Comparison of approximate inverse preconditioners for the fractional step Navier–Stokes equations

V. S. Djanali[1] S. W. Armfield[2] M. P. Kirkpatrick[3]

(Received 19 January 2011; revised 3 August 2011)

Abstract

Sparse approximate inverses are applied as preconditioners for the fractional step solution of the Navier–Stokes equations. An advantage of this method is that its implementation requires only matrix-vector products and hence is relatively easy to parallelise. Since the coefficients for the pressure Poisson equation are constant, sparse approximate inverses need to be constructed only once, and are recalled in the subsequent iterations. Using the three dimensional turbulent channel flow as a test case, this study shows that the sparse approximate inverse preconditioners have comparable sequential performance to the Incomplete Lower-Upper preconditioner with same amount of fill to the original coefficient matrix.

Contents

1 Introduction

In viscous incompressible flow simulations, fractional step methods are applied to the Navier–Stokes equations to simplify the coupling between the pressure and velocity. The computation of the pressure equation often consumes most of the total computation time of the Navier–Stokes equations, particularly for high accuracy solutions [1]. One way to accelerate the convergence is by preconditioning. Many studies have been conducted on preconditioned systems in various applications [3, 5, 6, 7, 8, 9, e.g.]. However, no single preconditioner is the most effective for every problem. A given preconditioner may perform very well in some cases, but fail in other cases [3, 11]. This study focuses on the effectiveness of various preconditioners used to solve the Navier–Stokes equation via the fractional step method.

1.1 Pressure Poisson equation

One of the difficulties in incompressible flow simulations is caused by the coupling between pressure and velocity in the momentum equations. Using Adams–Bashforth and Crank–Nicolson methods for the time discretisation of the advective and diffusive terms respectively, the Navier–Stokes equations with negligible body force is

$$\frac{v^{n+1}-v^{n}}{\Delta t}+\left[\frac{3}{2}H(v^{n})-\frac{1}{2}H(v^{n-1})\right]=-Gp^{n+1}+\frac{1}{2\mathrm{Re}}L(v^{n+1}+v^{n}),\tag{1}$$

$$Dv^{n+1}=0\,,\tag{2}$$

where H, G, L and D are the advection, gradient, Laplace and divergence operators, respectively. The discrete velocity is v, the discrete pressure p and the time level n.

One way to simplify the Navier–Stokes equations is to decouple the pressure and velocity terms, using a divergence free velocity constraint. This results in two separate equations, one involving momentum terms and the other involving pressure term. The latter is the Pressure Poisson Equation (PPE) shown in Equation (4). Armfield and Street [1] compared the time accuracy and efficiency of various fractional step schemes. Application of the fractional step method with second order pressure correction (P2) results in the equations

$$\frac{v^{*}-v^{n}}{\Delta t}+\left[\frac{3}{2}H(v^{n})-\frac{1}{2}H(v^{n-1})\right]=-Gp^{n}+\frac{1}{2\mathrm{Re}}L(v^{*}+v^{n}),\tag{3}$$

$$L\pi=\frac{1}{\Delta t}Dv^{*},\tag{4}$$

where π is the pressure correction. The intermediate velocity field, v^*, is not necessarily divergence free. A correction to v^* is then applied using the gradient of π to give a divergence free velocity, v^{n+1}, while π provides an update for the pressure, that is,

$$v^{n+1}=v^{*}-\Delta tG\pi\quad\text{and}\quad p^{n+1}=p^{*}+\pi\,.\tag{5}$$

In the solution of the PPE, the coefficient matrix, which is the discrete form of the Laplace operator, is constant. Computing the solution of the PPE may account for up to 95% of the runtime in high accuracy simulations [1]. Thus, accelerating the convergence of the PPE will significantly improve the overall efficiency of the Navier–Stokes solver. Preconditioning, implemented in the solution of PPE, can then greatly enhance the performance of the solver. Saad [11] comments that a good preconditioning method has a greater influence on the rate of convergence of the PPE than the selection of the iterative solver.

