On Robust Parallel Preconditioning for Incompressible Flow Problems

Timo Heister, Gert Lube, and Gerd Rapin

Abstract We consider time-dependent flow problems discretized with higher order finite element methods. Applying a fully implicit time discretization or an IMEX scheme leads to a saddle point system. This linear system is solved using a preconditioned Krylov method, which is fully parallelized on a distributed memory parallel computer.
We study a robust block-triangular preconditioner and beside numerical results of the parallel performance we explain and evaluate the main building blocks of the parallel implementation.

1 Introduction

The numerical simulation of time-dependent flow problems is an important task in research and industrial applications. The flow of Newtonian incompressible fluids is described by the system of the Navier-Stokes equations in a bounded domain $\Omega \subset \mathbb{R}^d$, $d = 2, 3$, where one has to find a velocity field $\mathbf{u} : [0, T] \times \Omega \to \mathbb{R}^d$ and a pressure field $p : [0, T] \times \Omega \to \mathbb{R}$ such that

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} - \nu \Delta \mathbf{u} + (\mathbf{u} \cdot \nabla)\,\mathbf{u} + \nabla p = \mathbf{f} \quad &\text{in} \quad (0, T] \times \Omega, \\ \nabla \cdot \mathbf{u} = 0 \quad &\text{in} \quad [0, T] \times \Omega. \end{aligned} \tag{1}$$

Timo Heister
Math. Dep., University of Göttingen, Germany, heister@math.uni-goettingen.de

Gert Lube
Math. Dep., University of Göttingen, Germany

Gerd Rapin
VW, Interior Engineering, Wolfsburg, Germany

Here $\mathbf{f} : (0,T] \times \Omega \to \mathbb{R}^d$ is a given force field, and ν is the kinematic viscosity. For brevity initial and boundary conditions are omitted. One has to cope with some modifications for turbulent flows, namely using $\nabla\cdot(2\nu S(\mathbf{u}))$ with $S(\mathbf{u}) := \frac{1}{2}(\nabla\mathbf{u}+\nabla\mathbf{u}^T)$ instead of $\nu\triangle\mathbf{u}$, variable and non-linear viscosity $\nu := \nu_{\text{const}} + \nu_t(\mathbf{u})$, and additional velocity terms from turbulence models.

In Section 2 this system of equations is discretized in space and time. The high spatial resolution needed for a typical domain $\Omega \subset \mathbb{R}^3$ leads to a number of unknowns in the order of millions of degrees of freedom. We need to calculate the solution at many time-steps, especially for optimization or inverse problems. This results in a demand for a robust and fast algorithm. We define such an algorithm in Section 3. The memory and performance requirements for the solution process can typically be met by a distributed memory cluster.

Let us state the requirements for the solver: *Flexibility*, to allow comparisons between different turbulence models, stabilization schemes, time discretizations, solvers, etc.. *Parallelization*, ranging from multicore workstations to clusters with hundred or more CPUs. *Scalability*, with respect to the number of CPUs and problem size.

Combining these three requirements is a challenge. Research codes are usually *flexible*, but often lack the other requirements. On the other hand commercial codes usually work with lowest order discretization and are not flexible enough. For higher accuracy and flexibility we favor a coupled approach for the saddle point system instead of a splitting scheme.

We use the standard Multiple Instruction, Multiple Data streams (MIMD) parallel architecture model. The basis for the parallel implementation are parallel linear algebra routines running on top of MPI to allow parallel assembling and solving of the linear systems. The data matrices and vectors are split row-wise between the CPUs (Section 4). We conclude the paper with numerical results in Section 5.

