No. 687 **May 2026**

On the Automation and Scaling of a Fictitious-Boundary Method for Non-Newtonian Extrusion Die Flows

M. Esser, S. Turek

ISSN: 2190-1767

On the Automation and Scaling of a Fictitious-Boundary Method for Non-Newtonian Extrusion Die Flows

Maximilian Esser[0009−0004−6009−7497] and
Stefan Turek[0000−0002−9740−6087]

Abstract We present a fast, robust and scalable solution approach for implicitly captured domains in the context of extrusion dies for highly viscous fluids, applicable for automated simulation pipelines for industrial relevant 3D problems. We also show the applicability and the challenges for the transition to multi-node GPU systems.

1 Overview

HPC simulations are an important part for the efficient development and design process of tools and machinery in a wide range of industrial applications. On one hand the complexity of real world problems combined with short development cycles warrant large computational requirements, on the other hand HPC simulation tools and systems typically require expert knowledge to facilitate a time and money efficient use. This discrepancy even grows in light of the HPC development in the past years, increasing compute power through higher number of compute cores as well as accelerator hardware, requiring scalable parallelized approaches. The idea to enable non-experts in HPC simulations to use these systems is behind the Strömungsraum® cloudservice from IANUS Simulation [1], providing an easy to use web interface, dispatching HPC simulations to suitable compute centers and providing a congested report of useful results back. We worked as part of the "StroemungsRaum" project on improving the time to solution of this interface.

Maximilian Esser
Department of Mathematics, TU Dortmund University, Vogelpothsweg 87, 44227 Dortmund, e-mail: maximilian.esser@tu-dortmund.de

Stefan Turek
Department of Mathematics, TU Dortmund University, Vogelpothsweg 87, 44227 Dortmund, e-mail: stefan.turek@mathematik.tu-dortmund.de

[1] https://ianus-simulation.de/stroemungsraum

The essential part for the automation is the abstraction of the input layer and the simulation layer by allowing the user to define the problem in his design space and providing an automated and robust way to extract the actual simulation space from this. While this abstraction is problem dependent, for the case of tool construction, like extrusion dies which are the example case for this article, the design space relies on a surface parametrization (for example in form of CAD files) describing the actual fluid domain only implicitly, as well as material parameters and typical operating points. For the simulation layer, this has to be transformed into the governing equations and typically some form of volume mesh or parametrization of sufficient quality to resolute the fluid domain while keeping the compute cost to a feasible level. To provide a useful automation pipeline, this gap has to be overcome in such a reliable manner, that human intervention is the exception.

Besides (semi-)automated meshing tools [1], [2], which have the disadvantage of typically generating very large base meshes for complex geometries due to mesh quality constraints while also relying on precise definitions of geometric features, there is a wide range of fictitious domain approaches that enable simulations on simpler or even carthesian meshes. In the context of finite elements, these are mainly based on cut cell methods [3], unfitted Nietsche methods [4] as well as Lagrange multiplier based methods [5], which balance ease of implementation, order of convergence and stability of the underlying solvers against each other. These methods typically introduce some form of additional stabilization terms often leading to issues for fast iterative solvers, which are required for large case simulations of stiff problems considered here. Additionally, due to complexity of the domain, a reduced regularity of the solution is to be expected, making higher degree polynomial approaches undesirable and warranting at least some form of local mesh adaptivity.

For these reason, our approach is based on a fictitious boundary method [6], [7], [8] in combination with automatically generated roughly adapted coarse grid meshes, allowing us to facilitate fast and scalable solvers in the form of a hierarchical geometric multigrid approach. Due to the lower convergence order, this approach leads to generally larger volume meshes, which is compensated by the improved solver efficiency, generality of the method and the very low requirement on the quality of the surface description.

To further improve the time to solution we facilitate hardware resources in form of multinode GPU clusters which constitute a steadily increasing share of compute resources in recent HPC systems. While adapting to these systems requires extra care on the numerical methods and their implementation, it also provides a huge opportunity to speed up simulation times due to the increased computational resources.

The presented method is implemented in the open source Finite Element Analysis Toolkit 3 (FEAT3) [2] developed at TU Dortmund University.

The article begins with a brief description of the simulation pipeline and defining the problem to be solved. After this, we describe our numerical method and show in section 4 test results for a test problem as well as scaling results on a GPU cluster. Finally we end the article with a short conclusion.