Many preconditioning techniques have been developed and in general divided into two types. The first type is a preconditioning matrix M that approximates the original coefficient matrix A, or $M \approx A$. Examples of this class are Jacobi, Gauss–Seidel, Symmetric Successive Over Relaxation (SSOR) and various forms of Incomplete Lower-Upper (ILU) preconditioners. While simple methods like Jacobi, Gauss–Seidel and SSOR are often not effective, ILU-variant preconditioners greatly improve the rate of convergence [11]. However, the implementation of ILU in the solver is highly sequential. The second type of preconditioner is the sparse approximate inverse, in which the preconditioner approximates the inverse of the original matrix, $M \approx A^{-1}$. The implementation of sparse approximate inverses in the solver requires only matrix-vector products in each iteration, and thus is fairly straightforward to parallelise.

1.2 Sparse approximate inverse preconditioners

A number of studies developed sparse approximate preconditioners [5, 6, 7, 8, 9]. The most popular are the Sparse Approximate Inverse (SPAI) method, which is based on the minimisation of the Frobenius norm, and the factorised sparse Approximate Inverse (AINV) method, which is based on the biconjugation algorithm.

The SPAI preconditioner developed by Grote and Huckle [8] is based on the minimisation of $\|AM - I\|$, for right preconditioning, or $\|MA - I\|$, for left preconditioning. Because the computation of of the 1-norm or the 2-norm is quite expensive, the minimum $\|AM - I\|$ is computed in the Frobenius norm, that is,

$$\|AM - I\|_F^2 = \sum_{k=1}^{n_k} \|(AM - I)e_k\|_2^2 \,, \tag{6}$$

where n_k is the number of columns. Each column of M, denoted m_k, is constructed independently in separate least squares problems,

$$\min_{m_k} \|Am_k - e_k\|_2, \qquad k = 1, \ldots, n_k, \tag{7}$$

with e_k the unit basis vector. If M is sparse, only a small number of least square problems need to be solved and, thus, in this case the construction of M will be fast. The construction initially starts with a certain sparsity pattern and subsequently adds the nonzero entries in M until it reaches the maximum number of nonzeros or until the 2-norm residual is below a specific value [8].

Benzi and Tůma [4, 5] developed the AINV method, which constructs factorised sparse approximate inverse preconditioners based on the biconjugation algorithm. The biconjugation algorithm computes two A-biconjugate sets of vectors, $\{\mathbf{z}_i\}_{i=1}^{n_k}$ and $\{\mathbf{w}_i\}_{i=1}^{n_k}$, such that $\mathbf{w}_i^T A\mathbf{z}_j = 0$ if and only if $i \neq j$. Defining Z and W to be matrices whose columns are formed from the A-biconjugate vectors,

$$Z = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_{n_k}] \qquad \text{and} \qquad W = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_{n_k}], \tag{8}$$

with $p_i = \mathbf{w}_i^T A\mathbf{z}_i \neq 0$, then

$$W^T AZ = D = \operatorname{diag}(p_1, p_2, \ldots, p_{n_k}). \tag{9}$$

Thus, the inverse of A is

$$A^{-1} = ZD^{-1}W^T = \sum_{i=1}^{n_k} \frac{\mathbf{z}_i \mathbf{w}_i^T}{p_i} \,. \tag{10}$$

To obtain a sparse approximation of A^{-1} a dropping strategy is used, removing small entries below a certain drop tolerance. This means that any value in M that is below the tolerance value is omitted. A smaller drop tolerance results in a denser M. Benzi and Tůma [5] detail the algorithm.

For a given number of nonzero entries, factorised methods (e.g., AINV) approximate the inverse better than nonfactorised methods (e.g., SPAI) [4, 7]. Benzi and Tůma [3] found that the most robust preconditioners are the ILU-variant methods, which help to solve systems that would not converge without preconditioning. The simplest variant of ILU, denoted ILU(0), is obtained by restricting the number of nonzeros of the lower and upper matrices to be equal to that of the original matrix A. Benzi and Tůma [3] further concluded that AINV and SPAI gave approximately the same rate of convergence as ILU(0), in terms of the number of iterations needed to converge.

