}\,. \end{aligned} \tag{34}$$

The right-hand-side can be computed accordingly. For instance, in 3D we get:

$$\vec{f} = \begin{pmatrix} 2x-y-z \\ 2y-z-x \\ 2z-x-y \end{pmatrix}\,. \tag{35}$$



(a) Quadrangles (b) Triangles (c) Locally Refined (d) Polygons

(e) Checkerboard (f) Kershaw (g) Voronoi (h) Random

Figure 4: Representation of some meshes from the benchmark.

This allows us to evaluate precisely the error made by PolyMAC. Concretely, we evaluate the (relative) error in velocity by

$$\mathrm{err}_v^2 \;=\; \frac{\|\vec{v}-\vec{v}_h\|_2^2}{\|\vec{v}\|_2^2} \approx \frac{\sum_{e\in E}\|\vec{v}_e-[\vec{v}]_e\|^2|e|}{\sum_e\|\vec{v}_e\|^2|e|} \tag{36}$$

where $\vec{v}_e$ is the analytical solution velocity $\vec{v}$ evaluated in the element e, $[\vec{v}]_e$ is the approximated solution built at the elements from the velocity at the faces by the relation (14) and $|e|$ is the volume of the element e. The (relative) error in pressure is computed in a similar fashion:

$$\mathrm{err}_p^2 \;=\; \frac{\|p-p_h\|_2^2}{\|p\|_2^2} \approx \frac{\sum_{e\in E}\|p_e-[p]_{\overline{e}}\|^2|e|}{\sum_e\|p_e\|^2|e|}\,. \tag{37}$$

The convergence rate between two successive mesh refinements l and $l+1$ can then be estimated

$$r \;=\; -d\frac{\log(\mathrm{err}_i[l+1]/\mathrm{err}_i[l])}{\log(\#E(l+1)/\#E(l))}\,, \tag{38}$$

where d is the dimension of the space, $\#E(l)$ is the number of elements in the l^{th} mesh and i takes the values v, p.

The results are shown in Figure 5. The L^2-error in velocity and pressure is shown along the evolution of the number of cells (to the power 1/2 or 1/3 according to the dimension) for all considered meshes. All the cases converge which suggests that PolyMAC is indeed a robust discretisation method. The asymptotic convergence rates for both the velocity and the pressure are reported in Table 2 to give a quantitative insight to the observations. In most cases, the convergence order of the velocity is about 1.0 as expected. It is interesting to note that some super-convergence is observed. This phenomenon has already been observed and documented on meshes with some regularity such as `Cartesian`, `Hexahedras`, or even `Locally Refined 3D` [11, 30, 7]. Here, however, we also observe that phenomenon for very deformed meshes such as `Kershaw 2D` and `Kershaw 3D`. The order of convergence of the

