Implementation and Validation of a 2D Incompressible SIMPLE Solver in Python

Emin KHALIFAYEV   Abel Inalegwu ONUH

École des Mines de Saint-Étienne, October 2025

1 Introduction

This project presents a Python-based simulation of a two-dimensional, steady, incompressible Poiseuille flow using the SIMPLE, short for the Semi-Implicit Method for Pressure-Linked Equations algorithm. The main objective is to obtain the numerical solution and compare it with the analytical velocity profile for laminar flow between parallel plates. In this work we looked at the effects of grid resolution, relaxation factors, and Reynolds number on solver stability and accuracy building up on the already existing framework for solving the 2-D flow equation developed in the lecture notes.

2 Numerical Methodology

2.1 Finite Volume Formulation

We discretized the steady incompressible Navier–Stokes equations using the finite volume method (FVM) on a staggered grid, with pressure $P_{i,j}$ stored at cell centers and velocities $u_{i+\frac{1}{2},j}$, $v_{i,j+\frac{1}{2}}$ located at the faces, in order to prevent checkerboard oscillations when compared to collocated grid. We have explicitly implemented the contribution of diffusive terms and convective terms, after what we have applied the central differences on both. In the case of convective flow, it was necessary to use the upwind scheme. Finally, substituting it into the integrated momentum balance produced a discrete transport relation for each control volume.

2.2 Pressure–Velocity Coupling: The SIMPLE Algorithm

To solve this newly-obtained discretized momentum equation, we had to couple momentum and continuity, and therefore we decided to use the Semi–Implicit Method for Pressure–Linked Equations (SIMPLE).

Here we briefly describe how the SIMPLE pressure-velocity coupling works. At each outer iteration k, the solver guessed (P^k, u^k, v^k), solved the momentum equations for provisional velocities (u^*, v^*), and introduced corrections (u', v', P') such that $u = u^* + u'$, $v = v^* + v'$, and $P = P^k + P'$.

A Poisson–type pressure–correction equation derived from continuity was then solved to remove local mass imbalance $b_{i,j}$, after which velocity fields were updated. Under–relaxation factors $\alpha_{u,v} = 0.7$ and $\alpha_p = 0.3$ stabilized convergence until normalized residuals fell below 10^{-6}.

2.3 Boundary Conditions and Corner Treatment

All boundary conditions were applied consistently to velocity and pressure fields. We enforced no–slip condition at the wall with $u = v = 0$ (Dirichlet), and zero–gradient condition at the outlet boundary(Neumann). The velocity at the left wall was fixed, and a reference pressure point was fixed to remove ambiguity (with the Poisson null space).

At solid walls, wall-neighbor coefficients were set to zero, approximating the Neumann pressure condition: $\frac{\partial P}{\partial n} = \frac{\partial P'}{\partial n} = 0$. (impermeable boundary).

Just like in the lecture nodes we have zeroed the missing neighboring fluxes in `solve_x_momentum.py` and `solve_y_momentum.py`, to stabilize corner nodes and instead absorb their "net" effect into the diagonal term a_P.

A key improvement from the lecture notes was explicitly implementing convective and diffusive terms, and including of density ρ in the pressure–correction coefficients and as one of the arguments of `solve_P.py`.

2.4 Implementation Structure and Debugging

We had plenty of errors during the early development, including dimensioning errors and incomplete logic boundary for boundary conditions, especially in link coefficients and wall terms. We were able to fix it.

We corrected these by re-deriving flux equations with density ρ where required, and reapplying boundary conditions at each iteration. Residual and symmetry checks validated consistency, while pure–diffusion tests confirmed correctness for $u = v = 0$.

We derived it from **mass conservation** ($\nabla \cdot (\rho \mathbf{u}) = 0$), whereas the lecture notes use **volume conservation** ($\nabla \cdot \mathbf{u} = 0$). For an incompressible fluid, these are physically the same, but we decided to take an extra challenge.

