# An all-at-once algebraic multigrid method for finite element discretizations of Stokes problem

Pierre-loïc Bacq, Stéphane Gounand, Artem Napov, Yvan Notay

# An all-at-once algebraic multigrid method for finite element discretizations of Stokes problem

Pierre-Loïc Bacq[1,*] Stéphane Gounand[2] Artem Napov[1]
Yvan Notay[1,†]

[1] *Service de Métrologie Nucléaire*
*Université Libre de Bruxelles*
*50, Av. F.D. Roosevelt, B-1050 Brussels, Belgium*

[2] *Université Paris-Saclay, CEA,*
*Service d'Etudes Mécaniques et Thermiques*
*91191, Gif-sur-Yvette, France*

Report GANMN 21-05

November 2021

**Abstract**

We consider numerical solution of finite element discretizations of the Stokes problem. We focus on the transform-then-solve approach, which amounts to first apply a specific algebraic transformation to the linear system of equations arising from the discretisation, and then solve the transformed system with an algebraic multigrid method. The approach has recently been applied to finite differences discretisations of the Stokes problem with constant viscosity, and has recommended itself as a robust and competitive solution method. In this work, we examine the extension of the approach to standard finite element discretizations of the Stokes problem, including problems with variable viscosity. The extension relies, on one hand, on the use of the successive over-relaxation method as a multigrid smoother for some finite element schemes. On the other hand, we present strategies that allow us to limit the complexity increase induced by the transformation. Numerical experiments show that our method is competitive compared to a state-of-the-art solver based on a block diagonal perconditioner and MINRES, and suggest that the transform-then-solve approach is also more robust. In particular, for problems with variable viscosity,

*Pierre-Loïc Bacq is Research Fellow of the Fonds de la Recherche Scientifique – FNRS; Pierre-Loic.Bacq@ulb.be

†Yvan Notay is Research Director of the Fonds de la Recherche Scientifique – FNRS; yvan.notay@ulb.be

the transform-then-solve approach demonstrates significant speed-up with respect to the block diagonal preconditioner.

**Key words.** solution of discrete Stokes problem, algebraic multigrid for Stokes, transform-then-solve approach, variable viscosity.

**AMS subject classification.** 65N22, 65F10

# 1 Introduction

We consider iterative solution of large sparse linear systems arising from some finite element discretizations of the Stokes problem

$$\begin{aligned} -\nabla \cdot \nu \left(\nabla \vec{u} + \nabla^t \vec{u}\right) + \nabla p &= \vec{f} \quad \text{in } \Omega\,, \\ \nabla \cdot \vec{u} &= 0 \quad \text{in } \Omega\,, \end{aligned} \tag{1.1}$$

with boundary conditions

$$\begin{aligned} \vec{u} &= \vec{w} \quad \text{on } \partial\Omega_D\ , \\ \nu \left(\nabla \vec{u} + \nabla^t \vec{u}\right) \cdot \vec{n} - \vec{n} p &= \vec{s} \quad \text{on } \partial\Omega_N\ . \end{aligned}$$

Here, $\vec{u}$ represents the velocity, $p$ represents the pressure, $\vec{f}$ is a forcing term, $\Omega \subset \mathbb{R}^d$ is a bounded two-dimensional ($d = 2$) or three-dimensional ($d = 3$) domain, $\vec{n}$ is the unit outward normal vector to the domains boundary $\partial\Omega = \partial\Omega_D \cup \partial\Omega_N$, and $\vec{w}$, $\vec{s}$ are known functions on $\partial\Omega_D$ and $\partial\Omega_N$, respectively; $\nu$ is the (kinematic) viscosity. Note that if the viscosity $\nu$ is constant, $\nabla \cdot \nu \left(\nabla \vec{u} + \nabla^t \vec{u}\right)$ can be replaced by $\nu \Delta \vec{u}$ and $\nu \left(\nabla \vec{u} + \nabla^t \vec{u}\right) \cdot \vec{n}$ by $\nu \frac{\partial \vec{u}}{\partial n}$; however, in this work we also consider problems with variable viscosity.

The solution of discretized Stokes problem is performed here with the help of multigrid methods [25, 10]. This choice is motivated by the optimal order convergence properties of multigrid methods for various PDE problems. These methods are based on the recursive use of a two-grid method, which itself is a combination of a *smoother*, typically a greedy iterative method, and a *coarse grid correction*, which amounts to solve a related, but smaller (or coarser) system. A multigrid method is obtained when the coarser system is in turn solved approximately with one or few iterations of the multigrid method, implying thus its recursive use; the recursion stops when the coarse system can be solved directly at a negligible cost. The hierarchy of progressively coarser grids can be obtained by discretizing the problem on a set of progressively refined meshes (geometric multigrid) or built based on the system matrix (algebraic multigrid, abbreviated as AMG in what follows). Here we focus in particular on AMG methods because geometric multigrid methods cannot be used if the problem is discretized on an unstructured mesh. AMG methods also represent a more suitable alternative to direct solvers as they do not require interaction between solver and discretization parts of the code.

Design of multigrid methods for discrete Stokes problems is not a trivial task. This is because the second equation of (1.1) has no pressure term, and therefore the corresponding diagonal block in the system matrix is either a zero matrix or a small-in-magnitude stabilization matrix. As a consequence, simple smoothing schemes, such as Jacobi or Gauss–Seidel iterations, are either undefined or non convergent for Stokes problems. Geometric multigrid methods get around this difficulty by using specific, more involved, smoothing, such as the Vanka smoother [26], the Braess-Sarazin smoother [4], incomplete Uzawa procedures [13, 8] or distributive smoothing [5, 31]. On the other hand, AMG methods typically rely on a simple smoothing scheme. This peculiarity, in addition to the fact that it is not straightforward to define a hierarchy of coarse systems for discrete Stokes problems, probably explains why only few attempts have been made to apply AMG methods to Stokes problems [14, 27, 28].

Therefore, when geometric multigrid cannot be used, the state-of-the-art alternative is to rely on block diagonal, block triangular and related preconditioners (see [3, 6, 22] and works cited therein). In particular, the block diagonal preconditioner with symmetric and positive definite diagonal blocks as proposed in [6, 9] is attractive because it is proved of optimal order for finite element discretizations and can be combined with the MINRES method [23], which provides minimal residual reduction without need of restarting. AMG methods for discrete Laplacians are then typically used as block solvers for specific blocks.

Recently, a new AMG approach for Stokes problems [20, 21] was proposed by the last author. The basic idea is to get around the intrinsic difficulty of Stokes problems by first applying an algebraic transformation which gives each diagonal block a structure of a discrete Laplacian, and then by solving the transformed system with an AMG method; in what follows, we refer to this approach as the *tranform-then-solve* approach. Note that AMG is used here in an all-at-once fashion since it is applied to the whole transformed system, and not to some of its blocks. In particular, in [21] the conditions for an optimal two-grid convergence of an associated multigrid method are established, and a practical multigrid method is proposed, which relies on a specific version of AGMG software [17] to solve the transformed system.

Now, the results in [21] cover mainly finite difference discretizations of (1.1) with constant viscosity. When adapting the approach to finite element discretizations, we noted two main difficulties. First, the Gauss–Seidel smoother used in [21] is not convergent for some finite element schemes[1]. Second, the increase in complexity due to the algebraic transformation is higher than what is observed for finite difference discretizations, calling into question the competitiveness of the approach. This increase is particularly important for some finite element schemes or for problems discretized on unstructured grids. On the other hand, as discussed in the next section, it is difficult to anticipate from previous results how the method will behave in case of variable viscosity.

In this paper, we address all of these issues. First, we show that robust convergence is obtained when using successive over-relaxation (SOR) smoothing instead of Gauss-Seidel in cases where the discretization of the velocity components is based on the $Q_2$ finite elements. Further, we propose two complementary strategies that significantly reduce the impact of the complexity increase due to the transformation.

Eventually, we compare this modified transform-then-solve approach to the state-of-the-art block-preconditioned MINRES method on a varied panel of problems. The results suggest that the transform-then-solve approach is always competitive while being significantly more robust on open flow problems. The improvement with respect to block-preconditioned MINRES is particularly important for problems with varying viscosity, despite the transform-then-solve approach then generally converges slower than when the viscosity is constant.

The remaining of the paper is organized as follows. In Section 2 we recall the transform-then-solve approach. Section 3 contains the description of our benchmark problems. In Section 4 we show that SOR smoothing schemes should be preferred for some finite element discretizations, whereas in Section 5 we present strategies that reduce the complexity

[1]Note that this does not contradict the aforementioned two-grid convergence theory, which holds for damped Jacobi smoothing. But, in practice, when Gauss–Seidel smoothing works, it brings a significant improvement.

increase induced by the transformation. In Section 6 we report on the comparison with the block-preconditioned MINRES method. Conclusions are drawn in Section 7.

# 2 Transform-then-solve approach

The transform-then-solve approach amounts to applying an algebraic transformation to the discretized Stokes problem in order to make each diagonal block suitable for an AMG method for discrete Laplacians. We therefore briefly recall in Section 2.1 the key AMG principles, before presenting the transform-then-solve approach in Section 2.2.

## 2.1 Algebraic multigrid

AMG methods are widely used black-box solvers and preconditioners for the linear systems $A\mathbf{u} = \mathbf{b}$ corresponding of various discrete PDEs problems, including discrete Laplacian-based problems. Like all multigrid methods, AMG achieve an optimal convergence behavior by using a hierarchy of progressively smaller (or coarser) versions of the problem. In AMG methods, such hierarchy is constructed in a back-box fashion during the *setup* stage, before it is used in the *solve* stage. More information about AMG methods can be found in [25, Appendix A].

The key operation during the setup stage is *coarsening*, that is, a construction of the coarser version of a problem. First, a prolongation operator $P$ is defined, which establishes the mapping from the coarse to the fine space:

$$\mathbf{v} = P\mathbf{v}_c\,,$$

where $\mathbf{v}$ is a $n$-dimensional fine vector and $\mathbf{v}_c$ is a $n_c$-dimensional coarse vector; the transpose $P^T$ of $P$ is then typically used to map from the fine to the coarse space. The $n_c \times n_c$ matrix of the coarse problem is then defined with Galerkin formula

$$A_c = P^T A P\,. \tag{2.1}$$

The multigrid hierarchy is obtained by starting with the original problem and repeatedly coarsening the coarsest problem in the hierarchy until this latter is small enough.

In the AMG framework, $P$ is defined automatically based on the information contained in the matrix $A$. Focusing on AMG methods for discrete Laplacians, classical AMG approaches divide the set of all unknowns into two subsets: $\mathcal{F}$ the set of unknowns strictly on the fine grid and $\mathcal{C}$ the set of unknowns both on the fine and coarse grid. The matrix $P$ is then defined in such a way that the variables in $\mathcal{C}$ keep the same values on the coarse and fine grid while the variables in $\mathcal{F}$ are interpolated from neighboring values from $\mathcal{C}$.