In the Navier–Stokes application, the important characteristic of the pressure correction equation is that the coefficient matrix of the system remains constant while the grid spacing is fixed. This suggests a benefit in using approximate inverses as preconditioners. Sparse approximate inverses need to be constructed and stored only once, and are recalled in each subsequent time step. An advantage of this method is that its implementation requires only matrix-vector products and hence is relatively easy to parallelise. Thus, this study focuses on testing the effectiveness of various preconditioners to enhance the performance of the pressure correction fractional step method used in the solution of the Navier–Stokes equations.

2 Methodology

Sparse approximate inverse preconditioners are implemented for the pressure correction equation in the P2 Navier–Stokes solver. Armfield and Street [1] gave details of the P2 pressure correction scheme. The test case is three dimensional turbulent flow in a channel with dimensions of $6.3 \times 2.0 \times 3.15$.

The flow is driven by a pressure gradient in the x-direction with $\partial P/\partial x = 1.0$ and Re $= 180$ (based on the friction velocity). The computational domain is generated with a structured, non-staggered mesh. The standard mesh spacing was detailed by Armfield et al. [2], and has a total of 53776 cells. Periodic boundary conditions are applied in the x and z directions, and no-slip walls at both y boundaries. The computation is also tested on a finer mesh, with cell widths in the x, y and z directions approximately half of the standard widths, giving a total of 293706 cells.

The equations are discretised using second order central differencing in space. Time discretisations for the advective terms use Adams–Basforth differencing, while Crank–Nicolson differencing is used for the diffusive terms. The iterative solvers used are Jacobi for the momentum equations, and GMRES for the PPE equation. To minimise memory, GMRES is restarted after five iterations. The preconditioners tested are SPAI, AINV and ILU(0). The preconditioners are all applied as right preconditioners, since they are found to be about 8% faster than left preconditioners in this case. The time steps chosen are 10^{-3} for the standard mesh and 5×10^{-4} for the finer mesh. The iterative solvers are considered converged when the L_2 norm of the residuals becomes less than 10^{-4}.

3 Results and discussion

Results are obtained on a single core of an Intel E8400 processor with 4 GB 1.6 GHertz DDR2 RAM. Figure 1 shows the time averaged velocity profiles of solutions with various preconditioners compared to the DNS spectral method of Moser et al. [10]. The preconditioner choice has no effect on the accuracy of the solution, since the convergence of the solver is held by the constant residual limit. This suggests that the use of preconditioners is mainly to accelerate the convergence, without affecting the accuracy of the solver, as expected. For the standard mesh, the solution of the PPE with no preconditioning requires more than 3000 iterations to converge at each time step. Significant improvement is achieved with preconditioning, reducing the number of iterations to less than approximately 100 at each time step. With preconditioning, the average PPE computing time is about 67% of the total solution time.

Figure 2 shows the average PPE computation time per time step using the SPAI preconditioned solver for various values of the tolerance parameter for the residual of $\|Am_k - e_k\|_2$ for all $k = 1, \ldots, n_k$. Overall, the PPE solver time is faster with lower SPAI residual. For the standard mesh, the computing time reaches a minimum at SPAI residual of around 0.15 and then increases at even lower SPAI residual. This may be explained by Figure 3, which shows the sparsity patterns of the matrix M for different SPAI residuals. Each picture represents the n_k by n_k matrix, where n_k also represents the number of total cells. The ratio of the number of nonzero entries in M to the number of nonzero entries in A is denoted by α. The pictures show that all the resulting matrices M are diagonally dominant, but the amount of fill for low SPAI residual is very high. This suggests that with low SPAI residual, the matrix M becomes very dense as a result of a better approximation of the dense matrix A^{-1}, and the matrix-vector multiplications in the solver require more floating point operations. Therefore, it is important to find the parameter which achieves the optimum performance for the preconditioning method. For this case, the optimum SPAI residual is 0.15. For the finer mesh, the optimum SPAI residual moves to 0.1. This suggests that the optimum SPAI residual changes with grid size, to lower values with finer meshes.

