IX International Conference on Computational Methods for Coupled Problems in Science and Engineering
COUPLED PROBLEMS 2021
E. Oñate, M. Papadrakakis and B. Schrefler (Eds)

PARTITIONED COUPLING VS. MONOLITHIC BLOCK-PRECONDITIONING APPROACHES FOR SOLVING STOKES-DARCY SYSTEMS

JENNY SCHMALFUSS[*⋆], CEDRIC RIETHMÜLLER[†⋆], MIRCO ALTENBERND[†], KILIAN WEISHAUPT[††] AND DOMINIK GÖDDEKE[†#]

[*]Institute for Visualization and Interactive Systems (VIS), University of Stuttgart
70569 Stuttgart, Germany. E-mail: jenny.schmalfuss@vis.uni-stuttgart.de

[†]Institute of Applied Analysis and Numerical Simulation (IANS), University of Stuttgart

[††]Institute for Modelling Hydraulic and Environmental Systems (IWS), University of Stuttgart

[#]Stuttgart Center for Simulation Science (SC SimTech), University of Stuttgart

Key words: time dependent Stokes-Darcy flow, iterative vs. direct methods, sub-solver optimization, partitioned coupling with preCICE

Abstract. We consider the time-dependent Stokes-Darcy problem as a model case for the challenges involved in solving coupled systems. Keeping the model, its discretization, and the underlying numerics for the subproblems in the free-flow domain and the porous medium constant, we focus on different solver approaches for the coupled problem. We compare a partitioned coupling approach using the coupling library preCICE with a monolithic block-preconditioned one that is tailored to different formulations of the problem. Both approaches enable the reuse of already available iterative solvers and preconditioners, in our case, from the DuMu^{x} framework. Our results indicate that the approaches can yield performance and scalability improvements compared to using direct solvers: Partitioned coupling is able to solve large problems faster if iterative solvers with suitable preconditioners are applied for the subproblems. The monolithic approach shows even stronger requirements on preconditioning, as standard simple solvers fail to converge. Our monolithic block preconditioning yields the fastest runtimes for large systems, but they vary strongly with the preconditioner configuration. Interestingly, using a specialized Uzawa preconditioner for the Stokes subsystem leads to overall increased runtimes compared to block preconditioners utilizing a more general algebraic multigrid. This highlights that optimizing for the non-coupled cases does not always pay off.

⋆ The first two authors contributed equally to this paper.

1 INTRODUCTION

Coupled systems of free flow adjacent to permeable media have a decisive role in many applications. Examples include the environmental sciences (soil water evaporation), medical contexts (intervascular exchange), material design (optimization of fuel cell water management) or technical applications (drying of perishable goods) to name just a few. Capturing the complex interplay between the two flow domains is essential, however, the governing systems of equations form a coupled problem which can become quite complex to solve. This even holds for single-phase-flow systems, such as a river flowing over its porous bed. In this paper, we deliberately restrict ourselves to a simple, stationary, single-phase-flow problem, i.e., creeping Stokes flow in the free-flow domain, while using Darcy's law for the porous domain. While limiting the physical complexity of our model, we focus on the numerical solution of the arising coupled system using either fully monolithic coupled schemes or a partitioned, iterative approach.

We build our contribution on the following observation: Practitioners, in particular in the modeling community, often rely on sparse direct solvers for the (linearized) subproblems, e.g., UMFPACK, PARDISO and SUPERLU, see [5] for an overview. This holds when Matlab's Backslash operator or its equivalent in SciPy are used, as they translate to one of these sparse direct solvers under the hood. Often this also applies to users of PDE software frameworks like DUMU$^{\text{X}}$ [12, 16], whose design is in fact aiming to minimize the users' burden of having to deal with every single aspect of the simulation pipeline.

Two issues in this context are often overlooked: First, sparse direct methods for the linear(ized) system(s) do not scale well in terms of compute time and memory. Second, the ill-conditioning of a fully assembled monolithic system can lead to severe trustworthiness issues in the solution. Both issues typically only appear after a model and its corresponding simulation pipeline have been set up, i.e., when test problems are exchanged for real-world scenarios. Table 1 exemplarily shows the fill-in factors in UMFPACK, when solving the monolithic variant of one of our model problems with the finite volume scheme described in Section 2. While the matrix density increases mostly linearly due to surface-to-volume arguments, the fill-in for the computed sparse LU decomposition is clearly nonlinear in terms of memory. Thus it translates to compute time for generating the decomposition, and subsequently to solving the linear system using the decomposition.

Table 1: Memory requirements for storing the sparse matrix and its decomposition measured as matrix entries per degree of freedom.

