A LARGE SCALE BENCHMARK FOR THE INCOMPRESSIBLE NAVIER-STOKES EQUATIONS∗

ZIZHOU HUANG†, TESO SCHNEIDER‡, MINCHEN LI§, CHENFANFU JIANG§, DENIS ZORIN†, AND DANIELE PANOZZO†

Abstract. Numerical methods for solving incompressible Navier-Stokes equations are widely applied and are among the most extensively studied. A plethora of approaches have been introduced targeting different applications and regimes; these methods vary in space discretization, choice of time integration scheme, or reduction of the non-linear equations to a sequence of simpler problems. While the theoretical properties of most of these variants are known, it is still difficult to pick the best option for a given problem, as practical performance of these methods has never been systematically compared over a varied set of geometries and boundary conditions.

We introduce a collection of benchmark problems in 2D and 3D (geometry description and boundary conditions), including simple cases with known analytic solution, classical experimental setups, and complex geometries with fabricated solutions for evaluation of numerical schemes for incompressible Navier-Stokes equations in laminar flow regime. We compare the performance of a representative selection of most broadly used algorithms for Navier-Stokes equations on this set of problems. Where applicable, we compare the most common spatial discretization choices (unstructured triangle/tetrahedral meshes and structured or semi-structured quadrilateral/hexahedral meshes).

The study shows that while the type of spatial discretization used has a minor impact on the accuracy of the solutions, the choice of time integration method, spatial discretization order, and the choice of solving the coupled equations or reducing them to simpler subproblems have very different properties. Methods that are directly solving the original equations tend to be more accurate than splitting approaches for the same number of degrees of freedom, but numerical or computational difficulty arise when they are scaled to larger problem sizes. Low-order splitting methods are less accurate, but scale more easily to large problems, while higher-order splitting methods are accurate but require dense time discretizations to be stable.

We release the description of the experiments and an implementation of our benchmark, which we believe will enable statistically significant comparisons with the state of the art as new approaches for solving the incompressible Navier-Stokes equations are introduced.

Key words. Large-Scale Benchmark, Incompressible Navier-Stokes Equations

AMS subject classifications. 76D05 Navier-Stokes equations for incompressible viscous fluids

1. Introduction. The computation of the motion of incompressible viscous fluids is ubiquitous in scientific computing and engineering applications. The corresponding incompressible Navier-Stokes equations are challenging to solve numerically, and a large number of approaches have been proposed to compute their solution on discrete domains.

While a large literature considered the theoretical properties of these methods, such as convergence, and applied these to a broad range of fluid problems, a systematic comparison of these techniques on a representative common set of benchmark problems is not available. Our work strives to fill this gap by introducing a large collection of benchmark problems, varying from simple cases with known analytical solution, to manufactured solutions on thousands of complex geometries, a collection

∗Submitted to the editors December 13, 2021.

Funding: This work was partially supported by the NSF CAREER award with number 1652515, the NSF grant IIS-1320635, the NSF grant DMS-1436591, the NSF grant 1835712, the NSF grant IIS-1943199, CCF-1813624, ECCS-2023780, the NSERC grant DGECS-2021-00461 and RGPIN-2021-03707, a gift from Adobe Research, and a gift from nTopology.

†NYU Courant, New York.

‡University of Victoria, British Columbia.

§UCLA Math Dept., California.
of reference implementations of representative solution methods, and an automated way of comparing results. The comparison is performed by matching a primary quantity of interest (e.g., running time) to a secondary quantity (e.g., accuracy): to achieve this, we automate the generation of discretizations for all our problems, thus enabling us to experimentally match the primary quantity and measure the secondary one.

We consider two classes of problems: (1) problems with known closed-form solutions, using fabricated solutions both on simple domains and on a large collection of hundreds of complex 2D and 3D geometries, and (2) a set of standard test problems for which a closed-form solution is unknown, and the error has to be studied with respect to fine-resolution numerical solutions.

1.1. Mesh Types. We consider three types of meshes:
- T unstructured triangular mesh (2d) or tetrahedral mesh in (3d);
- Q unstructured quadrilateral meshes (2d) and hexahedral meshes (3d);
- R regular 2d or 3d grid.

1.2. Spatial Discretization. We compare four approaches to spatial discretization (SD):
- FE A finite-element discretization using elements of possibly different order for velocity and pressure [50].
- FV A finite-volume formulation, (we use the Ansys FLUENT [5] implementation)
- FD A finite difference scheme using the Marker-and-Cell grid [29].

1.3. Time Integration. Spatial discretization methods are coupled with five time-integration methods (TD):
- BDF3 The Backward Differentiation Formula of order 3 [55].
- C The coupled time integration scheme in Ansys FLUENT [5].
- SL A splitting formulation [15] using FE/FD for diffusion and pressure projection, and a semi-Lagrangian scheme [49] for advection.
- FLIP A variant of the FLIP scheme [10] using FE/FD for diffusion and pressure projection, and particles for advection.
- AB2AM2 An explicit predictor-corrector method [38] that consists of a second order Adam-Bashforth (AB2) predictor and a modified second order Adam-Moulton (AM2) corrector.

1.4. Methods. Not all combinations of spatial and time discretizations are realizable. We selected six representative valid combinations commonly used research papers, and added the most popular solution used in industry. For each method, we test it on all mesh types, with the exception of finite differences, which require a regular grid:

| Acronym       | Spatial D. | Time D. | Mesh  |
|---------------|------------|---------|-------|
| FE21-BDF3     | FE21       | BDF3    | T, Q, R |
| FV-C          | FV         | C       | T, Q, R |
| FE11-SL       | FE11       | SL      | T, Q, R |
| FE11-FLIP     | FE11       | FLIP    | T, Q, R |
| FD-FLIP       | FD         | FLIP    | R     |
| FE11-AB2AM2   | FE11       | AM2AB2  | T, Q, R |
| FE21-AB2AM2   | FE21       | AM2AB2  | T, Q, R |

1.5. Summary of Results. Our study allows evaluating the combinations of time and spatial discretizations over multiple performance measures. (For a description of our experimental setup see Appendix B.)
Mesh Type. Different mesh types do not lead to noticeably different results on any of the settings that we tested. In particular, we did not observe measurable benefits for structured meshes (which are difficult to generate automatically) over unstructured meshes (which can be reliably generated automatically from a boundary description) for the type of solvers we have used.