The AINV results are similar to those of the SPAI. Figure 4 shows the average PPE solver time per time step for various AINV drop tolerances. In this study, absolute drop tolerance is used, meaning that any value in M that is below the tolerance value is omitted. A smaller drop tolerance results in a denser M. As in SPAI, the solver time reaches a minimum at a certain drop tolerance value. Again, a very dense M does not always represent an effective accelerator. For the standard mesh, the optimum AINV drop tolerance is 0.02. For the finer mesh, the optimum AINV drop tolerance shifts to 0.01, suggesting that the optimum value of the dropping parameter depends on the grid size, similar to that for SPAI. Figure 5 depicts examples of the AINV matrices for three

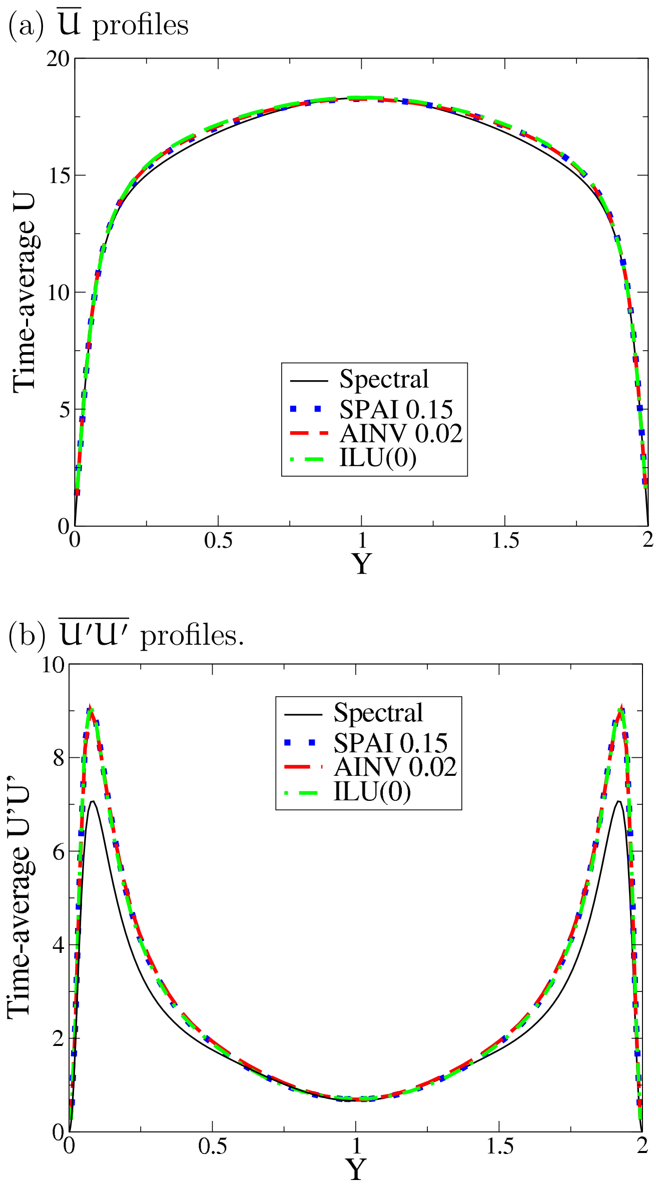


FIGURE 1: Time averaged velocity profiles compared to DNS spectral method [10].