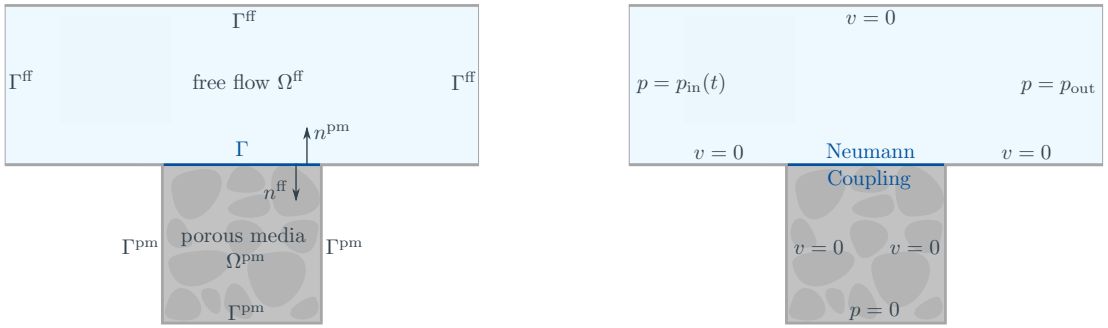
DoF	**156**	**1 056**	**10 100**	**102 720**	**1 001 000**
Discretization	6×6	16×16	50×50	160×160	500×500
System matrix	6.7	7.3	7.6	7.7	7.7
UMFPACK	19.9	38.6	78.6	145.7	241.4

In this paper, we demonstrate how carefully devised iterative and thus scalable solvers

can alleviate these issues for two different solution strategies for coupled problems: We consider both a partitioned coupling approach where the subproblems are solved alternately, and a monolithic approach that honors the saddle point structure of the system. The former is realized with the coupling library preCICE [6], while the latter is tailored to standard PDE frameworks like DuMu$^{\text{x}}$. An important part of our contribution is a thorough comparison of these fundamentally different approaches.

2 MODEL PROBLEM

We consider an instationary, coupled Stokes-Darcy two-domain problem. It comprises a free flow of an incompressible fluid over a porous medium, see Figure 1a.



(a) Coupled problem illustration. **(b)** Boundary conditions.

Figure 1: The instationary Stokes-Darcy problem and its boundary conditions.

We mark quantities that are associated with the free-flow domain with $^{\text{ff}}$, and use $^{\text{pm}}$ to denote the porous medium. The domains are denoted by Ω^{ff} and Ω^{pm}, and share the common boundary Γ. Boundaries that are not shared are called Γ^{ff} and Γ^{pm}. The normal vectors n^{ff} and n^{pm} are orthogonal to Γ and point outwards their respective domains. The time dependent quantities pressure p and velocity v are used to describe the flow in each domain. We use the transient and incompressible Stokes equations to model the free flow:

$$\frac{\partial v}{\partial t} + \nabla \cdot \left(-\nu \left(\nabla v^{\text{ff}} + \nabla v^{\text{ff,T}}\right) + \rho^{-1} p^{\text{ff}} I\right) = 0 \qquad \text{in } \Omega^{\text{ff}} \tag{1}$$

$$\nabla \cdot v^{\text{ff}} = 0 \qquad \text{in } \Omega^{\text{ff}} \tag{2}$$

Above, ρ and ν are the fluid density and kinematic viscosity, and I is a suitable identity map. In the porous medium, Darcy's law and the continuity equation are used:

$$v^{\text{pm}} = -K\mu^{-1}\nabla p^{\text{pm}} \qquad \text{in } \Omega^{\text{pm}} \tag{3}$$

$$\nabla \cdot v^{\text{pm}} = 0 \qquad \text{in } \Omega^{\text{pm}} \tag{4}$$

K is the intrinsic permeability of the porous medium and $\mu = \nu\rho$ the dynamic viscosity of the fluid. As coupling conditions [17], we use the continuity of the normal stresses (5), the

Beavers-Joseph-Saffman condition [21] in equation (6) and the continuity of the normal mass fluxes (7):

$$n \cdot [(pI - \tau)n]^{\mathrm{ff}} = [p]^{\mathrm{pm}} \qquad \text{on } \Gamma \tag{5}$$

$$[(v + \sqrt{K}(\alpha_{\mathrm{BJ}}\mu)^{-1}\tau n) \cdot t_{\mathrm{ff,pm}}]^{\mathrm{ff}} = 0 \qquad \text{on } \Gamma \tag{6}$$

$$[v \cdot n]^{\mathrm{pm}} = -[v \cdot n]^{\mathrm{ff}} \qquad \text{on } \Gamma \tag{7}$$

We use n for the normal of the respective flow component, τ for the viscous stresses and α_{BJ} is the Beavers-Joseph coefficient. Further, $t_{\mathrm{ff,pm}}$ is the basis of the tangent plane that describes the interface between Ω^{ff} and Ω^{pm}. To close this system, boundary conditions for the nonshared domain boundaries are illustrated in Figure 1b. Note that the pressure p_{in} on the left free-flow boundary changes over time.

The system of equations is discretized with a first-order backward Euler scheme in time, and finite volumes in space [16]. In the Darcy domain, a two-point flux approximation is used for the finite volume approximation of the pressure [13, Chap. 4]. In the Stokes domain, a staggered grid is used for the quantities pressure and velocity, and the fluxes are approximated with an upwind scheme [16, 22]. In summary, the discrete model, to be solved for every time step, has the form

$$Ax = b \quad = \quad \begin{pmatrix} A' & B' \\ C' & D' \end{pmatrix} \begin{pmatrix} x^{\mathrm{ff}} \\ p^{\mathrm{pm}} \end{pmatrix} = \begin{pmatrix} b^{\mathrm{ff}} \\ 0 \end{pmatrix} \tag{8}$$

$$= \quad \begin{pmatrix} \begin{bmatrix} V & B \\ C & 0 \end{bmatrix} & \begin{bmatrix} B'_1 \\ 0 \end{bmatrix} \\ \begin{bmatrix} C'_1 & 0 \end{bmatrix} & D' \end{pmatrix} \begin{pmatrix} \begin{bmatrix} v^{\mathrm{ff}} \\ p^{\mathrm{ff}} \end{bmatrix} \\ p^{\mathrm{pm}} \end{pmatrix} = \begin{pmatrix} \begin{bmatrix} g \\ 0 \end{bmatrix} \\ 0 \end{pmatrix}. \tag{9}$$

Formulation (8) is denoted as *two-domain* (td) formulation of the problem, because the matrix blocks correspond to the free-flow and porous-medium phase of the problem. Further, we dub equation (9) the *pressure-velocity* (pv) formulation, due to the correspondence of the matrix blocks to the variables pressure and velocity.

