Parallel Incompressible Navier-Stokes Solver with Stabilization using deal.II

Bibek Shrestha

Department of Mechanical, Aerospace, and Nuclear Engineering Rensselaer Polytechnic Institute, 110 8th St. Troy, NY 12180

Abstract

This project implements a parallel finite element solver for the incompressible Navier-Stokes equations, incorporating SUPG, PSPG, and LSIC stabilization methods using deal.II. The generalized-alpha time discretization was used for time integration. Polynomial manufactured solution method used to test for validation of the solver. Catch2 and Github actions were used to run automated tests.

Keywords: Navier-Stokes, Finite Element Method, deal.II, Generalized-alpha, SUPG, PSPG, LSIC

Developer's repository link: https://github.com/Bibek-ko-git/Projects/tree/main/Computation@Scale
Licensing provisions(please choose one): BSD 3-clause
Programming language: C++
Supplementary material: https://github.com/Bibek-ko-git/FEniCSx-for-NS/blob/main/FEMproject.pdf

1. Summary

InNSgen-αFEM is a C++ software package that provides a finite element implementation of the incompressible Navier-Stokes equations using the deal.II finite element library. The software implements a stabilized P2-P1 Taylor-Hood element formulation for spatial discretization and a generalized-alpha method for time stepping. The weak formulations is also supplemented by incorporating stabilization terms such as SUPG, PSPG and LSIC. The package aims to complete integration of a specialized module for solution verification through the method of manufactured solutions (MMS), enabling verification of the solver's accuracy and convergence properties. InNSgen-αFEM is designed for research purposes to extend the solver capabilities of OpenIFEM, offering a modular and extensible framework with parallel execution support via MPI and automated testing with Catch2 and github actions. The project is aimed to utilize the newfound knowledge earned from the Computation at Scale class to used the previously developed fluid solver in FENICSx and convert into C++ code using deal.II.

2. Statement of Need

Accurate and stable numerical solutions of the incompressible Navier-Stokes equations are essential in computational fluid dynamics. Stabilization techniques address numerical instabilities common with standard finite element discretizations. A robust Finite Element incompressible Navier Stokes is required in the Computational Fluid Mechanics lab at RPI to support the slightly compressible solver used by the lab students. As the current solver used by OpenIFEM blows up for a problem with a Reynolds number greater than 20.

3. Nature of problem

InNSgen-αFEM addresses the numerical solution of incompressible Navier-Stokes equations, which govern fluid flow in numerous engineering applications from aerodynamics to biomedical flows. These equations present significant computational challenges due to several factors:

- Nonlinearity: In the momentum equation the $(u \cdot \nabla u)$ introduces nonlinearity, requiring iterative solver.

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u}\right) = -\nabla p + \mu \Delta \mathbf{u} + \rho \mathbf{f}$$

- Incompressibility constraint: The divergence-free constraint continuity equation must be satisfied precisely to conserve mass.

$$\nabla \cdot \mathbf{u} = 0$$

- Saddle-point structure: As we are solving for both the equations the pressure-velocity coupling forms a saddle-point problem which requires careful discretization to avoid numerical instabilities

- Multi-scale phenomena: At high Reynolds number the fluid flows requires better mesh resolutions near boundary layers and vortices. Better resolutions are required to observe shock formation.

- Verification and Validation: Comparing results with an experimental study for a standard benchmark problems can provide a qualitative validation. But, robust verification requires a zero error while solving for a manufactures solution.

So the over all package would require a proper Finite Element, Difference or Volume Discretization, Matrix multiplication iterators or solvers, mesh refinement mechanisms, Parallelization and testing mechanisms.

The project aimed to solve a 2D pipe flow problem with a channel of length 2.2 m and height 0.41 m with an inlet velocity of 1m/s. Since it did not converge a polynomial manufactured solution case was tested to understand where the problem lies in the solver.

4. Solution method

The over all solution and methodology used for solving the problem with some of their code snippet is given below:

4.1. Spatial Discretization:

P2-P1 Taylor-Hood elements provide quadratic approximation for velocity and linear approximation for pressure, satisfying the Ladyzhenskaya-Babuška-Brezzi (LBB) condition for stability[1]. For convection-dominated flows, three stabilization techniques are implemented:

- Streamline Upwind Petrov-Galerkin (SUPG) to control oscillations in flow direction. It is a type of discontinuous Galerkin element which provides more weightage to the upstream part of the element in order to address numerical instabilities due to the advection[2].
- Pressure-Stabilizing Petrov-Galerkin (PSPG) to relax the inf-sup condition. It helps to address the instabilities associated with pressure-velocity coupling, allowing the use of equal-order interpolation for both pressure and velocity[3].
- Least-Squares Incompressibility Constraint (LSIC) to enhance mass conservation. It is typically employed with low-order finite elements to mitigate numerical instabilities[4].

