On Robust Parallel Preconditioning for Incompressible Flow Problems

Timo Heister, Gert Lube, and Gerd Rapin

Abstract We consider time-dependent flow problems discretized via 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 Krylow method, which is fully parallelized on a distributed memory parallel computer. We introduce 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

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

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

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

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. Together with the 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 solution 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 lack in the other two requirements. On the other hand commercial codes often work with lowest order discretizations and are not flexible enough. For higher accuracy and flexibility we favour a coupled approach for the saddle point system instead of a splitting scheme. This is not commonly used in parallel and scalable codes.

The parallel architecture is the standard Multiple Instruction, Multiple Data streams (MIMD) 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. This is realized by splitting the data of matrices and vectors 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. $\mathbf{u}^n$. Primarily 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. Note that 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 cost. An 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 the pressure-velocity coupling B.

3 The Solver

In our approach the system (4) is solved using the preconditioned Krylow subspace method FGMRES. This is a variant of the standard GMRES algorithm, for details see [6]. FGMRES can cope with a changing preconditioner in each iteration. This is needed in our case, because the preconditioner is not calculated explicitly as a matrix but is given as an implicit operator which uses iterative solvers internally.

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. The choice of the preconditioner is motivated by the fact that with exact evaluations of A and S the number of outer (F)GMRES steps is at most two, see [4]. The inverse can be calculated by

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

Therefore, each outer iteration requires the solution of two inner problems: the applications of $\widetilde{A}^{-1}$ and $\widetilde{S}^{-1}$. Additionally 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 qualtiy of $\widetilde{A}^{-1}$ and $\widetilde{S}^{-1}$ is adjustable at will, because the outer iteration converges in 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 low condition number of the matrix. Thus a BICGStab with Block-ILU preconditioning provides quite 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, in the case of reaction-dominated A we can simplify

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

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

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

and suitable boundary conditions, see [7]. Note that 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 for the Schur complement 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. Periodic boundary conditions for the velocity can be treated with periodic boundary conditions in (5), which provide good results, c.f. Section 5.

4 Implementation Overview

The implementation of the solver described in this paper is built on top of a collection of known libraries, see Figure 1. 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, iterative solvers and preconditioners. 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, one creates a graph, with cells as vertices and edges between two vertices if the corresponding cells are neighbors. This graph is partitioned into several mostly equal-sized sets, so that each CPU "owns" a number of cells. The library METIS minimizes the number of cutted edges. This reduces the amount of communication in parallel calculations. With the partition of the cells one can finally 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 assigend 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.

Among other things, the authors did some improvements in the way deal.II assigns these degrees of freedom, which result in a decrease of the imbalance of the number of degrees of freedom on the different CPUs up to 50%. This is done by making a (deterministic) pseudo-random choice.

The main loop is layed out 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 an implict discretization a fixed-point iteration, which is not done for the IMEX-scheme, surrounds the inner part. Finally the inner part consists of assembling and solving the linear system. The solution process is done by FGMRES as described before. In each iteration the preconditioner is applied once. Finally, the preconditioner consists of the preconditioned, inner solvers for A and S.

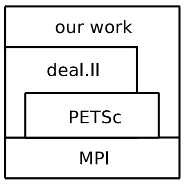

Fig. 1 Framework of used libraries. The basis is given by MPI, which is used by the libraries PETSc and deal.II.

5 Numerical Results

First we present a simple parallel scalability test for a vector-valued Poisson problem on the unit cube with homogeneous Dirichlet boundary conditions

$$-\triangle \mathbf{u} = \mathbf{f} \text{ in } \Omega, \quad \mathbf{u} = 0 \text{ on } \partial\Omega,$$

discretized using Q_2-elements. This problem is related to the A-block of the Oseen problem. It is solved via BiCGStab without preconditioning. The parallel speed-up versus the number of CPUs is compared in Figure 2. This test assures a good partitioning and correct algorithms for the following example. It also presents a practical upper bound for the performance. One observes a linear scaling of the solution process with respect to the number of CPUs if the problem size is large enough with respect to the number of CPUs. The communication overhead has to be sufficiently small compared to the computational work on each CPU. Additionally, the communication can often be done while doing local calculations and is hence hidden. This doesn't work for small local problem sizes.