2D vs 3D. The relative pros and cons of the methods we tested are similar in our 2D and 3D test cases, thus the conclusions of our study hold for both two- and three-dimensional problems. Using a hexahedral mesh in 3D for FE11-FLIP, FE11-SL has a minor downside, since they require inverting the trilinear geometric map using a numerical method, which has a small negative effect on runtime (the issue is not present in 2D, as the bilinear geometric map can be inverted in closed form).

Runtime Limitations. In our experiments, we were able to get a solution from methods that linearly discretize space and time within our time budget for all mesh resolutions we considered, but the accuracy of these methods is relatively low even on the finest mesh. Higher-order methods have issues with higher resolutions. FE21-BDF3 leads to linear systems for which iterative solvers may not converge to target accuracy, thus precluding the solution of large systems. FE21-AB2AM2 allows to solve for large number of DOFs but, due to the explicit time integration, the time step has to be decreased quadratically with respect to the edge length, thus making it impractical. FV-C is a good compromise, its convergence rate is lower than FE21-BDF3, but higher than all the linear methods, and it still scales well to larger problem sizes.

Accuracy. Methods using third-order time-integration schemes and second-order bases for the velocity are the only ones that can obtain high-accuracy solutions in a reasonable time due to their cubic convergence. However, they suffer from scalability problems due to either numerical difficulties or restrictions in the maximal time step.

Pressure Boundary Accuracy. Only FE21-BDF3 produces reliably accurate results: standard linear splitting schemes do not converge to a solution, high-order splitting schemes converge, but due to their limitations on the time step are impractical. This observation may be of importance for the choice of method for coupled fluid-structure problems.

Time Step Size. All methods have restrictions, due to the CFL condition, on their time step size. The methods that are less affected by this limitation are linear splitting schemes, which converge for large steps, but their accuracy is low. On the other end of the spectrum, higher-order explicit time splitting methods have better accuracy for a given grid resolution, but are only stable for time steps decreasing quadratically with edge length.

Viscosity. All methods perform similarly for fluids with high viscosity. For low viscosity, their solutions are widely different, making it challenging to evaluate their accuracy on test problems without analytical solutions.

We present our results and benchmarks first, and provide a more detailed analysis of suggested directions for future work in the conclusion.

2. Review of Spatial and Time Discretization Types. We briefly review some common existing algorithms for numerical solution of Navier-Stokes equations. We cover the discretization of space and time separately: while these need to be chosen in a compatible way, there is often some flexibility in how space and time discretization types are paired.

2.1. Incompressible Navier-Stokes Equations. Let \( \Omega \subset \mathbb{R}^d \), \( d \in \{2, 3\} \) be a domain with boundary \( \partial \Omega \), divided into parts \( \partial \Omega_D \subset \partial \Omega \) and \( \partial \Omega_N \subset \partial \Omega \) on which
Dirichlet and Neumann boundary conditions are applied (we only consider these two types of boundary conditions). The Navier-Stokes (NS) equations are formulated in terms of velocity

\[ u : \Omega \times (0, T) \rightarrow \mathbb{R}^d \]

of a fluid with kinematic viscosity \( \nu \) and density \( \rho \), with time \( t \in (0, T) \) and pressure

\[ p : \Omega \times (0, T) \rightarrow \mathbb{R} \].

The equations for \( u \) and \( p \) are:

\[
\begin{align*}
\frac{\partial u}{\partial t} + \rho(u \cdot \nabla)u - \nu \Delta u + \nabla p &= b & \text{on } \Omega \times (0, T) \\
\nabla \cdot u &= 0 & \text{on } \Omega \times (0, T) \\
\nabla \cdot u &= 0 & \text{on } \partial \Omega_D \times (0, T) \\
\nu \frac{\partial u}{\partial n} + pn &= g & \text{on } \partial \Omega_N \times (0, T) \\
u(0) = u_0 & \text{on } \Omega \times \{0\},
\end{align*}
\]

where \( b \) is an external volume force, \( n \) is the surface normal, \( u_0 \) is the initial condition, and \( d \) and \( g \) are the Dirichlet and Neumann conditions.

We consider discretization based on meshes: triangular and quadrilateral in 2D and tetrahedral and hexahedral in 3D. Thus for a given mesh size \( h \) (i.e., the maximum edge length on the mesh) the different methods compute approximations \( u_h(x,t) \) and \( p_h(x,t) \) of the solutions \( u \) and \( p \) of the NS equations.

### 2.2. Spatial Discretization.

#### Finite Elements.

The finite element method (FEM) \([57, 16]\) is a classical method used to solve PDEs, and large number of libraries and software exist (e.g., DOLFIN (FEniCS) \([2]\) Firedrake \([40]\), libMESH \([36]\), mFEM \([3, 41]\), getFEM \([46]\), Deal II \([6]\)).

The elements used for incompressible Navier-Stokes equations are typically the same as for the linear incompressible Stokes equation, e.g., \([26, 27]\). To ensure convergence, the finite element space should satisfy the inf-sup condition \([56, 11]\). Conforming elements, such as the Taylor-Hood elements \([56, 11, 27]\), and non-conforming elements such as Raviart-Thomas elements \([12]\) are commonly used. To solve the nonlinear problem, iterative penalty methods \([17, 56]\) and augmented Lagrangian methods \([42]\) were used.

In our comparison, we use the standard linear and quadratic Lagrange bases for tetrahedra \((P_1 \text{ and } P_2)\) and hexahedra \((Q_1 \text{ and } Q_2)\) \([57, 16]\), which is a special case of the Taylor-Hood element \([56]\). We use the standard mixed Galerkin formulation \([57, 16]\) with the full-order Gaussian quadrature for all our experiments.

#### Finite Volume.

In the finite volume (FV) method for fluid simulation (e.g., \([37]\)), the solution domain is subdivided into a finite number of small control volumes (or cells) by a mesh, which does not change in time for transient calculations. Velocity and pressure are stored in the same cell location (cell center in Ansys Fluent), which is usually called the collocated or non-staggered variable arrangement \([43]\).

The volume integrals in a partial differential equation that contain a divergence term are converted to surface integrals, using the divergence theorem. These terms are then evaluated as fluxes at the surfaces of each finite volume. Because the flux entering a given volume is identical to that leaving the adjacent volume, these methods are conservative.

We use the Ansys pressure-based FV solver and the Ansys laminar model \([5]\) in Ansys Fluent\([5]\). We use the pressure-velocity coupled scheme. For spatial dis-
cretization, we select the least-square cell-based scheme for gradients, the second-order scheme for pressure, and the second upwind implicit scheme for momentum. We select the second-order implicit scheme provided in Fluent as the time integrator [5].