4.2. Temporal discretization:

The generalized-alpha method offers second-order accuracy with controllable numerical dissipation, particularly suitable for transient simulations as it effectively damps high-frequency numerical oscillations[5]. The over all discretized Finite Element Equation to be solved is given below. The fundamental and derivations of the equation is covered in FEMproject.pdf mentioned earlier.

$$\begin{aligned}
&\int_\Omega \rho\ddot{\mathbf{u}}^{n+1-\alpha_m}\mathbf{w}\,d\Omega + \int_\Omega \rho\frac{\mathbf{u}^{n+1-\alpha_f}-\mathbf{u}^n}{\Delta t}\mathbf{w}\,d\Omega + (1-\alpha_f)\int_\Omega \rho(\mathbf{u}^n\cdot\nabla)\mathbf{u}^n\mathbf{w}\,d\Omega \\
&+\alpha_f\int_\Omega \rho(\mathbf{u}^{n+1}\cdot\nabla)\mathbf{u}^{n+1}\mathbf{w}\,d\Omega + (1-\alpha_m)\nu\int_\Omega \nabla\mathbf{u}^n:\nabla\mathbf{w}\,d\Omega + \alpha_m\nu\int_\Omega \nabla\mathbf{u}^{n+1}:\nabla\mathbf{w}\,d\Omega \\
&-(1-\alpha_f)\int_\Omega p^n\nabla\cdot\mathbf{w}\,d\Omega + \int_\Omega q\nabla\cdot\mathbf{u}^{n+1-\alpha_f}\,d\Omega - \alpha_f\int_\Omega p^{n+1}\nabla\cdot\mathbf{w}\,d\Omega \\
&\sum_{n_{el}}\int_{\Omega_{n_{el}}}\tau_{SUPG}[\vec{u}^{n+1-\alpha_f}\cdot\nabla\vec{v}][\vec{u}^{n+1-\alpha_f}\cdot\nabla\vec{u}^{n+1-\alpha_f}]\,d\Omega \\
&+\sum_{n_{el}}\int_{\Omega_{n_{el}}}\tau_{PSPG}[\vec{u}^{n+1-\alpha_f}\cdot\nabla\vec{v}]\nabla p^{n+1-\alpha_f}\,d\Omega + \\
&\sum_{n_{el}}\int_{\Omega_{n_{el}}}\tau_{LSIC}\nabla\cdot\mathbf{w}\nabla\cdot\mathbf{u}^{n+1-\alpha_f}\,d\Omega = \int_\Omega \mathbf{f}^{n+1-\alpha_f}\mathbf{w}\,d\Omega \quad (1)
\end{aligned}$$

4.3. Nonlinear solution:

Newton's method linearizes the system at each time step, with adaptive damping for enhanced robustness in challenging flow regimes.

Listing 1: Newton Solver

```
assemble_system();
// Compute residual norm
residual_norm = system_rhs.l2_norm();
if (newton_iteration == 0)
  initial_residual_norm = residual_norm;
// Check convergence
if (residual_norm < newton_tolerance || residual_norm < 1e-10)
  break;
// Solve nonlinear system
newton_update = 0;
solver.solve(system_matrix, newton_update,
             system_rhs, preconditioner);
// Apply damping
solution.add(damping, newton_update);
```

4.4. Linear Solvers:

GMRES or BiCGStab algorithms with ILU preconditioning efficiently solve the resulting sparse linear systems, leveraging Trilinos through deal.II.

Listing 2: GMRES with ILU preconditioning

```
// Initialize preconditioner with Trilinos
TrilinosWrappers::PreconditionILU::AdditionalData ilu_data;
ilu_data.overlap = 0;
ilu_data.ilu_fill = 10.0;
ilu_data.ilu_atol = 1e-8;
ilu_data.ilu_rtol = 1.0;
preconditioner.initialize(system_matrix, ilu_data);

// GMRES solver setup
SolverControl solver_control(1000, 1e-8 * system_rhs.l2_norm());
SolverGMRES<TrilinosWrappers::MPI::Vector>solver(solver_control);

// Solve linear system
solver.solve(system_matrix, newton_update,
             system_rhs, preconditioner);
```

4.5. Parallelization

Domain decomposition via MPI divides the computational domain across processors, with careful handling of ghost cells and parallel vector operations for consistent solutions across process boundaries. This is still incomplete as the solver is still having problem copying values into the ghost cells.