The second and main example is the simulation of "Homogeneous Decaying Isotropic Turbulence" and 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 3) 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 $\nu \approx 1.5$e-5 (air). As a turbulence model we choose a standard LES Smagorinsky model with $\nu_t = (C_s \triangle_h)^2 |S(u)|, \ \ |M| := (2M \cdot M)^{\frac{1}{2}}$. The energy dissipation in time is compared to experimental data from [5], see Figure 3, 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. The time was discretized using a second order IMEX-scheme with a time-step size of 0.0087.

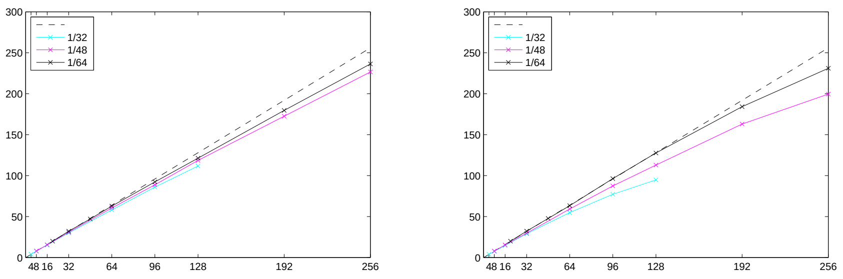

Fig. 2 left: speed-up of assembling for different mesh sizes; right: speed-up of solution process. The dashed line represents perfect linear speed-up.

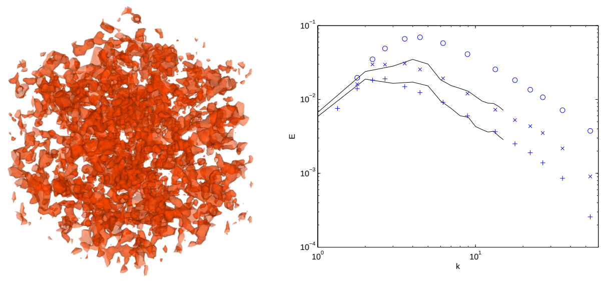

Fig. 3 left: iso-surface of initial velocity spectrum; right: energy spectra at $t = 0.87$ and $t = 2.00$ (upper and lower line) and corresponding experimental data (symbols) with starting value

There are several important numerical results. First the number of FGMRES iterations is independent of the number of CPUs. This is clear, since there is no difference to the serial algorithm. Then 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 $h = 1/16$, see Table 1, right. The scaling up to 16 processors for the solution process and up to 64 processors for assembling is quite reasonable and comparable to the scalability tests done before. The performance degrades for larger number of processors, especially in the solution process. There are two reasons for this. First, the problem size is getting to small. Second, the solver for the A-block, which takes most of the time of the whole solution process, does not scale linearly because of the Block-ILU preconditioner.

Table 2 shows calculations of the same problem, where the problem size and the number of degrees of freedom are increased at the same time - not with the same factors, though. Again, the scaling with the number of CPUs degrades with a large number of CPUs, nevertheless it shows the calculation

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	6

# CPUs	speed-up assembling	efficency assembling	speed-up solving	efficiency solving
4	1.00	100%	1.00	100%
8	1.93	96%	1.90	95%
16	3.71	93%	3.21	80%
32	6.97	87%	3.50	44%
64	12.15	76%	2.79	17%

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

# CPUs	1/h	# DoFs	ass./speed-up/Eff.			solve/speed-up/Eff.		
1	8	15468	3.8s	1.00	100%	2.01s	1.00	100%
4	16	112724	8.2s	3.42	85%	5.89s	2.47	62%
32	32	859812	13.5s	15.74	49%	8.41s	13.22	41%
128	48	2855668	54.2s	13.04	10%	11.1s	33.29	26%

of problems with a large number of degrees of freedom. The degradation compared to Table 1 can be explained with parts of the algorithm scaling worse than $O(n)$.

6 Summary and outlook

We presented 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. Nevertheless there is much room for improvement. A better solver for the A-Block, e.g. parallel Multigrid or a Domain Decomposition Method, combined with smaller mesh sizes should result in good scaling up to a higher number of CPUs.

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. Saad, Y.: Iterative Methods for Sparse Linear Systems, second edn. Society for Industrial and Applied Mathematics, Philadelphia, PA (2003)
7. Turek, S.: Efficient Solvers for Incompressible Flow Problems: An Algorithmic and Computational Approach. Springer, Berlin (1999)