

Combining Domain Decomposition and Multigrid Methods: A scalable interface between FROSch and FEAT

Stephan Köhler^[0000-0003-1015-8736],
Oliver Rheinbach^[0000-0002-9310-8533]

1 Introduction

Numerical simulations of fluid dynamics problems typically lead to discretized systems that feature a saddle point structure. Today, these discretized systems are often too large to be solved by direct methods. Consequently, iterative methods, especially Krylov subspace methods combined with multilevel preconditioners based on multigrid and domain decomposition (DD), offer a compelling alternative.

DD techniques provide a scalable preconditioning strategy by distributing the solution process across multiple compute nodes. These methods effectively exploit spatial locality and parallelism, making them well-suited for modern HPC systems. Multilevel or multigrid strategies are crucial components for scalability on architectures with large core counts.

We consider monolithic, multilevel versions of overlapping Schwarz domain decomposition preconditioners of the generalized Dryja-Smith-Widlund (GDSW) type [5, 4], as implemented in the FROSch [7] framework (Fast and Robust Overlapping Schwarz).

This research is conducted as part of the StroemungsRaum project which strives to extend the performance and scalability of the CFD software FEATFLOW [1] by the integration of highly scalable and efficient multigrid solvers with scalable domain decomposition techniques.

FEATFLOW is a modern C++ finite element framework developed at TU Dortmund, with a strong focus on scalable geometric multigrid solvers for large-scale PDE simulations. Within the StroemungsRaum project, it serves as the solver backbone of the IANUS Simulation software platform, which builds on FEATFLOW to deliver simulation-as-a-service.

Stephan Köhler · Oliver Rheinbach
Technische Universität Bergakademie Freiberg, Akademiestr. 6, 09596 Freiberg,
Germany, e-mail: stephan.koehler@math.tu-freiberg.de, oliver.rheinbach@math.tu-freiberg.de

This paper focuses on the software interface between the monolithic, multi-level FROSch preconditioner and the geometric multigrid methods provided by the FEATFLOW software library. The interface enables the use of GDSW-type preconditioners within FEATFLOW, as a building block for constructing efficient hybrid multigrid-domain decomposition solvers for fluid problems.

The geometric multigrid methods in FEATFLOW are highly efficient and scalable. However, for very complex geometries, the multigrid coarse problem can become too large to be solved directly. In such cases, FEATFLOW, as it has so far not featured a scalable coarse solver for saddle-point systems, falls back on unpreconditioned FGMRES. Our algebraic parallel monolithic, multilevel Schwarz method in FROSch can complement FEATFLOW by supplying such a coarse-level solver, as we demonstrate in this paper.

Using a new interface, FROSch can be used as preconditioner for the pressure Poisson, where the fluid problem is solved by an operator splitting approach, or as monolithic preconditioner for the original saddle point. In this paper, we focus on the application of FROSch to the coarse problem of the multigrid method, thus creating a hybrid multigrid-domain decomposition method. The monolithic multilevel DD preconditioners applied here are based on [6, 11] and [9, 15, 10, 12].

Let us describe the interface of FROSch and FEATFLOW. The main functionality of our interface is implemented in two classes: `TpetraCore` and `TpetraCoreStokes`. These classes implement the conversion from the CSR system matrix, provided by FEATFLOW, to a `Tpetra::CrsMatrix`, the setup of FROSch, and the matrix-vector multiplication, used by the Krylov solver provided by FEATFLOW. For the sake of simplicity, we describe the interface only for the monolithic preconditioner, `TpetraCoreStokes`, and make a few additional remarks for `TpetraCore`.