Listing 3: Parallelization using MPI

```
// Parallel distributed triangulation
parallel::distributed::Triangulation<dim>
                              triangulation{mpi_communicator};

// Update ghost values for parallel access
solution.update_ghost_values();

// Only iterate over locally owned and ghost cells
for (const auto &cell : dof_handler.active_cell_iterators()) {
  if (cell->is_locally_owned() || cell->is_ghost()) {
    // Cell operations...
  }
}

// Parallel reduction to get global maximum
double global_max = Utilities::MPI::max(local_max,
                                        mpi_communicator);
```

4.6. Verification

The Method of Manufactured Solutions analytically derives forcing terms that produce known solutions, enabling precise error quantification and convergence testing using automated Catch2 testing framework. Simple polynomial manufactured solution was used for the convergence study.

Listing 4: Polynomial manufactures solution

```
  // Simple manufactured solution - polynomial
  // u_x = x^2(1-x)^2 * y(1-y) * t
  // u_y = -x(1-x) * y^2(1-y)^2 * t
  // p = (x-0.5)*(y-0.5) * t
```

5. Results and Validation

The given equation was solved on a 1m × 1m square whose exact solution is as shown in following figure 1.

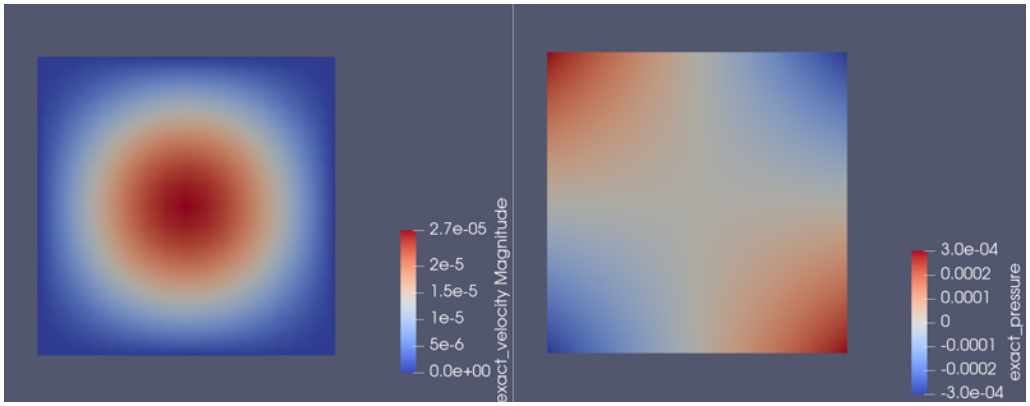

Figure 1: Exact solution of manufactures solution

The solution obtained on the sixth time step before the residual started blowing up is shown in figure 2. Also mesh refinement study was done using different mesh size 1,

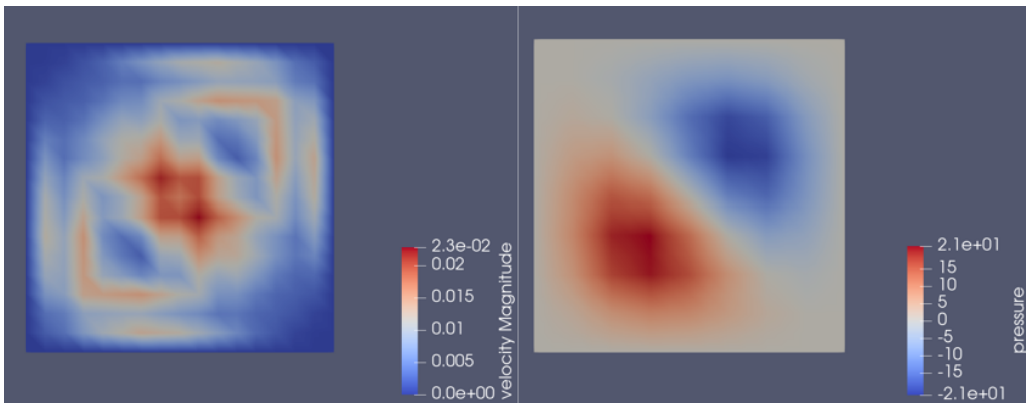

Figure 2: Obtained solution of manufactures solution

1/2, 1/4 and 1/8 with convergence study on each. The figure 3 shows the result of the study along with a standard change in L2-norm for a second order accurate case.