Finite Differences. In certain applications, high solution errors are more acceptable; in these cases, a structured and uniform spatial discretization with the lowest possible order of interpolation kernels for both velocity and pressure [13] is used. Many methods in this category adopt a linear velocity and constant pressure scheme. Since this discretization combination on collocated grids has spurious checkerboard or hour-glass modes and is not stable, it is common to use the staggered Marker-And-Cell (MAC) grid [29] with velocities discretized on the cell surfaces and pressure on the cell centers, that stabilizes the scheme. We use the MAC grid in our experiments with FD approaches. In Appendix B we describe how we use the MAC grid for complex geometries.

2.3. Time Discretization.

Implicit Time Integration. A common choice for high-order time integration is the backward differentiation formula (BDF) of order 3, applied to the full system of equations, in contrast to splitting methods explained below.

Ansys Coupled. The widely used software Ansys Fluent [5] uses a coupled method to solve the pressure-velocity coupling. The fully implicit coupling is achieved through an implicit discretization of pressure gradient terms in the momentum equations and an implicit discretization of the face mass flux, including the Rhie-Chow pressure dissipation terms. Unlike the SIMPLE [44, 33] and SIMPLEC [19] algorithms, it solves the momentum and pressure-based continuity equations together.

Linear Operator Splitting. Since directly solving the nonlinear time integration problem with incompressibility constraints is expensive, it is possible to use the Chorin’s projection [15, 53] to split the solve at each time step into predictor-corrector sub-steps where pressure terms only participate in the corrector step that handles incompressibility.

The predictor step is often further split into advection and diffusion for viscous fluids [13]. Diffusion step typically involves a global linear solve [13] or sometimes is computed using particles [47, 63]. The advection step suffers from significant numerical dissipation in the classic semi-Lagrangian method [49, 53]; ways to reduce this dissipation were extensively studied. For purely grid based methods, while MacCormack [39, 52] and BFECC [20] advect the solution forward and then backward in time to estimate and compensate error, Bimocq [45] resolve the flow map characteristics for periods longer than a single time step (as opposed to one step with semi-Lagrangian) to reduce dissipation. Hybrid Lagrange/Eulerian techniques like Particle-In-Cell (PIC) [28], Fluid-Implicit-Particle (FLIP) [10], AffinePIC [35, 18], and PolyPIC [25] explicitly track motion of particles in the fluid and transfer information between grid and particles, which is nearly dissipation-free, but can suffer from nonuniform particle distribution and also higher computational costs. Ding et al. [18] presents a quantitative study of accuracy of several different particle-based advection methods. Chorin projection itself causes dissipation: Zhang et al. [62] reduced numerical dissipation caused by the corrector step by estimating lost vorticity and adding it back into the fluid. Zehnder et al. [61] propose a simple but effective modification to the splitting scheme that is similar to midpoint rule integration to reduce the corrector error.

In this study, we evaluate the two advection methods, semi-Lagrangian (SL) [49]
and the Fluid-Implicit-Particle (FLIP) [10], which are typical for the Eulerian and Lagrangian views respectively. The original SL method is unconditionally stable, and the method is widely used in numerical weather prediction [7]. For a more thorough comparison among different advection methods, we refer to [45] and [18].

**High-Order Operator Splitting.** The computational advantages of commonly used linear operator splitting have the downside of limiting their rate of convergence to linear. To overcome this limitation, while retaining some of their benefits, high-order splitting schemes can be used instead [58, 30, 38, 21, 14]. In this study, we use the method proposed in [38] as a recent representative of this class. We briefly review this method for completeness.

[38] solves the incompressible NS equations using a split-step approach that decouples the solution of the velocity from the solution of the pressure. The split-step method [30, 38] is an explicit predictor-corrector method that consists of a second-order Adam-Bashforth (AB2) predictor and a modified second-order Adam-Moulton (AM2) predictor. We implemented this method with the TN boundary condition and divergence damping in [38].

**2.4. Convergence Rates.** We implemented the mesh-based methods using the PolyFEM library [50]. For FV-C, we rely on the implementation in Ansys [5]. For FD-FLIP, we use a separate framework specialized for MAC-grids. We report the convergence of the five different methods on a simple square domain (Figure 1). As expected, FE21-BDF3 and FE21-AB2AM2 converges cubically, FV-C and FE11-AB2AM2 quadratically, and the other three methods linearly. Since the different methods lead to different system sizes and non-linear optimization, we also compare the time required to solve to a given error (Figure 2).

**3. Commonly Used Test Problems.** We study five two-dimensional standard test cases to cover most common variations of fluid flow problems and various boundary conditions: driven-cavity (classical problem with only Dirichlet boundary conditions, Section 3.1), airfoil (complex internal boundary, Section 3.2), open cavity (different viscosities, Section 3.3), vortex street (complex turbulent behavior, Section 3.4), and a drag force computation (Section 3.5). For a complete description of our experimental setup we refer to Appendix B.

**Evaluation.** We compute a numerical reference solution using a denser mesh using the same CFL number (i.e., we use a smaller time step) and a more accurate method:
that is, we define the error as
\[ e^2 = \int_{\Omega} \| u_h(x,T) - u^*(x,T) \|^2 \, dx, \]
where \( u_h(x,T) \) is the solution of a given method and \( u^*(x,T) \) is the reference solution computed using FE32-BDF4 on a denser mesh. This scheme uses \( P_3/Q_3 \) basis for velocity and \( P_2/Q_2 \) basis for pressure, and BDF order 4 for time integration. Note that this scheme uses both a time and space discretization of higher order than all the methods we include in the study, in addition to being applied to a denser mesh. We note that for many applications other error measures are relevant (e.g., error in velocity gradients, determining the fluid forces). However, for many commonly used low-order methods, velocity is discontinuous, and convergence of gradients can be shown only in specific discrete norms, i.e., effectively, in addition to numerical method itself, one needs to choose a discretization of the gradient to define the norm [24]. This makes fair comparisons between discretizations and methods more difficult as the optimal choice of velocity gradient discretization may be method-dependent. For this reason, we focus on the solution error. We also note that the equation residual provides a stringent test for how well the equation is satisfied, but can be easily applied only to higher-order methods for which both gradients and second derivatives can be computed pointwise; we present some experiments in Appendix A, for a particular choice of approximation of the high order derivatives.