2 Discretization

We start by semi-discretizing the continuous equation (1) in time. The solution $(\mathbf{u}, p)$ and the data $\mathbf{f}$ are expressed only at discrete time-steps $0 = t_0 < t_1 < \ldots < t_{max} = T$ of the time interval $[0,T]$, denoted by the superscript n, e.g. $\mathbf{u}^n$. We consider two different discretization schemes, the typical *implicit time discretization* and an implicit-explicit (short *IMEX*) scheme, c.f. [1]. The fully *implicit time discretization* leads to a sequence of non-linear stationary problems of the form

$$\begin{aligned} -\nu\triangle\mathbf{u}^n + c\mathbf{u}^n + (\mathbf{u}^n\cdot\nabla)\mathbf{u}^n + \nabla p^n &= \hat{\mathbf{f}}(\mathbf{u}^{n-1}, p^{n-1}), \\ \nabla\cdot\mathbf{u}^n &= 0, \end{aligned} \tag{2}$$

where $c \in \mathbb{R}$ is a reaction coefficient related to the inverse of the time-step size $\tau_n := t_{n+1} - t_n$ and $\hat{\mathbf{f}}$ is a modified right-hand side. Many time discretizations fit into this implicit scheme, for instance implicit Euler, BDF(2) or diagonal-implicit Runge-Kutta schemes. The non-linear system (2) is linearized by a fixed-point or Newton-type iteration. Hence, we have to solve a sequence of linear systems with a given divergence-free field $\mathbf{b}$ in the convective term $(\mathbf{b} \cdot \nabla)\mathbf{u}$.

The iteration for the non-linearity in (2) implies high computational costs. Explicit time-stepping is not desirable because of the strong restrictions on the time-step size. A remedy is to treat the non-linear term $(\mathbf{u}^n \cdot \nabla)\mathbf{u}^n$ in an explicit way, while the remainder of the equation is kept implicit. These methods are called *IMEX-schemes*. An elegant option is to combine an explicit Runge-Kutta scheme for the convection and an diagonal-implicit scheme, as used above, for the rest. With this method, the non-linearity disappears.

Thus, in both cases we end up with the solution of stationary Oseen problems:

$$\begin{aligned} -\nu\triangle\mathbf{u} + c\mathbf{u} + (\mathbf{b} \cdot \nabla)\mathbf{u} + \nabla p &= \mathbf{f}, \\ \nabla \cdot \mathbf{u} &= 0, \end{aligned} \tag{3}$$

which are discretized via Galerkin FEM on quadrilateral meshes with continuous, piece-wise (tensor-) polynomials Q_k of order $k > 0$. The inf-sup-stability is ensured using a Taylor-Hood pair Q_{k+1}/Q_k for velocity and pressure. This stable discretization leads to a finite-dimensional, linear saddle point system

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix} \tag{4}$$

with finite element matrices A containing diffusion, reaction, and convection and B containing the pressure-velocity coupling.

3 The Solver

The system (4) is solved using the preconditioned Krylov subspace method FGMRES. This is a variant of the standard GMRES algorithm, see [12, 13]. FGMRES can cope with a changing preconditioner in each iteration. This is required because the preconditioner is not calculated explicitly as a matrix but is given as an implicit operator which uses iterative solvers internally. The usage of FGMRES in the context of flow problems is also described in detail in [9]. System (4) is preconditioned with an operator P^{-1} of block triangular type:

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} P^{-1} \begin{pmatrix} v \\ q \end{pmatrix} = F \quad \text{with} \quad P^{-1} = \begin{pmatrix} \widetilde{A} & B^T \\ 0 & \widetilde{S} \end{pmatrix}^{-1}.$$

Here approximations $\widetilde{A}^{-1} \approx A^{-1}$ and $\widetilde{S}^{-1} \approx S^{-1}$ for the Schur complement $S := -BA^{-1}B^T$ are used. With exact evaluations of A and S the number of outer (F)GMRES steps is at most two, see [4]; this motivates the choice of the preconditioner. The inverse can be calculated by

$$P^{-1} = \begin{pmatrix} \widetilde{A}^{-1} & -\widetilde{A}^{-1}B^T\widetilde{S}^{-1} \\ 0 & \widetilde{S}^{-1} \end{pmatrix} = \begin{pmatrix} \widetilde{A}^{-1} & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} I & B^T \\ 0 & -I \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & -\widetilde{S}^{-1} \end{pmatrix}.$$

Each outer iteration requires the solution of two inner problems: the applications of $\widetilde{A}^{-1}$ and $\widetilde{S}^{-1}$, and there is one matrix-vector product with the matrix B^T.