Source Term (Mass/Volume Imbalance)
From the Lecture Notes (Volume Flux Imbalance, Eq. 3.30):

$$b_{i,j} = \left(u^*_{x,i,j} - u^*_{x,i+1,j}\right)\Delta y + \left(u^*_{y,i,j} - u^*_{y,i,j+1}\right)\Delta x \tag{1}$$

From the Python Code (Mass Flux Imbalance):

$$b_{i,j} = \rho\left(u^*_{x,e} - u^*_{x,w}\right)\Delta y + \rho\left(u^*_{y,n} - u^*_{y,s}\right)\Delta x \tag{2}$$

Pressure Equation Link Coefficients (a_p)
From the Lecture Notes (Eq. 3.31-3.35):

$$a_{p,e} = \frac{\Delta y^2}{a_{x,o}(i+1,j)}, \quad a_{p,w} = \frac{\Delta y^2}{a_{x,o}(i,j)} \tag{3}$$

$$a_{p,n} = \frac{\Delta x^2}{a_{y,o}(i,j+1)}, \quad a_{p,s} = \frac{\Delta x^2}{a_{y,o}(i,j)} \tag{4}$$

$$a_{p,o} = a_{p,e} + a_{p,w} + a_{p,n} + a_{p,s} \tag{5}$$

From the Python Code (via Influence Coefficients d):

$$d_e = \frac{\Delta y^2}{a_{x,o_e}}, \quad d_w = \frac{\Delta y^2}{a_{x,o_w}}, \quad d_n = \frac{\Delta x^2}{a_{y,o_n}}, \quad d_s = \frac{\Delta x^2}{a_{y,o_s}} \tag{6}$$

$$a_{p,e} = \rho d_e, \quad a_{p,w} = \rho d_w, \quad a_{p,n} = \rho d_n, \quad a_{p,s} = \rho d_s \tag{7}$$

$$a_{p,o} = a_{p,e} + a_{p,w} + a_{p,n} + a_{p,s} \tag{8}$$

Velocity Correction
From the Lecture Notes (Eq. 3.26):

$$u^k_{x,i,j} = u^*_{x,i,j} - \frac{1}{a_{x,o}}(P'_{i,j} - P'_{i-1,j})\Delta y \tag{9}$$

From the Python Code (`correct_ux.py`), where $d_i = \Delta y / a_{x,o_i}$:

$$u_{x,i} = u^*_{x,i} + d_i\left(P'_{i-1} - P'_i\right) \tag{10}$$

To improve runtime, all loops were replaced by NumPy vectorized operations, accelerating computation by roughly 50–100× and enabling efficient parametric studies. A short video demonstration of the full simulation (runtime ≈ 10 s) is available at: `youtube.com/watch?v=cs763mlg21Y`.

It must be noted that the simulation shows the convergence of simple over cycles (iterations) not to be confused with the development of the flow in time

3 Results and Discussion

3.1 Convergence History

The convergence behavior of the SIMPLE solver is illustrated in Fig. 1, which shows the logarithmic decay of normalized residuals for the velocity and pressure–correction equations with iteration count. Residuals decreased super-exponentially over first two orders of magnitude (from 10^0 to 10^-2) and first ∼400 iterations.

After that the residuals started to decrease exponentially monotonically until eventually converging at 1026 iterations confirming that the numerical solution had reached the steady state.

