CHEMICAL ENGINEERING TRANSACTIONS
VOL. 70, 2018
A publication of AIDIC
The Italian Association of Chemical Engineering
Online at www.aidic.it/cet
Guest Editors: Timothy G. Walmsley, Petar S. Varbanov, Rongxin Su, Jiří J. Klemeš

ISBN 978-88-95608-67-9; **ISSN** 2283-9216
DOI: 10.3303/CET1870242

On Improving Computational Efficiency of Simplified Fluid Flow Models

Vojtěch Turek

Sustainable Process Integration Laboratory – SPIL, NETME Centre, Faculty of Mechanical Engineering, Brno University of Technology – VUT Brno, Technická 2, 616 69 Brno, Czech Republic
turek@fme.vutbr.cz

Single-core computational efficiency of several iterative methods for numerical solution of nonsymmetric linear systems originating from simplified fluid flow models is evaluated together with different preconditioning techniques in terms of solution behaviour and the overall rate of convergence. A previously developed simplified 3D CFD model, which involves the solution of linear systems of orders usually from units to tens of thousands, is used to generate test cases. The respective Java software application employs the Parallel Colt linear algebra library. This is to ensure that the corresponding sparse matrix computations needed in different solution and preconditioning methods are carried out in as efficient a manner as possible to minimize the influence of the computer implementation itself. Measures (warm-up phase etc.) are taken to eliminate the effects of JVM-specific behaviour such as JIT compilation.

1. Introduction

Improving performance is crucial when it comes to computational methods used to evaluate objective functions in direct optimization algorithms. Because the simplified 3D CFD model proposed by Turek et al. (2016) is intended for just this purpose, i.e., fast evaluation of many possible flow system geometries in the early stage the apparatus design process, finding ways to further shorten computational time and improve convergence should be considered. This paper aims to do this by applying preconditioning techniques to the numerical methods used to obtain solutions of the underlying systems of linearized equations.
Even though the simplified 3D CFD model has a lot in common with the classical CFD models implemented in various simulation tools, it utilises a largely simplified and rather coarse mesh while turbulence is also modelled in a simplified manner. Consequently, the numerical behaviour of the model is more erratic and significantly more prone to divergence. The respective preconditioning techniques must therefore be selected carefully with respect to the specifics of the model. This, however, is not possible by just relying on literature discussing the classical CFD approach such as the monographs by Saad (2003) or Bruaset (1995) focusing on iterative methods or preconditioning techniques, respectively. Similarly, numerical performance-oriented investigations (e.g. Norris, 2000) or various papers presenting the respective preconditioning performance comparisons (e.g. Ma, 2000) are of limited use here because they were carried out with the classical CFD models. In other words, the only way to properly choose preconditioning techniques for the numerical solution methods involved in the simplified CFD model is to perform the respective benchmarks.
Additionally, only single-core performance is considered in this paper. The reason for such a limitation is that, given the simplified meshes containing relatively small amounts of cells, the computational overhead resulting from splitting a task over multiple cores would be significant. It is therefore much more economical in terms of CPU time to limit each computation to a single core and run these computations asynchronously on multiple cores within a suitable optimisation algorithm.

2. Benchmarking procedure

The paper by Turek et al. (2016), which discussed in detail the simplified 3D CFD model, also presented a Java software where this model had been implemented. Because originally the software did not include any

Please cite this article as: Turek V., 2018, On improving computational efficiency of simplified fluid flow models , Chemical Engineering Transactions, 70, 1447-1452 DOI:10.3303/CET1870242

preconditioning or benchmarking capabilities, these had been added. In keeping with (Goetz, 2005), however, the following measures were taken so that the results were not affected by Java-specific behaviour:

- 30 warm-up runs (i.e., full computations from initialization to converged state including all the related diagnostic printouts, data saving, etc.) were always carried out. This was to make sure that all Java initializations and compilations have been finished before the timing phase.
- The code was run with “XX:+PrintCompilation” and “verbose:gc” JVM options to check whether enough warm-up runs were carried out (see the item above). Similarly, “Xbatch” was used to serialize the compiler with the application.
- Even though the benchmarks were run with no other applications being opened at the same time, 50 test runs were carried out to eliminate the effect of various tasks which the OS performs in the background. Means and standard deviations were then calculated from the obtained data (no outliers were encountered in any of the tests).