There are several reasons for choosing a coupled approach. Using a projection method would introduce a CFL-like condition restricting the maximum time-step size. The main advantage of projection type methods (computational speed) can be simulated by only applying the preconditioner with a simple iteration method with a fixed number of steps (e.g. one). The result is comparable to a projection step method. Furthermore, the coupled approach fits better to higher order methods. Finally, this method has the advantage that the approximation quality of $\widetilde{A}^{-1}$ and $\widetilde{S}^{-1}$ is adjustable at will; the outer iteration converges either way.

The A-block forms a vector-valued convection-diffusion-reaction problem, which is a lot larger than the Schur complement. It is non-symmetric due to the convection part and the vector components may be coupled as a result of modifications for turbulent calculations, c.f. Section 1. An important part is the (strong) reaction term, which results in the low condition number of the matrix. Thus a BiCGStab solver with algebraic multi-grid preconditioning through BoomerAMG, [8], provides good results for $\widetilde{A}^{-1}$.

The approximation of the Schur complement $\widetilde{S}^{-1}$ is more difficult, because $S = -BA^{-1}B^T$ is dense and hence cannot be built explicitly as a matrix. Fortunately, the reaction-dominated A can be simplified with the mass matrix M_u:

$$S^{-1} \approx \left[B(cM_u)^{-1}B^T\right]^{-1} = c\left(BM_u^{-1}B^T\right)^{-1}.$$

We approximate $p = \widetilde{S}^{-1}q$ by a pressure Poisson problem:

$$-\frac{1}{c}\triangle p = q \tag{5}$$

and suitable boundary conditions, see [14]. The correct boundary conditions stem from $BM_u^{-1}B^T$, which cannot be applied directly. As an approximation there are Neumann boundary conditions applied in (5) where Dirichlet data is applied to the velocity in (1). Vice versa if Neumann boundary conditions are given in (1), homogeneous Dirichlet boundary conditions are applied in the Schur complement, (5). Periodic boundary conditions for the velocity can be treated with periodic boundary conditions in (5), which provide good results, c.f. Section 5.

The block triangular structure has been used for years, a good general overview is given in [4]. The form of the preconditioner is already described in [10], although we neclect the viscosity term in the Schur complement. A discussion of block preconditioners for flow problems can be found in [6] and [11]. Using FGMRES instead of e.g. GMRES proved to be a huge advantage not discussed there, but is motivated in [9].

4 Implementation Overview

The implementation of the solver described in this paper is built on top of a collection of known libraries. The basis is given by an MPI implementation for the parallel communication and the library PETSc, see [2], which supplies us with data structures and algorithms for scalable parallel calculations: matrices, vectors, and iterative solvers. The finite elements, mesh handling and assembling are performed by deal.II, see [3], which directly interfaces with the linear algebra objects from PETSc.

For the parallel calculations the rows in the system matrix have to be partitioned, such that each row is stored on exactly one CPU. This can be done with the following algorithm: first, create a graph, with cells as vertices and edges between two vertices if the corresponding cells are neighbors. This graph is partitioned into mostly equal-sized sets, such that each CPU “owns” a number of cells. The library METIS minimizes the number of cut edges. This reduces the amount of communication in parallel calculations. With the partition of the cells one can assign the owner for each degree of freedom. If two neighboring cells are owned by different CPUs, degrees of freedom on the shared face have to be assigned to one or the other CPU. By controlling this allocation one tries to balance the number of local rows per CPU. This improves the scalabilty of the solution process. The authors improved the way deal.II assigns these degrees of freedom, which decreases the imbalance of the number of degrees of freedom by up to 50%. This is done by making a (deterministic) pseudo-random choice.

The main loop is structured as follows: the outermost loop is the time stepping. For each time step the inner loop is repeated for each stage of the time discretization. For a fully implicit time discretization a fixed-point iteration surrounds the inner part. Finally the inner part consists of assembling and then solving the linear system with FGMRES. In each iteration the preconditioner is applied once. Finally, the preconditioner consists of the preconditioned inner solvers for A and S.