We acknowledge that using a dense mesh with FE32-BDF4 as “ground-truth” potentially might give an unfair advantage to FE21-BDF3. We selected FE32-BDF4 as it gives the lower errors in the experiments having a ground truth solution, and it is reasonable to expect that in other cases its accuracy remains high on fine meshes.

**Boundary Geometry.** To facilitate the comparison for FD-FLIP (where complex boundaries need to be captured with a level-set), and to avoid favoring unstructured meshes which are easier to adapt to complex boundaries, we choose most of our domains to have straight boundaries. For all methods except FD-FLIP we run experiments on both a regular quad mesh and an unstructured triangular mesh. We adapt the mesh size for every experiment to ensure that all the methods have similar running times and report the resulting errors. Neumann boundary conditions are imposed differently for different methods as discussed in Appendix C.

**Time step.** We use \( dt = C_2 h^2 \) for FE11-AB2AM2, \( dt = C_3 h^2 \) for FE21-AB2AM2, and \( dt = C_1 h \) for all other methods, where \( h \) is the mesh size and the choices of constants \( C_i \) are in Table 1.
Table 1

| Problem       | Driven Cavity | Airfoil | Open Cavity | Vortex Street | Drag Force |
|---------------|---------------|---------|-------------|---------------|------------|
| $C_1$         | 0.1           | 0.036   | 0.2         | 0.29          | 0.12       |
| $C_2$         | 0.125         | 0.0013  | 0.5         | 0.082         | 0.014      |
| $C_3$         | 0.0625        | 0.0013  | 0.5         | 0.082         | 0.014      |

3.1. Driven Cavity. For this test, the domain is a unit square, the initial velocity is zero, zero Dirichlet boundary condition is imposed on the left/bottom/right side, and $(\sin(\frac{\pi}{2}t), 0)$ on the top side. We run the experiments with $\nu = 0.1$, $T = 2$. For this simple setup, all methods produce similar results (Figure 3). In this experiment, FV-C has a slightly smaller error compared with the other methods. Our experiments show that FV-C is the best method (measured by time versus accuracy) at the coarser resolutions (Figure 4).

3.1.1. Boundary Condition Smoothness. The boundary condition for this test in a standard form is not continuous; consequently, high-order methods like FV-C and FE21-BDF3 do not reach the theoretical convergence rate. To study how the smoothness of the boundary condition affects the convergence, we solve the driven-cavity problem with three different boundary conditions for the top boundary:

\[ v = (1, 0), \quad v = (4(1 - x)x, 0), \quad \text{and} \quad v = \left(50\exp \left(\frac{1}{x(1 - x)}\right), 0\right). \]

Figure 5 shows that for discontinuous Dirichlet boundary condition, no method can achieve convergence rate higher than one. As we increase the continuity of the boundary conditions to $C^0$ FE21-BDF3, FV-C, FE11-AB2AM2, and FE21-AB2AM2 show quadratic convergence, and finally, when we use a smooth function, FE21-BDF3 and FE21-AB2AM2 have cubic convergence. Since the solve time of the different methods is mostly independent from the boundary conditions, the reduction in the convergence rate affects the error corresponding to a given time budget (Figure 6): for $v = 1$ all methods are similar, and as smoothness of the boundary condition increases FE21-BDF3 and FE21-AB2AM2 gets more efficient.

3.2. Airfoil. In this experiment (Figure 7), we embed the NACA 5012 airfoil in a background mesh and specify left-to-right inflow velocity of $v = 1 - e^{-5t}$ and zero initial velocity. We apply zero Dirichlet boundary condition on the airfoil surface. We run our simulations with $\nu = 0.0002$ and $T = 4$. For this more complex geometry, the methods produce different results (Figure 7, closeups). FE11-FLIP and FE11-SL exhibit additional numerical viscosity, compared to FD-FLIP. We note that our error measure in this case is not able to capture some of the major artefacts in the solutions computed with higher-order splitting methods. When using AB2AM2, the necessary number of time steps is very high, thus to match the wall-clock time for different methods, we have to use coarse meshes leading to large errors and artefacts.

3.3. Open Cavity. For this test, we compare different methods for several viscosities (Figure 8). The fluid flows from left to right; the velocity is $v = (1 - e^{-5t}, 0)$.
on the left-most vertical boundary, the right side has zero Neumann boundary condition, and the remaining boundaries have zero velocity. For low Reynolds number, all methods exhibit similar behavior. As the Reynolds number increases, FE11-SL and FE11-FLIP have higher numerical viscosity, and the qualitative stream behavior is different, with FE11-FLIP showing non-physical turbulence-like behavior in the middle. These differences have a small impact on the error, which stays within one order of magnitude independently from $\nu$. 

Fig. 3. Result of the driven cavity at time $t = T = 2$. 

A LARGE SCALE BENCHMARK FOR INCOMPRESSIBLE NAVIER-STOKES
3.4. Vortex Street. We run five methods on a pipe with an obstacle located off-axis, to generate a vortex. The fluid has viscosity 0.001, the top, bottom, and circle have zero Dirichlet boundary conditions, velocity on the left side is \( u(0,y) = (6(1 - e^{-5t})(0.41 - y)y/0.1681,0) \), and zero Neumann on the right side \(^1\). Figure 9 shows the results at the end of the simulation, when the vortex is fully formed. For a qualitative comparison, we plot the \( y \)-component of the velocity along a horizontal line passing through the middle of the domain (Figure 10); all methods manifest similar behavior, and the dissipation is similar. It is interesting to note that some methods have a significant “phase shift” with respect to others.

The behavior of the different methods varies depending on the mesh resolution (Figure 11): only FE21-BDF3, FV-C, and FE21-AB2AM2 exhibit similar dissipation on coarse meshes. In contrast, the other methods have high numerical dissipation at coarse resolution.

For this specific example, we also consider a quad boundary layer mesh: the resolution of the quad mesh is higher around the obstacle and gets coarser on the left-hand side of the domain. As the boundary-adapted structure of the quad mesh was hand-crafted, we can only increase its resolution by uniform refinement, and thus it is impossible to tune the resolution to match running time across different methods. Additionally, FD-FLIP cannot take advantage of the adaptivity as it is designed to work on regular grids. Figure 12 shows that different methods produce similar results (i.e., the errors are within an order of magnitude) on the adaptive quad mesh; however,

\(^1\)We used the setup from http://www.featflow.de/en/benchmarks/cfdbenchmarking/flow/dfg_benchmark2_re100.html
A LARGE SCALE BENCHMARK FOR INCOMPRESSIBLE NAVIER-STOKES

\[ v = 1 \]

\[ v = \frac{4(1 - x)x}{x(1 - x)} \]

\[ v = 50 \exp\left(\frac{1}{x(1-x)}\right) \]

**Fig. 6.** Error vs. running time of all methods in the driven cavity problem with different boundary conditions.

**Fig. 7.** Simulation results for \( T = 4 \) for the airfoil mesh. Bottom shows a closeup of the back of the wing. The error is computed only in a subset of the domain containing the airfoil \((-0.2, 1.2] \times [-0.2, 0.25])\).