Alternatively, aggregation-based AMG approaches define $P$ based on small non-overlapping subsets of fine unknowns. Those subsets are called aggregates and a coarse unknown is associated to each aggregate. The matrix $P$ is then defined in such a way that the value of each coarse unknown is attributed to all the fine unknowns of the corresponding aggregate. In this work, we use an aggregation-based AMG method for reasons explained in [21] and briefly recalled in the next subsection.

During the solve stage, the application of an AMG preconditioner amounts to combining at each level the solution of a coarser problem, or *coarse-grid correction*, with an application of a greedy iterative method, or *smoothing* (we speak of pre-smoothing if it precedes the coarse-grid correction step, and post-smoothing if it follows this step), except for the coarsest level, where the problem is solved with a (sparse) direct solver. In particular, Gauss-Seidel iterations is typically used as a smoother in AMG methods; more details on these are given in Section 4.

## 2.2 Transform-then-solve

The finite element discretization of (1.1) (see, e.g., [6]) leads to a linear system of the form

$$A_0 \begin{pmatrix} \mathbf{u} \\ \mathbf{p} \end{pmatrix} = \begin{pmatrix} F & B^T \\ -B & C \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \mathbf{p} \end{pmatrix} = \begin{pmatrix} \mathbf{b_u} \\ \mathbf{b}_p \end{pmatrix} . \tag{2.2}$$

In this system, when the viscosity $\nu$ is constant, $F$ is typically composed of $d$ decoupled diagonal blocks, with $d = 2$ or $d = 3$, according to the dimensions of $\Omega \subset \mathbb{R}^d$; each such block then corresponds to a scalar discrete Laplacian operator that acts on a velocity component and, hence, $F$ is a symmetric positive definite (SPD) matrix. $B$ and $B^T$ are, respectively, the discrete divergence and the discrete gradient, and $C$ is either a zero matrix or a (small-in-magnitude, symmetric nonnegative definite) stabilization term needed for some discretizations. In particular, when $B$ is not full rank, $C$ is required to be positive definite on the null space of $B^T$ for the matrix $A_0$ to be nonsingular [3]. Note that we changed the sign of the last block of equations compared with what is considered in most references.

The transformation proposed in [21] is induced by the following change of variables:

$$\begin{pmatrix} \mathbf{u} \\ \mathbf{p} \end{pmatrix} = \begin{pmatrix} I & -D_{\mathbf{u}}^{-1} B^T \\ & I \end{pmatrix} \begin{pmatrix} \widehat{\mathbf{u}} \\ \widehat{\mathbf{p}} \end{pmatrix} , \tag{2.3}$$

where $D_{\mathbf{u}}$ is a diagonal matrix, typically the diagonal part of $F$ (or more generally $D_{\mathbf{u}} = \eta \operatorname{diag}(F)$ for some scaling factor $\eta$, but in practice $\eta = 1$ appears near optimal). The system matrix then becomes

$$A = \begin{pmatrix} F & (I - F D_{\mathbf{u}}^{-1}) B^T \\ -B & C + B D_{\mathbf{u}}^{-1} B^T \end{pmatrix} = \begin{pmatrix} F & \widehat{B}^T \\ -B & \widehat{C} \end{pmatrix} . \tag{2.4}$$

Unlike with the initial system matrix $A_0$ in (2.2), the diagonal blocks $F$ and $\widehat{C}$ of the transformed matrix $A$ from (2.4) are both suitable for AMG methods for discrete Laplacians. Regarding the top left $F$ block, as already said, it is typically block diagonal with each diagonal block corresponding a scalar discrete Laplacian operator. As of the bottom right $\widehat{C} = C + B D_{\mathbf{u}}^{-1} B^T$ block, it is dominated by the $B D_{\mathbf{u}}^{-1} B^T$ term since $C$ is either zero or a small-in-magnitude stabilization term. The $B D_{\mathbf{u}}^{-1} B^T$ term is essentially the product of a discrete divergence with a discrete gradient and is therefore similar to a discrete Laplacian, as explicitly seen in [20].

In [21] is then proposed a relatively standard all-at-once AMG preconditioner for the transformed matrix $A$ (2.4), using *unknown-based* coarsening to cope with the fact that the

matrix stems from coupled PDEs; that is, the different types of unknowns (velocity components and pressure) are coarsened separately, each one on the basis of the corresponding diagonal block in the (transformed) system matrix. The prolongation matrix has thus the block diagonal form

$$P \;=\; \begin{pmatrix} P_{\mathbf{u}} & \\ & P_p \end{pmatrix} \;, \tag{2.5}$$

and the coarse grid matrix is obtained by the Galerkin formula $A_c = P^T A\, P$. The accompanying two-grid analysis further proves optimal convergence properties of a simpler variant using a single smoothing step of Jacobi smoother.

It is worth noting that, whereas the approach may in principle be used with any type of AMG method, some investigations reveal that aggregation-based AMG has to be prioritized because classical AMG leads to loss of stability at coarser levels [21, 29]. This leads in [21] to the proposal of solving the transformed system (2.4) with a specific variant of the AGMG software [17]. Indeed, this software implements an aggregation-based AMG method (see [15, 18, 19]), while the specific variant enforces unknown-based coarsening and uses Gauss–Seidel smoothing.

Regarding problems with variable viscosity, little can be anticipated about the results of the transform-then-solve approach for these problems. Of course, the approach can still be applied since it is purely algebraic. Moreover, since $F$ is in general still SPD, the main convergence result in [21] remains valid because it is formulated in an algebraic fashion. However, it is unclear how do behave the involved constants, since the analyses of these provided in [20, 21] essentially refer to model problems with constant coefficients.

Moreover, for the approach to be successful, it is important that the AMG method for the top left block $F$ exhibits an efficient convergence. In case of variable viscosity, the term whose discretization leads to the $F$ block expands

$$-\nabla \cdot \nu \left(\nabla \vec{u} + \nabla^t \vec{u}\right) \;=\; -\nabla \cdot (\nu \nabla \vec{u}) - \nabla \cdot \left(\nu \nabla^t \vec{u}\right) \,.$$

Thus, compared with the constant viscosity case, there is an additional term which makes $F$ typically no longer block diagonal (i.e., all the velocity components are coupled). As a result, there is in general no guarantee that AMG methods for discrete Laplacians preserve their efficiency in the presence of this term.

# 3 Test problems

The transform-then-solve approach is studied here with the help of the test problems that we now describe. The considered test problems are grouped in two sets, based on the software used to generate them. The main properties of the problems are given in Table 1. Note, in particular, that most of the standard finite element schemes for Stokes problems are represented in the considered test set.

The first set is made up of problems generated with the IFISS software [24]. These are defined on structured 2D grids with various pairs of finite elements and are described in [6]. Each problem is discretized using four finite element schemes: $Q_1 - P_0$, $Q_1 - Q_1$, $Q_2 - P_1$ and $Q_2 - Q_1$. `Cavity`, `Channel` and `Collide` problems are defined on the $[-1, 1] \times [-1, 1]$ square domain, whereas `Step` problem is defined on a L-shaped domain. Further, `Cavity`

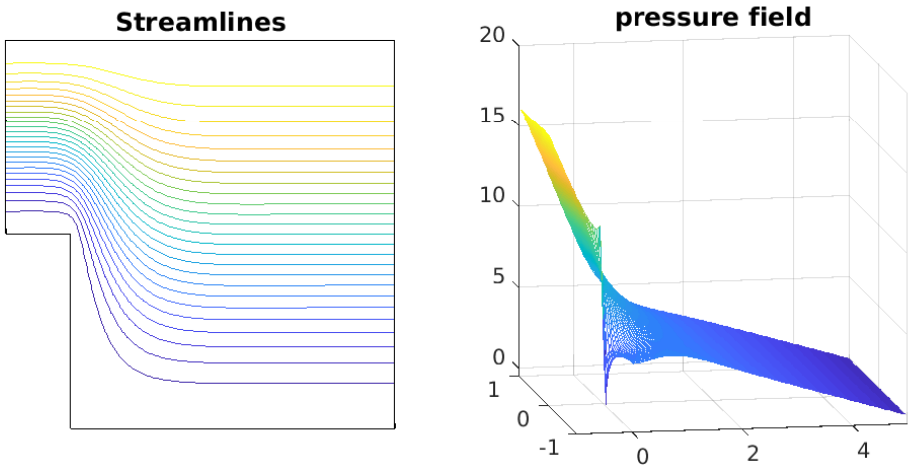


Figure 1: Streamlines and pressure field for the `Step` problem.

problem describes a fluid in a square box (closed flow) whose lid moves from left to right; all boundary conditions are of Dirichlet type. `Channel` problem describes a fluid flow through a square from left to right; the velocity profile is imposed on the left boundary and a Neumann boundary condition is imposed on the right side (open flow). `Collide` problem describes two fluid flows going in opposite directions and colliding in the middle of the domain; practically, two opposed velocity profiles are imposed on the left and right boundaries (closed flow). Eventually, `Step` problem describes a fluid flow over a backward step; a velocity profile is imposed on the left boundary via a Dirichlet condition while the outflow is described via a Neumann condition (open flow). Streamlines and pressure field for `Step` problem discretized with $Q_1 - P_0$ finite elements are illustrated in Figure 1. The viscosity for all these problems is set to $\nu = 1$ and, hence, constant.

Note that in Table 1 we present the data for the smallest considered mesh size, which is also the one used whenever not otherwise specified. In some tests reported below, we check the scalability of the method by including results with mesh size twice as large. This smaller problem size is then referred to as $S_1$, whereas the default size is referred to as $S_2$.

The second set of problems is generated by the Cast3M software [1], developed for structural mechanics and fluid dynamics modelization. It contains both two and three dimensional problems discretized on unstructured grids, except for the `Weld_pool` problem for which the grid is structured but the boundary of the domain is irregular, as illustrated in Figure 2 (left).

More specifically, `Square_a` and `Square_b` (respectively `Cube`) problems are defined on a square (respectively cube) domain and the flow is closed: those problems are similar in nature to the `Cavity` problem of IFISS. Their viscosity is also set to $\nu = 1$.