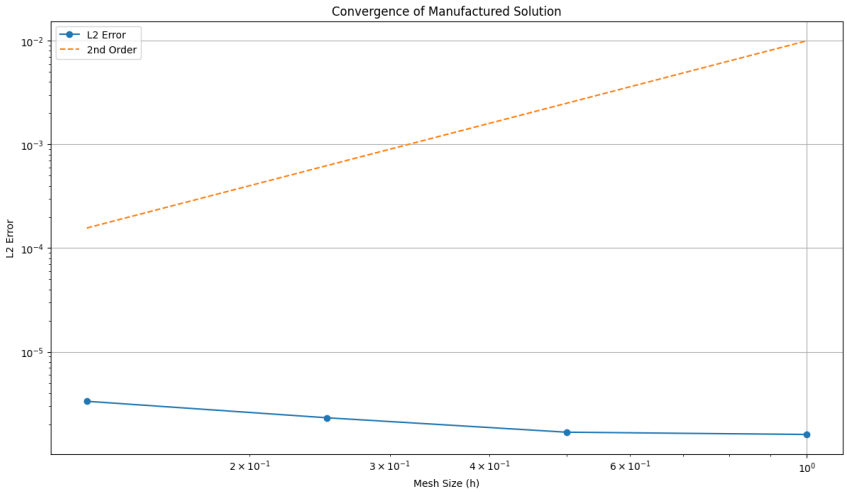

Figure 3: Mesh refinement convergence study

Convergence test shows that the solver is not accurate and error is increasing with decreasing mesh size which is not optimal. It demonstrates that the current version of solver is not accurate and needs more work before applying to solve complex problems.

6. Conclusion

The developed solver lack accuracy and robustness as the manufactured solution is not seem to converge. The MPI integration is incomplete as since the solver fails to run with MPI. These current inefficiency and incomplete version is not suitable for solving any complex fluid dynamics simulations.

7. Additional comments: restrictions and unusual features

7.1. Limitations:

- the whole derivations of solution is limited to 2D geometries
- no adaptive mesh refinement, refinement study was done by changing the mesh length and running the solver for each case
- Fixed physical properties line viscosity and density

7.2. Parallel ghost points handling:

Values assigned to the parallel ghost points that access the vector data on the cell is causing parallel execution error.

7.3. Manufactured solution implementation and modular package:

The software package is aimed to be made modular which can handle more test cases or fluid simulation example with the same root solver.

References

[1] Tobias Gelhard, Gert Lube, Maxim A. Olshanskii, and Jan-Hendrik Starcke. Stabilized finite element schemes with lbb-stable elements for incompressible flows. *Journal of Computational and Applied Mathematics*, 177(2):243–267, 2005. ISSN 0377-0427. doi: https://doi.org/10.1016/j.cam.2004.09.017. URL `https://www.sciencedirect.com/science/article/pii/S0377042704004236`.

[2] Alexander Nelson Brooks. *A Petrov-Galerkin Finite Element Formulation for Convection Dominated Flows*. PhD thesis, California Institute of Technology, 1981. URL `https://thesis.library.caltech.edu/430/4/brooks_an_1981.pdf`.

[3] Tayfun Tezduyar and Sunil Sathe. Stabilization parameters in supg and pspg formulations. *Journal of Computational and Applied Mathematics*, 2023. URL `https://www.tafsm.org/PUB_PRE/jALL/j106-JCAM-SP.pdf`. Accessed: 2024-12-10.

[4] Yuxiang Liang and Shun Zhang. Least-squares methods with nonconforming finite elements for general second-order elliptic equations. *Journal of Scientific Computing*, 2023. URL `https://link.springer.com/article/10.1007/s10915-023-02246-x`. Accessed: 2024-12-10.

[5] A. Lovrić, Wulf G. Dettmer, Chennakesava Kadapa, and Djordje Perić. A new family of projection schemes for the incompressible navier–stokes equations with control of high-frequency damping. *Computer Methods in Applied Mechanics and Engineering*, 339:160–183, 2018. ISSN 0045-7825. doi: 10.1016/j.cma.2018.05.006. URL `https://www.sciencedirect.com/science/article/pii/S0045782518302494`.