FE11-FLIP, FV-C, and FD-FLIP are significantly faster.

### 3.5. Drag Force

For quantitative comparisons, we embed a circle of radius \( r = 0.05 \) into a large square background mesh (the size of the square is 1) to avoid boundary effects near the obstacle. The flow moves with velocity \( v = 1 \) from left to right (imposed as a Dirichlet condition on the boundary of the square) and measure drag force on the circle (zero Dirichlet boundary condition on the circle boundary).

To compute the drag force more accurately, we generate an adaptive mesh with high resolution close to the circle. Figure 13 shows a comparison of the drag force for three viscosity values. We note that FD-FLIP is designed to work with regular non-adaptive grids, which explains the high error at coarse resolutions.

For high viscosity, only FE21-BDF3 has low (below 1%) error, while for \( \nu = 0.001 \), both FE21-BDF3 and FV-C have similar errors. For the same running time, FE21-BDF3 has lower error than other methods. We note that we could not compute certain solutions for FE21-AB2AM2 as it did not terminate after our cutoff wall-clock time of 48 hours.

### 4. Large-Scale Benchmark

To quantify method errors on a large set of complex geometries, we use setups with known closed-form solutions: Taylor-Green vortex in 2D and the method of manufactured solutions in 3D [48]. In the latter case, we start from a target closed-form solution \( u \), compute the corresponding right-hand side
\[ \nu = 0.0001, \nu = 0.0005, \nu = 0.002, \nu = 0.005, \nu = 0.01 \]

**Fig. 8.** Result of the open driven cavity at time \( t = T = 6 \) for different viscosities.

**Fig. 9.** Result of the vortex street on a triangular mesh at time \( t = T = 6 \).
equations can be obtained without introducing artificial forcing terms. We use the following manufactured solution in 3D, following [23]. It is one of the few cases for which exact fully three-dimensional solution of the incompressible NS equations can be obtained without introducing artificial forcing terms.

\[ u_{2D}(x_1, x_2, t) = (\cos(2\pi x_1) \sin(2\pi x_2), -\sin(2\pi x_1) \cos(2\pi x_2))e^{-8\nu \pi^2 t} \]

\[ p_{2D}(x_1, x_2, t) = -\frac{1}{4} (\cos(4\pi x_1) + \cos(4\pi x_2))e^{-16\nu \pi^2 t} \]

We use the following manufactured solution in 3D, following [23]. It is one of the few cases for which exact fully three-dimensional solution of the incompressible NS equations can be obtained without introducing artificial forcing terms.

\[ u_{3D}(x_1, x_2, x_3, t) = -e^{-\nu t}(e_1^x \sin(x_2 + x_3) + e_3^x \cos(x_1 + x_2), e_2^x \sin(x_3 + x_1) + e_1^x \cos(x_2 + x_3), e_3^x \sin(x_1 + x_2) + e_2^x \cos(x_3 + x_1)) \]

\[ p_{3D}(x_1, x_2, x_3, t) = -\frac{1}{2} e^{-2\nu t}(e^{2x_1} + e^{2x_2} + e^{2x_3} + 2 \sin(x_1 + x_2) \cos(x_3 + x_1)e^{x_2 + x_3} + 2 \sin(x_2 + x_3) \cos(x_1 + x_2)e^{x_1 + x_3} + 2 \sin(x_3 + x_1) \cos(x_2 + x_3)e^{x_1 + x_2}) \]
with viscosity $\nu = 0.01$, $T = 0.1$.

We evaluate the performance of all methods (except FD-FLIP) on a large, diverse set of realistic domains, for two- and three-dimensional problems. We excluded FD-FLIP as it is designed to work on regular grids, and the error introduced by the implicit approximation of the boundary would lead to an unfair comparison.

For every domain, we generate the quad-/hex-meshes first, as they are more challenging to generate for automatic tools, and then create corresponding tri-/tet-meshes with a matching number of vertices. For conciseness, we report only time versus $L^2$ error plots; the other metrics can be explored in our interactive plot at https://cims.nyu.edu/gcl/fluids/.

4.1. 2D Dataset. We create our 2D dataset using the 20k OpenClip dataset introduced in [31]. For each model, we create a quadrilateral mesh using the instant meshing algorithm [32], using the option to preserve and align to the mesh boundary. In Figure 14, left, we report the average element scaled Jacobian $q$ [54, Section 4.9, Section 5.15] for quadrilaterals. To assess the quality of the meshes we also compute the percentage of low quality element (Figure 14, right): that is ratio of elements without $0.5 < q < 2/\sqrt{3}$ for triangles and $0.3 < q < 1$ quadrilaterals over the total number of elements.

We then drop quadrilateral meshes with flipped elements, meshes with disconnected components, and non-manifold meshes. To keep the total running time acceptable, we drop the meshes that are too dense, which results in 522 quadrilateral
A large scale benchmark for incompressible Navier-Stokes.

Fig. 14. Quality (left) and percentage of low quality elements (right) in our 2D dataset.

Triangular meshes.

Quadrilateral meshes.

Fig. 15. Comparisons among four methods on the 2D dataset.

meshes. For each quadrilateral mesh, we obtain a corresponding triangular mesh with the same number of vertices by splitting every quadrilateral into two triangles (we keep the shorter among the two diagonals). The meshes are normalized, so that the Reynolds number is the same for different meshes. For FE11-AB2AM2 and FE21-AB2AM2 we use $dt = 0.1h^2$, while for other methods $dt = 0.1h$.

Figure 15, left, shows the results of the four different methods on the whole 2D dataset (triangle and quad meshes combined). The plot shows that FE21-BDF3 and FE21-AB2AM2 generally yield the lowest error for a given wall-clock time, closely followed by FV-C for higher resolutions. For every method, we compare the performance of triangle and quadrilateral meshes. Figure 16 shows that all methods have
similar performance for both mesh types.
For completeness, we also measure the performance using DOFs instead of solver time, as this is an indirect measure of problem complexity not affected by the choice of the specific numerical solver. The result does not show any major difference with respect to wall-clock time (Figure 15, right).