Further, `Arc_2D`, `Arc_3D_a`, `Arc_3D_b`, and `Weld_pool` problems arise when modeling welding processes; in particular, `Arc_2D`, `Arc_3D_a`, and `Arc_3D_b` problems correspond to modeling of welding arcs, and are open flow problems, whereas `Weld_pool` models a

| | $\frac{n}{10^5}$ | $\frac{nnz}{n}$ | $d$ | $\nu_{\min}$ | $\nu_{\max}$ | mesh | flow | $\mathrm{FE}_{\mathbf{u}}$ | $\mathrm{FE}_p$ | stab. |
|---|---|---|---|---|---|---|---|---|---|---|
| `Cavity` $Q_1 - P_0$ | 7.9 | 12.2 | 2 | 1 | 1 | S | C | $Q_1$ | $P_0$ | U |
| $Q_1 - Q_1$ | 7.9 | 20.4 | 2 | 1 | 1 | S | C | $Q_1$ | $Q_1$ | U |
| $Q_2 - P_1$ | 7.2 | 20.4 | 2 | 1 | 1 | S | C | $Q_2$ | $P_1$ | S |
| $Q_2 - Q_1$ | 5.9 | 24.3 | 2 | 1 | 1 | S | C | $Q_2$ | $Q_1$ | S |
| `Channel` $Q_1 - P_0$ | 7.9 | 12.2 | 2 | 1 | 1 | S | O | $Q_1$ | $P_0$ | U |
| $Q_1 - Q_1$ | 7.9 | 20.4 | 2 | 1 | 1 | S | O | $Q_1$ | $Q_1$ | U |
| $Q_2 - P_1$ | 7.2 | 20.4 | 2 | 1 | 1 | S | O | $Q_2$ | $P_1$ | S |
| $Q_2 - Q_1$ | 5.9 | 24.3 | 2 | 1 | 1 | S | O | $Q_2$ | $Q_1$ | S |
| `Collide` $Q_1 - P_0$ | 7.9 | 12.2 | 2 | 1 | 1 | S | C | $Q_1$ | $P_0$ | U |
| $Q_1 - Q_1$ | 7.9 | 20.4 | 2 | 1 | 1 | S | C | $Q_1$ | $Q_1$ | U |
| $Q_2 - P_1$ | 7.2 | 20.4 | 2 | 1 | 1 | S | C | $Q_2$ | $P_1$ | S |
| $Q_2 - Q_1$ | 5.9 | 24.3 | 2 | 1 | 1 | S | C | $Q_2$ | $Q_1$ | S |
| `Step` $Q_1 - P_0$ | 21.7 | 12.3 | 2 | 1 | 1 | S | O | $Q_1$ | $P_0$ | U |
| $Q_1 - Q_1$ | 21.7 | 20.5 | 2 | 1 | 1 | S | O | $Q_1$ | $Q_1$ | U |
| $Q_2 - P_1$ | 19.9 | 20.5 | 2 | 1 | 1 | S | O | $Q_2$ | $P_1$ | S |
| $Q_2 - Q_1$ | 16.3 | 24.4 | 2 | 1 | 1 | S | O | $Q_2$ | $Q_1$ | S |
| `Square_a` | 13.3 | 17.3 | 2 | 1 | 1 | U | C | $P_2$ | $P_1$ | S |
| `Square_b` | 12.8 | 16.1 | 2 | 1 | 1 | U | C | $P_1$ | $P_1$ | U |
| `Arc_2D(cst)` | 1.1 | 18.5 | 2 | 1 | 1 | U | O | $P_1$,$Q_1$ | $P_1$,$Q_1$ | U |
| `Cube` | 7.6 | 36.2 | 3 | 1 | 1 | U | C | $P_1$ | $P_1$ | U |
| `Arc_3D_a(cst)` | 2.2 | 61.5 | 3 | 1 | 1 | U | O | $R_1$,$Q_1$ | $R_1$,$Q_1$ | U |
| `Arc_3D_b(cst)` | 5.4 | 62.6 | 3 | 1 | 1 | U | O | $R_1$,$Q_1$ | $R_1$,$Q_1$ | U |
| `Weld_pool(cst)` | 13.3 | 158.5 | 3 | 1 | 1 | S | C | $Q_2$ | $Q_1$ | S |
| `Arc_2D` | 1.1 | 18.5 | 2 | $5.8\,10^{-5}$ | $2.7\,10^{-4}$ | U | O | $P_1$,$Q_1$ | $P_1$,$Q_1$ | U |
| `Arc_3D_a` | 2.2 | 96.8 | 3 | $3.8\,10^{-5}$ | $2.7\,10^{-4}$ | U | O | $R_1$,$Q_1$ | $R_1$,$Q_1$ | U |
| `Arc_3D_b` | 5.4 | 99.4 | 3 | $3.8\,10^{-5}$ | $2.7\,10^{-4}$ | U | O | $R_1$,$Q_1$ | $R_1$,$Q_1$ | U |
| `Weld_pool` | 13.3 | 200.9 | 3 | $2.5\,10^{-3}$ | 1.7 | S | C | $Q_2$ | $Q_1$ | S |

Table 1: Main properties of the considered test problems, $n$ being the number of unknowns and $nnz$ the number of nonzero entries of the initial system matrix $A_0$. Column with $d$ as a header indicates if the problem is two- ($d = 2$) or three-dimensionnal ($d = 3$); $\nu_{\min}$ and $\nu_{\max}$ columns indicate, respectively, minimal and maximal value of viscosity $\nu$, "mesh" column indicates if the mesh is structured (S) or unstructured (U); "flow" column indicates if the flow is open (O) or closed (C); columns $\mathrm{FE}_{\mathbf{u}}$ and $\mathrm{FE}_p$ indicate the finite element scheme used for the discretization of, respectively, velocity and pressure; $R_1$ designates the linear-bilinear shape functions for prismatic elements; "stab." indicates if the discretization is stable (S) or has been stabilized by adding a stabilization term to the pressure equation (U).

weld pool and is a closed flow problem. The involved fluids have viscosity varying with temperature which is nonuniform during the simulated process; this is illustrated with Figure 2 (right). Hence these problems naturally present variable viscosity, whose range depends on the temperature profile. Besides, we found it interesting to also consider the constant viscosity variant of these problems, although these are artificial. On the one hand, this allows us to better assess the robustness with respect to variable viscosity. On

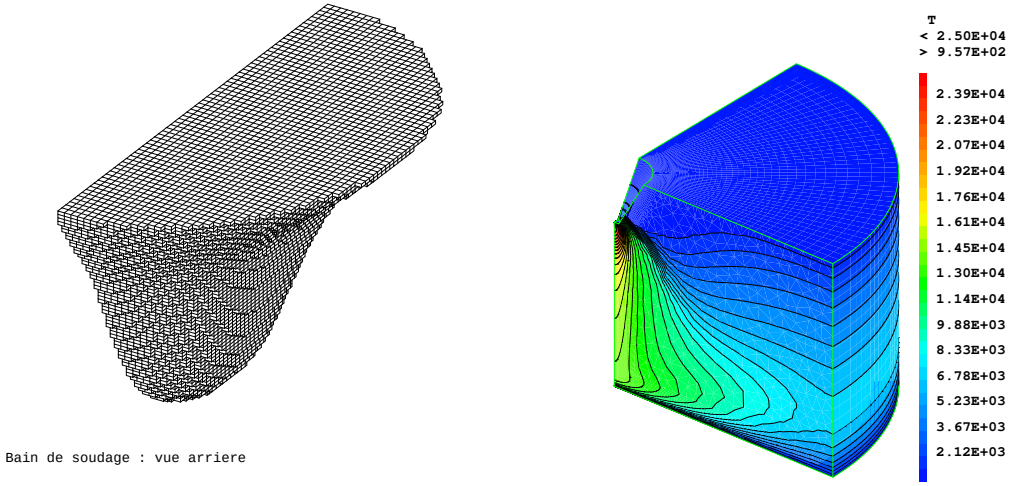


Figure 2: Discretization grid for `Weld_pool(cst)` problem (left) and temperature distribution for `Arc_3D_b` problem (right); in this latter case, the viscosity is related to the temperature via the relation pictured in figure 3.

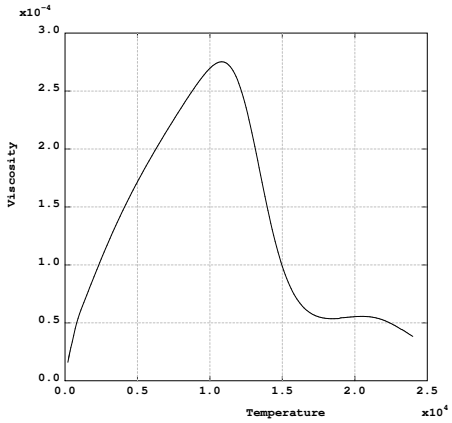


Figure 3: Viscosity of argon as a function of temperature used in `Arc_2D`, `Arc_3D_a`, and `Arc_3D_b` problems.

the other hand, the resulting discrete equations are representative of systems that one may obtain when modeling constant viscosity fluids on unstructured grids with similar boundary conditions.

# 4 Smoothing

Here we address the smoother convergence issues encountered with the transform-then-solve approach for some finite element schemes. We recall that the original transform-then-solve approach in [21] is based on the AGMG software in unknown-based coarsening

| | SOR (automatic $\omega$) | | | | Gauss–Seidel | |
|---|---|---|---|---|---|---|
| | $S_1$ | | $S_2$ | | $S_1$ | $S_2$ |
| | #it | ($\omega$) | #it | ($\omega$) | #it | #it |
| `Cavity` $Q_1 - P_0$ | 13 | (0.68) | 17 | (0.60) | 12 | 13 |
| $Q_1 - Q_1$ | 14 | (0.61) | 14 | (0.61) | 11 | 12 |
| $Q_2 - P_1$ | 20 | (0.52) | 18 | (0.52) | 16 | 16 |
| $Q_2 - Q_1$ | 22 | (0.50) | 22 | (0.50) | $> 99$ | $> 99$ |
| `Channel` $Q_1 - P_0$ | 13 | (0.68) | 14 | (0.60) | 11 | 11 |
| $Q_1 - Q_1$ | 13 | (0.61) | 13 | (0.61) | 12 | 11 |
| $Q_2 - P_1$ | 20 | (0.52) | 20 | (0.52) | 17 | 17 |
| $Q_2 - Q_1$ | 23 | (0.50) | 23 | (0.50) | $> 99$ | $> 99$ |
| `Collide` $Q_1 - P_0$ | 15 | (0.68) | 19 | (0.60) | 13 | 15 |
| $Q_1 - Q_1$ | 16 | (0.61) | 17 | (0.61) | 14 | 14 |
| $Q_2 - P_1$ | 24 | (0.52) | 24 | (0.52) | 18 | 19 |
| $Q_2 - Q_1$ | 24 | (0.50) | 25 | (0.50) | $> 99$ | $> 99$ |
| `Step` $Q_1 - P_0$ | 17 | (0.68) | 18 | (0.60) | 15 | 16 |
| $Q_1 - Q_1$ | 16 | (0.61) | 15 | (0.61) | 13 | 13 |
| $Q_2 - P_1$ | 24 | (0.52) | 23 | (0.52) | 19 | 19 |
| $Q_2 - Q_1$ | 22 | (0.50) | 34 | (0.50) | $> 99$ | $> 99$ |
| `Square_a` | | | 23 | (0.50) | | 18 |
| `Square_b` | | | 30 | (0.56) | | 36 |
| `Arc_2D(cst)` | | | 72 | (0.69) | | 80 |
| `Cube` | | | 29 | (0.32) | | 22 |
| `Arc_3D_a(cst)` | | | $> 99$ | (0.56) | | 88 |
| `Arc_3D_b(cst)` | | | 39 | (0.56) | | 31 |
| `Weld_pool(cst)` | | | 28 | (0.36) | | $> 99$ |

Table 2: Number of iterations for the transform-then-solve approach with SOR and Gauss–Seidel smoothing; for SOR smoother, the automatically computed value of $\omega$ is indicated under parenthesis.

mode with Gauss-Seidel smoothing. We report in Table 2 the number of iterations needed by the corresponding solver to reduce the relative residual error by the factor of $10^{-6}$. Only problems with constant viscosity are considered, since we want the ensure robustness of the considered method primarily with respect to these problems; methods performance for variable viscosity problems is assessed in Section 6.