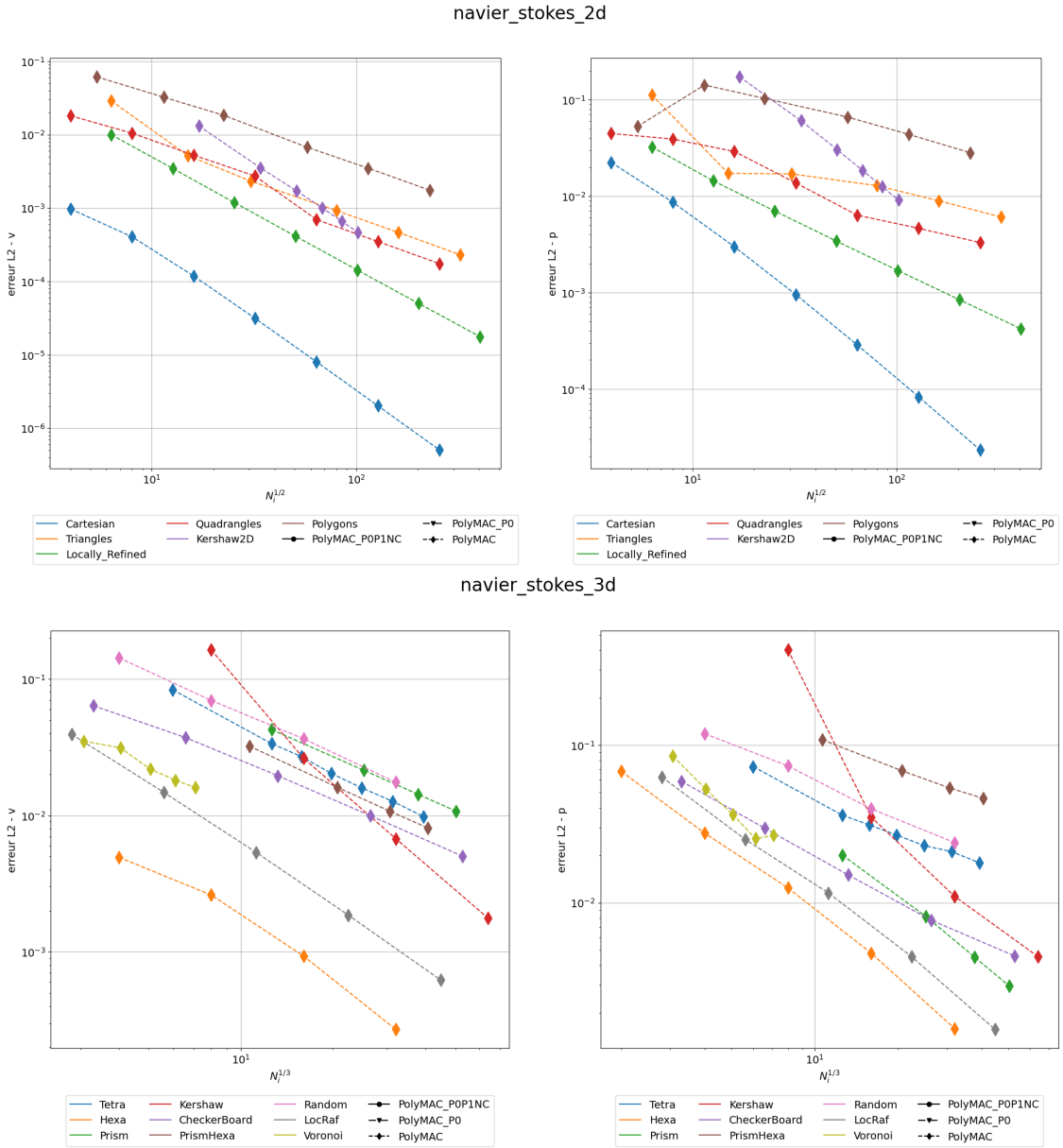


Figure 5: L^2-error in velocity (left) and pressure (right) in 2D (top) and 3D (bottom).

pressure are between 0.5 and 2.0 and very mesh-dependent. Globally, the results are as expected and coherent with the litterature [7]. Note that the super-convergence on `Kershaw 2D` and `Kershaw 3D` is also happening for the pressure, with a convergence rate close to 2.0: PolyMAC seems especially robust on deformed meshes.

3.1 Industrial case

One of the objectives of PolyMAC is to be used in an industrial context, where geometries can be much more complex than the unit square or cube. As a consequence, the meshes used are usually unstructured and very deformed. In this work, we consider the flow through an assembly of nuclear rods. One such mesh is represented in Figure 6. Its structure is much more challenging than the meshes from the FVCA benchmarks, notably there does not exist an analytical solution. The cylindrical holes in the domain represent the fuel rods around which the fluids flows. Furthermore, each rod is circled by a spacer wire that is also visible in Figure 6. This spacer is a known difficulty when modelling sodium-cooled fast reactor

	Velocity	Pressure
Cartesian	2.00	1.83
Triangles	1.00	0.55
Quadrangles	1.00	0.49
Polygons	0.99	0.63
Locally Refined 2D	1.51	1.00
Kershaw 2D	1.92	1.75
Hexahedras	1.78	1.59
Locally Refined 3D	1.58	1.53
Kershaw 3D	1.93	1.27
Checkerboard	0.99	0.75
Voronoi	0.98	1.86
Tetrahedras	1.11	0.70
Random	1.06	0.71
Prism	1.00	1.46
PrismHexa	0.98	0.54

Table 2: Order of convergence in velocity and pressure of the PolyMAC method on the different meshes described in Table 1.

and liquid-metal reactors [37, 27]. It is also a challenge for the mesher which tends to give high aspect ratio cells along it. In this article, we consider tetrahedric cells.

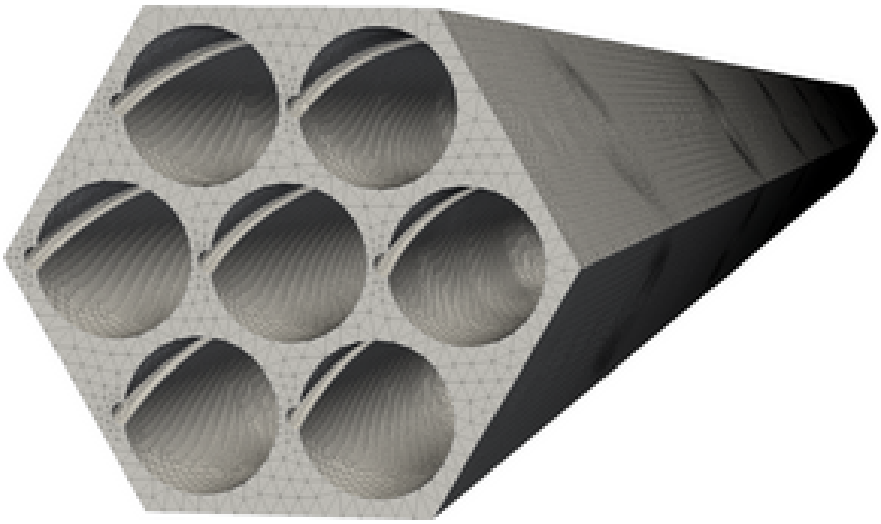


Figure 6: Representation of a meshed assembly from the Rapsodie reactor.