4.2. 3D Dataset. We use hexahedral meshes from the dataset described in [51], excluding those requiring more than 150GB of memory for the linear system solve: for every mesh, we generate a tetrahedral mesh with similar mesh size. We opt for matching mesh size \( h \) (differently from vertex count as in [51]) as for transient problems \( dt \) is related to \( h \), and the running time depends on \( dt \). This leads to a dataset of 730 meshes. The meshes are normalized to fit the unit cube, so that the Reynolds number is consistent for different meshes. For FE11-AB2AM2 and FE21-AB2AM2 we use \( dt = 0.01h^2 \), while for other methods \( dt = 0.1h \). As in 2D, in Figure 17 left, we measure the scaled Jacobian \( q \) [54, Section 6.13, Section 7.11] of the meshes in our dataset. In Figure 17 right, we report the percentage of low quality elements; an high quality element has \( 0.5 < q < 1 \).

Figure 18 shows that moving from 2D to 3D affects all methods in a similar way: FE21-BDF3 is still the most accurate method (both in terms of time and DOFs versus error), closely followed by FV-C. FE21-AB2AM2 becomes slower as it requires more time steps.

In three dimensions, the difference between tetrahedral and hexahedral meshes is minor with the peculiar case of FE11-FLIP for which tetrahedral elements outperform hexahedral ones (Figure 19). This is likely caused by the requirement of FE11-FLIP to invert the trilinear geometric map several times for each timestep, which we compute with Newton’s method.

5. Concluding Remarks. We introduced a benchmark to quantitatively evaluate approaches for solving the incompressible Navier-Stokes equations for moderate Reynolds numbers. We included both test cases with closed-form or fabricated solutions, and more realistic scenarios for which an analytic solution is unknown. Overall, our dataset contains 1044 2D test cases and 730 3D test cases.
We believe that our work will enable future research on improving numerical
methods for incompressible Navier-Stokes equations. In particular:

1. New approaches can be easily tested against representative methods for the most popular spatial and time discretizations.

2. Scalability of new methods can be evaluated in a statistically meaningful way, using a common quantity of interest, such as running time, as reference. This applies to both problems with closed-form solutions on simple domains, and more realistic setups in complex geometries.

3. Our preliminary study on using point-wise residuals of the strong form to evaluate the quality of methods in a discretization-agnostic way shows that discretization artefacts are an important factor for pointwise quantities, es-

![Tetrahedral meshes.](image1)

![Hexahedral meshes.](image2)

**Fig. 17.** Quality (left) and percentage of low quality elements (right) in our 3D dataset.

**Fig. 18.** Comparisons among 4 methods on the 3D dataset.
especially for low-order methods (Appendix A), confirming that extracting accurate pointwise velocities and pressures from these solutions is an important problem on its own.

5.1. Venues for Future Work. Overall, we could not identify an existing solver allowing to solve larger-scale problems with high-order convergence rates (both on the interior and on the boundary of the domain), which has potential for solving Navier-Stokes equations more efficiently, for a given target accuracy, especially for higher viscosities and in smoother domains. This could be achieved by extending existing methods in different ways:

Linear System Solver for FE21-BDF3. The FE21-BDF3 algorithm performs consistently well in all our tests, with the exception of struggling to scale to large systems. Developing a preconditioner for this system to address the scaling issues is a promising direction for future work.

Implicit High-Order Splitting. The approaches based on high-order splitting perform very similarly to FE21-BDF3, while allowing to timestep problems with a large number of DOFs. However, since they are based on explicit time integration, they require small time steps, also making them impractical for large problems. We are not aware of high-order implicit-splitting time integration schemes for this type of discretizations, making this an interesting venue for future work.

Boundary Conditions and Accuracy. Different method impose boundary conditions in different ways, with a significant impact on the accuracy close to the domain boundary. Improving boundary condition formulations may be a computationally inexpensive approach to achieving better performance.

Benchmark Extension. Our benchmark is restricted to incompressible, laminar flows. We believe extending our experimental approach to compressible flows, boundary layer meshes, and turbulent regimes are all important and exciting directions for future work.

5.2. Open-Source Release. The code, data, and scripts to reproduce all our results will be released with an open source licence after publication.
Acknowledgments.

REFERENCES

[1] C. Alappat, A. Basermann, A. R. Bishop, H. Fehske, G. Hager, O. Schenk, J. Thies, and G. Wellein, A recursive algebraic coloring technique for hardware-efficient symmetric sparse matrix-vector multiplication, ACM Trans. Parallel Comput., 7 (2020).

[2] M. S. Alnæs, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. E. Rognes, and G. N. Wells, The FEniCS project version 1.5, Archive of Numerical Software, 3 (2015).

[3] R. Anderson, J. Andrej, A. Barker, J. Bramwell, J.-S. Camier, J. C. V. Dobrev, Y. Dudouit, A. Fisher, T. Kolev, W. Pazner, M. Stowell, V. Tomov, I. Akkerman, J. Dahm, D. Medina, and S. Zampini, MFEM: A modular finite element methods library, Computers & Mathematics with Applications, 81 (2021), pp. 42–74, https://doi.org/10.1016/j.camwa.2020.06.009.

[4] ANSYS, Ansys fluent user’s guide, 2019r1, section 7.3.2 07/25/2021.

[5] ANSYS, Ansys fluent - cfd software, ansys, 2016.

[6] D. A. Dupont and Y. Liu, Back and forth error compensation and correction methods for removing errors induced by uneven gradients of the level set function, Journal of Computational Physics, 190 (2003), pp. 311–324.

[7] R. Codina, An iterative penalty method for the finite element solution of the stationary navier-stokes equations, Computer Methods in Applied Mechanics and Engineering, 110 (1993), pp. 237–262.

[8] O. Ding, T. Shinar, and C. Schroeder, Affine particle in cell method for mac grids and fluid simulation, Journal of Computational Physics, 408 (2020), p. 109311.

[9] J. P. V. Doormaal and G. D. Raithby, Enhancements of the simple method for predicting incompressible fluid flows, Numerical Heat Transfer, 7 (1984), pp. 147–163.

[10] T. F. Dupont and Y. Liu, Back and forth error compensation and correction methods for removing errors induced by uneven gradients of the level set function, Journal of Computational Physics, 190 (2003), pp. 311–324.

[11] W. E and J.-G. Liu, Projection method i: Convergence and numerical boundary layers, SIAM J. Numer. Anal, 32 (1995), pp. 1017–1057, https://doi.org/10.1137/0732047.