For consistency and efficiency’s sake, all the involved numerical routines and sparse matrix computations utilised the Parallel Colt linear algebra library (Wendykier and Nagy, 2010). Moreover, the benchmarks were run on two disparate machines to find if this made any difference in terms of single core computational efficiency. These were as follows: (a) server: Intel Xeon E5-2698 v4 with 128 GB RAM, (b) laptop: Intel Core i5-6300U with 16 GB RAM.

2.1 Evaluated solution and preconditioning methods

The following commonly used iterative methods for numerical solution of linear systems were tested: conjugate gradient (CG) (Hestenes and Stiefel, 1952), conjugate gradient on the normal residuals (CGNR) (Saad, 2003), and bi-conjugate gradient stabilized with minimization of residuals over L-dimensional subspaces, L = 1, …, 8, (BiCGstab(L)) (Sleijpen and Fokkema, 1993). Other methods with smoother convergence such as generalized minimal residual (GMRES) (Saad and Schultz, 1986), quasi-minimal residual (QMR) (Freund and Nachtigal, 1991), or iterative refinement (IR) (Moler, 1967) were not considered because their rates of convergence generally are much slower than those of CG-based methods.

The solution methods listed above were used either with no preconditioner, or with one of these five preconditioner types when applicable: diagonal (i.e., Jacobi) (van der Vorst, 2003), incomplete LU factorization (ILU) (Meijerink and van der Vorst, 1977), dual-threshold incomplete LU factorization (ILUT) (Saad, 1994), symmetric successive overrelaxation (SSOR) (Young, 1977), and incomplete Cholesky factorization (ICC) (Manteuffel, 1980). Please note that in this study only the default relaxation parameters of 1.0 for the SSOR forward and backward sweeps were used, i.e., effectively the method reverted to the Gauss-Seidel form.

As for newer solution methods and preconditioning techniques, these generally are various multigrid implementations using several increasingly coarsened grids (e.g. Ruggiu et al., 2018) or are explicitly tailored to multi-core usage (e.g. Esmaily et al., 2018). The former approach is not feasible in case of the already much simplified meshes, while the latter one is out of the scope of this paper for the reasons discussed earlier.

2.2 Test cases

The flow system used for benchmarking purposes was a Z-arranged one with identical cuboid headers (width × height × length = 40 × 40 × 380 mm) and a tube bundle containing two inline rows of 20 tubes each. The respective tube length was 500 mm while the inner diameter was 10 mm. Two meshes of different finenesses were considered. The coarser one consisted of roughly 6,000 cells while the finer one contained roughly 25,000 cells. Boundary conditions were as follows: mass flow inlet with the rate set to 0.5 kg/s of water at 300 K and pressure outlet with the boundary pressure of 101,325 Pa. Heat flux of 25 kW was defined solely on the tube walls if the energy equation was enabled, otherwise all the walls were adiabatic. Considering other CFD model parameters, steady-state simulations were carried out using the SIMPLEC pressure-velocity coupling (van Doormaal and Raithby, 1984), Power Law discretization scheme (Patankar, 1981), and standard scaled residual limits (10^{-3} for continuity and momentum, 10^{-6} for energy).

Because CG and ICC require symmetric, positive-definite matrices, these two methods were only used for the pressure correction equation. Although SSOR together with its forward-only or backwards-only versions were, too, originally meant for such matrices, they have been shown by Birken et al. (2013) and Woźnicki (2001), respectively, to work quite well even when the respective matrix was not symmetric. What is more, construction of the SSOR preconditioning matrix cannot lead to breakdown (contrary to incomplete factorization methods) and therefore this technique was evaluated with all types of equations.