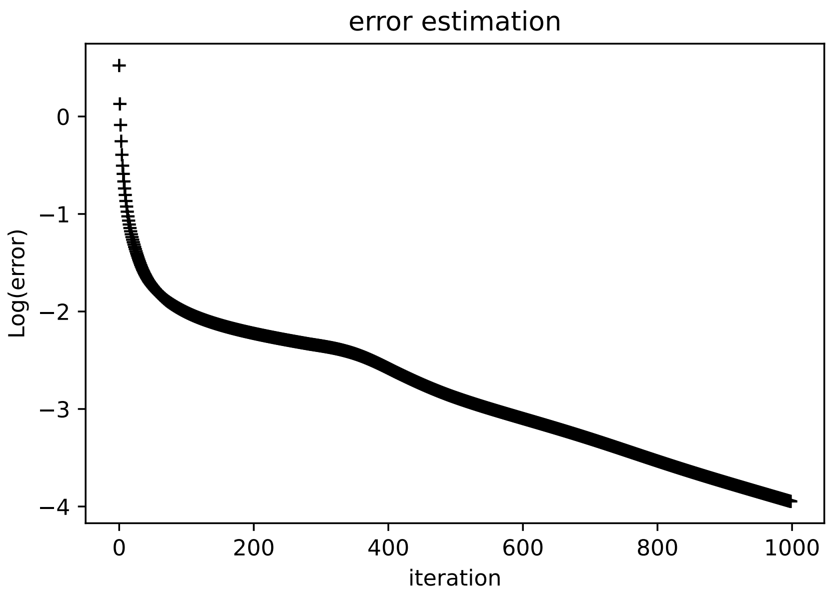
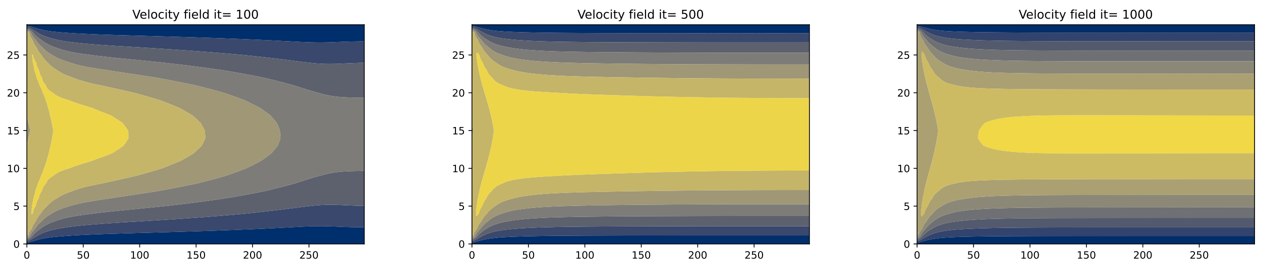


Figure 1: Convergence history showing residual decay of velocity and pressure equations.

This is the characteristic and expected behavior of steady laminar flows solved with the SIMPLE algorithm.Physically, this convergence trend indicates that no unbalanced viscous or pressure forces remained throughout the domain and that continuity was enforced such that the net mass flux in each control volume was near zero.

3.2 Velocity Field Evolution

The evolution of the horizontal velocity field through the SIMPLE iterations is illustrated in Fig. 2. At iteration 100, the velocity profile remained strong at the entrance of the channel and weak near the right wall with minimal development of the parabolic profile. At iteration 500 we could see that the velocities are greater, closer to the centerline everywhere in the channel, but there is no clear transition, the flow is homogeneous. At iteration 1000, we can see a smooth transition of the uniform flow at the entrance of the channel to a fully developed Poiseuille profile towards the right wall, characterized by a steady, laminar flow with maximum velocity at the centerline and zero velocity at the walls ($u = 0$ at both walls, $\partial u/\partial x = 0$ at the outlet).

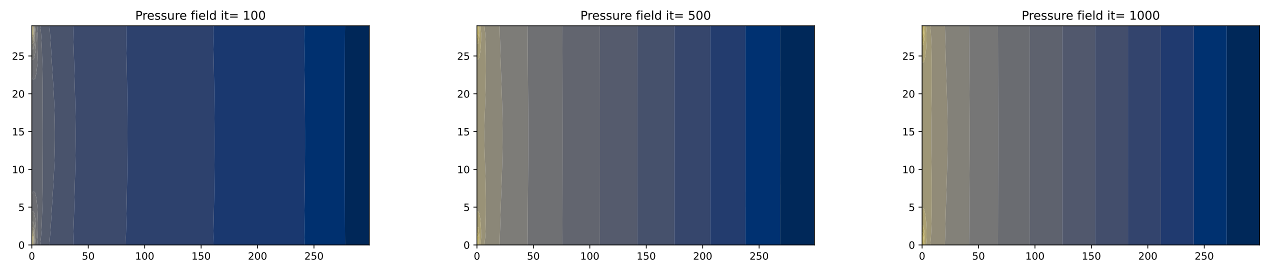


(a) Iteration 100 (b) Iteration 500 (c) Iteration 1000

Figure 2: Evolution of the velocity field during SIMPLE iterations.

3.3 Pressure Field Evolution

The development of the pressure field during the SIMPLE iterations is shown in Fig. 3.