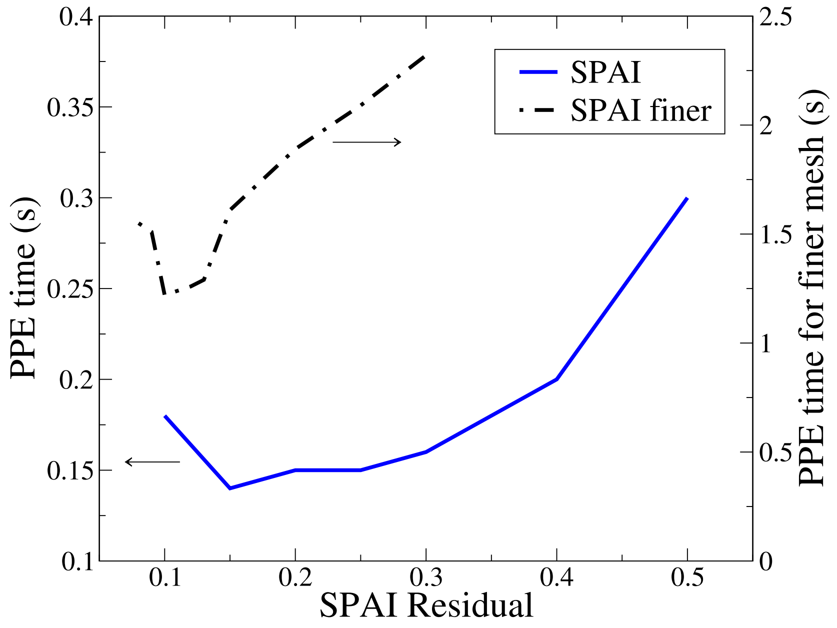
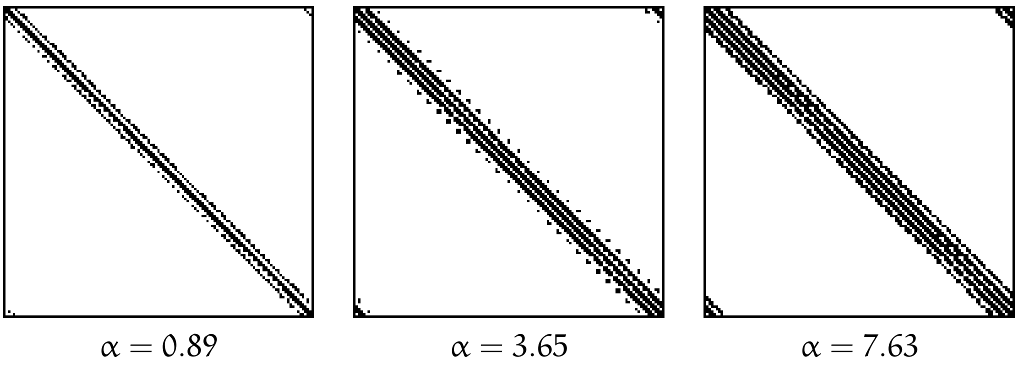


Figure 2: Averaged SPAI preconditioned solver time per time step.



(a) SPAI residual = 0.3 (b) SPAI residual = 0.15 (c) SPAI residual = 0.1

Figure 3: Preconditioner matrices for different SPAI residual parameters.

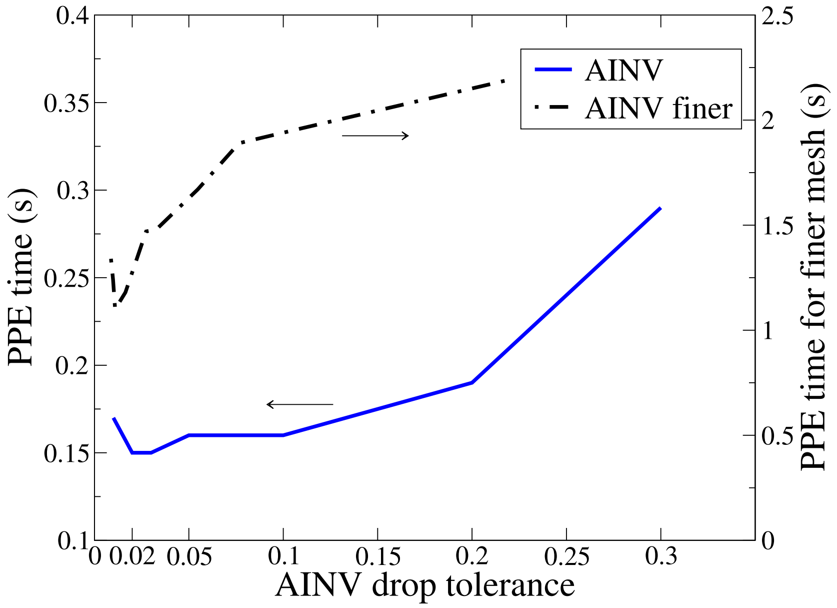
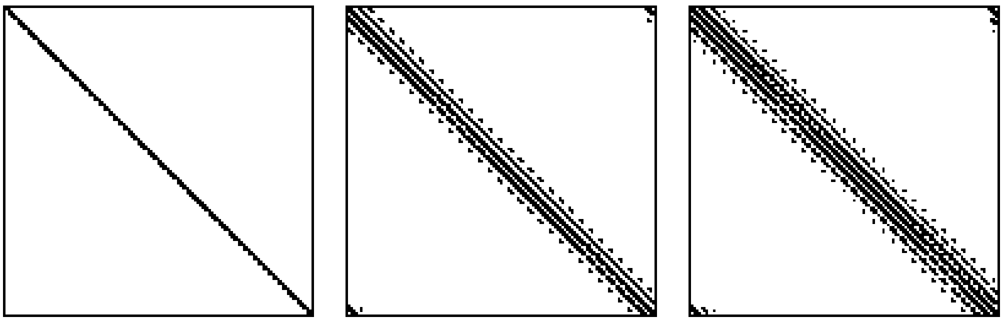