Clearly, the method does not converge within 100 iterations for $Q_2 - Q_1$ discretizations. Therefore, in Table 2 we also report the number of iterations when using the default AGMG smoother for unsymmetric problems, which is SOR with a relaxation parameter $\omega$ automatically computed by the code.

Here it is worth recalling what these methods are. Gauss–Seidel iterations update the approximate solution $\mathbf{u}$ as follows:

*Computation of the residual:* $\mathbf{r} = \mathbf{b} - A\,\mathbf{u}$ *;*
*Update of the approximate solution:* $\widetilde{\mathbf{u}} = \mathbf{u} + \mathbf{v}$ *with* $\mathbf{v} = M_A^{-1}\mathbf{r}$ *,*

where $M_A$ is either the lower triangular part (forward Gauss–Seidel) or the upper triangular part (backward Gauss–Seidel) of the system matrix $A$; "forward" (resp. "backward") stems from the fact that the associated linear system is solved using "forward" (resp. "backward") substitution.

In general, a multigrid method can perform well only if its smoother defines a convergent iterative process. Of course, some error modes will converge very slowly, which motivates the coarse grid correction. But it is important to avoid any divergent mode. AGMG uses 1 step of forward Gauss–Seidel as pre-smoother and 1 step of backward Gauss–Seidel as post-smoother, which is always convergent for SPD matrices.

The convergence of Gauss–Seidel is no longer guaranteed for nonsymmetric matrices, but robustness is in general improved using SOR instead of Gauss–Seidel. With this, one performs the same iteration, but the diagonal of $M_A$ is exchanged for $\omega^{-1}\mathrm{diag}(A)$ with $\omega < 1$; or, otherwise stated, $(\omega^{-1}-1)\mathrm{diag}(A)$ is added to the lower or upper triangular part of $A$. (Observe that SOR coincides with Gauss–Seidel when $\omega = 1$.) For nonsymmetric matrices, AGMG automatically computes some relevant $\omega$ using some heuristics, which however tend to favor robustness over speed.

Going back to Table 2, one sees that the transform-then-solve approach with such an SOR smoother and automatically computed $\omega$ is indeed robust except in one case, but often requires slightly more iterations than Gauss–Seidel, with a notable exception for $Q_2 - Q_1$ finite element discretizations.

To better understand the difficulties associated with $Q_2 - Q_1$ finite elements, we resort to Local Fourier Analysis (LFA). We refer to [30] for technical details, and just remind the main philosophy: LFA is a tool which enables a cheap computation of exact quantities associated with two-grid methods, such as the asymptotic convergence factor; however, to apply LFA, the problem has to be simplified to a constant coefficient PDE problem on a uniform, typically Cartesian, grid with periodic boundary conditions. Thus, LFA allows one to cheaply obtain exact quantities for a nearby problem, including when the mesh size parameter goes to zero, whereas similar computation for problems with realistic boundary conditions are restricted to relatively small, and hence not necessarily representative, sizes. Despite its partly heuristic nature, LFA is recognized as a reliable tool for determining the potential of multigrid methods [25].

The LFA convergence parameters are given in Table 3 for the case where one uses model boxwise aggregation for both velocity and pressure components (which is close to the aggregation generated by AGMG in such cases), while, as in AGMG, the pre-smoothing consists in a single forward SOR sweep and the post-smoothing – in a single backward SOR sweep.

The second column gives the LFA asymptotic convergence factor $\rho_{\mathrm{SOR}}$ of the standalone SOR smoothing step (i.e. no coarse grid correction is applied here) as a function of relaxation parameter $\omega$. Logically, the asymptotic convergence factor $\rho_{\mathrm{SOR}}$ is at best practically equal to 1, indicating that the smoother alone cannot converge at an acceptable speed. However, it is interesting to observe that $\rho_{\mathrm{SOR}} > 1$ for $\omega \geq 0.9$; that is, the smoother alone then does not define a convergent iterative process, which opens the door to the failure of the associated multigrid method. Since SOR smoothing with $\omega = 1$ coincides with Gauss–Seidel smoothing, this essentially confirms the previous observations and explains the convergence issues of the original transform-then-solve approach.

| $\omega$ | $\rho_{\mathrm{SOR}}$ | $\rho_{\mathrm{TG}}$ |
|---|---|---|
| 0.1 | 1.00 | 0.89 |
| 0.2 | 1.00 | 0.80 |
| 0.3 | 1.00 | 0.73 |
| 0.4 | 1.00 | 0.67 |
| 0.5 | 1.00 | 0.64 |
| 0.6 | 1.00 | 0.62 |
| 0.7 | 1.00 | 0.58 |
| 0.8 | 1.00 | 0.57 |
| 0.9 | 1.28 | 0.57 |
| 1.0 | 1.87 | 1.60 |

Table 3: LFA results for $Q_2 - Q_1$ discretizations of 2D Stokes problems (and for the mesh size parameter going to zero) as a function of SOR relaxation parameter $\omega$ ; $\rho_{\mathrm{SOR}}$ and $\rho_{\mathrm{TG}}$ are the asymptotic convergence factors of, respectively, the standalone SOR smoothing step (i.e., a successive application of one forward and one backward SOR iteration) and the associated two-grid method.

The third column gives the LFA asymptotic convergence factor $\rho_{\mathrm{TG}}$ of the two-grid method. The results indicate that the convergence issues of the smoother for $\omega = 1$ do carry over to the two-grid method. Next, they suggest that the fastest convergence correspond to the values of $\omega$ around 0.7–0.9; the choice $\omega = 0.9$ should however be treated with care, since the standalone smoother is then already divergent. Further, note that these values are larger than the automatically computed $\omega$ values reported in Table 2, these latter being around 0.5 for $Q_2 - Q_1$ discretizations.

Eventually, in Table 4 the observations based on LFA are corroborated with the actual convergence of the transform-then-solve approach with SOR smoother for all discretizations using $Q_2$ for velocities. The results then suggest that SOR with $\omega = 0.7$ is significantly better than that with the automatically computed value of $\omega$ . Since Gauss–Seidel does not work for $Q_2 - Q_1$ discretizations, this is the choice approach for such problems. Moreover, it can be used for all discretizations using $Q_2$ for velocities, the results being equivalent to that with Gauss-Seidel for $Q_2 - P_1$ discretizations. Further results show that this is, however, not true for other discretizations, where Gauss–Seidel performs better.

We thus end up with the following simple rule: use SOR smoothing with $\omega = 0.7$ if the discretization of the velocities is based on $Q_2$, and Gauss–Seidel smoothing in all other cases. Note that usually which discretization has been used is known in the code that calls the solver for linear system solution; in case it is unknown, we suggest to use SOR smoothing with $\omega = 0.7$ which is robust in all cases considered here.

# 5 Improving the computational complexity

Another issue is a significant increase in complexity observed when the transform-then-solve approach is applied to some problems. It is reflected in the fact that the transformed

| $\omega$ | Auto | 0.6 | 0.7 | 0.8 | 1.0 |
|---|---|---|---|---|---|
| `Cavity` $Q_2 - Q_1$ | 22 | 20 | 19 | 35 | $> 99$ |
| $Q_2 - P_1$ | 18 | 16 | 15 | 14 | 16 |
| `Channel` $Q_2 - Q_1$ | 23 | 21 | 22 | 36 | $> 99$ |
| $Q_2 - P_1$ | 20 | 18 | 17 | 16 | 17 |
| `Collide` $Q_2 - Q_1$ | 25 | 22 | 20 | 30 | $> 99$ |
| $Q_2 - P_1$ | 24 | 18 | 20 | 19 | 19 |
| `Step` $Q_2 - Q_1$ | 34 | 26 | 21 | 30 | $> 99$ |
| $Q_2 - P_1$ | 23 | 22 | 20 | 19 | 19 |
| `Weld_pool(cst)` | 28 | 24 | 23 | $> 99$ | $> 99$ |

Table 4: Number of iterations for the transform-then-solve approach with SOR smoothing for $Q_2 - Q_1$ and $Q_2 - P_1$ discretizations for various values of the relaxation parameter $\omega$. Column with "Auto" header corresponds to the automatically computed values of $\omega$; that with $\omega = 1.0$ corresponds to Gauss–Seidel smoothing.

matrix $A$ may have significantly more nonzero entries than the initial matrix $A_0$. Since the cost of the numerical solution with AGMG is roughly speaking proportional to the number of nonzero entries in the system matrix (in this case $A$), this increase may penalize the transform-then-solve approach, compared with other methods that work with the initial matrix $A_0$. We define the *transformation complexity* as

$$\begin{aligned} \mathrm{cpl}(A) &= \frac{nnz(A)}{nnz(A_0)} \\ &= \frac{nnz(F) + nnz(B) + nnz(\widehat{B}) + nnz(\widehat{C})}{nnz(A_0)} \\ &= 1 + \frac{nnz(\widehat{B}) - nnz(B)}{nnz(A_0)} + \frac{nnz(\widehat{C}) - nnz(C)}{nnz(A_0)} \\ &= 1 + \mathrm{cpl}(\widehat{B}) + \mathrm{cpl}(\widehat{C}) \,, \end{aligned}$$

where $\widehat{B}$, $\widehat{C}$ are as in (2.4), and where

$$\mathrm{cpl}(\widehat{B}) = \frac{nnz(\widehat{B}) - nnz(B)}{nnz(A_0)} \quad , \qquad \mathrm{cpl}(\widehat{C}) = \frac{nnz(\widehat{C}) - nnz(C)}{nnz(A_0)} \,.$$

The values of $\mathrm{cpl}(A)$, $\mathrm{cpl}(\widehat{B})$ and $\mathrm{cpl}(\widehat{C})$ are reported in Table 5. The results show that the penalty is actually important, especially for $Q_2 - P_1$ discretizations and unstructured meshes. Interestingly, in all cases, $\mathrm{cpl}(\widehat{C})$ remains modest and significantly below $\mathrm{cpl}(\widehat{B})$, whose contribution is always "to blame" when $\mathrm{cpl}(A)$ goes above 2.