Here, we solve the *time-dependent* incompressible Navier-Stokes equations

$$\begin{cases} \partial_t \vec{v} - \nu \Delta \vec{v} + (\vec{v} \cdot \nabla)\vec{v} + \nabla p &= \vec{f} \quad \text{in } \Omega\,, \\ \nabla \cdot \vec{v} &= 0 \quad \text{in } \Omega\,. \end{cases} \tag{39}$$

The associated boundary conditions specify the pressure at the inlet and the outlet and the velocity along the walls - the nuclear rods are associated to walls. More precisely, we have

$$\begin{aligned} P_{in} &= 10^5 Pa \\ P_{out} &= P_{in} - \rho g z \\ \vec{v}_{wall} &= (0,0,0)\,, \end{aligned}$$

Refinement	1	2	3	4	5	6	7	8
Cells ($\times 10^4$)	0.9	1.8	4.9	21.8	43.5	80.9	127.0	462.4
Matrix Size ($\times 10^5$)	0.3	0.5	1.5	6.4	13.5	24.2	38.1	138.7

Table 3: Sizes of the successive refinements of the Assembly mesh.

where g is the gravitational acceleration and ρ the density of the fluid - here, of water in normal conditions - while z is the height of the assembly. The initial condition is stated as

$$\vec{v}_0 \;=\; (0,0,0)\,.$$

The time discretisation is done by implicit Euler scheme and yields:

$$\begin{cases} \dfrac{\vec{v}^{n+1}-\vec{v}^{n}}{\Delta t} - \nu\Delta\vec{v}^{n+1} + (\vec{v}^{n}\cdot\nabla)\vec{v}^{n+1} + \nabla p^{n+1} \;=\; \vec{f} \quad \text{in } \Omega\,, \\ \nabla\cdot\vec{v}^{n+1} \;=\; 0 \quad \text{in } \Omega\,. \end{cases} \tag{40}$$

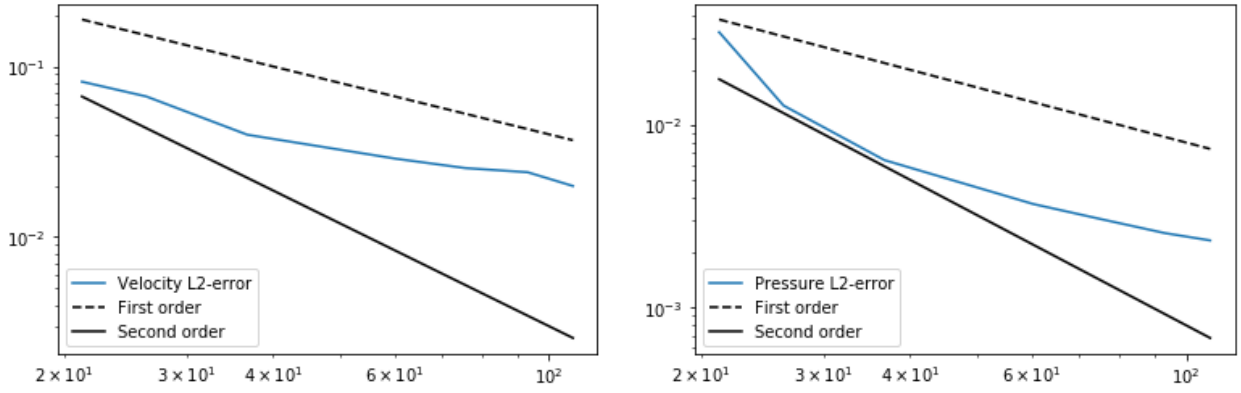
As before, the equations are discretised by the TRUST [14] software on an Intel(R) Xeon(R) Gold 5222 CPU @ 3.80GHz. We use a series of more and more refined meshes whose sizes are shown in Table 3.

On each mesh, the simulation is run until the norm of the time derivative of the velocity is below some threshold that was set at 10^{-8}. The solution on the finest mesh $(\vec{v}_f, p_f)$ is considered as the reference solution and the error made by the coarser meshes is estimated with respect to the couple $(\vec{v}_f, p_f)$. In practice, the computation of err_v and err_p is done by the same formulas (36) and (37) and the convergence rate is estimated as previously with the expression (38). The results are represented in the Figure 7. The blue continuous line represents the evolution of the relative error with the mesh size: on the left, is the velocity error and on the right, the pressure error. The black dashed line represents the first order of convergence and the continuous black line, the second order. The computed convergence rate is indicated in the legends of the Figure 7. As can be seen, the order of convergence is close to 1.0 for the velocity, as was observed on FVCA cases. The order of convergence for the pressure is much higher, close to 3.0. However, the smallest mesh is quite coarse and does not reflect precisely the geometry of the assembly. If we take it out of the study, the convergence rate drops to 0.99, much closer to what we expect from the previous analysis. These results show the robustness of PolyMAC in an industrial context.

4 Numerical Resolution

The key advantage of PolyMAC is its robustness, especially on very large unstructured and distorted meshes. As presented in the previous section, results are still robust even on meshes and geometry of industrial complexity. However, this advantage cannot be fully taken advantage of if we do not have an efficient method to solve the resulting linear systems. It is especially important in an industrial context where one is required to work with (very) large systems. In this work, we consider matrix sizes up to 10^7 unknowns.

The system of linear equations resulting from the discretisation of the incompressible Navier-Stokes equations by PolyMAC is given by Equation (31). This system presents a velocity-pressure saddle-point which makes it hard to solve by iterative methods - see [4, 20] for recent approaches to tackle such systems. To get around this difficulty, alternative approaches have been developed by engineers and mathematicians; a famous example is the



(a) Velocity convergence order: 0.93 (b) Pressure convergence order: 3.24

Figure 7: Convergence curves associated to the velocity (a) and the pressure (b) for the Assembly industrial case.

SIMPLE method developed in the 70s by Spalding and Patankar [34]. In this work, we consider the PISO method introduced in 1984 by Issa [25, 26]. The common idea of these methods is to split the resolution of the global system into several steps.