(a) Iteration 100 (b) Iteration 500 (c) Iteration 1000

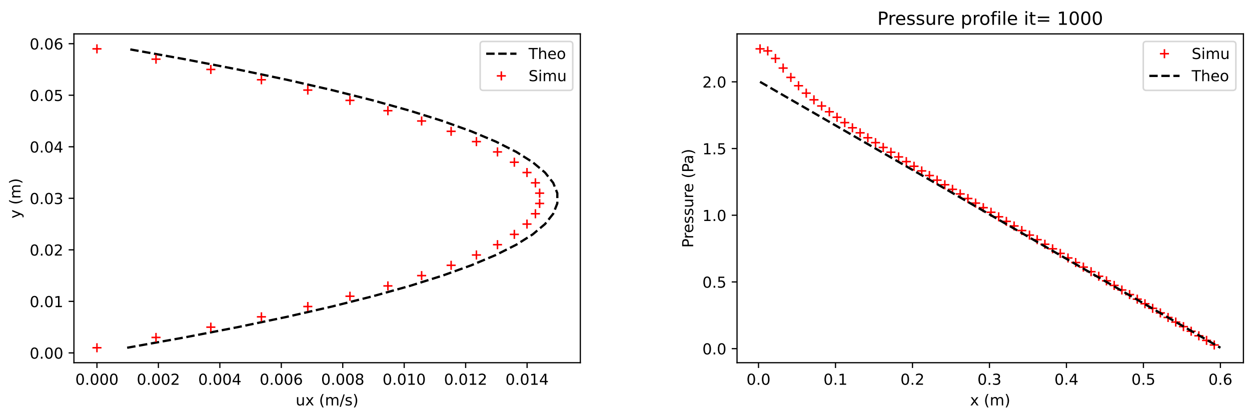
Figure 3: Pressure field evolution through SIMPLE iterations.

At iteration 100, the contours were irregular with a steep gradient near the entrance of the channel and shallow gradients towards the end, meaning that the flow is not fully developed. By iteration 500, the field had smoothed $\frac{\partial P}{\partial x} \approx 0$, and by iteration 1000, a stable gradient formed from the upper-left to the lower-right corner $\frac{\partial P}{\partial x} = 0$. There was not much differences between the iterations 500 and 1000, because the residual loss is already small enough, making all further improvements incrementally smaller. Overall, the pressure decay is nice and smooth downstream, which confirms the physical correctness and convergence of the SIMPLE pressure correction.

3.4 Common Sense: Validation of Numerical Accuracy

Here we compared the numerical velocity and pressure profiles obtained from the SIMPLE solver against analytical Poiseuille solutions in Fig. 4.

Qualitatively, the analytical velocity for fully developed laminar flow follows a parabolic distribution, while the pressure decreases linearly along the channel. Quantitatively, the RMSE$_u$ is below 3%, while the RMSE$_p$ is below 2% and the global continuity residual, $\max(\nabla \cdot \vec{u}) < 10^{-5}$, meaning that the algorithm correctly pictures the physics for the standard case such as Poiseuille flow.



(a) Velocity profile at steady state (Iteration 1000). (b) Pressure profile at steady state (Iteration 1000).

Figure 4: Comparison of numerical and analytical profiles for velocity and pressure at convergence.

This can be confirmed visually by looking at the Figure 4. To achieve better results we can further decrease the tolerance at the cost of number of iterations.

A complete visualization of the iterative evolution of both velocity and pressure fields is presented in (Appendix), illustrating the progressive stabilization of the SIMPLE solution.

4 Conclusion

The truth is SIMPLE is a very robust and powerful solver. Central differences method is precise, and the staggered grid and upwind scheme prevent oscillations. Some minor errors may exist, and this throws some exceptions. Minor warnings such as the division by zero warnings is largely inconsequential, as it only occurs for the first iteration and have no effect on convergence. Minor deviations observed near the corners were attributed to grid diffusion and simplified boundary approximations, yet, remained within acceptable tolerance. From the physics point of view, the Navier-Stokes equations were modified, the equations governing the flow and the ones solved are not exactly the same. However it seems to approximate the underlying physics very well. At the end of the day, as George Box said (and reinforced in the lecture notes) all models are wrong but some are useful. The fact that the code executes in a runtime of seconds on a low-end laptop is already big win.

5 Bonus: Hyperparameter Study

A parametric study was conducted to assess the effect of grid aspect ratio, grid size, relaxation factors, and Reynolds number on solver convergence and stability. By varying one parameter at a time ("ceteris paribus") we were able to trace the effect of Reynolds number, the grid size, aspect ratio, velocity and pressure relaxation on the number of iterations, convergence rate, final residual, and RMSE of the velocity and pressure profiles.