5 Numerical Results

We present the simulation of "Homogeneous Decaying Isotropic Turbulence" which is a widespread turbulence benchmark. The domain is given by a cube $[0, 2\pi]^3$ with periodic boundary conditions. A starting value (isotropic random velocity, see Figure 1) from a given energy spectrum (calculated via Fourier transform) is prescribed. The problem has a Taylor-scale Reynolds number of Re_λ =150 and the viscosity is $\nu \approx$ 1.5e-5 (air). As a turbulence model we choose a standard LES Smagorinsky model with $\nu_t = (C_s \triangle_h)^2 |S(u)|, \quad |M| := (2M \cdot M)^{\frac{1}{2}}$. The energy dissipation in time is compared to experimental data from [5], see Figure 1, right. The calculations were done with Q_2/Q_1 elements on a mesh with 16^3 cells and the Smagorinsky constant $C_s = 0.17$. Here the filter-width $\triangle_h$ is given by h. This constant was not optimized but the results show good agreement to experimental data. For time discretization we apply a second order IMEX-scheme with a time-step size of 0.0087. The outer FGMRES residual is chosen as 1e-7 to the starting residual, whereas the inner residuals are set to 1e-5 (also relative).

There are several important numerical results. The number of outer FGMRES iterations is independent of the number of CPUs, because there is no difference to the serial algorithm. The number of iterations is independent of the mesh size and lies between 5 to 6 iterations, see Table 1, left. This proves that the preconditioner design works well and the accuracy of $\widetilde{A}^{-1}$ and $\widetilde{S}^{-1}$ is sufficient.

Now, we consider the so-called *strong scalability*, where the number of CPUs n is increased while the mesh size is kept fixed at $h = 1/32$, see Table 1, right. The scaling up to 64 processors is quite reasonable. The performance degrades slightly for larger number of processors, especially in the solution process. There are two reasons for this. First, the problem size is getting too small for the local calculations to dominate the communication costs. Second,

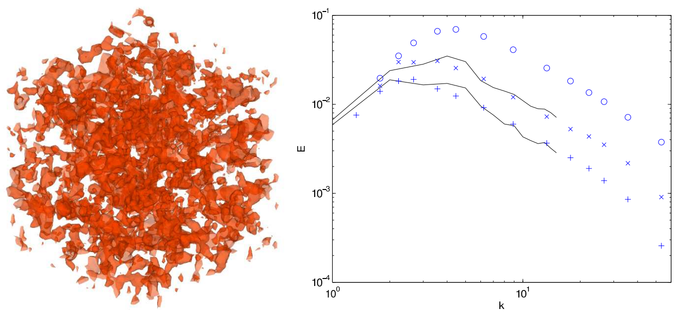

Fig. 1 left: iso-surface of initial velocity spectrum; right: energy spectra at $t = 0.87$ and $t = 2.00$ (upper and lower line) and corresponding experimental data (symbols) with starting value.

Table 1 left: number of FMGRES iterations with respect to mesh size; right: speed-up and efficiency of assembling and solving.

1/h	# DoFs	# It.
8	2312	5
16	112724	5
32	859812	5
48	2855668	6
64	6714692	5

# CPUs	speed-up assembling	efficency assembling	speed-up solving	efficiency solving
4	1.00	100%	1.00	100%
8	1.92	96%	1.72	86%
16	3.70	93%	2.91	73%
32	6.33	79%	4.69	59%
64	12.79	79%	7.39	46%

the solver for the A-block, which takes the most time in the solution process, does not scale linearly.

Table 2 shows the *weak scalability*, where the problem size increases together with the number of CPUs. This keeps the number of degrees of freedom per CPU nearly constant. The results are satisfying and efficiency only degrades slightly with a large number of CPUs.