3 PARTITIONED COUPLING APPROACH

Partitioned coupling approaches are a common strategy to solve coupled problems. In our setting, this means that the flow fields in the two flow domains are calculated separately, and the coupling between the subdomains is ensured by exchanging information over the sharp interface Γ. The benefit of this approach is that existing, optimized solvers for the subdomains can be used. We rely on DuMu$^{\mathrm{x}}$ for the subdomain solvers, and preCICE for the coupling.

Looking at the two-domain formulation (8), it is clear that boundary conditions on the common interface Γ need to be exchanged in order to get a well-defined solution. For this, we use a serial implicit coupling technique [9] where the subdomain problems are solved sequentially and the boundary values for the other domain are written after the

Figure 2: Subdomain coupling scheme implemented with preCICE.

solution step is performed. The coupling procedure is depicted in Figure 2 and comprises a Dirichlet-Neumann coupling between the subdomains. We start the coupling by solving the free-flow problem, to determine Dirichlet pressure values on the interface Γ. The porous-medium-flow solver then determines Neumann velocity values on the interface. Thus, the pressure in the Darcy domain is fixed at the coupling interface Γ, and the Dirichlet-Neumann coupling leads to a well-defined solution. In more detail, let k be the coupling iteration index and $v_k^{\text{pm},\Gamma}$ the normal velocity at the interface Γ. The free-flow solver $\mathcal{S}^{\text{ff}}$ computes a new flow state, which leads to an updated pressure $p_{k+1}^{\text{ff},\Gamma}$ on the interface. With this updated pressure, the porous-medium-flow solver $\mathcal{S}^{\text{pm}}$ computes a new flow state, which then leads to an updated normal velocity $v_{k+1}^{\text{pm},\Gamma}$ on the interface. When we combine the two interface equations

$$\left.\begin{aligned} \mathcal{S}^{\text{ff}}\left(v_k^{\text{pm},\Gamma}\right) &= p_{k+1}^{\text{ff},\Gamma} \\ \mathcal{S}^{\text{pm}}\left(p_{k+1}^{\text{ff},\Gamma}\right) &= v_{k+1}^{\text{pm},\Gamma} \end{aligned}\right\} \quad \Leftrightarrow \quad \mathcal{S}^{\text{pm}}\left(\mathcal{S}^{\text{ff}}\left(v_k^{\text{pm},\Gamma}\right)\right) = v_{k+1}^{\text{pm},\Gamma}, \tag{10}$$

we can interpret the coupling scheme as an iterative solver for the fixed-point problem

$$\mathcal{S}^{\text{pm}}\left(\mathcal{S}^{\text{ff}}\left(v^{\text{pm},\Gamma}\right)\right) = v^{\text{pm},\Gamma} \quad \Leftrightarrow \quad R\left(v^{\text{pm},\Gamma}\right) := \mathcal{S}^{\text{pm}}\left(\mathcal{S}^{\text{ff}}\left(v^{\text{pm},\Gamma}\right)\right) - v^{\text{pm},\Gamma} = 0. \tag{11}$$

The scheme stops when the interface values converge, i.e., the fixed-point problem is solved to a prescribed accuracy. We emphasize that when the residual R is sufficiently close to zero, we recover the monolithic solution.

Solving the fixed-point problem (11) with a Picard fixed-point iteration is prone to divergence for problems with strong instabilities or oscillations. In order to improve stability and convergence speed, fixed-point acceleration methods enrich the Picard iteration. These methods are applied as a post-processing step that we denote as $\mathcal{I}^{\text{post}}$. Figure 3 illustrates our accelerated fixed-point iteration. Now, the flow update from the porous-medium solver is denoted by $\tilde{v}_{k+1}^{\text{pm},\Gamma}$ (previously: $v_{k+1}^{\text{pm},\Gamma}$), as it is the solution *before* the improvement by $\mathcal{I}^{\text{post}}$. The post-processing scheme receives $\tilde{v}_{k+1}^{\text{pm},\Gamma}$ from the porous-medium solver and computes an improved velocity $v_{k+1}^{\text{pm},\Gamma}$. This new velocity depends on the current value and a history of previously calculated values. For our experiments, we choose the inverse least-squares interface quasi-Newton method [10] for the post-processing. This method approximates the inverse Jacobian of the residual operator R of the nonlinear coupling

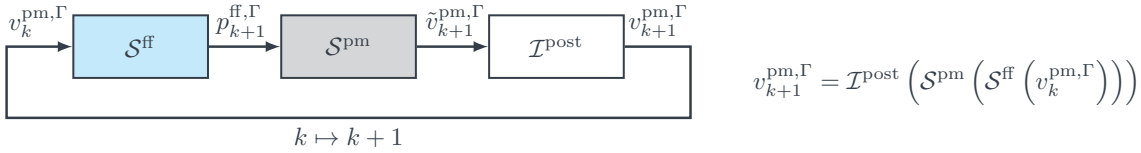


Figure 3: preCICE coupling scheme with enabled post-processing.