2 Stokes Problem

We consider the Stokes equation on a domain $\Omega \subset \mathbf{R}^d$, $d \in \{2, 3\}$. The weak formulation of the Stokes equation is given by: Find $u \in V_g$ and $p \in Q$ such that $\mu \int_{\Omega} \nabla u : \nabla v \, dx - \int_{\Omega} \operatorname{div} v \, p \, dx = \int_{\Omega} f v \, dx$ for all $v \in V_0$, $-\int_{\Omega} \operatorname{div} u \, q \, dx = 0$ for all $q \in Q_0$, with Dirichlet boundary condition $u = g$ and Neumann boundary condition $\frac{\partial u}{\partial n} - p n = 0$, where n denotes the outward pointing normal vector. The discretized system can be written as the saddle point system

$$\mathcal{A}x := \begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} F \\ 0 \end{bmatrix} =: \mathcal{F}, \quad (2.1)$$

where the matrix A corresponds to the velocity and B^T to the coupling between the velocity and pressure.

2.1 Monolithic Overlapping Schwarz Preconditioner

We decompose the computational domain Ω into N nonoverlapping subdomains Ω_i , $i = 1, \dots, N$ and denote the corresponding subdomain with k layers of overlap with Ω'_i . We denote the local saddle point problem of Ω'_i with \mathcal{A}_i , i.e., \mathcal{A}_i is the restriction of \mathcal{A} onto the overlapping subdomain Ω'_i . The GDSW coarse spaces for (2.1), see [6], can be written as $\mathcal{B}_{GDSW} = \Phi \mathcal{A}_0^{-1} \Phi^T + \sum_{i=1}^N \mathcal{R}_i^T \mathcal{A}_i^{-1} \mathcal{R}_i$, where $\mathcal{A}_0 := \Phi^T \mathcal{A} \Phi$ and $\mathcal{R}_i := \begin{bmatrix} R_{i,u} & 0 \\ 0 & R_{i,p} \end{bmatrix}$. The columns of Φ are the coarse basis functions, $R_{i,u}$ denotes the restriction of the velocity onto the i -th subdomain, and $R_{i,p}$ the restriction of the pressure [6, 11]. Let us remark that our benchmark problems involve inflow and outflow boundary conditions ensuring a uniquely determined pressure.

For the three-level monolithic approach [12], we replace \mathcal{A}_0^{-1} by the application of the monolithic two-level overlapping Schwarz preconditioners. This is simplified by the fact that FROSch can be built almost entirely algebraically. Replacing the exact factorization of the coarse problem of the two-level overlapping Schwarz preconditioner recursively, leads to a multilevel monolithic preconditioner; see [8, 15, 10].

3 Software Interface

Our interface follows the design of the FEATFLOW interface to BoomerAMG [13], which is routinely employed in FEATFLOW to the coarse pressure Poisson equation, but cannot be used effectively for saddle-point systems.

FEATFLOW handles the parallel distribution of (2.1) and the assembling process. The matrix \mathcal{A} follows a row-wise distribution of local saddle point problems across the MPI ranks, with one subdomain per rank, which is a permutation of (2.1). Our interface is nonintrusive because we have restricted ourselves to using the parallel CSR data.

TpetraCoreStokes The `TpetraCoreStokes` constructor takes the global dof offset, numbers of owned/global velocity and pressure dofs, first owned DOF indices, and a `FROSchParameterList` for configuration. These parameters are stored in private variables prefixed with underscore (e.g., `_no_velo`). The class inherits from `TpetraCore`, which implements the pressure Poisson interface; separate classes are needed since multi-filed problems like Stokes require different setup than single-field problems.

The conversion from dof data to `Tpetra::CsrMatrix` is handled by `init_core(const IT_* row_ptr, const IT_* col_idx)`, which takes CSR format pointers. First, we create a `Tpetra::Map` encoding the parallel row distribution:

Listing 1 Create _map

```
1 std::vector<GO> dvec(_no_dofs);
2 for(IndexType i=0; i<_no_velo; ++i)
```

```

3   dvec.at(i) = static_cast<GO>(_fo_velo+i);
4   for(IndexType i=0; i<_no_pres; ++i)
5     dvec.at(_no_velo+i) = static_cast<GO>(_ng_velo+_fo_pres+i);
6   const Teuchos::ArrayView<const GO> dview = Teuchos::arrayViewFromVector(dvec)
7   ;
  _map = Teuchos::rcp(new Tpetra::Map(_ng_velo+_ng_pres,dview,0,_comm));

```

This renumbers rows of \mathcal{A} to match (2.1) as required by FROSch, without changing the parallel distribution.