The set of test cases (i.e., combinations of the solution and preconditioning methods) was built in successive steps with respect to the obtained results. First, all feasible solution methods were evaluated on the server using the coarser mesh without preconditioning being applied to find the baseline performances. These combinations were subsequently also tested with only the pressure solver, or only the momentum solvers, being preconditioned. Promising combinations of the solution and preconditioning methods that were thus

obtained were evaluated in the third step. The fourth step in the benchmarking process constituted testing of the best combinations found so far, but this time with the energy equation being enabled and solved both without any preconditioning (again to find baseline performances) and with various preconditioners in place. This totalled to 934 cases.
Any combination which failed to converge within 1,000 iterations or 10 minutes was deemed unsuitable. The remaining 214 combinations were tested using the finer mesh, for which the suitability thresholds were 2,000 iterations or 20 minutes. The best 10 combinations in terms of (a) flow only and (b) flow and energy transport were in the end also evaluated using each mesh on the laptop along with the corresponding baseline combinations (i.e., without any preconditioning being applied). This constituted 60 additional cases. In summary, 1,148 cases were evaluated on the server and 60 cases on the laptop.

3. Results

The results are split into four categories covering the two meshes (coarser and finer) and two simulation scenarios (flow only and flow & energy transport). Overall rate of convergence is discussed first and then a few comments on smoothness of convergence and solution behaviour in general are provided.

3.1 Overall rate of convergence

Figures 1 through 4 show the ten best-performing combinations of the solution and preconditioning methods for each mesh and simulation scenario. Each of the graphs contains mean computational time and the corresponding standard deviation obtained using both the server and the laptop. Average improvements over the respective baseline combinations (those with no preconditioning being applied) are also provided. These values are the averages of the speedups obtained using the two machines and, if t_A denoted the actual computational time and t_B the baseline time, are computed as $t_B / t_A - 1$.
As can be seen from Figure 1, which corresponds to flow only and the coarser mesh, the best approach was to solve the pressure correction equation with CG preconditioned using ILU and the momentum equations with BiCGstab(3) using the same preconditioner. A converged solution was reached in 3.0 s with this setup. Compared to the respective average baseline time of 15.7 s, this yields the improvement of 4.0. It is also apparent that in almost all the listed cases the laptop performance was slightly better. The reason for this might lie in the fact that the laptop CPU core ran during benchmarks at much higher clock rate than the server CPU core (2.93 GHz vs. 2.20 GHz).

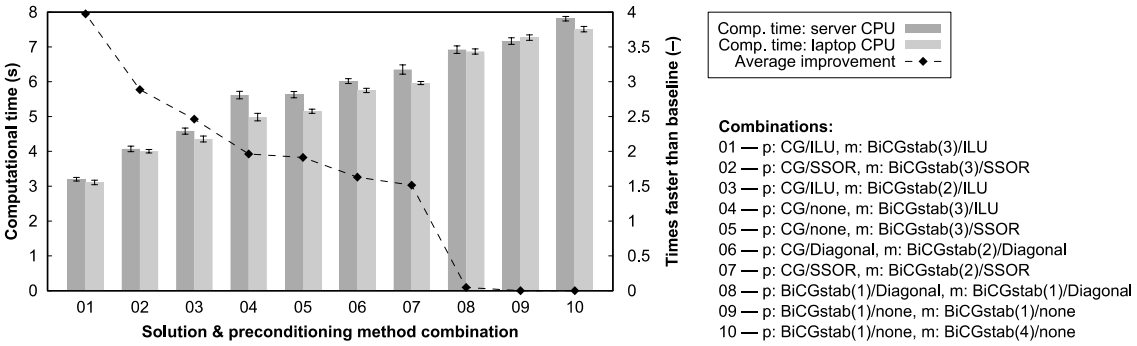

Figure 1: Ten best-performing combinations of solution and preconditioning methods in case of simulation of only the flow using the coarser mesh (roughly 6,000 cells)