6 Summary and outlook

We recap our plans stated in the introduction and critically look what we accomplished in this paper. The development of a highly scalable parallel Navier-Stokes solver is underway. The parallel scalability is shown with the numerical results, but is constrained to a larger number of CPUs for several reasons. Some parts of deal.II are not yet parallellized, e.g., mesh handling and management of the degrees of freedom. The result is degraded performance and an increased demand on memory for a larger number of CPUs. This is visible starting at around 100 CPUs. With [7] we see good scaling up to thousands of CPUs with respect to computational costs and memory requirements. The goal of flexibility is solved in large parts. On the one hand we are able to compare different time discretizations, finite element or-

Table 2 Weak scalability of assembly- and solution-process w.r.t. increasing number of CPUs and number of degrees of freedom (time and efficiency)

# CPUs	1/h	# DoFs	assembly		solving	
6	24	368572	20.07s	100%	44.86s	100%
16	32	859812	18.21s	96%	49.14s	80%
54	48	2855668	19.16s	90%	49.02s	79%
128	64	6714692	19.98s	86%	54.64s	70%

ders and turbulence models. On the other hand we only look at instationary problems with small viscosities. Extending and testing the solver for a broad spectrum of test problems is still work in progress. The different regime of stationary and convection dominated flow poses challenges. The performance of the algebraic multi-grid for the A-Block needs to be verified there.

We present a flexible, parallel, and scalable solver framework for the solution of the incompressible Navier-Stokes equations. The numerical results prove that the design of the preconditioner is promising.

Acknowledgements T. Heister is partly supported by the DFG through GK 1023.

References

1. Ascher, U.M., Ruuth, S.J., Spiteri, R.J.: Implicit–explicit Runge–Kutta methods for time-dependent partial differential equations. Applied Numerical Mathematics: Transactions of IMACS **25**(2–3), 151–167 (1997)
2. Balay, S., Buschelman, K., Gropp, W.D., Kaushik, D., Knepley, M.G., McInnes, L.C., Smith, B.F., Zhang, H.: PETSc Web page (2009). http://www.mcs.anl.gov/petsc
3. Bangerth, W., Hartmann, R., Kanschat, G.: deal.II — a General Purpose Object Oriented Finite Element Library. ACM Transactions on Mathematical Software **33**(4), 27 (2007)
4. Benzi, M., Golub, G.H., Liesen, J.: Numerical Solution of Saddle Point Problems. Acta Numerica **14**, 1–137 (2005)
5. Comte-Bellot, G., Corrsin, S.: Simple eulerian time correlation of full- and narrow-band velocity signals in grid generated isotropic turbulence. J. Fluid Mech. **48**, 273–337 (1971)
6. Elman, H.C., Silvester, D.J., Wathen, A.J.: Finite Elements and Fast Iterative Solvers with Applications in Incompressible Fluid Dynamics. Numerical Mathematics and Scientific Computation. Oxford University Press, Oxford, New York (2005)
7. Heister, T., Burstedde, C., Kronbichler, M., Bangerth, W.: Algorithms and Data Structures for Massively Parallel Generic Finite Element Codes (in preparation)
8. Henson, V.E., Yang, U.M.: Boomeramg: a parallel algebraic multigrid solver and preconditioner. Appl. Numer. Math. **41**(1), 155–177 (2002).
9. John, V.: Large Eddy Simulation of Turbulent Incompressible Flows: Analytical and Numerical Results for a Class of LES Models, *Lect. Notes Comput. Sci. Eng.*, vol. 34. Springer, Berlin (2004)
10. Loghin, D., Wathen, A.J.: Schur complement preconditioners for the Navier–Stokes equations. Int. J. Num. Meth. in Fluids **40**(3–4), 403–412 (2002)
11. Olshanskii, M.A., Vassilevski, Y.V.: Pressure schur complement preconditioners for the discrete oseen problem. SIAM J. Sci. Comput. **29**(6), 2686–2704 (2007).
12. Saad, Y.: A Flexible Inner-Outer Preconditioned GMRES Algorithm. Tech. Rep. 91-279, Minnesota Supercomputer Institute, University of Minnesota (1991).
13. Saad, Y.: Iterative Methods for Sparse Linear Systems, second edn. Society for Industrial and Applied Mathematics, Philadelphia, PA (2003)
14. Turek, S.: Efficient Solvers for Incompressible Flow Problems: An Algorithmic and Computational Approach. Springer, Berlin (1999)