This observation motivates two complementary strategies proposed here to limit the impact of the transformation on the complexity of the method. The first acts at the fine level, and is based on a clever implementation of Gauss-Seidel or SOR smoothing iterations for matrices of the form (2.4), which does not require forming $\widehat{B}^T$ and allows one to save most of the computational cost associated with the increase of the number of nonzero entries in the top right block.

| | cpl($A$) | cpl($\widehat{B}$) | cpl($\widehat{C}$) |
|---|---|---|---|
| `Cavity` $Q_1 - P_0$ | 1.81 | 0.65 | 0.16 |
| $Q_1 - Q_1$ | 1.78 | 0.52 | 0.26 |
| $Q_2 - P_1$ | 2.43 | 1.08 | 0.36 |
| $Q_2 - Q_1$ | 1.62 | 0.51 | 0.11 |
| `Channel` $Q_1 - P_0$ | 1.81 | 0.65 | 0.16 |
| $Q_1 - Q_1$ | 1.78 | 0.52 | 0.26 |
| $Q_2 - P_1$ | 2.43 | 1.08 | 0.36 |
| $Q_2 - Q_1$ | 1.62 | 0.51 | 0.11 |
| `Collide` $Q_1 - P_0$ | 1.81 | 0.65 | 0.16 |
| $Q_1 - Q_1$ | 1.78 | 0.52 | 0.26 |
| $Q_2 - P_1$ | 2.43 | 1.08 | 0.36 |
| $Q_2 - Q_1$ | 1.62 | 0.51 | 0.11 |
| `Step` $Q_1 - P_0$ | 1.81 | 0.65 | 0.16 |
| $Q_1 - Q_1$ | 1.78 | 0.52 | 0.26 |
| $Q_2 - P_1$ | 2.43 | 1.07 | 0.36 |
| $Q_2 - Q_1$ | 1.62 | 0.50 | 0.11 |
| `Square_a` | 3.36 | 1.61 | 0.75 |
| `Square_b` | 1.75 | 0.50 | 0.25 |
| `Arc_2D(cst)` | 1.77 | 0.51 | 0.26 |
| `Cube` | 2.52 | 1.13 | 0.39 |
| `Arc_3D_a(cst)` | 2.40 | 1.04 | 0.37 |
| `Arc_3D_b(cst)` | 2.45 | 1.07 | 0.38 |
| `Weld_pool(cst)` | ?? 1.83 | 0.77 | 0.06 |

Table 5: Transformation complexity with the contributions of the two transformed blocks.

This strategy cannot be used at coarser levels, since the coarse grid matrices are obtained by the Galerkin formula $A_c = P^T A\, P$ and thus do not have the form (2.4) (see Section 5.2 for details). However, in multigrid algorithms, we have the flexibility to replace the coarse grid matrix by a close approximation as long at this does not have a significant impact on the convergence.

The second strategy exploits this flexibility, and consist in defining the coarse grid matrix by applying the Galerkin formula to a sparsified version of the transformed system matrix (2.4). This sparsified version is obtained by replacing in $A$ the top right block $\widehat{B}^T$ by the original block $B^T$ . In Section 5.2 we justify why this seemingly crude approximation may indeed have only a limited impact on the convergence rate, while the numerical results in Section 6 show that it has in practice nearly no impact at all.

Note that, when combining both strategies, the transformed top right block $\widehat{B}^T$ never need to be formed explicitly.

## 5.1 Fine level

In this section we present an efficient way to implement the Gauss-Seidel and SOR smoothers for the matrices of the form (2.4). This implementation reduces the extra cost associated with the use of the transformed block $\widehat{B}^T$ to that of a matrix multiplication with the lower or the upper triangular part of the top left block $F$. The extra cost associated with the transformed block $\widehat{C}$ is not reduced but, as already noticed, this cost is anyway not dominant. Note that a similar approach for Jacobi smoother is presented in [20]; however, since the solver for the transform-then-solve approach converges significantly faster with Gauss–Seidel or SOR smoothing, the approach in [20] is of limited use for the present work.

Now, AGMG uses a more advanced implementation of the Gauss–Seidel and SOR smoothing than that sketched in Section 4. Indeed, instead of computing the residual with the formula $\mathbf{r} = \mathbf{b} - A\,\mathbf{u}$, it uses a more economical residual update, as summarized in the following two substeps:

*Update of the approximate solution:* $\widetilde{\mathbf{u}} = \mathbf{u} + \mathbf{v}$ *with* $\mathbf{v} = M_A^{-1}\mathbf{r}$ *;*
*Update of the residual:* $\widetilde{\mathbf{r}} = \mathbf{r} - A\,\mathbf{v} = -N_A\mathbf{v}$ *with* $N_A = A - M_A$ *.*

As written in the preceding section, $M_A$ is either the lower triangular part (forward Gauss-Seidel) or the upper triangular part (backward Gauss–Seidel) of the system matrix $A$, or the same with a diagonal correction (backward or forward SOR). This implies that the update of the residual is cheaper with the second equation in the second line, since, e.g., if $M_A$ is the lower triangular part of $A$, $N_A$ is its strict upper part, hence multiplication by $N_A$ costs less that a multiplication by $A$ itself.

In the following, we consider both forward and backward Gauss–Seidel, AGMG using one forward sweep as pre-smoothing and one backward sweep as post-smoothing. We consider explicitly only Gauss–Seidel for the clarity of the presentation, the extension to SOR being straightforward since using SOR instead of Gauss–Seidel requires only to update the diagonals of $M_A$ and $N_A$.

**Forward Gauss–Seidel/SOR.** Here

$$M_A \;=\; \begin{pmatrix} L_F & \\ -B & L_{\widehat{C}} \end{pmatrix}\,, \qquad N_A \;=\; \begin{pmatrix} N_F & (I - F\,D_{\mathbf{u}}^{-1})\,B^T \\ & N_{\widehat{C}} \end{pmatrix}\,, \tag{5.1}$$

where $L_i$ is the lower triangular part of the block $i$ and $N_i$ its strict upper part, with $i \;=\; F, \widehat{C}$.

Computing $M_A^{-1}\mathbf{r}$ in this case does not involve $\widehat{B}^T$, hence a straightforward implementation suffices. On the other hand

$$\begin{pmatrix} \mathbf{w_u} \\ \mathbf{w}_p \end{pmatrix} \;=\; N_A \begin{pmatrix} \mathbf{v_u} \\ \mathbf{v}_p \end{pmatrix}$$

can be computed as follows, remembering that $F = L_F + N_F$:

$$\begin{aligned}
\hat{\mathbf{w}}_\mathbf{u} &= B^T \mathbf{v}_p \\
\tilde{\mathbf{w}}_\mathbf{u} &= D_\mathbf{u}^{-1} \hat{\mathbf{w}}_\mathbf{u} \\
\mathbf{w}_\mathbf{u} &= N_F(\mathbf{v}_\mathbf{u} - \tilde{\mathbf{w}}_\mathbf{u}) + \hat{\mathbf{w}}_\mathbf{u} - L_F \tilde{\mathbf{w}}_\mathbf{u} \\
\mathbf{w}_p &= N_{\widehat{C}} \mathbf{v}_p \,.
\end{aligned}$$

One sees that the only significant additional cost, compared with the situation where the top left block of $N_A$ would be just $B^T$, is an additional multiplication by $L_F$ at the third line.

**Backward Gauss–Seidel.** Here

$$M_A \;=\; \begin{pmatrix} U_F & (I - F D_\mathbf{u}^{-1}) B^T \\ & U_{\widehat{C}} \end{pmatrix} \,, \qquad N_A \;=\; \begin{pmatrix} N_F & \\ -B & N_{\widehat{C}} \end{pmatrix} \,, \tag{5.2}$$

hence only the computation of $M_A^{-1}\mathbf{r}$ requires adaptations. Those we propose solve linear systems

$$M_A \begin{pmatrix} \mathbf{v}_\mathbf{u} \\ \mathbf{v}_p \end{pmatrix} \;=\; \begin{pmatrix} \mathbf{r}_\mathbf{u} \\ \mathbf{r}_p \end{pmatrix}$$

using

$$\begin{aligned}
\mathbf{v}_p &= U_C^{-1} \mathbf{r}_p \\
\mathbf{w}_1 &= B^T \mathbf{v}_p \\
\mathbf{w}_2 &= D_\mathbf{u}^{-1} \mathbf{w}_1 \\
\mathbf{v}_\mathbf{u} &= \mathbf{w}_2 + U_F^{-1}(\mathbf{r}_\mathbf{u} - \mathbf{w}_1 + N_F \mathbf{w}_2) \,,
\end{aligned}$$

the last equation holding because

$$U_F^{-1} F \, \mathbf{w}_2 \;=\; U_F^{-1}(U_F + N_F)\mathbf{w}_2 \;=\; \mathbf{w}_2 + U_F^{-1} N_F \mathbf{w}_2 \,.$$

Here, compared with the situation where the top right block would be just $B^T$, the only significant additional cost is a multiplication by $N_F$, i.e., the strict lower part of $F$.

## 5.2 Coarse levels

The coarse grid matrix is defined by the Galerkin formula (2.1) which, for the prolongation $P$ given by (2.5) and for the system matrix $A$ given by (2.4), is computed as

$$A_c \;=\; \begin{pmatrix} P_\mathbf{u}^T & \\ & P_p^T \end{pmatrix} A \begin{pmatrix} P_\mathbf{u} & \\ & P_p \end{pmatrix} \;=\; \begin{pmatrix} P_\mathbf{u}^T F \, P_\mathbf{u} & P_\mathbf{u}^T (I - F \, D_\mathbf{u}^{-1}) \, B^T P_p \\ -P_p^T B \, P_\mathbf{u} & P_p^T \widehat{C} \, P_p \end{pmatrix}$$

Clearly, with $F_c = P_\mathbf{u}^T F \, P_\mathbf{u}$, $D_c = \mathrm{diag}(F_c)$ and $B_c = P_p^T B \, P_\mathbf{u}$, one will in general not have ${P_\mathbf{u}}^T (I - F D_\mathbf{u}^{-1}) B^T P_p$ equal to $(I - F_c D_c^{-1}) B_c^T$. Hence the implementation described above cannot be used at coarser levels.