Figure 4: Averaged AINV preconditioned solver time per time step.

drop tolerances on the standard mesh. Interestingly, although the resulting patterns are highly diagonal, they are slighty different from those obtained from SPAI (shown in Figure 3) with about the same amount of fill.

Figure 6 shows the averaged preconditioned solver time per time step against α, the ratio of fill in the matrix M to the matrix A. Comparing the effectiveness of sparse approximate inverse preconditioners, the overall performance of AINV is slightly better than that of SPAI for the finer mesh computation. In contrast, for the standard mesh, SPAI performs better than AINV for very sparse patterns. The optimum α, the optimum ratio of the number of nonzero entries in M to the number of nonzero entries in A, which corresponds to the optimum preconditioner parameter, doubles as the cell size is reduced by approximately half in the x, y and z directions. The performance of sparse approximate inverse preconditioners are also compared to ILU(0), which is


$\alpha = 0.92$
(a) AINV drop $= 0.2$

$\alpha = 4.26$
(b) AINV drop $= 0.02$

$\alpha = 7.40$
(c) AINV drop $= 0.01$

Figure 5: AINV matrices for different absolute drop tolerance parameters.

known to be an effective preconditioner [11]. The ILU(0) preconditioned solver time is much lower than those of the sparse approximate inverses. This suggests that ILU(0) is the best preconditioner on a single processor, having the smallest computing time per time step, and less fill than that of the optimum for the sparse approximate inverse methods.

4 Concluding remarks

This study shows that for the pressure Poisson equation used in the pressure correction Navier–Stokes solver (P2), the sparse approximate inverse preconditioners provide comparable sequential performance to ILU(0). Sparse approximate inverses require only matrix-vector products in their implementation on the solvers, and thus will be of benefit in parallel computing. The sparse approximate inverse methods considered have about the same performance, with AINV slightly outperforming SPAI on the finer mesh computation, for the same number of fills in the case tested. Future work will be undertaken on the parallel implementation of the sparse approximate inverse preconditioners for the Navier–Stokes solvers.