equation, based on input-output relations. Then, it performs Newton-like update steps where a norm minimization is carried out. For more details on post-processing schemes and their implementation, see [9] and [6] respectively. To determine when the iteration can be stopped, we use the relative convergence measures

$$\left\| p_{k+1}^{\mathrm{ff},\Gamma} - p_k^{\mathrm{ff},\Gamma} \right\|_2 < \varepsilon \left\| p_{k+1}^{\mathrm{ff},\Gamma} \right\|_2 \qquad \text{and} \qquad \left\| \tilde{v}_{k+1}^{\mathrm{pm},\Gamma} - v_k^{\mathrm{pm},\Gamma} \right\|_2 < \varepsilon \left\| \tilde{v}_{k+1}^{\mathrm{pm},\Gamma} \right\|_2 . \tag{12}$$

Our choice of post-processing method and convergence measure is based on [15]. There, it is shown that for a similar model problem, the coupling approach as outlined above is consistent and converges to the monolithic solution. This finding is the basis for our convergence study in Section 5. As solvers $\mathcal{S}^{\mathrm{ff}}$ and $\mathcal{S}^{\mathrm{pm}}$, we use problem specific preconditioned iterative subdomain solvers in order to benefit from their smaller memory footprint, which results in a better numerical scaling with respect to the problem size.

preCICE follows a pure library approach, and is called from within DuMu$^{\mathrm{x}}$ by the participating solvers, and not by the general framework. Through the library approach, the coupling is minimally invasive and provides a black-box functionality that allows us to use optimized solvers for the two domains.

4 MONOLITHIC BLOCK PRECONDITIONING

As an alternative to the partitioned approach, we consider a block-preconditioning strategy to iteratively solve the linear system in its entirety, allowing to consider both formulations (8) and (9) separately. The saddle point structure of A' implies one of the diagonal matrix blocks to be zero, which prevents the direct application of many preconditioning techniques like simple splitting based schemes [20, Chap. 10.2] or incomplete LU (ILU) preconditioning [14, 18]. To address this, we apply two types of block-preconditioning schemes that use exchangeable preconditioners for the respective matrix blocks. This allows to select the preconditioners for the matrix blocks based on structural or model-based properties of the block. The two considered block-preconditioning approaches are a *block-Jacobi* $\mathcal{P}_{\mathrm{BJ}}$ and a *block-Gauss-Seidel preconditioning scheme* $\mathcal{P}_{\mathrm{BGS}}$, which themselves are formulated to regard the linear system either as the 2×2 block matrix (8) or as the 3×3 block matrix (9).

To construct the $\mathcal{P}_{\mathrm{BJ}}$ preconditioners, we consider the block diagonal of the matrices (8) and (9). Additionally incorporating all block lower triangular parts is the basis for the $\mathcal{P}_{\mathrm{BGS}}$ preconditioners. Block-'inverting' those reduced block matrices yields the block

preconditioners. The procedure to acquire the $\mathcal{P}_{\mathrm{BGS}}$ preconditioner is similar in spirit to [7]. There are two points to note: Firstly, an exact block inversion requires the exact inverses of the diagonal blocks, which is infeasible for preconditioners. We thus replace the inverses of the blocks A', D' and V with preconditioners for these blocks that approximate the action of the exact inverses on the quantities of interest. We denote these block specific preconditioners by $\mathcal{P}_{A'}$, $\mathcal{P}_{D'}$ and $\mathcal{P}_V$. Secondly, we formally replace the zero block on the diagonal of the reduced matrix (9) with an identity matrix, preventing the 'inversion' of the zero block. In our implementation, this means that no preconditioner is applied to this block. We thus obtain general $\mathcal{P}_{\mathrm{BJ}}$ and $\mathcal{P}_{\mathrm{BGS}}$ preconditioner formulations.

In the implementation, a variety of concrete preconditioners for the respective matrix blocks can be used. For the two-domain formulation (8), this yields the *two-domain block-Jacobi preconditioner* $\mathcal{P}^{\mathrm{td}}_{\mathrm{BJ}}(\mathcal{P}_{A'}, \mathcal{P}_{D'})$ and the *two-domain block-Gauss-Seidel preconditioner* $\mathcal{P}^{\mathrm{td}}_{\mathrm{BGS}}(\mathcal{P}_{A'}, \mathcal{P}_{D'})$. Both depend on suitable preconditioners $\mathcal{P}_{A'}$ and $\mathcal{P}_{D'}$ for the blocks A' and D':

$$\mathcal{P}^{\mathrm{td}}_{\mathrm{BJ}}(\mathcal{P}_{A'}, \mathcal{P}_{D'}) \coloneqq \begin{pmatrix} \mathcal{P}_{A'} & 0 \\ 0 & \mathcal{P}_{D'} \end{pmatrix}, \qquad \mathcal{P}^{\mathrm{td}}_{\mathrm{BGS}}(\mathcal{P}_{A'}, \mathcal{P}_{D'}) \coloneqq \begin{pmatrix} \mathcal{P}_{A'} & 0 \\ -\mathcal{P}_{D'} C' \mathcal{P}_{A'} & \mathcal{P}_{D'} \end{pmatrix} \tag{13}$$