The results were very similar when the finer mesh was used (see Figure 2). Again, the best combination was CG/ILU for the pressure correction equation and BiCGstab(3)/ILU for the momentum equations, with the respective computational time being 55.7 s. Given this combination's baseline time of 308.6 s, the average speedup was 4.5. Furthermore, Figure 2 shows that even in this case the laptop CPU slightly overperformed the server one.
When the energy equation was enabled while dealing with the coarser mesh (see Figure 3), in all the ten best cases CG/ILU was used to solve the pressure correction equation; however, BiCGstab(2)/ILU overperformed BiCGstab(3)/ILU. As for the energy solver, the best combination used BiCGstab(2)/SSOR. The corresponding speedup was 3.6 (on average, computational time decreased from 36.7 s to 8.0 s). The notable increase in computational time over the flow-only scenarios was caused primarily by much more iterations being needed for the energy scaled residual to reach the standard limit of 10^{-6}. Additionally, here the server CPU performed slightly better than the laptop one. This might be because larger amounts of varied data and

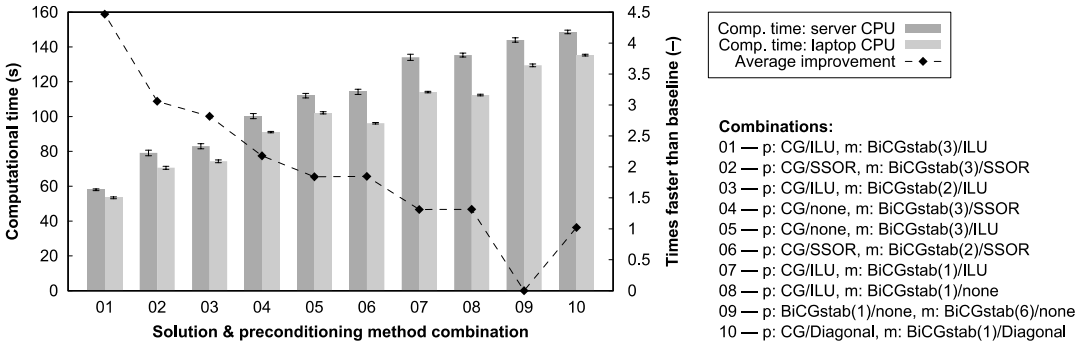

Figure 2: Ten best-performing combinations of solution and preconditioning methods in case of simulation of only the flow using the finer mesh (roughly 25,000 cells)

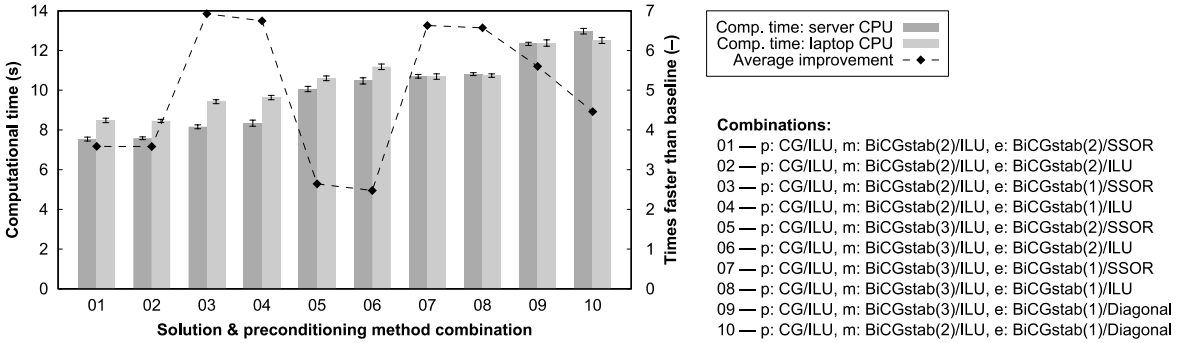

Figure 3: Ten best-performing combinations of solution and preconditioning methods in case of simulation of flow and energy transfer using the coarser mesh (roughly 6,000 cells)

instructions were being handled and the fact that the server CPU had an order of magnitude more L1, L2, and L3 caches.