4.1 The PISO method

The Pressure-Implicit with Splitting Operators - PISO for short - algorithm was developed by Issa in the 80s. The PISO method implements a non-iterative method to solve coupled transport equations. The method is made of two complementary steps: the predictor and the corrector step. Note that the corrector step can be repeated to increase the accuracy of the solver with respect to the time step [25]. PolyMAC is a low-order method and in this context, one corrector step satisfies our accuracy requirements.

The aim of the predictor step is to compute the intermediate velocity $[v]_F^\star$ and vorticity $[\omega]_A^\star$ by solving the momentum equation with the pressure $[p]_{\overline{E}}^t$ at the previous time step t. In other words, the method requires to solve the system

$$\begin{pmatrix} \frac{M^{(f)}}{\Delta t} + C^F([v]_F^t) & M^{(f)} R^F \\ R^A M^{(f)} & -\frac{1}{\nu} M^{(a)} \end{pmatrix} \begin{pmatrix} [v]_F^\star \\ \nu\,[\omega]_A^\star \end{pmatrix} = \begin{pmatrix} \frac{M^{(f)}}{\Delta t}\,[v]_F^t - G\,[p]_{\overline{E}}^t \\ 0 \end{pmatrix}. \tag{41}$$

Note that the resulting velocity $[v]_F^\star$ is not necessarily divergence-free.

In the corrector step, we compute a correction $[\delta v]_F$ to the velocity so that $[v]_F^{t+\Delta t} = [v]_F^\star + [\delta v]_F$ is divergence-free. In the same time, we compute $[\delta p]_{\overline{E}}$ such that $[p]_{\overline{E}}^{t+\Delta t} = [p]_{\overline{E}} + [\delta p]_{\overline{E}}$. To simplify the presentation of the derivation of the corrector step, we write the Navier-Stokes equations in velocity and pressure only.

We are looking for $[v]_F^{t+\Delta t}$ that is divergence free. To do so, we rewrite the momentum equation as

$$\frac{M^{(f)}}{\Delta t}([v]_F^{(t+\Delta t)} - [v]_F^\star) + C^F([v]_F^t)\,[v]_F^\star - \nu\Delta_h\,[v]_F^\star + G\,[p]_{\overline{E}}^{(t+\Delta t)} = f^t, \tag{42}$$

where we write the discrete Laplacian as Δ_h which is expressed as the curl of the vorticity in practice. The only implicit term is the pressure $[p]_{\overline{E}}^{(t+\Delta t)}$. Remember that the momentum equation solved in the predictor step is written as

$$\frac{M^{(f)}}{\Delta t}([v]_F^\star - [v]_F^t) + C^F([v]_F^t)\,[v]_F^\star - \nu\Delta_h\,[v]_F^\star + G\,[p]_{\overline{E}}^{(t)} = f^t. \tag{43}$$

Substracting Equation (43) to Equation (42) and taking the divergence of the result, we get

$$-D\left[\delta v\right]_F + \Delta t \Delta_h \left[\delta p\right]_{\overline{E}} \;=\; 0\,. \tag{44}$$

Taking advantage of the fact that $[v]_{t+\Delta t}$ is divergence free, we can rewrite this last equation as

$$\Delta t \Delta_h \left[\delta p\right]_{\overline{E}} = -D\left[v\right]_F^\star\,, \tag{45}$$

which can also be rewritten as

$$\begin{pmatrix} M^{(f)} & G \\ -D & 0 \end{pmatrix} \begin{pmatrix} \left[\delta v\right]_F \\ \Delta t \left[\delta p\right]_{\overline{E}} \end{pmatrix} \;=\; \begin{pmatrix} 0 \\ -D\left[v\right]_F^\star \end{pmatrix}\,, \tag{46}$$

which ultimately yields

$$\begin{array}{rcl} \left[v\right]_F^{t+\Delta t} & = & \left[v\right]_F^\star + \left[\delta v\right]_F\,, \\ \left[p\right]_{\overline{E}}^{t+\Delta t} & = & \left[p\right]_{\overline{E}}^t + \left[\delta p\right]_{\overline{E}}\,. \end{array}$$

4.2 Iterative approach to the Correction Step

Applying the PISO-like method presented in the previous section to the linear system (31) arising from PolyMAC discretisation requires to solve the two systems (41) and (46). Both systems present a saddle-point but the system (41) can be solved efficiently by a GMRES solver preconditioned by ILU and does not cause any significant difficulties.

However, the system (46) sometimes proves harder to solve even though its structure is rather simple and usual preconditioner for saddle-point systems are assumed to work efficiently. Classical iterative solvers for systems such as (46) usually rely on block preconditioners associated to a Krylov method such as GMRES - see [8] for a review. A block-diagonal preconditioner will have the followind structure:

$$\begin{pmatrix} \tilde{M}^{(f)} & \\ & \tilde{S} \end{pmatrix}\,, \tag{47}$$

where $\tilde{M}^{(f)}$ is a good approximation of the matrix $M^{(f)}$ while $\tilde{S}$ approximates the Schur complement $S = D(M^{(f)})^{-1}G$.

In the present case, $M^{(f)}$ is SPD and can be assimilated to a mass matrix in practice. For such cases, D_M - the diagonal part of $M^{(f)}$ - provides in general a good approximation of $M^{(f)}$. A typical block preconditioner would be

$$\begin{pmatrix} D_M & \\ & DD_M^{-1}G \end{pmatrix}\,, \tag{48}$$

where the second block is then similar to a discrete Laplacian since it is essentially the product of a discrete divergence D and a discrete gradient G. When the preconditioner is applied, this block can be efficiently inverted by a standard algebraic multigrid (AMG) method.