Likewise, we obtain the *pressure-velocity block-Jacobi preconditioner* $\mathcal{P}^{\mathrm{pv}}_{\mathrm{BJ}}(\mathcal{P}_V, \mathcal{P}_{D'})$ and the *pressure-velocity block-Gauss-Seidel preconditioner* $\mathcal{P}^{\mathrm{pv}}_{\mathrm{BGS}}(\mathcal{P}_V, \mathcal{P}_{D'})$. They depend on preconditioners $\mathcal{P}_V$ and $\mathcal{P}_{D'}$ for the blocks V and D':

$$\mathcal{P}^{\mathrm{pv}}_{\mathrm{BJ}}(\mathcal{P}_V, \mathcal{P}_{D'}) \coloneqq \begin{pmatrix} \mathcal{P}_V & 0 & 0 \\ 0 & \mathrm{I} & 0 \\ 0 & 0 & \mathcal{P}_{D'} \end{pmatrix}, \quad \mathcal{P}^{\mathrm{pv}}_{\mathrm{BGS}}(\mathcal{P}_V, \mathcal{P}_{D'}) \coloneqq \begin{pmatrix} \mathcal{P}_V & 0 & 0 \\ -\mathrm{I}C\mathcal{P}_V & \mathrm{I} & 0 \\ -\mathcal{P}_{D'} C_1' \mathcal{P}_V & 0 & \mathcal{P}_{D'} \end{pmatrix} \tag{14}$$

The differences between the four variants directly influence the possible choices of the sub-preconditioners $\mathcal{P}_{A'}$, $\mathcal{P}_{D'}$ and $\mathcal{P}_V$, as well as the computational efficiency of the whole block preconditioner. We begin with the difference between the two-domain $\mathcal{P}^{\mathrm{td}}_{*}$ and pressure-velocity $\mathcal{P}^{\mathrm{pv}}_{*}$ preconditioner formulations. The two-domain block preconditioners treat the saddle point structure of A' as one block. This permits specialized saddle point preconditioning techniques $\mathcal{P}_{A'}$, like an Uzawa preconditioner [11], which may lead to better results due to their construction for the specific structure. The pressure-velocity formulation explicitly treats the saddle point structure by introducing an identity preconditioner on the critical diagonal block. This allows using a wider range of preconditioning techniques for $\mathcal{P}_V$ and $\mathcal{P}_{D'}$, like splitting techniques, ILU(p) or algebraic multigrid (AMG) [2, 4] preconditioners. Also, this weakens the prerequisites on the preconditioners and allows using techniques that are not specialized for the problem's structure.

Comparing the $\mathcal{P}^{*}_{\mathrm{BJ}}$ and $\mathcal{P}^{*}_{\mathrm{BGS}}$ preconditioners, the sparsity pattern of the block-Jacobi matrices suggests that their application requires fewer computational steps. The Gauss-Seidel preconditioners use additional coupling entries below the diagonal. Intuitively, one may expect an improved conditioning with a formulation that makes use of such additional information. However, this comes at the cost of more computation in the preconditioner

application and setup. Without numerical experiments it is unclear whether the envisioned improvement of the system's condition leads to shorter solution times compared to a preconditioner that is cheaper in its application but less capable to improve the systems condition number.

We highlight that preconditioners are commonly implemented as their application to vectors, i.e., for a vector x, it is of the form $\mathcal{P}(x)$. If such an implementation is already given, applying the block preconditioners to a blocked vector is a simple block matrix-vector product. This allows using existing preconditioner implementations within the blocked preconditioner. We use the ones available in DuMu$^{\text{x}}$ for our experiments.

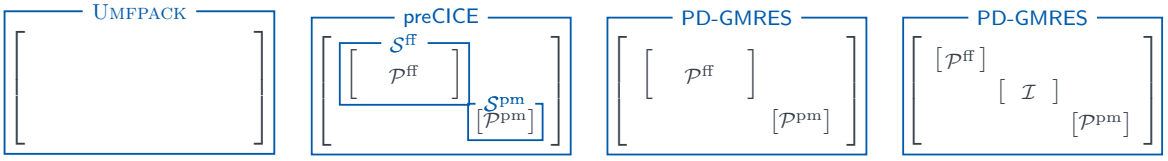
5 COMPARISON AND RESULTS

We now assess the iterative solution of a coupled Stokes-Darcy system with partitioned coupling and block preconditioning, compared to the direct solver UMFPACK [8]. We compare the runtime for solving increasingly large systems, and also comment on the memory requirement for all approaches. In the model problem from Section 2, we set $K = 10^{-6}\text{m}^2$ and $\alpha_{\text{BJ}} = 1.0$. The simulation is stopped at $t_{\text{end}} = 50 \cdot 10^5\text{s}$, with a time step size of $\text{d}t = 2 \cdot 10^5\text{s}$. A time dependent pressure difference is applied between the left and right boundary, which changes in the form of a half cosine-wave, with a maximum difference of 10^{-9}Pa. The Stokes domain is a 1×3 rectangle over a 1×1 square for the Darcy domain, see also Figure 1. Each 1×1 square uses the same number of spatial cells.