Table 1 summarizes the principal findings, while detailed trends and full plots are provided in the Appendix.

Table 1: Summary of parametric study results and physical interpretation.

Parameter	**Observation**	**Interpretation**
Grid size	Converged iterations rose from ∼900 (coarse) to ∼1250 (fine); RMSE$_u$ dropped from 7.1×10^{-4} to 3.8×10^{-4}.	Finer grid leads to longer convergence but smaller residual. Accuracy improved with minimal cost increase beyond 300×300.
Grid aspect ratio (GAR)	Optimum near 10 (longer pipe); ($i_{\max}/j_{\max} > 10$).	Convergence slowed and RMSE rose for elongated cells
Velocity relaxation (α_{uv})	Stable for 0.6−0.9, optimum near 0.8; larger values caused mild oscillations.	Adequate under-relaxation ensured stable SIMPLE momentum prediction, but increase number of iterations. Relaxation > 1 is non-physical does not converge.
Pressure relaxation (α_p)	Stable for 0.2−0.3; instability observed beyond 0.5.	Over-correction of pressure can cause oscillations. Just as before the relaxation > 1 is non-physical does not converge.
Reynolds number (Re)	Iterations increased from ∼575 (Re=1) to ∼1350 (Re=200).	Higher Re introduces non-linearity. While we were able to simulate creeping flow, for Re > 200 the algorithm did not converge

References

[1] S. Martin, *Computational Fluid Dynamics – Introduction*, Lecture notes, Mines Saint-Étienne, France, September 22, 2025.

[2] CFD Online Encyclopedia, *Computational Fluid Dynamics (CFD) Wiki*, Available at: https://www.cfd-online.com/Wiki/Main_Page, accessed October 2025.

[3] S. V. Patankar, *Numerical Heat Transfer and Fluid Flow*, Hemisphere Publishing Corporation, 1980.

[4] P. Wesseling, *Principles of Computational Fluid Dynamics*, Springer-Verlag, 2000.

[5] S. Mazumder, *Detailed Lecture on the SIMPLE Algorithm*, YouTube, Available at: https://www.youtube.com/watch?v=JUbHKZRupnE, accessed October 2025.

[6] *Introduction to the SIMPLE Structure*, YouTube, Available at: https://www.youtube.com/watch?v=OOILoJ1zuiw, accessed October 2025.

Appendix

During testing, we also experimented with our own variation of the SIMPLE algorithm, which we will refer to as EMPLE or (Explicit Method for Pressure-Linked Equations). The hypothesis was that we could double down on the main assumption of SIMPLE (neglecting the contribution of velocities of neighboring nodes) and in a similar fashion neglect the contributions of the neighboring nodes when solving for pressure, with the idea in mind that we could speed-up the convergence and switch from the semi-implicit approach to a fully-explicit formulation. Although the attempt to speed-up the SIMPLE was unsuccessful, it is interesting to consider the reason. The nature of the pressure field is elliptic (meaning that pressure at a point depends on the pressure everywhere else in the system). We tried to calculate pressure at a point and broke the elliptic coupling of pressure and mass flux, generating non–physical (NaN) results. This confirmed that the standard implicit SIMPLE is very robust as it survived the test of time.

Iterative Evolution Panels

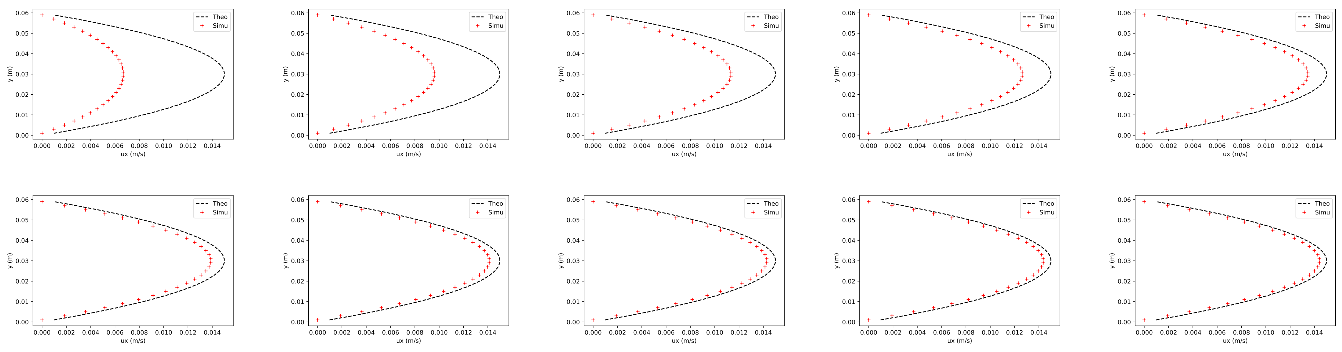


Figure 5: Transverse Poiseuille velocity profiles (iteration 100–1000).