Nevertheless, this standard preconditioner fails to yield an efficient solver for the correction step on the more challenging meshes, as is illustrated in the next Section. The main reason is a loss of diagonal dominance of the matrix $M^{(f)}$ when the cells get distorted: the stencil of $M^{(f)}$ becomes wider and $M^{(f)}$ cannot be approximated by its diagonal D_M anymore. Note that this issue, to the extent of our knowledge, has not been much studied in the litterature related to iterative solvers for saddle point problems. A solution would be

to replace the top left block of P by a better approximation of $M^{(f)}$, for instance an AMG approximation. However, this cannot be done for the inverse of D_M in the approximation of the Schur complement and the results in Section 4.3 will show that using the diagonal D_M in the approximation of the Schur complement is not adequate.

To illustrate the difficulties met with *classical* preconditioners for saddle-point systems, we consider the following block triangular matrix

$$P_{Clas} = \begin{pmatrix} \tilde{M^{(f)}} & \\ -D & DD_M^{-1}G \end{pmatrix}, \tag{49}$$

where $\tilde{M^{(f)}}$ is an AMG approximation of $M^{(f)}$. Notay showed in [32] that such a preconditioner is a good indicator of the efficiency one can expect from a wide range of widely used preconditioners, such as inexact Uzawa or symmetric block Gauss-Seidel. If $M^{(f)}$ and S are well approximated by the diagonal blocks, all these preconditioners yield similar results. On the contrary, if one of these preconditioners struggle, one can expect all of them to struggle. Note that we could have used a block diagonal preconditioner such as (48) to illustrate this behavior. However, in the following, the preconditioner we propose is itself block triangular so we choose P_{Clas} as in Equation (49) to ease the comparison.

We propose a more robust preconditioner for systems like (46) when classical saddle-point preconditioners fail. The key idea is to apply an algebraic transformation to the system (46) and to solve the transformed system instead. We hereafter call this method the *transform-then-solve approach*, since it is similar to what is done in [33, 2, 4]. The solution to the original system is recovered by applying the inverse transformation. The transformation is defined by:

$$\widehat{A} = \begin{pmatrix} M^{(f)} & G \\ -D & \end{pmatrix} \begin{pmatrix} I & -D_M^{-1}G \\ & I \end{pmatrix} = \begin{pmatrix} M^{(f)} & (I - M^{(f)}D_M^{-1})G \\ -D & DD_M^{-1}G \end{pmatrix}. \tag{50}$$

We write $\widehat{C} = DD_M^{-1}G$ and $\widehat{G} = (I - M^{(f)}D_M^{-1})G$. It is interesting to note that $\widehat{C}$ is similar to a discrete Laplacian, since it is essentially the product of a discrete divergence and a discrete gradient. Note that the off-diagonal blocks are first order discrete derivatives. The system (50) has then a structure that is similar to a Stokes system for which *optimal* preconditioners are well-known [16]. State-of-the-art preconditioners are defined by

$$\begin{pmatrix} Q & \\ & \tilde{\widehat{C}} \end{pmatrix}, \tag{51}$$

where Q is the mass matrix[3] which is spectrally equivalent to the Schur complement for Stokes problems [16]. Since $\widehat{C}$ is close to a discrete Laplacian, it is standard to use an AMG method to precondition that block. To approximate the mass matrix Q, it is natural to use the matrix $M^{(f)}$. Since the Schur complement of $\widehat{A}$ is $M^{(f)} + \widehat{G}\widehat{C}^{-1}D$, we choose $Q = 2M^{(f)}$. Finally, to improve the efficiency of the preconditioning, we can consider a block triangular preconditioner such as

$$\begin{pmatrix} 2M^{(f)} & \widehat{G} \\ & \tilde{\widehat{C}} \end{pmatrix}. \tag{52}$$

Indeed, there is no benefit in keeping a symmetric preconditioner since the matrix $\widehat{A}$ is non-symmetric. Furthermore, the application of the triangular preconditioner implies almost no

[3]Note that the unknowns are reversed with respect to Stokes problems: the Laplacian here acts on the pressure.

extra cost with respect to the block diagonal one: only one matvec operation with $\widehat{G}$ and one vector addition are needed. No additional memory is required.

It is possible to simplify the preconditioner (52) by replacing $M^{(f)}$ with its diagonal D_M and we get the preconditioner

$$P_{Trans} = \begin{pmatrix} 2D_M & \widehat{G} \\ & \widehat{\widetilde{C}} \end{pmatrix}. \tag{53}$$

Note that this simplification is not in contradiction with the difficulties related to the loss of diagonal dominance we reported at the start of the Section 4.2. The transformation has brought back weight on the diagonal, notably *via* the block $\widehat{C}$, which allows to replace $M^{(f)}$ by D_M without putting the convergence of the iterative method at risk.

The theoretical analysis of this approach is outside the scope of this paper and will be the subject of future work. We provide a heuristic justification of this approach in the next section *via* numerical results.