As solvers, we either use UMFPACK, or preconditioned versions of the iterative solvers PD-GMRES [19] or Bi-CGSTAB [23]. As preconditioners we use an AMG method [2, 4], Uzawa-iterations [11] or an ILU(0) factorization [14, 18]. PD-GMRES uses $m_{\text{init}} = m_{\text{min}} = 3$ and $m_{\text{step}} = 5$, other parameters are chosen according to the original publication. DUNE-ISTL's non-smooth aggregation AMG is used as solver or preconditioner, and performs one V-cycle [3, 4]. Pre- and post-smoothing is a single Gauss-Seidel iteration each with $\omega = 1$. In our setup, we restrict ourselves to UMFPACK as coarse grid solver, and limit the hierarchy to 3 levels, which results in comparatively large coarse grid problems for the velocity blocks at scale. Our standard Uzawa configuration is an inexact Uzawa that executes one Richardson iteration where the optimal relaxation parameter ω_{opt} is estimated via power iteration. In the inexact case, the AMG method as specified above is used as solver, while UMFPACK is used for the exact Uzawa iteration (Uzawa$_e$). All simulations are run on a single core of an AMD EPYC 7551P CPU with 2.0 GHz.

To assess the runtime scaling of our different approaches, we increase the number of degrees of freedom. As baseline, we solve the linear system (9) with the direct solver UMFPACK. The evaluations tested for the two iterative schemes are listed in Table 2, and schematically illustrated in Figure 4. The iterative methods are stopped when the residual's norm is in the same order of magnitude as the UMFPACK's residual.

Figure 5 shows the measured runtime scaling behavior. To allow a comparison between the approaches, we choose the preconditioners for the subsystems to be either AMG or Uzawa. We observe that using iterative methods pays off in terms of runtime already for



(a) UMFPACK **(b)** preCICE **(c)** Preconditioner $\mathcal{P}^{\mathrm{td}}$ **(d)** Preconditioner $\mathcal{P}^{\mathrm{pv}}$

Figure 4: Evaluation setup conceptual visualization. Solvers are marked in blue around the matrix block they are applied to, preconditioners $\mathcal{P}$ are marked within their block.

Table 2: Evaluation setup comparison. Cells marked with *n.a.* are not applicable.

Name	Iterative	Monolithic	Solver			Preconditioner		
			System	Stokes	Darcy	System	Stokes	Darcy
UMFPACK	✗	✓	UMFPACK	*n.a.*	*n.a.*	*n.a.*	*n.a.*	*n.a.*
preCICE UMFPACK	✓	✗	preCICE	UMFPACK	UMFPACK	*n.a.*	*n.a.*	*n.a.*
preCICE $\mathcal{P}$(Uzawa$_e$, AMG)	✓	✗	preCICE	PD-GMRES	Bi-CGSTAB	*n.a.*	Uzawa-exact	AMG
preCICE $\mathcal{P}$(Uzawa, AMG)	✓	✗	preCICE	PD-GMRES	Bi-CGSTAB	*n.a.*	Uzawa	AMG
PD-GMRES $\mathcal{P}^{\mathrm{pv}}_{\mathrm{BJ}}$(AMG, AMG)	✓	✓	PD-GMRES	*n.a.*	*n.a.*	B-Jacobi	AMG	AMG
PD-GMRES $\mathcal{P}^{\mathrm{pv}}_{\mathrm{BGS}}$(AMG, AMG)	✓	✓	PD-GMRES	*n.a.*	*n.a.*	B-Gauss-Seidel	AMG	AMG
PD-GMRES $\mathcal{P}^{\mathrm{td}}_{\mathrm{BJ}}$(Uzawa, AMG)	✓	✓	PD-GMRES	*n.a.*	*n.a.*	B-Jacobi	Uzawa	AMG
PD-GMRES $\mathcal{P}^{\mathrm{td}}_{\mathrm{BGS}}$(Uzawa, AMG)	✓	✓	PD-GMRES	*n.a.*	*n.a.*	B-Gauss-Seidel	Uzawa	AMG
PD-GMRES $\mathcal{P}^{\mathrm{td}}_{\mathrm{BJ}}$(Uzawa, ILU(0))	✓	✓	PD-GMRES	*n.a.*	*n.a.*	B-Jacobi	Uzawa	ILU(0)
PD-GMRES $\mathcal{P}^{\mathrm{td}}_{\mathrm{BGS}}$(Uzawa, ILU(0))	✓	✓	PD-GMRES	*n.a.*	*n.a.*	B-Gauss-Seidel	Uzawa	ILU(0)

moderate problem sizes with 10^4 degrees of freedom, benefiting from their better numerical scaling with respect to the problem size n. Partitioned coupling with preconditioned iterative solvers is able to outperform UMFPACK for large n, while using the partitioned coupling approach with UMFPACK for both subsystems is not beneficial and always slower than directly applying UMFPACK to the monolithic system. The performance of our block-preconditioning approach yields the fastest runtimes for large systems, but varies strongly with the preconditioner configuration. Interestingly, we observe that using the specialized Uzawa preconditioner for Stokes in $\mathcal{P}^{\mathrm{td}}_{*}$ leads to increased runtimes compared to the less specialized $\mathcal{P}^{\mathrm{pv}}_{*}$ block preconditioner with two AMG preconditioners. In general, the $\mathcal{P}^{*}_{\mathrm{BJ}}$ configurations lead to slightly improved runtimes compared to the corresponding $\mathcal{P}^{*}_{\mathrm{BGS}}$ preconditioners.