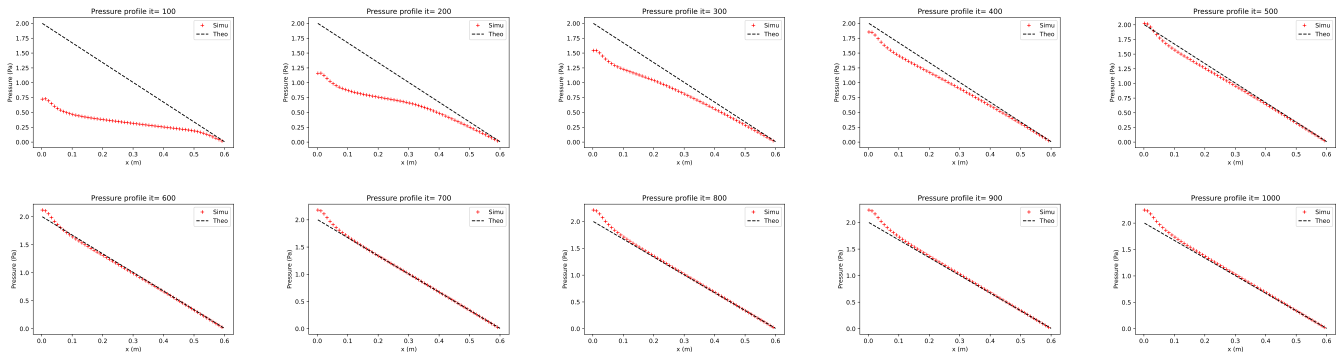


Figure 6: Streamwise pressure profile evolution from iteration 100 to 1000. The contours progressively smooth out, showing a transition from early numerical irregularities to a fully developed linear pressure gradient, confirming steady laminar behavior.