4.3 Results

In this section, we compare the transform-then-solve preconditioner P_{Trans} described in the previous section with a classical block diagonal preconditioner P_{Clas}. The results are shown in Tables 4, 5 and 6. We consider the number of iterations needed by GMRES [39] to decrease the relative residual by a factor of 10^{-6}. We set the maximal number of iterations to 200 and if the method does not converge within this limit, we indicate > 199 in the results. If the method diverges (increase of the relative residual), we put / in the results. The first column of each table indicates the relative size of each problem: 1 corresponds to the smallest system and 7 to the largest one. For each problem, the size of the largest system is shown in Table 1 and in Table 3 for `Assembly`. `Assembly` is by far the largest system to solve, with more than 10^7 unknowns. For each problem, the number of iterations corresponding to the so-called *Classical* approach are reported in the column *Class.* while the one corresponding to the transform-then-solve approach is in the column *Trans.*.

The comparison between preconditioners were realised on a Dell computer with an Inter(R) Core(TM) i9-10900 CPU @ 2.80GHz. Both preconditioners were implemented in Python with the module SciPy [42]. As indicated above, the system was solved with a Krylov subspace method, here, GMRES [39] since the system (50) is not symmetric. The implementation of the Classical preconditioner (48) requires the inversion of both diagonal blocks. This was done with PyAMG [6] parametrised with the Ruge-Stuben algorithm [38] - see [41] for a more modern presentation. The transform-then-solve preconditioner (53) also uses PyAMG with the Ruge-Stuben algorithm for the inversion of the $\widehat{C}$ block, while the inversion of D_M does not require any special treatment.

The Table 4 gathers the 2D problems (except `Kershaw 2D` which is shown in Table 5). For these cases, both approaches work well and converge to the tolerance largely within the maximal number of iterations. Typically, the transform-then-solve approach converges as fast as the classical approach. However, the transform-then-solve approach is more costly because of the transformation and the classical approach would likely be the most efficient. Looking at Tables 5 and 6, we get a different picture. Some cases behave similarly as previously: `Hexahedras`, `Checkerboard`, `Voronoi` and `Prism`. However, the cases `Kershaw 2D`, `Kershaw 3D`, `Random` and `Tetrahedras` show a different behavior: at first, for smaller sizes of the systems, the classical method converges. Then there is a degradation of the convergence and for the larger sizes, the maximal number of iterations is reached. On the other hand, for those same cases, the transform-then-solve approach always converge within

	Cartesian		Triangles		Quadrangles		Polygons		LR 2D	
Size	Class.	Trans.	Class.	Trans.	Class.	Trans.	Class.	Trans.	Class.	Trans.
1	3	3	11	9	7	7	9	8	8	8
2	5	4	12	12	10	11	11	10	9	10
3	5	4	13	12	10	10	9	10	11	9
4	5	5	14	15	10	11	10	11	10	10
5	5	5	18	13	9	10	11	12		11
6	5	6			9	10		13		
7	10	6				12				

Table 4: Comparison of the number of iterations needed to decrease the relative residual by a factor of 10^{-6} by the so-called Classical preconditioner (Class.) and the transform-then-solve approach (Trans.).

	Kershaw 2D		Hexahedras		Kershaw 3D		Checkerboard		Voronoi	
Size	Class.	Trans.	Class.	Trans.	Class.	Trans.	Class.	Trans.	Class.	Trans.
1	41	56	2	2	145	59	8	8	21	11
2	57	54	4	4	150	60	11	25	10	14
3	62	58	5	4	> 199	109	15	18	13	13
4	70	63	14	5	> 199	112	17	18	14	16
5	> 199	63		4				26	17	12
6	> 199	64								

Table 5: Comparison of the number of iterations needed to decrease the relative residual by a factor of 10^{-6} by the so-called Classical preconditioner (Class.) and the transform-then-solve approach (Trans.).

the maximal number of iterations. This shows the robustness of the transform-then-solve approach. The most impressive result concerns the `Assembly` case: the classical approach diverges, while the transform-then-solve method converges for all sizes. The reduction of the number of iterations observed between size 1 and 3 is much likely due to the fact that the mesh is too coarse to accurately represent the continuous system and the resulting matrices may have a more complicated structure as a result.

To summarise those observations, the classical preconditioner and the transform-then-solve approach yield equivalent results in the simpler cases: for the 2D problems (except `Kershaw 2D`) and and the 3D problems that show some regularity (`Hexahedras`, `Checkerboard`). For more complex cases, the transform-then-solve approach is better suited and much more robust. Note that this is in agreement with the results described in [2] where the robustness of a similar approach was highlighted for Stokes problems. The most impressive results are obtained for the industrial problem `Assembly` where our approach allows us to get results, whereas the classical preconditioner fails to converge.

5 Conclusions

This article has two main objectives. On the one hand, we propose a compatible Finite Volume discretisation scheme for the incompressible Navier-Stokes on general meshes called PolyMAC. On the other hand, we study the specific challenges related to the numerical

	Random		Prism		Tetrahedras		Assembly	
Size	Class.	Trans.	Class.	Trans.	Class.	Trans.	Class.	Trans.
1	16	18	46	24	25	22	/	88
2	51	22	36	34	27	28	/	49
3	/	68	52	32	88	26	/	25
4	/	80	54	32	> 199	27	/	23
5					> 199	29	/	28
6							/	33
7							/	33

Table 6: Comparison of the number of iterations needed to decrease the relative residual by a factor of 10^{-6} by the so-called Classical preconditioner (Class.) and the transform-then-solve approach (Trans.).

resolution of the linear systems arising from the discretisation.