The CSR graph construction includes also column renumbering to match (2.1).

The subdomain overlap can be computed algebraically for single-field problems by inspecting the matrix graph. For Stokes (2.1), we extract the velocity block to compute the overlap; no overlap is needed for the pressure since FEATFLOW's default Q2-P1-discontinuous elements have discontinuous pressure. This leads to constant coarse basis functions for the pressure in FROSch.

Matrix-Vector Multiplication The matrix-vector multiplication respects FEATFLOW's internal storage using correction vector `_vec_cor` and defect vector `_vec_def`. The function `solve_frosch` (Listing 2) wraps these vectors with Thyra objects and applies the FROSch preconditioner via `Thyra::apply<SC>(...)`. This implementation works for both `TpetraCoreStokes` and `TpetraCore` due to inheritance.

Listing 2 Implementation of the matrix-vector multiplication

```

1 void solve_frosch(void* cr,void* pre) {
2   TpetraCore *core = reinterpret_cast<TpetraCore *>(cr);
3   FROSchPrecond *fprecond = reinterpret_cast<FROSchPrecond *>(pre);
4   auto thMap = Thyra::tpetraVectorSpace<SC,L0,G0,NO>(core->_vec_cor->getMap()
5   );
6   auto thDMap = Thyra::tpetraVectorSpace<SC,L0,G0,NO>(Tpetra::
7   createLocalMapWithNode<L0,G0,NO>(core->_vec_cor->getNumVectors(), core
8   ->_vec_cor->getMap()->getComm());
9   auto thX = Teuchos::rcp(new Thyra::TpetraMultiVector<SC,L0,G0,NO>());
10  thX->initialize(thMap,thDMap,core->_vec_cor);
11  thMap = Thyra::tpetraVectorSpace<SC,L0,G0,NO>(core->_vec_def->getMap());
12  thDMap = Thyra::tpetraVectorSpace<SC,L0,G0,NO>(Tpetra::
13  createLocalMapWithNode<L0,G0,NO>(core->_vec_def->getNumVectors(), core
14  ->_vec_def->getMap()->getComm());
15  auto thB = Teuchos::rcp(new Thyra::TpetraMultiVector<SC,L0,G0,NO>());
16  thB->initialize(thMap,thDMap,core->_vec_def);
17  Teuchos::RCP<Thyra::LinearOpBase<SC>> LinPrecOp = fprecond->ThyraPrec->
18  getNonconstUnspecifiedPrecOp();
19  Thyra::apply<SC>(*LinPrecOp,Thyra::NOTRANS,*(thB.getConst()),thX.ptr());
20 }

```

Parameterlist The wrapper class `FROSchParameterList` is used to stores relevant FROSch options consistent with FEATFLOW. The corresponding `Teuchos::ParameterList` is constructed by `create_core`.

Generalization Currently, our approach can be applied to fluid problems with a discontinuous pressure discretization. To extend the interface to the case of continuous pressure, FEATFLOW needs to provide the information which pressure dofs are shared between which subdomains, since this cannot be computed algebraically from the saddle point problem (2.1). In principle, using the same approach, a extension to general saddle point problem would be possible.

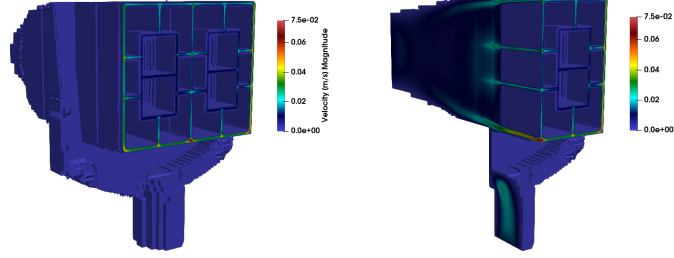


Fig. 1 Visualization of the magnitude of the velocity of an incompressible Navier-Stokes simulation with a Carreau-type material model on an extrusion die. There are two inflow boundaries and one outflow boundary.