In Figure 6 we show that tweaking the preconditioner configurations has the potential to further speed up the runtime, especially for the block-preconditioning approaches. While UMFPACK scales roughly as $O(n \cdot \log(n))$, our partitioned and block-preconditioned approaches suggest a linear runtime increase with respect to the problem size n. In general, this behavior is also expected for the $\mathcal{P}^{\mathrm{pv}}_{*}$(AMG, AMG) approaches but due to our setup we see an increase in runtime to a level similar to the UMFPACK setting. This is caused by our restriction to use UMFPACK as coarse grid solver in the preconditioner, and limiting the multigrid hierarchy to 3 levels: The resulting comparatively large coarse grid problems for the velocity blocks start to dominate the overall runtime. Increasing the multigrid hierarchy and/or switching to more efficient iterative solver for the

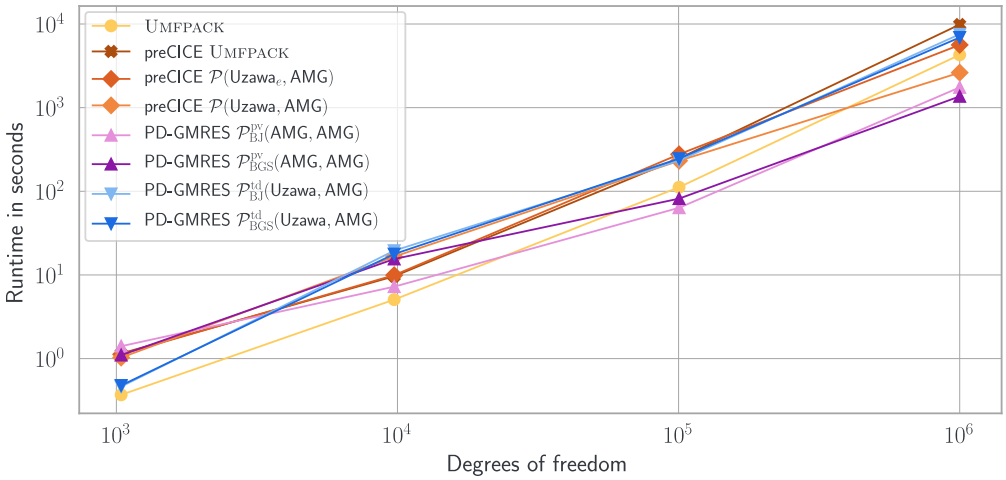


Figure 5: Runtime comparison of the direct solver UMFPACK and iterative solvers with partitioned coupling and block preconditioning.

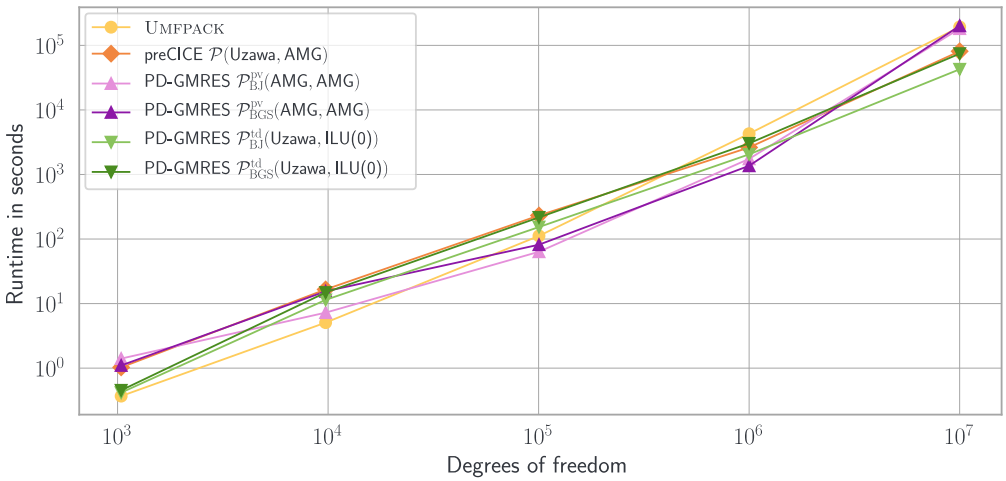


Figure 6: Runtime comparison of the best performing solver configurations.

coarse grid problems is expected to reduce the runtimes to or below the level of our $\mathcal{P}^{\text{td}}_{*}(\text{Uzawa}, \text{ILU}(0))$ approaches.

In terms of memory requirements both considered approaches are very similar when configured to use the same iterative solver(s) and preconditioners. Then, the memory consumption is dominated by the auxiliary vectors used by the iterative solvers to solve the linear system - in parts or as a whole. If limited memory is an issue, the partitioned coupling approach has the advantage to solve one subsystem at a time, requiring only the memory for solving the current subsystem.

6 CONCLUSION AND FUTURE WORK

Our experiments clearly indicate that both partitioned coupling and block-preconditioning approaches yield superior performance compared to using a sparse direct solver. This holds already for moderate problems sizes in single-threaded computations, and we expect the benefits to be substantially larger in parallel computations. Our implementation in DuMu^x is very general, and can in principle also be applied for the nonlinear Navier-Stokes case, or for coupled flows involving more physics. Initial experiments show that our sophisticated coupling/preconditioning techniques are then obligatory, as simple monolithic iterative schemes fail due to the severe ill-conditioning.