However, multigrid algorithms can remain efficient when the coarse grid matrix is replaced by a nearby matrix; in particular, the sparsification of the coarse grid matrix is discussed in [7]. Here, because most of the increase of the complexity is associated with the top right block $\widehat{B}^T = (I - F\,D_{\mathbf{u}}^{-1})\,B^T$, we want to sparsify the corresponding coarse block $P_{\mathbf{u}}^T\,(I - F\,D_{\mathbf{u}}^{-1})\,B^T P_p$. We tested several options and found that overall the best results were obtained replacing this block with $P_{\mathbf{u}}^T B^T P_p = B_c^T$. That is, the sparsified coarse grid matrix $A_{\mathrm{sp},c}$ is still obtained via the Galerkin formula, but applied to a modified matrix:

$$A_{\mathrm{sp},c} \;=\; \begin{pmatrix} P_{\mathbf{u}}^T & \\ & P_p^T \end{pmatrix} A_{\mathrm{sp}} \begin{pmatrix} P_{\mathbf{u}} & \\ & P_p \end{pmatrix} \quad \text{with } A_{\mathrm{sp}} \;=\; \begin{pmatrix} F & B^T \\ -B & \widehat{C} \end{pmatrix}. \tag{5.3}$$

As will be seen in the next section, the number of iterations is only mildly affected by this sparsification, and sometimes it even decreases. This property has likely much to do with the fact that $A_{\mathrm{sp}}$ is always an excellent approximation of $A$, as show in the following theorem which proves that all eigenvalues of $A_{\mathrm{sp}}^{-1}A$ are relatively well clustered around 1.

**Theorem 5.1.** *Let $A$ be a $n \times n$ nonsingular matrix of the form (2.4) with $F$ and $D_{\mathbf{u}}$ being SPD matrices and $C$ being a symmetric nonnegative definite matrix which is positive definite on the null space of $B^T$. Further, let $A_{\mathrm{sp}}$ be as in (5.3). The eigenvalues $\lambda_i(A_{\mathrm{sp}}^{-1}A)$ of $A_{\mathrm{sp}}^{-1}A$ are real and there holds*

$$\frac{1}{1+\gamma_{\mathbf{u}}} \;\le\; \lambda_i(A_{\mathrm{sp}}^{-1}A) \;\le\; 1\,, \quad i = 1, \ldots, n\,, \tag{5.4}$$

*where*

$$\gamma_{\mathbf{u}} \;=\; \lambda_{\max}(D_{\mathbf{u}}^{-1/2} F D_{\mathbf{u}}^{-1/2}) \tag{5.5}$$

*is the maximal eigenvalue of $D_{\mathbf{u}}^{-1/2} F D_{\mathbf{u}}^{-1/2}$.*

*Proof.* Since $F$ is non singular, both $A$ and $A_{\mathrm{sp}}$ can be factored as follows

$$\begin{aligned} A &= \begin{pmatrix} I & \\ -BF^{-1} & I \end{pmatrix} \begin{pmatrix} F & \\ & S \end{pmatrix} \begin{pmatrix} I & F^{-1}\widehat{B}^T \\ & I \end{pmatrix}, \\ A_{\mathrm{sp}} &= \begin{pmatrix} I & \\ -BF^{-1} & I \end{pmatrix} \begin{pmatrix} F & \\ & S_{\mathrm{sp}} \end{pmatrix} \begin{pmatrix} I & F^{-1}B^T \\ & I \end{pmatrix}, \end{aligned}$$

where $S \;=\; C + BF^{-1}B^T$ and $S_{\mathrm{sp}} \;=\; C + BF^{-1}B^T + BD_{\mathbf{u}}^{-1}B^T$. Note that since $F$ and $D_{\mathbf{u}}$ are SPD, and since $C$ is positive definite on the null space of $B^T$, both $S$ and $S_{\mathrm{sp}}$ are SPD. Hence, we can write

$$A_{\mathrm{sp}}^{-1}A \;=\; \begin{pmatrix} I & -F^{-1}B^T \\ & I \end{pmatrix} \begin{pmatrix} I & \\ & S_{\mathrm{sp}}^{-1}S \end{pmatrix} \begin{pmatrix} I & F^{-1}\widehat{B}^T \\ & I \end{pmatrix} \;=\; \begin{pmatrix} I & * \\ & S_{\mathrm{sp}}^{-1}S \end{pmatrix}. \tag{5.6}$$

Denoting with $n_{\mathbf{u}}$ and $n_p$ the dimension of $F$ and $C$, respectively, the $n_{\mathbf{u}}$ first eigenvalues of $A_{\mathrm{sp}}^{-1}A$ are therefore equal to 1, and the $n_p$ remaining eigenvalues are those of $S_{\mathrm{sp}}^{-1}S$. Now, since $S$ and $S_{\mathrm{sp}}^{-1}$ are SPD, the eigenvalues of $S_{\mathrm{sp}}^{-1}S$ are real and positive. Further,

since $S_{\rm sp} - S$ is symmetric nonnegative, $\lambda_{\max}(S_{\rm sp}^{-1}S) \leq 1$. On the other hand, using Courant-Fischer theorem [2], there holds

$$\begin{aligned}
\lambda_{\min}(S_{\rm sp}^{-1}S) &= \min_{\mathbf{z}\in\mathbb{R}^{n_p}} \frac{\mathbf{z}^T S\,\mathbf{z}}{\mathbf{z}^T S_{\rm sp}\,\mathbf{z}} = \min_{\mathbf{z}\in\mathbb{R}^{n_p}} \frac{\mathbf{z}^T(C + BF^{-1}B^T)\mathbf{z}}{\mathbf{z}^T(C + B(D_{\mathbf{u}}^{-1} + F^{-1})B^T)\mathbf{z}} \\
&\geq \min_{\mathbf{w}\in\mathbb{R}^{n_{\mathbf{u}}}} \frac{\mathbf{w}^T F^{-1}\mathbf{w}}{\mathbf{w}^T(D_{\mathbf{u}}^{-1} + F^{-1})\mathbf{w}} = \min_{\mathbf{u}\in\mathbb{R}^{n_{\mathbf{u}}}} \frac{\mathbf{u}^T D_{\mathbf{u}}^{1/2} F^{-1} {D_{\mathbf{u}}}^{1/2}\mathbf{u}}{\mathbf{u}^T\mathbf{u} + \mathbf{u}^T {D_{\mathbf{u}}}^{1/2} F^{-1} {D_{\mathbf{u}}}^{1/2}\mathbf{u}} \\
&\geq \frac{{\gamma_{\mathbf{u}}}^{-1}}{1 + {\gamma_{\mathbf{u}}}^{-1}} = \frac{1}{1+\gamma_{\mathbf{u}}}\,.
\end{aligned}$$

■

The typical value of $\gamma_{\mathbf{u}}$ is around 2; see also the discussion in [20, 21], where this parameter is used in the theoretical analyses. For such values of $\gamma_{\mathbf{u}}$, the lower bound (5.4) is around $1/3$, which corresponds to a good clustering.

Of course, ideally, one would need to prove the clustering of the eigenvalues of $A_{{\rm sp},c}^{-1}A_c$ at the coarse level since what is actually done is substituting $A_c$ with $A_{{\rm sp},c}$ and not $A$ with $A_{\rm sp}$. Unfortunately, we cannot prove such a nice algebraic result for $A_{{\rm sp},c}^{-1}A_c$, but the results reported in Figure 4 suggest (as heuristically expected) that the initial clustering is preserved at the coarse level even though the eigenvalues become then complex (but with a relatively small imaginary part).

# 6 Numerical results

In this section, we compare the transform-then-solve approach (including its sparsified version) to a state-of-the-art method for Stokes problems. For constant viscosity problems, it is the preconditioned MINRES method as described in [6] with block diagonal preconditioner of the form

$$M = \begin{pmatrix} M_F & \\ & \frac{1}{\nu}Q \end{pmatrix}, \tag{6.1}$$

where $M_F$ is an approximation of the block $F$, while $Q$ is the diagonal of the mass matrix on the pressure space and $\nu$ the viscosity. Note that, contrarily to the method presented here, this preconditioner is not purely algebraic as it requires to adapt the discretization process to obtain (the diagonal of) the mass matrix on the pressure space, which is otherwise not needed.

In [6] it is showed that this preconditioner leads to optimal order convergence if $M_F$ is itself an optimal order approximation of $F$. Because $F$ is block diagonal with discrete Laplacians as diagonal blocks, such optimal order approximations are usually obtained with one step of a multigrid method, and in the context of this work it is natural to consider AMG codes for this purpose. We consider two such codes: AGMG [18, 15, 19] and BoomerAMG [11] from Hypre [12]. For AGMG, we use the "block" version of the code that ensure unknown-based coarsening to make sure that different velocity components are not aggregated together (this has no effect when the $F$ is block diagonal).

All in all, the application of the preconditioner comes then down to one iteration of an AMG solver (AGMG or BoomerAMG) for the velocity part, and the multiplication by a diagonal matrix for the pressure part.

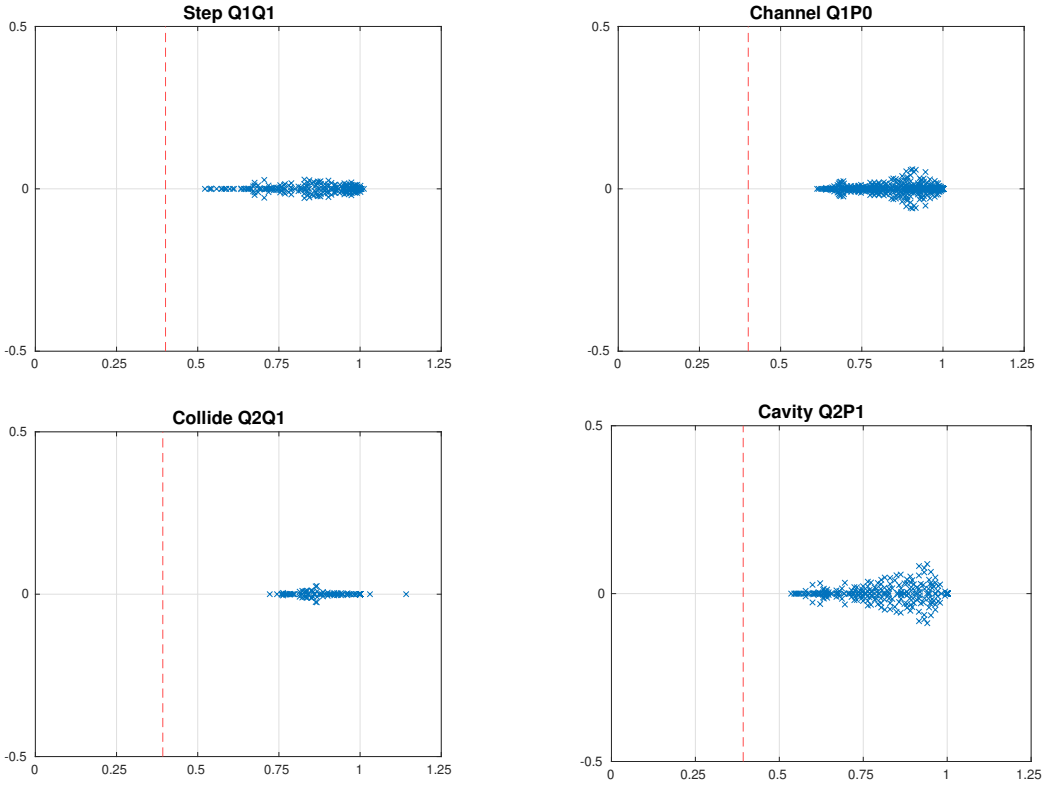


Figure 4: Representation of the spectrum of $A_{\mathrm{sp},c}^{-1}A_c$ in the complex plane for different finite element discretizations and different problems: the blue "×" represent the eigenvalues, while the vertical dashed red lines cross the real axis at the value $x = 1/(1+\gamma_{\mathbf{u}})$ of the lower bound (5.4) .