In the first part of the article, we detail how each continuous operator is discretised and we deduce the discretisation of the Navier-Stokes equations. In particular, we specify the structure of the resulting linear system. In the next section, we study the convergence of PolyMAC on a broad set of meshes taken from different FVCA conferences. We make sure that the method converges both in velocity and pressure and we compute the asymptotic convergence rate. As expected, the order of convergence is around 1.0 for both the pressure and velocity. Super-convergence is observed for some meshes, but most notably for the distorted cases `Kershaw 2D` and `Kershaw 3D`. We then apply PolyMAC to a problem of industrial complexity that was studied by the CEA: `Assembly` represents an assembly of nuclear rods that has been used in practice for safety studies and is representative of the difficulties met in practice in the industry. We start by checking that the convergence properties assessed for the benchmark are preserved for the `Assembly` case: a convergence order of about 1.0 is indeed observed. This highlights the robustness of the PolyMAC scheme.

In a second time, we focus on the resolution of the linear system in an industrial context. Since the systems may be large, the use of direct solvers is prohibited and we need robust and efficient iterative solvers. The system presents a double saddle-point that is well-known to be challenging for iterative solvers. To get around this difficulty, we implement a PISO-like method. We now have two smaller systems to solve and only the second one causes problems. We start by showing that classical preconditioners fail to converge when the underlying meshes used in the discretisation are distorted. We then develop an approach called the transform-then-solve approach inspired by the works [33, 2] to improve the robustness of the preconditioner. The results are impressive: all cases now converge in a reasonable number of iterations. The convergence of the `Assembly` case is especially fast on finer meshes.

To conclude, we present a robust approach to solve incompressible Navier-Stokes equations, from the discretisation step to the resolution of the linear system. The main asset of the approach is its robustness: PolyMAC converges on meshes of any shape. However, when considering distorted meshes, the resolution of the linear systems may be surprisingly difficult: the transform-then-solve approach allows to solve those difficulties.

References

[1] F. P. Ali Beik and M. Benzi, *Iterative methods for double saddle point systems*, SIAM Journal on Matrix Analysis and Applications, 39 (2018), pp. 902–921.

[2] P. Bacq, S. Gounand, A. Napov, and Y. Notay, *An All-at-once Algebraic Multigrid Method for Finite Element Discretizations of Stokes Problem*, Int. J. Numer. Meth. Fluids, 95 (2023), pp. 193–214.

[3] P.-L. Bacq, A. Gerschenfeld, and M. Ndjinga, *PolyMAC: Staggered Finite Volume Methods on General Meshes for Incompressible Navier–Stokes Problems*, in Finite Volumes for Complex Applications X, A. F. et al., ed., vol. Volume 1, Elliptic and Parabolic Problems, Switzerland, 2023, Springer Proceedings in Mathematics and Statistics, pp. 879–888.

[4] P.-L. Bacq and Y. Notay, *A new semi-algebraic two-grid method for Oseen problems*, SIAM J. Sci. Comput., n/a (2022). https://doi.org/10.1137/21M1429011.

[5] P. Baxandall and H. Liebeck, *Vector Calculus*, Clarendon Press, 1986.

[6] N. Bell, L. N. Olson, J. Schroder, and B. Southworth, *PyAMG: Algebraic Multigrid Solvers in Python*, Journal of Open Source Software, 8 (2023), p. 5495.

[7] R. Beltman, M. Anthonissen, and B. Koren, *Conservative Polytopal Mimetic Discretization of the Incompressible Navier–Stokes Equations*, Journal of Computational and Applied Mathematics, 340 (2018), pp. 443–473.

[8] M. Benzi, G. H. Golub, and J. Liesen, *Numerical Solution of Saddle Point Problems*, Acta Numerica, 14 (2005), p. 1–137.

[9] D. Bestion, *System Code Models and Capabilities*, in Thicket-2008, IAEA, 2008, p. 486.

[10] D. Bestion, *11 - the Structure of System Thermal-Hydraulic (SYS-TH) Code for Nuclear Energy Applications*, in Thermal-Hydraulics of Water Cooled Nuclear Reactors, F. D'Auria, ed., Woodhead Publishing, 2017, pp. 639–727.

[11] J. Bonelle, *Compatible Discrete Operator Schemes on Polyhedral Meshes for Elliptic and Stokes Equations*, PhD thesis, Université Paris-Est, 2014.

[12] J. Bonelle and A. Ern, *Analysis of compatible discrete operator schemes for the stokes equations on polyhedral meshes*, IMA Journal of Numerical Analysis, 35 (2014), pp. 1672–1697.

[13] F. Boyer and P. Omnes, *Benchmark Proposal for the FVCA8 Conference: Finite Volume Methods for the Stokes and Navier–Stokes Equations*, in Finite Volumes for Complex Applications VIII - Methods and Theoretical Aspects, C. Cancès and P. Omnes, eds., Cham, 2017, Springer International Publishing, pp. 59–71.

[14] CEA, *TRUST platform*. https://github.com/cea-trust-platform, 2022.

[15] A. J. Chorin, *A numerical method for solving incompressible viscous flow problems*, Journal of Computational Physics, 2 (1967), pp. 12–26.

[16] H. C. Elman, D. J. Silvester, and A. J. Wathen, *Finite Elements and Fast Iterative Solvers with Applications in Incompressible Fluid Dynamics*, Numerical Mathematics and Scientific Computation, Oxford University Press, 2014.