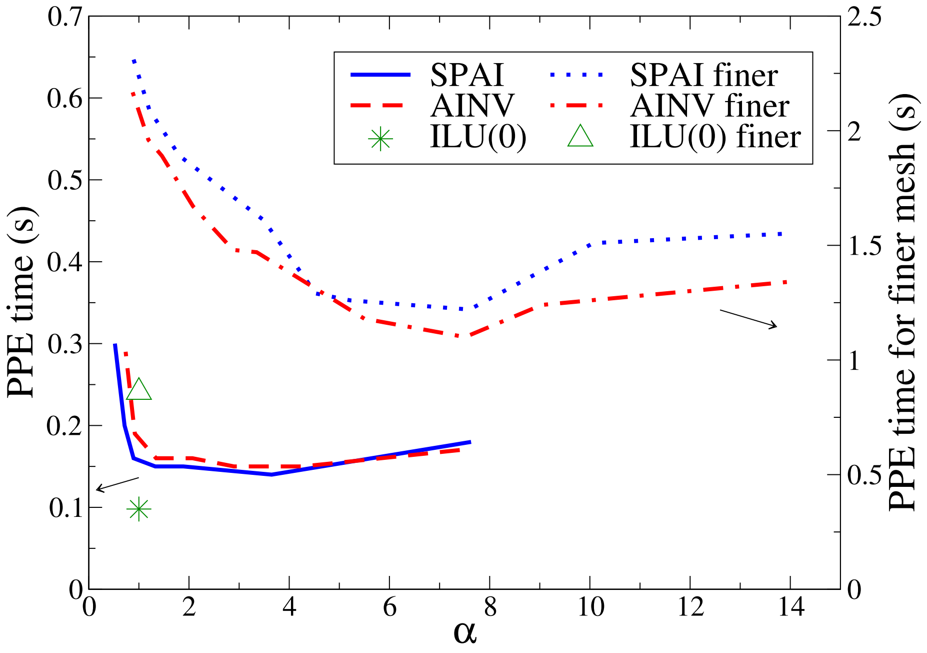


Figure 6: Averaged preconditioned solver time per time step compared to ILU(0).

Acknowledgements The first author acknowledges the Faculty for The Future programme of the Schlumberger Foundation for funding this study.

References

[1] S. Armfield and R. Street. An analysis and comparison of the time accuracy of fractional-step methods for the Navier–Stokes equations on staggered grid. *Int. J. Numer. Methods Fluids*, 38:255–282, 2002. doi:10.1002/fld.217 C582, C583, C584, C586

[2] S. W. Armfield, N. Williamson, M. P. Kirkpatrick, and R. Street. A divergence free fractional-step method for the Navier–Stokes equations on non-staggered grids. *ANZIAM J.*, 51:C654–C667, 2010. `http://anziamj.austms.org.au/ojs/index.php/ANZIAMJ/article/view/2627` C587

[3] M. Benzi and M. Tůma. A comparative study of sparse approximate inverse preconditioners. *Appl. Numer. Math*, 30:305–340, 1998. `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.54.1340` C582, C586

[4] M. Benzi and M. Tůma. Numerical experiments with two approximate inverse preconditioners. *BIT*, 38:234–241, 1998. `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.51.9389` C585, C586

[5] M. Benzi and M. Tůma. A sparse approximate inverse preconditioner for nonsymmetric linear systems. *SIAM J. Sci. Comput.*, 19(3):968–994, 1998. doi:10.1137/S1064827595294691 C582, C584, C585, C586

[6] R. Bru, J. Cerdán, J. Marín, and J. Mas. Preconditioning sparse nonsymmetric linear systems with the Sherman–Morrison formula. *SIAM J. Sci. Comput.*, 25(2):701–715, 2003. doi:10.1137/S1064827502407524 C582, C584

[7] E. Chow and Y. Saad. Approximate inverse preconditioners via sparse-sparse iterations. *SIAM J. Sci. Comput.*, 19(3):995–1023, 1998. doi:10.1137/S1064827594270415 C582, C584, C586

[8] M. J. Grote and T. Huckle. Parallel preconditioning with sparse approximate inverses. *SIAM J. Sci. Comput.*, 18(3):838–853, May 1997. doi:10.1137/S1064827594276552 C582, C584, C585

[9] L. Y. Kolotilina and A. Y. Yeremin. Factorized sparse approximate inverse preconditionings I. Theory. *SIAM J. Matrix Anal. Appl.*, 14(1):45–58, 1993. doi:10.1137/0614004 C582, C584

[10] R. D. Moser, J. Kim, and N. N. Mansour. Direct numerical simulation of turbulent channel flow up to $\mathrm{Re}_\tau = 590$. *Phys. Fluids*, 11(4):943–945, 1999. doi:10.1063/1.869966 C587, C589

[11] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, Philadelphia, PA, USA, 2nd edition, 2003. C582, C584, C592

Author addresses

1. **V. S. Djanali**, School of Aerospace, Mechanical and Mechatronic Engineering, The University of Sydney, Sydney, Australia.
   mailto:vivien_s@me.its.ac.id
2. **S. W. Armfield**, School of Aerospace, Mechanical and Mechatronic Engineering, The University of Sydney, Sydney, Australia.
3. **M. P. Kirkpatrick**, School of Aerospace, Mechanical and Mechatronic Engineering, The University of Sydney, Sydney, Australia.