Parametric (Hyperparameter) Study Details

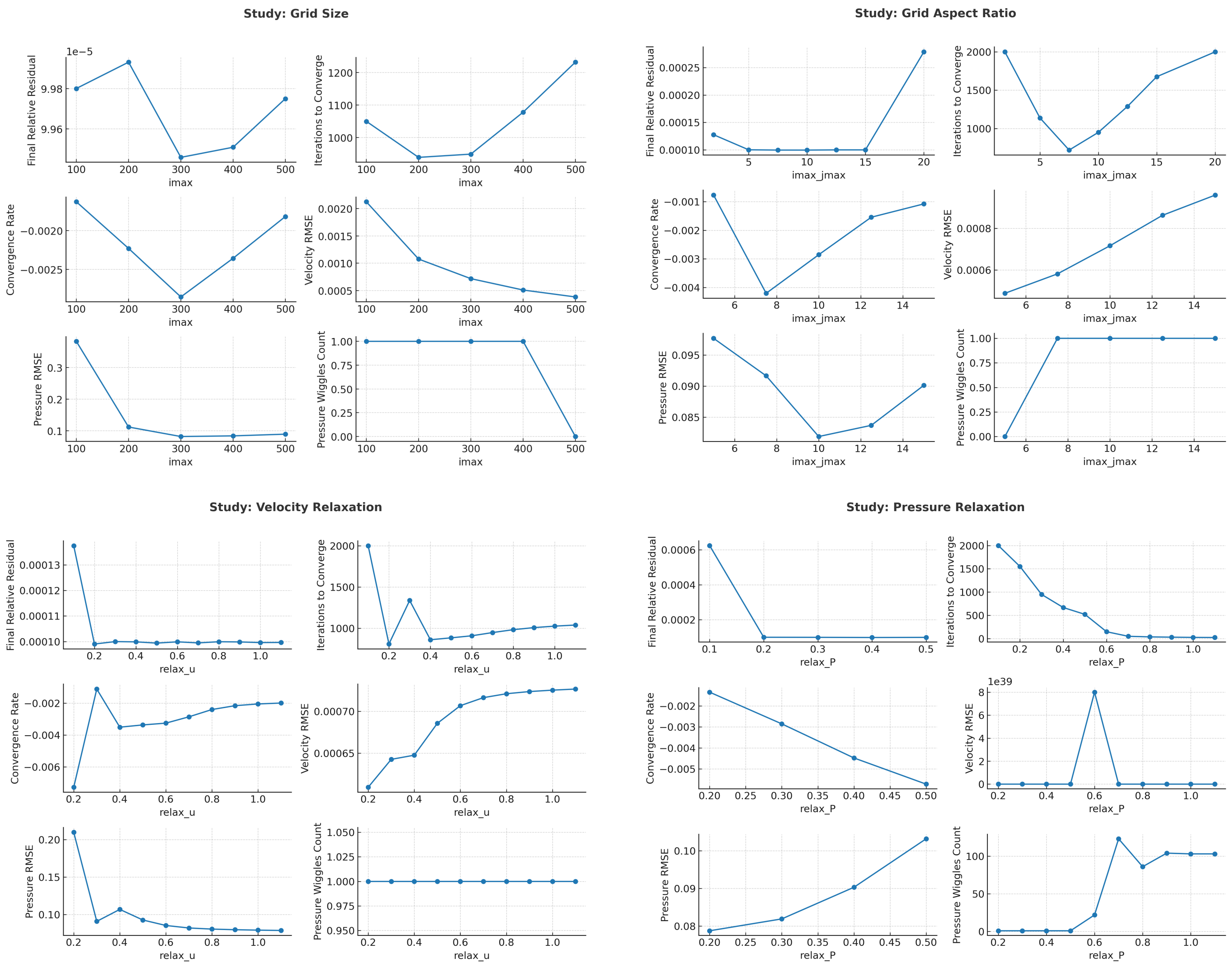


Figure 7: Parametric (hyperparameter) studies: influence of grid size, grid aspect ratio, and relaxation parameters on solver convergence and RMSE trends. Optimal stability was obtained at $\alpha_{uv} = 0.8$, $\alpha_p = 0.2$, and near-square grid aspect ratios.

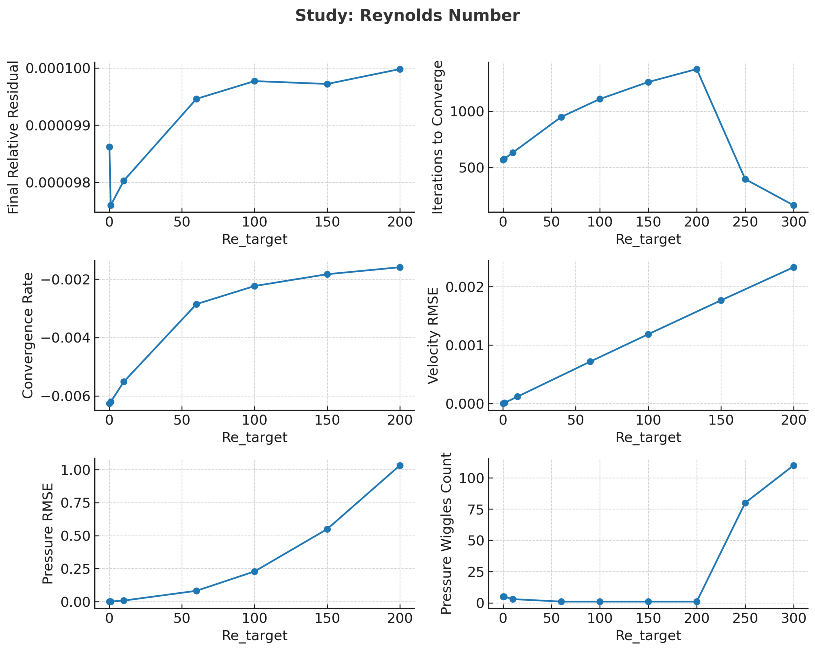


Figure 8: Effect of Reynolds number on convergence and residual evolution. Higher Re increased nonlinearity, slowing convergence but maintaining physical accuracy.