For problems with variable viscosity, as discussed in [9] the natural extension of this preconditioner is obtained by exchanging $\frac{1}{\nu}Q$ for $Q_\nu$ , the diagonal of the "mass type matrix" defined by

$$(M_\nu^{(\mathrm{mass})})_{i,j} \;=\; \int_\Omega \frac{1}{\nu}\psi_j\psi_i \mathrm{d}\Omega\,, \tag{6.2}$$

where $\psi_i$ are the pressure finite element functions. The diagonal block $M_F$ is defined, as in the constant coefficient case, using one step of an AMG method. However, as noted in the last two paragraphs of Section 2, the AMG methods used for that purpose may loose part of their efficiency because of the additional terms in the discrete equations.

Regarding the transform-then-solve approach, we use a modified version of the AGMG software, implementing the transformation and the features described in Section 5. Here again, we use the "block" version to enforce unknown-based coarsening, that is, to aggregate separately velocity and pressure, but also the different velocity components. For nonsymmetric systems AGMG uses GCR and the restart parameter is set to 50.

The test problems are as those described in Section 3. In all cases we solve the linear system with the zero vector as initial approximation and stop iterations when the relative residual norm is below $10^{-6}$. The computations have been run sequentially on an Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz. Results are reported in Tables 6 and 7.

| | Number of iterations | | | | Solution time | | | |
|---|---|---|---|---|---|---|---|---|
| | M+A | M+B | TaS | TaS sp | M+A | M+B | TaS | TaS sp |
| Cavity $Q_1 - Q_1$ | 46 | 31 | 12 | 12 | 4.3 | 3.1 | 2.2 | 2.2 |
| $Q_1 - P_0$ | 48 | 31 | 13 | 14 | 4.1 | 3.0 | 2.0 | 2.0 |
| $Q_2 - P_1$ | 44 | 29 | 17 | 17 | 3.7 | 3.6 | 3.1 | 2.9 |
| $Q_2 - Q_1$ | 57 | 45 | 19 | 20 | 4.6 | 5.5 | 2.7 | 2.8 |
| Channel $Q_1 - Q_1$ | 56 | 38 | 11 | 12 | 5.3 | 3.8 | 2.2 | 2.3 |
| $Q_1 - P_0$ | 48 | 31 | 14 | 15 | 4.1 | 2.9 | 2.4 | 2.4 |
| $Q_2 - P_1$ | 53 | 34 | 17 | 18 | 4.4 | 4.3 | 3.1 | 3.1 |
| $Q_2 - Q_1$ | 62 | 50 | 19 | 20 | 4.9 | 6.1 | 2.7 | 2.8 |
| Collide $Q_1 - Q_1$ | 49 | 33 | 14 | 14 | 4.5 | 3.3 | 2.6 | 2.6 |
| $Q_1 - P_0$ | 50 | 33 | 15 | 16 | 4.3 | 3.0 | 2.3 | 2.3 |
| $Q_2 - P_1$ | 47 | 30 | 20 | 22 | 4.0 | 3.9 | 3.7 | 3.8 |
| $Q_2 - Q_1$ | 63 | 49 | 20 | 23 | 5.1 | 5.9 | 2.9 | 3.2 |
| Step $Q_1 - Q_1$ | 85 | 50 | 15 | 14 | 22.3 | 14.5 | 8.1 | 7.3 |
| $Q_1 - P_0$ | 91 | 49 | 16 | 17 | 21.6 | 12.9 | 7.5 | 7.5 |
| $Q_2 - P_1$ | 89 | 47 | 21 | 24 | 21.5 | 16.4 | 10.6 | 11.6 |
| $Q_2 - Q_1$ | 91 | 62 | 21 | 21 | 20.6 | 20.9 | 8.4 | 8.1 |
| Square_a | 56 | 33 | 18 | 20 | 11.2 | 7.9 | 9.6 | 9.8 |
| Square_b | 52 | 34 | 35 | 45 | 8.0 | 6.1 | 13.3 | 17.8 |
| Arc_2D(cst) | 204 | 191 | 44 | 41 | 2.8 | 2.9 | 1.5 | 1.3 |
| Cube | 55 | 52 | 22 | 23 | 10.1 | 9.0 | 11.6 | 9.8 |
| Arc_3D_a(cst) | 519 | 855 | 185 | 153 | 32.8 | 63.3 | 33.7 | 23.6 |
| Arc_3D_b(cst) | 666 | 882 | 109 | 59 | 97.9 | 168.8 | 52.8 | 23.7 |
| Weld_pool(cst) | 1657 | 893 | 19 | 21 | 1401.0 | 1015.9 | 34.6 | 29.4 |
| Arc_2D | 411 | 485 | 45 | 73 | 7.4 | 10.7 | 2.2 | 2.9 |
| Arc_3D_a | 981 | 828 | 184 | 178 | 115.6 | 159.7 | 42.8 | 36.2 |
| Arc_3D_b | 961 | 848 | 150 | 169 | 296.2 | 444.9 | 94.1 | 91.9 |
| Weld_pool | 372 | 328 | 17 | 17 | 725.8 | 929.6 | 58.0 | 52.1 |
| Weld_pool(cst) | 204 | 159 | 21 | 23 | 147.8 | 153.9 | 32.5 | 26.6 |

Table 6: Number of iterations and solution time; "M+A" stands for preconditioned MINRES with AGMG to approximate the $F$ block;, "M+B" stands for preconditioned MINRES with BoomerAMG to approximate the $F$ block; "TaS" stands for transform-then-solve method; "TaS sp" stands for transform-then-solve with sparsification of the coarse grid matrices.

We first discuss the results for problems with constant viscosity; this correspond to all but the last 4 rows in Tables 6 and 7.

Let us start by comparing the two preconditioners for MINRES. Looking at Table 6, we see that MINRES associated with AGMG (M+A) requires more iterations than associated with BoomerAMG (M+B), except for the `Arc_3D_a(cst)` and `Arc_3D_b(cst)` problems. On the other hand, both variants perform overall similarly regarding the solution time, with variations from problem to problem. This is in agreement with the fact that BoomerAMG

| | Setup time | | | | Total time | | | |
|---|---|---|---|---|---|---|---|---|
| | M+A | M+B | TaS | TaS sp | M+A | M+B | TaS | TaS sp |
| `Cavity` $Q_1 - Q_1$ | 0.4 | 1.0 | 1.5 | 1.3 | 4.7 | 4.1 | 3.8 | 3.5 |
| $Q_1 - P_0$ | 0.4 | 1.0 | 1.2 | 0.9 | 4.5 | 4.0 | 3.2 | 2.9 |
| $Q_2 - P_1$ | 0.4 | 1.1 | 1.7 | 1.1 | 4.1 | 4.8 | 4.8 | 4.0 |
| $Q_2 - Q_1$ | 0.4 | 1.1 | 1.1 | 0.9 | 5.1 | 6.6 | 3.8 | 3.6 |
| `Channel` $Q_1 - Q_1$ | 0.4 | 0.9 | 1.6 | 1.3 | 5.7 | 4.8 | 3.8 | 3.6 |
| $Q_1 - P_0$ | 0.4 | 1.0 | 1.2 | 1.0 | 4.5 | 3.9 | 3.6 | 3.3 |
| $Q_2 - P_1$ | 0.4 | 1.0 | 1.7 | 1.1 | 4.8 | 5.3 | 4.8 | 4.2 |
| $Q_2 - Q_1$ | 0.4 | 1.0 | 1.1 | 0.9 | 5.4 | 7.1 | 3.9 | 3.6 |
| `Collide` $Q_1 - Q_1$ | 0.4 | 1.0 | 1.5 | 1.2 | 4.9 | 4.3 | 4.1 | 3.8 |
| $Q_1 - P_0$ | 0.4 | 1.0 | 1.2 | 0.9 | 4.7 | 4.0 | 3.5 | 3.2 |
| $Q_2 - P_1$ | 0.4 | 1.1 | 1.7 | 1.1 | 4.4 | 5.0 | 5.4 | 4.9 |
| $Q_2 - Q_1$ | 0.4 | 1.1 | 1.1 | 0.8 | 5.6 | 7.1 | 4.0 | 4.1 |
| `Step` $Q_1 - Q_1$ | 0.9 | 2.3 | 4.0 | 3.3 | 23.3 | 16.8 | 12.2 | 10.6 |
| $Q_1 - P_0$ | 0.9 | 2.3 | 3.1 | 2.4 | 22.5 | 15.2 | 10.6 | 9.9 |
| $Q_2 - P_1$ | 1.1 | 2.5 | 4.3 | 2.8 | 22.7 | 18.9 | 14.9 | 14.4 |
| $Q_2 - Q_1$ | 1.1 | 2.6 | 2.9 | 2.1 | 21.8 | 23.4 | 11.4 | 10.3 |
| `Square_a` | 1.1 | 2.5 | 5.5 | 3.9 | 12.3 | 10.3 | 15.2 | 13.7 |
| `Square_b` | 0.7 | 2.2 | 3.9 | 3.4 | 8.7 | 8.3 | 17.2 | 21.1 |
| `Arc_2D(cst)` | 0.1 | 0.2 | 0.3 | 0.2 | 2.9 | 3.1 | 1.8 | 1.6 |
| `Cube` | 1.1 | 3.5 | 9.8 | 7.8 | 11.2 | 12.5 | 21.4 | 17.6 |
| `Arc_3D_a(cst)` | 0.4 | 1.2 | 3.5 | 2.8 | 33.1 | 64.5 | 37.2 | 26.5 |
| `Arc_3D_b(cst)` | 0.9 | 3.2 | 9.0 | 7.2 | 98.8 | 172.1 | 61.8 | 30.9 |
| `Weld_pool(cst)` | 6.4 | 17.1 | 18.6 | 13.2 | 1407.4 | 1033.0 | 53.2 | 42.7 |
| `Arc_2D` | 0.1 | 0.3 | 0.4 | 0.3 | 7.5 | 10.9 | 2.6 | 3.2 |
| `Arc_3D_a` | 1.0 | 2.5 | 3.9 | 3.2 | 116.6 | 162.2 | 46.6 | 39.5 |
| `Arc_3D_b` | 2.4 | 7.0 | 10.3 | 8.4 | 298.6 | 451.9 | 104.4 | 100.4 |
| `Weld_pool` | 16.2 | 37.2 | 26.7 | 22.8 | 742.0 | 966.8 | 84.6 | 74.8 |
| `Weld_pool(cst)` | 5.5 | 14.9 | 15.9 | 11.6 | 153.3 | 168.8 | 48.7 | 38.2 |

Table 7: Setup time and total time (i.e., setup time + solution time); the abbreviations for the considered methods are the same as in Table 6.

implements a more accurate but more costly preconditioner than AGMG. It is then also not surprising, as seen in Table 7, that MINRES+AGMG requires lower setup time. All in all, it turns out that, if one excludes the `Arc_3D_a(cst)` and `Arc_3D_b(cst)` problems, MINRES+AGMG is slightly slower than MINRES+BoomerAMG if one considers the mean solution time, and slightly faster if one considers the mean total time. Including the `Arc_3D_a(cst)` and `Arc_3D_b(cst)` problems makes MINRES+AGMG the clear winner. This lower performance of BoomerAMG on these configurations confirms the observation made in [16] that BoomerAMG may sometimes lack robustness.