Results related to the finer mesh, which are shown in Figure 4, are quite similar to those shown in Figure 3 in terms of the pressure correction and momentum solvers. Still, there are significant differences in the utilized energy solvers – one can see many combinations with various unpreconditioned versions of BiCGstab(*L*) and, in case of the last combination, CGNR with diagonal preconditioning in place. The speedups, therefore, vary notably, i.e., combinations with all solvers being preconditioned perform roughly 10 to 17 times faster than their baseline variants, while with unpreconditioned energy solver the speedups are only about 3. In case of the best-performing combination, the average computational time dropped from 722.7 s to 69.2 s. It should also be mentioned that a much larger percentage of cases resulted without user intervention in divergence when the energy equation was enabled. Given the purpose of the original simulation tool and its having to be as autonomous as possible, the respective combinations were discarded as unsuitable.

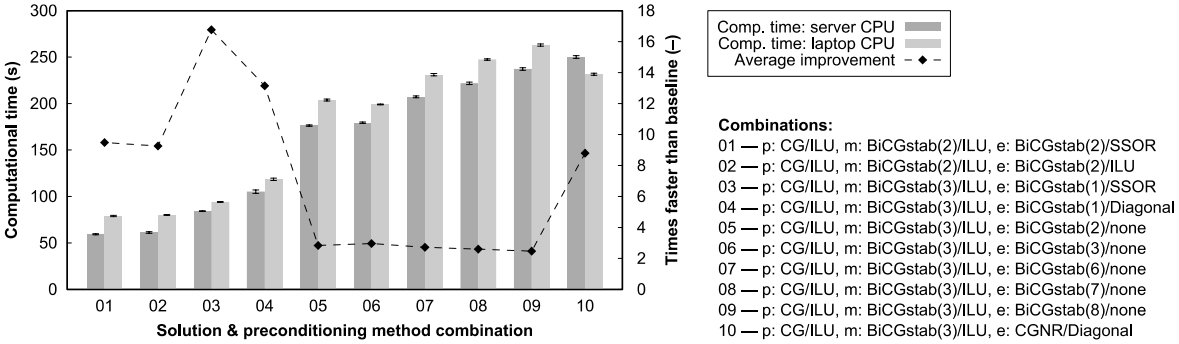

Figure 4: Ten best-performing combinations of solution and preconditioning methods in case of simulation of flow and energy transfer using the finer mesh (roughly 25,000 cells)

As for the ICC and ILUT preconditioning methods, these proved to be wholly unsuitable due to their slow convergence. What is more, these two preconditioners were markedly more likely to lead to divergence than ILU or SSOR.

3.2 Smoothness of convergence & general solution behaviour

In general, if a combination of preconditioned solvers reached a converged solution, then the convergence was faster, yet not necessarily smoother, than in case of the corresponding unpreconditioned combination (see Figures 5 and 6). However, it was also true that divergence was much more common among preconditioned solvers than when unpreconditioned solvers were employed. ILU provided marginally smoother convergence than SSOR and significantly smoother convergence than diagonal preconditioning. Considering BiCGstab(*L*), smoothness of convergence increased with increasing value of *L* (see Figure 5).