[17] R. Eymard, T. Gallouët, and R. Herbin, *Finite volume methods*, in Solution of Equation in $\mathbb{R}^n$ (Part 3), Techniques of Scientific Computing (Part 3), vol. 7 of Handbook of Numerical Analysis, Elsevier, 2000, pp. 713–1018.

[18] R. Eymard, G. Henry, R. Herbin, F. Hubert, R. Klöfkorn, and G. Manzini, *3D Benchmark on Discretization Schemes for Anisotropic Diffusion Problems on General Grids*, in Finite Volumes for Complex Applications VI Problems & Perspectives, J. Fořt, J. Fürst, J. Halama, R. Herbin, and F. Hubert, eds., Berlin, Heidelberg, 2011, Springer Berlin Heidelberg, pp. 895–930.

[19] R. Eymard and R. Herbin, *A staggered finite volume scheme on general meshes for the Navier-Stokes equations in two space dimensions*, International Journal on Finite Volumes, 2 (2005), pp. 1–18.

[20] P. Farrell, L. Mitchell, L. Scott, and F. Wechsung, *A reynolds-robust preconditioner for the scott-vogelius discretization of the stationary incompressible navier-stokes equations*, The SMAI journal of computational mathematics, 7 (2021), p. 75–96.

[21] J. H. Ferziger and M. Perić, *Computational Methods for Fluid Dynamics*, Springer, Berlin, 2nd ed., 2002.

[22] V. Girault and P.-A. Raviart, *Finite Element Methods for Navier-Stokes Equations: Theory and Algorithms. Berlin-Heidelberg-New York-Tokyo, Springer-Verlag 1986. X, 374 S., 21 Abb., DM198,–. ISBN 3-540-15796-4 (Springer Series in Computational Mathematics 5)*, Springer Berlin-Heidelberg, 1986.

[23] F. H. Harlow and J. E. Welch, *Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface*, The Physics of Fluids, 8 (1965), pp. 2182–2189.

[24] R. Herbin and F. Hubert, *Benchmark on discretization schemes for anisotropic diffusion problems on general grids*, in Finite volumes for complex applications V, ISTE, ed., France, June 2008, Wiley, pp. 659–692.

[25] R. Issa, *Solution of the implicitly discretised fluid flow equations by operator-splitting*, Journal of Computational Physics, 62 (1986), pp. 40–65.

[26] R. Issa, A. Gosman, and A. Watkins, *The computation of compressible and incompressible recirculating flows by a non-iterative implicit scheme*, Journal of Computational Physics, 62 (1986), pp. 66–82.

[27] Y. Li, L. Meng, Z. Huang, S. Li, D. Wang, B. Liu, Y. Zhang, L. Zhang, and W. Jiang, *Study on the effects from spacer wires on coolant flow within a ciads fuel assembly*, Annals of Nuclear Energy, 183 (2023), p. 109647.

[28] K. Lipnikov, G. Manzini, and M. Shashkov, *Mimetic finite difference method*, Journal of Computational Physics, 257 (2014), pp. 1163–1227. Physics-compatible numerical methods.

[29] J. Marshall, A. Adcroft, C. Hill, L. Perelman, and C. Heisey, *A finite-volume, incompressible navier stokes model for studies of the ocean on parallel computers*, Journal of Geophysical Research: Oceans, 102 (1997), pp. 5753–5766.

[30] R. Milani, *Schémas de discrétisation Compatible Discrete Operator pour les équations de Navier–Stokes d'un fluide incompressible en régime instationnaire*, theses, Université Paris-Est, Dec. 2020.

[31] B. Niceno, *A three dimensional , finite volume method for incompressible navier stokes equations on unstructured , staggered grids*, 2013.

[32] Y. Notay, *A new analysis of block preconditioners for saddle point problems*, SIAM J. Matrix Anal. Appl., 35 (2014), pp. 143–173.

[33] ——, *Algebraic Multigrid for Stokes Equations*, SIAM J. Sci. Comput., 39 (2017), pp. S88–S111.

[34] S. Patankar and D. Spalding, *A Calculation Procedure for Heat, Mass and Momentum Transfer in Three-dimensional Parabolic Fws*, International Journal of Heat and Mass Transfer, 15 (1972), pp. 1787–1806.

[35] B. Perot, *Conservation properties of unstructured staggered mesh schemes*, Journal of Computational Physics, 159 (2000), pp. 58–89.

[36] R. Rannacher, *Finite Element Methods for the Incompressible Navier-Stokes Equations*, Birkhäuser Basel, Basel, 2000, pp. 191–293.

[37] W. Raza and K.-Y. Kim, *Comparative analysis of flow and convective heat transfer between 7-pin and 19-pin wire-wrapped fuel assemblies*, Journal of Nuclear Science and Technology, 45 (2012), pp. 653–661.

[38] J. W. Ruge and K. Stüben, *4. Algebraic Multigrid*, 1987, pp. 73–130.

[39] Y. Saad and M. H. Schultz, *GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems*, SIAM J. Sci. Stat. Comput., 7 (1986), p. 856–869.

[40] J. C. Strikwerda, *Finite difference methods for the stokes and navier–stokes equations*, SIAM Journal on Scientific and Statistical Computing, 5 (1984), pp. 56–68.

[41] K. Stüben, *A review of algebraic multigrid*, Journal of Computational and Applied Mathematics, 128 (2001), pp. 281–309. Numerical Analysis 2000. Vol. VII: Partial Differential Equations.

[42] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, *SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python*, Nature Methods, 17 (2020), pp. 261–272.