[12] 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, 2005.

[13] C. R. Erie and D. A. Steinman, Exact fully 3d navier-stokes solutions for benchmarking, International Journal for Numerical Methods in Fluids, 19 (1994), pp. 369–375.

[14] R. Eymard, T. Gallouët, and R. Herbin, Finite volume methods, Handbook of numerical
analysis, 7 (2000), pp. 713–1018.
[25] C. Fu, Q. Guo, T. Gast, C. Jiang, and J. Teran, A polynomial particle-in-cell method, ACM Transactions on Graphics (TOG), 36 (2017), pp. 1–12.
[26] V. Girault and P.-A. Raviart, Theory and Approximation of the Navier-Stokes Problem, Springer Berlin Heidelberg, Berlin, Heidelberg, 1986, pp. 278–367.
[27] R. Glowinski, Finite element methods for incompressible viscous flow, in Numerical Methods for Fluids (Part 3), vol. 9 of Handbook of Numerical Analysis, Elsevier, 2003, pp. 3–1176.
[28] F. H. Harlow, The particle-in-cell method for numerical solution of problems in fluid dynamics, tech. report, Los Alamos Scientific Lab., N. Mex., 1962.
[29] F. H. Harlow and J. E. Welch, Numerical calculation of time-dependent viscous incompressible flow with free surface, The physics of fluids, 8 (1965), pp. 2182–2189.
[30] W. D. Henshaw and N. A. Petersson, A Split-Step Scheme for the Incompressible Navier-Stokes Equations, World Scientific, 2003, pp. 108–125.
[31] Y. Hu, T. Schneider, X. Gao, Q. Zhou, A. Jacobson, D. Zorin, and D. Panozzo, Triwild: Robust triangulation with curve constraints, ACM Trans. Graph., 38 (2019).
[32] W. Jakob, M. Tarini, D. Panozzo, and O. Sorkine-Hornung, Instant field-aligned meshes, ACM Transactions on Graphics (Proceedings of SIGGRAPH ASIA), 34 (2015).
[33] H. Jasak, Error analysis and estimation for the finite volume method with applications to fluid flows, PhD thesis, University of London, 1996.
[34] C. Jiang, C. Schroeder, A. Selle, J. Teran, and A. Stomakhin, The affine particle-in-cell method, ACM Transactions on Graphics (TOG), 34 (2015), pp. 1–10.
[35] C. Jiang, C. Schroeder, and J. Teran, An angular momentum conserving affine-particle-in-cell method, Journal of Computational Physics, 338 (2017), pp. 137–164.
[36] B. S. Kirk, J. W. Peterson, R. H. Stogner, and G. F. Carey, libMesh: A C++ Library for Parallel Adaptive Mesh Refinement/Coarsening Simulations, Engineering with Computers, 22 (2006), pp. 237–254.
[37] O. Kolditz, Computational Methods in Environmental Fluid Mechanics, Springer, Berlin, Heidelberg, 01 2002, https://doi.org/10.1007/978-3-662-04761-3.
[38] L. Li, A split-step finite-element method for incompressible navier-stokes equations with high-order accuracy up-to the boundary, Journal of Computational Physics, 408 (2020), p. 109274.
[39] R. W. MacCormack, The effect of viscosity in hypervelocity impact cratering, Journal of spacecraft and rockets, 40 (2003), pp. 757–763.
[40] A. T. T. McRae, G.-T. Bercea, L. Mitchell, D. A. Ham, and C. J. Cotter, Automated generation and symbolic manipulation of tensor product finite elements, SIAM Journal on Scientific Computing, 38 (2016), pp. S25–S47.
[41] MFEM: Modular finite element methods [Software]. mfem.org, https://doi.org/10.11578/dc.20171025.1248.
[42] T.-I.-S. Michel Fortin, R. Glowinski, Augmented Lagrangian methods: applications to the numerical solution of boundary-value problems, Studies in mathematics and its applications 15, North-Holland, 1983.
[43] F. Moukalled, L. Mangani, and M. Darwish, The Finite Volume Method in Computational Fluid Dynamics, Springer International Publishing, 2016.
[44] S. Patankar and D. Spalding, A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows, International Journal of Heat and Mass Transfer, 15 (1972), pp. 1787 – 1806.
[45] Z. Qu, X. Zhang, M. Gao, C. Jiang, and B. Chen, Efficient and conservative fluids using bidirectional mapping, ACM Transactions on Graphics (TOG), 38 (2019), pp. 1–12.
[46] Y. Renard and K. Poulios, Getfem: Automated fe modeling of multiphysics problems based on a generic weak form language, ACM Trans. Math. Softw., 47 (2020), https://doi.org/10.1145/3412849.
[47] E. Rivoalen and S. Huberson, The particle strength exchange method applied to axisymmetric viscous flows, Journal of Computational Physics, 168 (2001), pp. 519–526.
[48] K. Salari and P. Knupp, Code verification by the method of manufactured solutions, tech. report, Sandia National Lab., 06 2000.
[49] J. S. Sawyer, A semi-lagrangian method of solving the vorticity advection equation, Tellus, 15 (1963), pp. 336–342.
[50] T. Schneider, J. Dumas, X. Gao, D. Zorin, and D. Panozzo, Polyfem. https://polyfem.github.io/, 2019.
[51] T. Schneider, Y. Hu, X. Gao, J. Dumas, D. Zorin, and D. Panozzo, A large scale comparison of tetrahedral and hexahedral elements for finite element analysis, 2019, https://arxiv.org/abs/1903.09332.
Appendix A. $L^2$ Residual. $L^2$ norm of the pointwise PDE residual can be, in principle, used to measure the accuracy of different numerical methods in the cases when no analytic solution known, as it directly checks how well the original PDE is satisfied. The complication in direct residual evaluation is that it requires approximation of first and second derivatives of the solution, which is a non-trivial problem for commonly used low-order methods.

The $L^2$ norm of the residual can be defined as

$$R^2(u, p) = \int_0^T \int_\Omega \left\| \frac{\partial u}{\partial t} + \rho (u \cdot \nabla) u - \nu \Delta u + \nabla p \right\|^2$$

To compute this residual for a given numerical solution $u_h, p_h$, we need to evaluate the derivatives and integrate them over both time and space. To treat all methods uniformly we sample velocities and pressure on vertices and compute a cubic spline approximation to the solution.