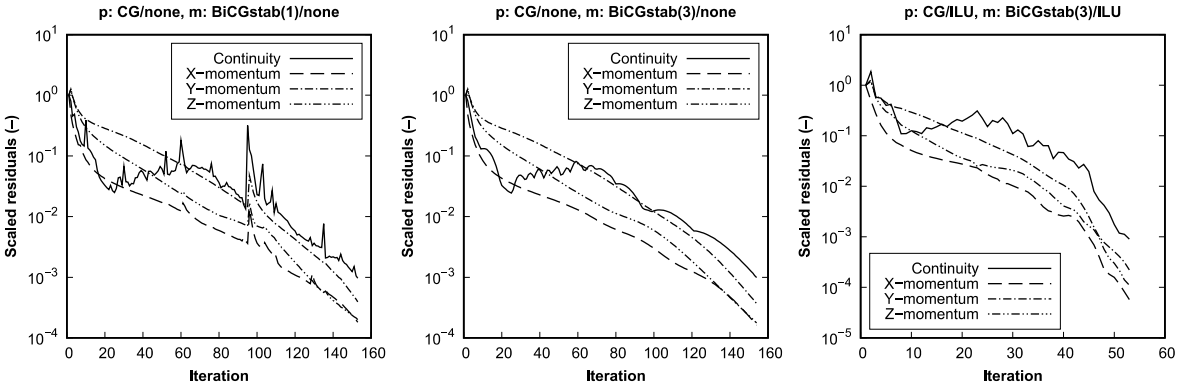

Figure 5: Comparison of scaled residual plots corresponding to three different combinations of solution and preconditioning methods in case of simulation of only the flow using the coarser mesh (roughly 6,000 cells)

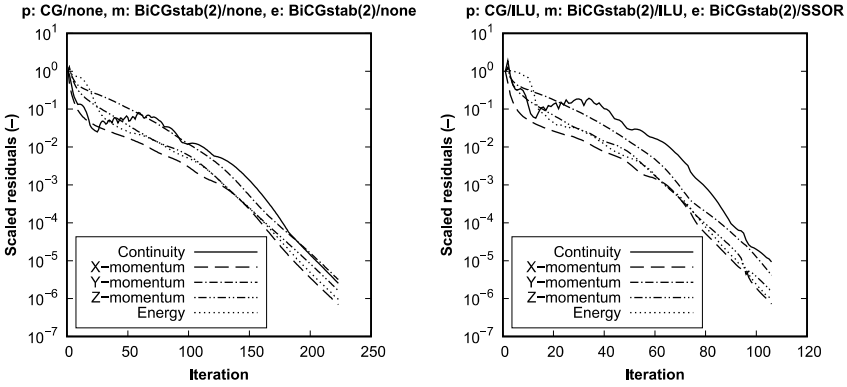

Figure 6: Comparison of scaled residual plots corresponding to a combination of unpreconditioned solution methods and the same methods preconditioned using ILU (pressure correction, momentum) and SSOR (energy) in case of the coarser mesh (roughly 6,000 cells)

4. Conclusions

The data presented in the previous paragraphs clearly suggest that in terms of the discussed simplified 3D CFD simulations best results are provided by different combinations of solution and preconditioning methods depending on whether only the flow or flow and energy transport are modelled. As for flow only, the best performance was reached using CG as the pressure correction solver together with BiCGstab(3) for momentum, both with ILU preconditioning being applied. The speedups compared to the respective baseline combination were ca. 4–5 depending on mesh fineness. If energy transport was included into the simulations, then the best results were obtained using CG/ILU for pressure correction, BiCGstab(2)/ILU for momentum, and BiCGstab(2)/SSOR or BiCGstab(2)/ILU for energy. The speedups corresponding to this combination were ca. 4 for the coarser mesh and ca. 10 in case of the finer mesh. Still, further tests should be carried out with respect to the selection of relaxation parameter $\omega \in (0, 2)$ in SSOR which, if chosen properly, can drastically improve performance.

Considering the two different CPUs, it is obvious that they performed almost identically in terms of single-core computations involving the small simplified meshes.