4 Model problem

As a model problem, we consider a highly viscous non-Newtonian Fluid modeled by an incompressible stationary Stokes problem with a Carreau-type material model [3], with inflow and outflow boundary conditions: For given $\Omega \subset \mathbb{R}^d$, find $\mathbf{u} \in H^1(\Omega)$, $p \in L^2(\Omega)$ with

$$\begin{aligned} 0 &= -\nabla p + \nabla \cdot \boldsymbol{\sigma}(\mathbf{u}, p) \quad \text{in } \Omega, & \nabla \cdot \mathbf{u} &= 0 \quad \text{in } \Omega, & \mathbf{u} &= \mathbf{g} \quad \text{on } \partial\Omega_{\text{in}}, \\ \boldsymbol{\sigma}(\mathbf{u}, p)\mathbf{n} &= 0 \quad \text{on } \partial\Omega_{\text{out}}, & \mathbf{u} &= 0 \quad \text{on } \partial\Omega \setminus (\partial\Omega_{\text{in}} \cup \partial\Omega_{\text{out}}), \end{aligned}$$

where the constitutive law is defined as $\boldsymbol{\sigma} = -p\mathbf{I} + 2\nu(\mathbf{u})\frac{1}{2}(\mathbf{u} + (\nabla\mathbf{u})^T)$, where $\nu(\mathbf{u}) = \nu_0 \left(1 + (\lambda\dot{\gamma})^2\right)^{\frac{n-1}{2}}$ and $\dot{\gamma} = \sqrt{2(\nabla\mathbf{u} + (\nabla\mathbf{u})^T) : (\nabla\mathbf{u} + (\nabla\mathbf{u})^T)}$ and $\nu_0 \gg 1$. Note that the model has a nonlinearity in the constitutive law.

We use Q2 finite elements for the velocity and P1-discontinuous for the pressure. Our computational domain Ω is an extrusion die; see Figure 1. The model problem is solved by the FEATFLOW library using a geometric multigrid, see [14, 16]. For the solution of the multigrid coarse problem, we compare a (restarted) FGMRES Krylov method preconditioned by our parallel monolithic multilevel FROSch (cf. [12]) with the default coarse solver in FEATFLOW, which is an unpreconditioned (restarted) FGMRES, if the coarse problem is too large to be solved directly.

Note that the coarse problem of the multigrid method is solved only to a low accuracy, i.e., we use a stopping criterion of 10^{-1} for the reduction of the relative residual by FROSch. For the outer iterations, we use an alternating Picard-Newton algorithm which is the limit of an implicit-explicit scheme [2].

All computations were performed on the Fritz cluster at FAU Erlangen-Nürnberg.

In Table 1, we report the results for our simulations where we use either a two-level (256 MPI ranks) or three-level (2 048 and 16,384 MPI ranks) monolithic overlapping Schwarz preconditioner for the multigrid coarse problem.

In Table 2, we report the results for the same problem setup using an unpreconditioned FGMRES, which was, up to now, the only option available in FEATFLOW for large coarse saddle point problems.

Table 1 Simulation of the flow in an extrusion die for three different meshes; geometric multigrid method with FMGRES as coarse solver, preconditioned by FROSch. When FEATFLOW and FROSch run on 256 MPI ranks, two-level monolithic overlapping Schwarz is used. When they run on 2048 and 16384 MPI ranks, three-level monolithic overlapping Schwarz is applied.