7 ACKNOWLEDGEMENTS

This work was financially supported by the German Research Foundation (DFG), within the Collaborative Research Center on Interface-Driven Multi-Field Processes in Porous Media (SFB 1313, Project Number 327154368).

References

[2] Bastian, P., Blatt, M., and Scheichl, R. "Algebraic multigrid for discontinuous Galerkin discretizations of heterogeneous elliptic problems". *Numerical Linear Algebra with Applications*. (2012) **19**(2):367–388.

[3] Blatt, M. and Bastian, P. "The Iterative Solver Template Library". *International Workshop on Applied Parallel Computing*. Springer. (2006) 666–675.

[4] Blatt, M., Ippisch, O., and Bastian, P. *A Massively Parallel Algebraic Multigrid Preconditioner Based on Aggregation for Elliptic Problems with Heterogeneous Coefficients*. Tech. rep. (arXiv:1209.0960). arXiv, (2012).

[5] Bollhöfer, M., Schenk, O., Janalík, R., Hamm, S., and Gullapalli, K. *State-of-the-Art Sparse Direct Solvers*. Tech. rep. (arXiv:1907.05309). arXiv, (2019).

[6] Bungartz, H.-J., Lindner, F., Gatzhammer, B., Mehl, M., Scheufele, K., Shukaev, A., and Uekermann, B. "preCICE – A fully parallel library for multi-physics surface coupling". *Computers and Fluids*. (2016) **141**:250–258.

[7] Cai, M., Mu, M., and Xu, J. "Preconditioning Techniques for a Mixed Stokes/Darcy Model in Porous Media Applications". *Journal of Computational and Applied Mathematics*. (2009) **233**(2):346–355.

[8] Davis, T. A. "UMFPACK User Guide". *Website: http://www.suitesparse.com*. (2018).

[9] Degroote, J. "Partitioned Simulation of Fluid-Structure Interaction". *Archives of Computational Methods in Engineering*. (2013) **20**:185–238.

[10] Degroote, J., Bathe, K.-J., and Vierendeels, J. "Performance of a new partitioned procedure versus a monolithic procedure in fluid-structure interaction". *Computers & Structures*. (2009) **87**(11):793–801.

[11] Elman, H. C. and Golub, G. H. "Inexact and Preconditioned Uzawa Algorithms for Saddle Point Problems". *SIAM Journal on Numerical Analysis*. (1994) **31**(6):1645–1661.

[12] Flemisch, B., Darcis, M., Erbertseder, K., Faigle, B., Lauser, A., Mosthaf, K., Müthing, S., Nuske, P., Tatomir, A., Wolff, M., and Helmig, R. "DuMux: DUNE for Multi-{Phase, Component, Scale, Physics, ...} Flow and Transport in Porous Media". *Advances in Water Resources*. (2011) **34**(9):1102–1112.

[13] Grüninger, C. *Numerical Coupling of Navier–Stokes and Darcy Flow for Soil-Water Evaporation*. Eigenverlag des Instituts für Wasser- und Umweltsystemmodellierung der Universität Stuttgart, (2017).

[14] Hysom, D. and Pothen, A. "Level-Based Incomplete LU Factorization: Graph Model and Algorithms". *SIAM Journal on Matrix Analysis and Applications*. (2002).

[15] Jaust, A., Weishaupt, K., Mehl, M., and Flemisch, B. "Partitioned Coupling Schemes for Free-Flow and Porous-Media Applications with Sharp Interfaces". *Finite Volumes for Complex Applications IX - Methods, Theoretical Aspects, Examples*. Springer, (2020) 605–613.

[16] Koch, T., Gläser, D., Weishaupt, K., Ackermann, S., Beck, M., Becker, B., Burbulla, S., Class, H., Coltman, E., Emmert, S., et al. "DuMux 3—An Open-Source Simulator for Solving Flow and Transport Problems in Porous Media with a Focus on Model Coupling". *Computers & Mathematics with Applications*. (2021) **81**:423–443.

[17] Layton, W. J., Schieweck, F., and Yotov, I. "Coupling Fluid Flow with Porous Media Flow". *SIAM Journal on Numerical Analysis*. (2002) **40**(6):2195–2218.

[18] Meijerink, J. A. and Vorst, H. A. van der. "An Iterative Solution Method for Linear Systems of Which the Coefficient Matrix Is a Symmetric M-matrix". *Mathematics of Computation*. (1977) **31**:148–162.

[19] Núñez, R. C., Schaerer, C. E., and Bhaya, A. "A Proportional-Derivative Control Strategy for Restarting the GMRES(m) Algorithm". *Journal of Computational and Applied Mathematics*. (2018) **337**:209–224.

[20] Saad, Y. *Iterative Methods for Sparse Linear Systems*. Second. SIAM, (2003).

[21] Saffman, P. G. "On the boundary condition at the surface of a porous medium". *Studies in Applied Mathematics*. (1971) **50**(2):93–101.

[22] Schneider, M., Weishaupt, K., Gläser, D., Boon, W. M., and Helmig, R. "Coupling Staggered-Grid and MPFA Finite Volume Methods for Free Flow/Porous-Medium Flow Problems". *Journal of Computational Physics*. (2020) **401**:109012.

[23] Vorst, H. A. van der. "Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems". *SIAM Journal on scientific and Statistical Computing*. (1992) **13**(2):631–644.