Acknowledgment

This research has been supported by EU project Sustainable Process Integration Laboratory – SPIL, funded as project No. CZ.02.1.01/0.0/0.0/15_003/0000456, by Czech Republic Operational Programme Research, Development, and Education, Priority 1: Strengthening capacity for quality research.

References

Birken P., Gassner G., Haas M., Munz C.-D., 2013, Preconditioning for modal discontinuous Galerkin methods for unsteady 3D Navier–Stokes equations, Journal of Computational Physics, 240, 20–35.

Bruaset A.M., 1995, A Survey of Preconditioned Iterative Methods, CRC Press, Boca Raton, FL, USA

Esmaily M., Jofre L., Mani A., Iaccarino G., 2018, A scalable geometric multigrid solver for nonsymmetric elliptic systems with application to variable-density flows, Journal of Computational Physics, 357, 142–158.

Freund R., Nachtigal N., 1991, QMR: a quasi-minimal residual method for non-Hermitian linear systems, Numerische Mathematik, 60, 315–339.

Goetz B., 2005, Java theory and practice: anatomy of a flawed microbenchmark, IBM Corporation <www.ibm.com/developerworks/java/library/j-jtp02225> accessed 10.01.2018.

Hestenes M.R., Stiefel E., 1952, Methods of conjugate gradients for solving linear systems, Journal of Research of the National Bureau of Standards, 49, 409–436.

Ma S., 2000, Comparisons of the ILU(0), Point-SSOR, and SPAI preconditioners on the CRAY-T3E for nonsymmetric sparse linear systems arising from PDEs on structured grids, The International Journal of High Performance Computing Applications, 14, 39–48.

Manteuffel T.A., 1980, An incomplete factorization technique for positive definite linear systems, Mathematics of Computation, 34, 473–497.

Meijerink J.A., van der Vorst A.H., 1977, An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix, Mathematics of Computation, 31, 148–162.

Moler C.B., 1967, Iterative refinement in floating point, Journal of the ACM, 14, 316–321.

Norris S.E., 2000, A Parallel Navier Stokes Solver for Natural Convection and Free Surface Flow, PhD Thesis, University of Sydney, Australia.

Patankar S.V., 1981, A calculation procedure for two-dimensional elliptic situations, Numerical Heat Transfer, 4, 409–425.

Ruggiu A.A., Weinerfelt P., Nordström J., 2018, A new multigrid formulation for high order finite difference methods on summation-by-parts form, Journal of Computational Physics, 359, 216–238.

Saad Y., 1994, ILUT: a dual threshold incomplete LU factorization, Numerical Linear Algebra with Applications, 1, 387–402.

Saad Y., 2003, Iterative Methods for Sparse Linear Systems, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

Saad Y., Schultz M., 1986, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM Journal on Scientific and Statistical Computing, 7, 856–69.

Sleijpen G.L., Fokkema D.R., 1993, BiCGstab(l) for linear equations involving unsymmetric matrices with complex spectrum, Electronic Transactions on Numerical Analysis, 1, 11–32.

Turek V., Fialová D., Jegla Z., 2016, Efficient flow modelling in equipment containing porous elements, Chemical Engineering Transactions, 52, 487–492.

van der Vorst H.A., 2003, Iterative Krylov Methods for Large Linear Systems, Cambridge University Press, Cambridge, UK.

van Doormaal J.P., Raithby G.D., 1984, Enhancement of the SIMPLE method for predicting incompressible fluid flows, Numerical Heat Transfer, 7, 147–163.

Wendykier P., Nagy J.G., 2010, Parallel Colt: a high-performance Java library for scientific computing and image processing, ACM Transactions on Mathematical Software, 37, 31:1–31:22.

Woźnicki Z.I., 2001, On performance of SOR method for solving nonsymmetric linear systems, Journal of Computational and Applied Mathematics, 137, 145–76.

Young D.M., 1977, On the accelerated SSOR method for solving large linear systems, Advances in Mathematics, 23, 215–271.