#Dofs	#MPI ranks	#Coarse dofs	total time [in s]	coarse solve [in s]	#Kryl. it. (cum.)
13 840 823	256	2 145 423	369	185	521
99 344 386	2 048	13 840 823	539	312	666
742 949 702	16 384	99 344 387	594	341	668

We report the problem size (*#Dofs*), the number of MPI ranks (*#MPI ranks*), the problem size of the multigrid coarse problem (*#Coarse dofs*), the total runtime of the simulation (*Total time*), the time to solve the coarse problem (*Coarse solve*), and the cumulative Krylov iterations (*#Kryl. it. (cum.)*). Note that the timings for the coarse problem do not include the setup of the FROSch preconditioner. The setup is, however, included in the total runtime. For instance, the total runtime for 2048 subdomains of 539s, which includes the setup time of 168s, is a factor of two faster than the total runtime of the unpreconditioned version of 1057s.

Due to the complex die geometry, PARMETIS produces imbalanced partitions with highly varying subdomain sizes, resulting in some load imbalance.

For the simulation with 256 MPI ranks in Table 1 (first row) and Table 2 (first row), the multigrid coarse problem has approximately 68000 cells. Preconditioning FMGRES by FROSch, see Table 1, needs 521 iterations whereas the unpreconditioned FMGRES needs over 20000 iterations. However, the total runtime is higher when using FROSch (369s compared to 327s) due to the setup cost of the Schwarz preconditioner which is not amortized, later.

The results for 2048 MPI ranks are presented in Table 1 (second row) and Table 2 (second row). Preconditioning FMGRES by monolithic three-level FROSch results in a cumulative total of 666 Krylov iterations, whereas the unpreconditioned FMGRES needs over 60000 iterations; see Table 1. Using the overlapping Schwarz preconditioner, we need 312s for the solution of coarse problem, which is more than 3 times faster compared the unpreconditioned FMGRES. The total runtime is nearly 2 times faster when using FROSch, despite the significant cost for the setup of the preconditioner.

The computation using 16384 MPI are presented in Table 1 (third row) and Table 2 (third row). Preconditioning FMGRES by monolithic three-level FROSch results in a cumulative total of 668 Krylov iterations to solve the coarse problem, whereas the unpreconditioned FMGRES exceeds the computation wall time of 25 minutes. The iteration numbers of 668 iterations (16384 ranks), 666 iterations (2048) MPI ranks and 521 iterations (256 MPI ranks) show that the method is numerically scalable. The total runtimes of 594s for 16384 MPI ranks, compared to 539s for 2048 MPI ranks and 369s for 256 MPI ranks show that our hybrid multigrid-domain decompose method is sufficiently parallel scalable and also applies when the default coarse saddle point problem fails.

Table 2 Simulation of the flow in an extrusion die for three different meshes; geometric multigrid method with *unpreconditioned* FMGRES as coarse solvers. For the largest problem the walltime was exceeded.

#Dofs	#MPI ranks	#Coarse dofs	total time [in s]	coarse solve [in s]	#Kryl. it. (cum.)
13 840 823	256	2 145 423	327	252	20 959
99 344 386	2 048	13 840 823	1 057	993	60 922
742 949 702	16 384	99 344 387	>1 500	-	-

5 Summary

We successfully implemented an interface between FEATFLOW and FROSch in a way that allows to use FROSch as a building block in the geometric multigrid methods of FEATFLOW. Using our monolithic multilevel overlapping Schwarz preconditioner, the efficiency and scalability of the fluid simulations in FEATFLOW can be extended. Our approach allows significantly larger coarse problems, which are necessary, e.g., for complex computational domains.

Acknowledgements This work is part of StroemungsRaum which is funded by the German Federal Ministry of Education and Research (BMFTR) under grant no. 16ME0708 as part of the SCALEXA initiative. SCALEXA is a German strategic initiative for advancing high-performance computing (HPC) software development in the exascale era. The authors gratefully acknowledge the scientific support and HPC resources provided by the Erlangen National High Performance Computing Center (NHR@FAU) of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) under the NHR project k107ce. NHR funding is provided by federal and Bavarian state authorities. NHR@FAU hardware is partially funded by the German Research Foundation (DFG) - 440719683. We would like to thank Peter Zajac and Maximilian Esser from TU Dortmund for their part of the implementation of the FEAT and FROSch interface and the implementation of the numerical examples and Otto Mierka for the meshes.