Let us now compare the transform-then-solve ("TaS") approach with its sparsified ver-

sion ("TaS sp"). Table 6 indicates that in terms of number of iterations both methods are essentially similar except, surprisingly, for the `Arc_3D_b(cst)` problem where the sparsified version is significantly better. This constitutes an a posteriori justification of the analysis developed in Section 5.2. However, the solution times do not yield a significant difference. If one excludes the `Arc_3D_b` problem, the mean solution time is only slightly smaller for the sparsified variant.

However, this latter variant is the clear winner if one includes the `Arc_3D_b(cst)` problem, and also if one considers the total time, including thus the setup time (see Table 7). The sparsified approach indeed saves time mostly during the setup phase. This is due to the fact that the successive coarse grid matrices have significantly fewer entries and are therefore cheaper to build. Note that this also alleviates the memory requirements of the method.

Table 6 indicates that, in all cases but one, "TaS sp" converges faster than MINRES in terms of iterations, but not always fast enough to compensate its higher cost. Looking more closely at the total times, we see that "TaS sp" is faster for structured 2D problems (IFISS test suite), especially for the `Step` problems. When coming to more difficult configurations, however, the results seem problem dependent. Roughly speaking, if either of the MINRES variants is able to solve the system in less than 2-3 times the number of iterations needed by the transform-then-solve approach, it is generally the winner thanks to its lower cost by iteration and also its lower setup time. This essentially happens for closed flow problems with unstructured grids `Square_a`, `Square_b` and `Cube`. In the other cases, which include all open flow problems, preconditioned MINRES clearly lacks of robustness while the transform-then-solve approach offers stable performance allowing it to obtain remarkable gains.

To conclude for constant viscosity problems, preconditioned MINRES is to be recommended only for closed flows problems on unstructured grids for which it is relatively robust while "TaS sp" is penalized by its higher complexity. In the other cases, "TaS sp" is faster either because the structure of the grid helps to limit its additional complexity or because preconditioned MINRES struggles due to an apparent lack of robustness for problems with open flows and/or complex geometries. In these cases, the speed up offered by "TaS sp" can be really remarkable.

Regarding problems with variable viscosity, except for the problem `Weld_pool`, "TaS sp" requires significantly more iterations than for the constant viscosity counterpart, without too much surprise in view of the discussion in the last two paragraphs of of Section 2. However, it solves robustly all problems, while preconditioned MINRES struggles even further, leading in all cases to an increase of the speed up offered by "TaS sp", which ranges from 2.4 to 9.2 when comparing with MINRES+AGMG, and from 3.4 to 11.9 when comparing with MINRES+BoomerAMG.

# 7 Conclusions

We have proposed an extension of the transform-then-solve approach from [21] to the finite element discretizations of the Stokes problem. The approach amounts to first apply a specific algebraic transformation to the linear system of equations, and then solve the

transformed system with the AGMG method. The extension addresses two issues arising with the approach in the finite element context. The first issue is the convergence problems observed for $Q_2 - Q_1$ finite element discretizations. The second issue is the complexity increase related to the use of the algebraic transformation, which is potentially penalizing for some problems.

We solve the first issue by proposing a simple adaptation of the smoothing strategy, replacing the original Gauss-Seidel smoother by SOR smoother with $\omega = 0.7$ when $Q_2$ finite elements are used for the velocities.

Regarding the second issue, we identified the top right block as being responsible for most of complexity increase, and we developed two complementary strategies to alleviate its impact. At the fine level, we propose an implementation allowing us to save most of the induced extra cost. At coarse levels, we consider a sparsified version of the matrix.

Numerical results show that our approach is robust for constant viscosity problems, requiring similar numbers of iterations for a wide range of problems, from standard fluid dynamics problems to more complicated welding processes. Moreover, the results also indicate that the convergence of the sparsified version is similar to that of the original one. Somehow unexpectedly, the sparsified version performs even better than the original on two of the most challenging problems. As expected, the total time is lower with the sparsified version.

We finally compare our approach with a state-of-the-art solver based on MINRES with optimal order block diagonal preconditioner. The results suggest that our approach is significantly more robust, leading to remarkable improvements when the reference method does not converge satisfactorily, whereas, otherwise, this latter is competitive thanks to its lower complexity. In particular, preconditioned MINRES seems to struggle with open flow problems on complex geometries. Our approach is also to be recommended when the pressure mass matrix is not available, since then relevant block preconditioners can in general not be defined.

These conclusions are reinforced when considering problems with variable viscosity. While the proposed approach exhibits a slower convergence for these problems, the slowdown is more moderate than for the reference solver, leading to remarkable gains.

# Acknowledgment

The work benefited from the support of the Fonds de Recherche Scientifique-FNRS (Belgium) under grant n° J.0084.16 .

# References

[1] *Cast3M software and documentation.* `http://www-cast3m.cea.fr/`.

[2] O. Axelsson, *Iterative Solution Methods*, Cambridge University Press, Cambridge, UK, 1994.

[3] M. Benzi, G. H. Golub, and J. Liesen, *Numerical solution of saddle point problems*, Acta Numer., 14 (2005), pp. 1–137.

[4] D. Braess and R. Sarazin, *An efficient smoother for the stokes problem*, Applied Numerical Mathematics, 23 (1997), pp. 3–19. Multilevel Methods.

[5] A. Brandt and O. Livne, *Multigrid Techniques: 1984 Guide with Applications to Fluid Dynamics, Revised Edition*, SIAM, 2011.

[6] H. Elman, D. Silvester, and A. Wathen, *Finite Elements and Fast Iterative Solvers*, Oxford University Press, Oxford, 2005.

[7] R. D. Falgout and J. B. Schroder, *Non-Galerkin coarse grids for algebraic multigrid*, SIAM Journal on Scientific Computing, 36 (2014).

[8] F. J. Gaspar, Y. Notay, C. W. Oosterlee, and C. Rodrigo, *A simple and efficient segregated smoother for the discrete Stokes equations*, SIAM J. Sci. Comput., 36 (2014), pp. A1187–A1206.

[9] P. Grinevich and M. Olshanskii, *An iterative method for the Stokes-type problem with variable viscosity*, SIAM J. Sci. Comput., 31 (2009), pp. 3959–3978.

[10] W. Hackbusch, *Multi-grid Methods and Applications*, Springer, Berlin, 1985.

[11] V. E. Henson and U. M. Yang, *BoomerAMG: A parallel algebraic multigrid solver and preconditioner*, Appl. Numer. Math., 41 (2002), pp. 155–177.

[12] *Hypre software and documentation*. `www.llnl.gov/casc/hypre/`.

[13] J. Maitre, F. Musy, and P. Nigon, *A fast solver for the stokes equations using multigrid with a Uzawa smoother*, in Advances in Multi-Grid Methods, D. Braess, W. Hackbusch, and U. Trottenberg, eds., Notes on Numerical Fluid Mechanics, Vol 11, Braunschweig, 1985, Vieweg, pp. 77–83.

[14] B. Metsch, *Algebraic Multigrid (AMG) for Saddle Point Systems*, dissertation, Institut für Numerische Simulation, Universität Bonn, Bonn, Germany, 2013.

[15] A. Napov and Y. Notay, *An algebraic multigrid method with guaranteed convergence rate*, SIAM J. Sci. Comput., 34 (2012), pp. A1079–A1109.

[16] ——, *Algebraic multigrid for moderate order finite elements*, SIAM J. Sci. Comput., 36 (2014), p. A1678–A1707.

[17] Y. Notay, *AGMG software and documentation*. `http://agmg.eu`.

[18] ——, *An aggregation-based algebraic multigrid method*, Electron. Trans. Numer. Anal., 37 (2010), pp. 123–146.

[19] ——, *Aggregation-based algebraic multigrid for convection-diffusion equations*, SIAM J. Sci. Comput., 34 (2012), pp. A2288–A2316.

[20] ——, *A new multigrid approach for Stokes problems*, Numer. Math., 132 (2016), pp. 51–84.

[21] ——, *Algebraic multigrid for Stokes equations*, SIAM J. Sci. Comput., 39 (2017), pp. S88–S111.

[22] ——, *Convergence of some iterative methods for symmetric saddle point linear systems*, SIAM J. Matrix Anal. Appl., 40 (2019), pp. 122–146.

[23] C. C. Paige and M. A. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629.

[24] D. Silvester, H. Elman, and A. Ramage, *Incompressible Flow and Iterative Solver Software (IFISS) version 3.2*, May 2012. `http://www.manchester.ac.uk/ifiss/`.

[25] U. Trottenberg, C. W. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, London, 2001.

[26] S. Vanka, *Block-implicit multigrid solution of Navier-Stokes equations in primitive variables*, J. Comput. Physics, 65 (1986), pp. 138–158.

[27] M. Wabro, *Algebraic Multigrid Methods for the Numerical Solution of the Incompressible Navier-Stokes Equations*, dissertation, Johannes Kepler Universität Linz, 2003.

[28] ——, *AMGe-coarsening strategies and application to the Oseen equations*, SIAM J. Sci. Comput., 27 (2005), pp. 2077–2097.

[29] R. Webster, *Stability of algebraic multigrid for Stokes problems*, Int. J. Numer. Meth. Fluids, 71 (2013), pp. 488–505.

[30] R. Wienands and W. Joppich, *Practical Fourier Analysis for Multigrid Methods*, Chapman & Hall/CRC Press, Boca Raton, Florida, 2005.

[31] G. Wittum, *Multi-grid methods for Stokes and Navier-Stokes equations*, Numer. Math., 54 (1989), pp. 543–563.