To evaluate how the derivative approximation affects $R$, we compute it for the Taylor-Green vortex and compare it against the $L^2$ error (Figure 20). To make sure that the derivative error is not dominant, we also plot the residual computed from sampled exact solution as reference (Figure 21).

In Figure 21, we note that all seven methods have different convergence rates between two plots. Besides, in Figure 20, only FE21-BDF3 is consistent between the two plots.

The reason why the residual of both FE11-FLIP and FD-FLIP is not decreasing under refinement is, most likely, because these methods use randomly sampled particles. The random sampling introduces noise in the numerical solution, which is then magnified by our local spline-based scheme for derivative approximation. For other methods, this may be due to discrete solutions not yielding sufficiently accurate estimates of the derivatives needed for residual evaluation.

Appendix B. Experimental Setup.

All experiments are run on individual nodes of an HPC cluster each using two
Fig. 20. Convergence plots of residual and L^2 error against wall-clock time.

Fig. 21. Convergence plots of residual and L^2 error against mesh size.

Intel Xeon Platinum 8268 24C 205W 2.9GHz Processors and 48 threads. We limit the runtime to 48 hours. All methods considered in this study, with the exception of the FV method that uses Ansys Fluent [5], and FD-FLIP, are part of the PolyFEM library [50], and use Pardiso [1, 9, 8] direct solver.

FV-C. We use the pressure-based solver and the laminar model in Ansys Fluent [5]. The coupled scheme is used for the pressure-velocity coupling. For the spatial discretization, we use the least square cell-based scheme for gradients, the second-order scheme for pressure, and the second-order upwind scheme for momentum. We use the second-order implicit scheme provided in Fluent as the transient formulation. The convergence criteria is set to 10^{-6} for the continuity and velocity residual. Other parameters are set to default values.

FE21-BDF3. The non-linear equation is solved via Newton’s method, whose tolerance is set to 10^{-8}. To ease convergence, we first use Picard iterations [22] up to 10^{-3} tolerance.

FE11-SL, FE11-FLIP. We briefly overview [53], focusing on the specific discretization choices that we use in our comparison. For time step n with velocity \( u^n \), the advection step aims at integrating

\[
\frac{\partial u}{\partial t} = -(u \cdot \nabla)u
\]
over time by $\Delta t$ to obtain an intermediate velocity $\bar{u}^n$, where semi-Lagrangian-type methods \cite{49, 39, 20, 45} or particle-in-cell methods\cite{28, 10, 34, 25} are often applied. After advection, body forces are integrated into the velocity field via the forward Euler scheme. The diffusion step resolves the viscosity terms (smoothing the current velocity field) and obtains $\tilde{u}^n$ via

$$\frac{\rho \tilde{u}^n - \bar{u}^n}{\Delta t} = \nu \Delta \tilde{u}^n.$$ 

Finally, the pressure projection step projects the velocity field back to a divergence-free state:

$$\frac{u^{n+1} - \tilde{u}^n}{\Delta t} = -\frac{1}{\rho} \nabla p^{n+1} \quad \text{s.t.} \quad \nabla \cdot u^{n+1} = 0.$$ 

We use the $P_1$ or $Q_1$ basis for both velocity and pressure in the FEM discretization. The advection requires to find the element containing a given point and compute its local coordinates in that element. We use spatial hashing \cite{60} to identify a small set of candidate elements. We then compute the local coordinates with respect to each, and identify the element for which these are within the valid $[0, 1]$ range. For triangular and tetrahedral meshes, local coordinates are computed by solving a small linear system; for quadrilateral and hexahedral meshes, the bi-linear and tri-linear map is inverted using Newton’s method.

In the pressure projection step, we solve the Laplacian equation by FE to get the pressure field. We then compute the gradient of pressure at each node by averaging the gradient in all adjacent elements.

**FD-FLIP.** FLIP is used for advection, where the specific ODE per FLIP particle is solved via RK-3. During the RK-3 solve, when velocity is queried at a point outside of the domain, we project the sample point to the closest point on the domain boundary to interpolate velocity. If a FLIP particle goes outside of domain, we simply ignore its contribution. As FLIP particle resampling is performed at the beginning of each time step to ensure $2^d$ particles per cell ($d = 2$ or $3$ is the dimension of space), a nearly uniform particle distribution is always maintained at the expense of introducing a small amount of random noise. After advection, Dirichlet boundary conditions are enforced by overwriting the boundary velocities.

Diffusion and pressure projection linear systems are solved using Pardiso \cite{1, 9, 8}. For diffusion, velocities are the unknowns; we enforce boundary conditions with linear equality constraints, requiring the boundary velocity DOF at the face centers and the boundary velocity interpolation at the edge centers are all equal to the prescribed values. On the other hand, boundary conditions in the pressure projection system follows the standard scheme \cite{13}: a volume fraction weighted gradient operator is applied to better resolve solid boundaries that are not grid-aligned.

**Appendix C. Neumann boundary conditions.**

The zero Neumann boundary condition is enforced differently for different methods.

**FE21-BDF3.** The zero Neumann boundary condition $\nu \frac{\partial u}{\partial n} + pm = 0$ is naturally enforced on $\partial \Omega_N$ and no special treatment is needed.

**FD-FLIP.** To enforce the zero Neumann boundary condition $\frac{\partial u}{\partial n}$ on the boundary $\partial \Omega_N$ is constrained to be zero, which can be directly enforced in the diffusion step. In the pressure projection step, this is treated by setting the ghost pressure outside the domain at the boundary $\partial \Omega_N$ to zero.
**FE11-SL, FE11-FLIP.** Similar to FD-FLIP, $\frac{\partial u}{\partial n}$ on the boundary $\partial \Omega_N$ is constrained to be zero in the diffusion step, which is naturally enforced under FE spatial discretizations. In the pressure projection step, pressure on the boundary $\partial \Omega_N$ is set to zero.

**FE11-AB2AM2, FE21-AB2AM2.** Similar to FE11-SL and FE11-FLIP, $\frac{\partial u}{\partial n}$ on the boundary $\partial \Omega_N$ is constrained to zero in the predictor and corrector step of velocity. When we solve pressure based on the velocity, pressure on the boundary $\partial \Omega_N$ is set to zero.

**FV-C.** In Ansys Fluent, pressure outlet boundary [4] conditions are used to define the static pressure at flow outlets. We observed that FE with zero Neumann boundary and FV with zero pressure outlet boundary converge to the same result in the test problems. Thus, in all our experiments, we use this observation when imposing zero Neumann on FV simulations.