References

1. FEATFLOW (2025). URL <http://www.featflow.de>
2. Barrenechea, G., Castillo, E., Pacheco, D.: Implicit-explicit schemes for incompressible flow problems with variable viscosity. *SIAM J. Sci. Comput.* **46**(4), A2660–A2682 (2024)
3. Carreau, P.J.: Rheological equations from molecular network theories. *Trans. Soc. Rheol.* **16**(1), 99–127 (1972)
4. Dohrmann, C.R., Klawonn, A., Widlund, O.B.: Domain decomposition for less regular subdomains: overlapping Schwarz in two dimensions. *SIAM J. Numer. Anal.* **46**(4), 2153–2168 (2008)
5. Dohrmann, C.R., Klawonn, A., Widlund, O.B.: A family of energy minimizing coarse spaces for overlapping Schwarz preconditioners. In: *Domain Decompos. Methods Sci. Eng. XVII, LNCSE*, vol. 60, pp. 247–254. Springer, Berlin (2008)
6. Heinlein, A., Hochmuth, C., Klawonn, A.: Monolithic overlapping schwarz domain decomposition methods with gds coarse spaces for incompressible fluid flow problems. *SIAM J. Comput.* **41**(4), C291–C316 (2019)

7. Heinlein, A., Klawonn, A., Rajamanickam, S., Rheinbach, O.: FROSch: A fast and robust overlapping Schwarz domain decomposition preconditioner based on Xpetra in Trilinos. In: Domain Decompos. Methods Sci. Eng. XXV, pp. 176–184. Springer International Publishing, Cham (2020)
8. Heinlein, A., Klawonn, A., Rheinbach, O., Röver, F.: A three-level extension of the GDSW overlapping Schwarz preconditioner in two dimensions. In: Advanced Finite Element Methods with Applications: Selected Papers from the 30th Chemnitz Finite Element Symposium 2017, pp. 187–204. Springer International Publishing, Cham (2019)
9. Heinlein, A., Klawonn, A., Rheinbach, O., Röver, F.: A three-level extension of the gdsw overlapping schwarz preconditioner in three dimensions. In: Int. Conf. Domain Decompos. Methods, pp. 185–192. Springer (2018)
10. Heinlein, A., Rheinbach, O., Röver, F.: A multilevel extension of the gdsw overlapping schwarz preconditioner in two dimensions. *Comput. Methods Appl. Math.* **23**(4), 953–968 (2023).
11. Hochmuth, C.: Parallel overlapping schwarz preconditioners for incompressible fluid flow and fluid-structure interaction problems. Ph.D. thesis, Universität zu Köln (2020)
12. Köhler, S., Rheinbach, O.: Monolithic multi-level overlapping schwarz solvers for fluid problems (2025)
13. Meier-Yang, U., et al.: Boomeramg: a parallel algebraic multigrid solver and preconditioner. *Appl. Numer. Math.* **41**(1), 155–177 (2002)
14. Münster, R., Mierka, O., Turek, S.: Finite Element-Fictitious Boundary Methods (FEM-FBM) for 3d particulate flow. *Int. J. Numer. Methods Fluids* **69**(2), 294–313 (2012)
15. Röver, F.: Multi-level Extensions for the Fast and Robust Overlapping Schwarz Preconditioners. Ph.D. thesis, Technische Universität Bergakademie Freiberg (2023)
16. Turek, S., Wan, D., Rivkind, L.S.: The fictitious boundary method for the implicit treatment of dirichlet boundary conditions with applications to incompressible flow simulations. In: Challenges in Scientific Computing-CISC 2002: Proceedings of the Conference Challenges in Scientific Computing Berlin, October 2–5, 2002, pp. 37–68. Springer (2003)