[2] https://github.com/tudo-math-ls3/feat3

2 Simulation Pipeline for Extrusion Die Flows

Given a domain $\Omega \subset \mathbb{R}^d$, $d = 3$ described by a surface parametrization $\Gamma = \partial\Omega$, we want to find the solution $\mathbf{u} : \Omega \to \mathbb{R}^d$, $p : \Omega \to \mathbb{R}$ to the steady incompressible Navier-Stokes equations with a shear-rate dependent consecutive law [9]

$$(\mathbf{u} \cdot \nabla)\,\mathbf{u} - \nabla \cdot \sigma\,(\mathbf{u}, p) = \mathbf{f} \qquad \text{in } \Omega, \tag{1}$$

$$\nabla \cdot \mathbf{u} = 0 \qquad \text{in } \Omega, \tag{2}$$

$$\mathbf{u} = \mathbf{u}_{\text{in}} \text{ on } \Gamma_{\text{in}}, \quad \mathbf{u} = 0 \text{ on } \Gamma_{\text{wall}}, \tag{3}$$

$$(\mathbf{D}(\mathbf{u}) - p\mathbf{I}) \cdot \mathbf{n} = 0 \qquad \text{on } \Gamma_{\text{out}}, \tag{4}$$

$$\sigma\,(\mathbf{u}, p) = 2\nu(\dot{\gamma})\mathbf{D}(\mathbf{u}) - p\mathbf{I}, \tag{5}$$

with $\mathbf{D}(\mathbf{u}) = \frac{1}{2}(\nabla\mathbf{u} + \nabla\mathbf{u}^T)$ the symmetric deformation tensor, $\dot{\gamma} = \sqrt{2\mathbf{D}(\mathbf{u}) : \mathbf{D}(\mathbf{u})}$ the shear-rate, $\nu(\dot{\gamma})$ the shear-rate dependent viscosity, $\mathbf{f}$ a right hand side and Γ_{in}, Γ_{out}, $\Gamma_{\text{wall}} \subset \Gamma$ respectively the inflow boundary with a fixed inflow $\mathbf{u}_{\text{in}}$, the wall boundary with no-slip condition and outflow boundary with a force free outflow in normal direction. For the scope of this article, assume $\nu \in C^1$, $\nu > 0$ and $\int_{\mathbb{R}_+} \nu^2 < \infty$ for a well posed problem, but in praxis it might be necessary to clamp the shear-rates for particular materials or have more restrictions on the growth of the viscosity [10]. Furthermore, to simplify the scope of this work, assume $\nu >> 1$, which is a sound assumption for laminar extrusion die flows. Since the rheology is dominated by the shear-rate depending viscosity, we can safely drop the convective term. Note it is straightforward to reincorporate the convection back into our approach.

Let $\Omega_b \supseteq \Omega$ be the minimal fitting bounding box of the fluid domain. For the simulation pipeline to work, assume the outflow to be axis aligned as well as the inflow and outflow boundaries lying on the faces of Ω_b, i.e. $\Gamma_{\text{in}}, \Gamma_{\text{out}} \subset \partial\Omega_b$. This is generally not a restriction for extrusion dies, since for physical setups inflow and outflow parametrizations can easily be extended such that above condition holds. For multiple or curved outflow planes additional care for the outflow condition has to be considered, which we will ignore for the scope of this article.

By providing a parametrization for the inflow condition $\mathbf{u}_{\text{in}}$ and an extruded surface parametrization Γ_{ext}, such that inflow and outflow are shifted outside of Ω_b in normal direction, only the no-slip condition has to be implicitly prescribed on the background domain Ω_b.

The actual pipeline then consists of three phases.

1. **Preprocessing**. Through a REST API we load all necessary input and generate an triangulation of the extruded surface Γ_{ext}. For the given boundingbox Ω_b we also construct a simple hexahedral background coarse mesh T_H, compare figure 1 based on conformal refinement templates [11], [12]. The choice of starting mesh as well as refinement prediction is based on a combination of the required min gap resolution and the amount of surface Γ_{ext} inside of the mesh elements. The mesh T_H is constructed in such a way, that $k > 0$ additional uniform refinements lead to the desired resolution of the smallest flow channels of Ω, $h, H > 0$ denoting the fine, resp. coarse, grid resolution of our mesh. This construction also provides the

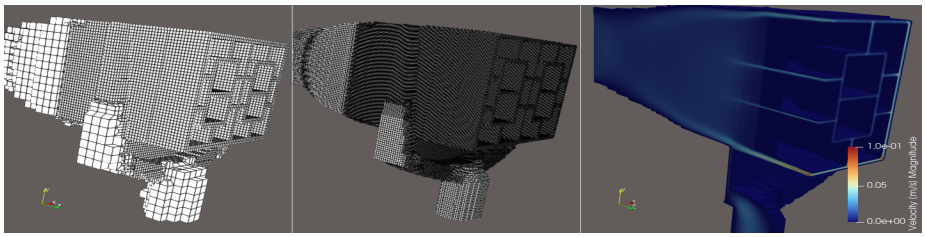

Fig. 1 Illustration of Scalexa benchmark geometry. Left: Large coarse grid with 69.000 elements. Middle: Level 2 refinement with 2 million elements. Right: Example velocity distribution.

hierarchy for a geometric multigrid solver with $k + 1$ levels. The number of levels is essential for the stability of the method: Too small generally leads to large coarse grid problems and therefore an inefficient solver, while too many levels lead to under resolved coarse-grids and a degradation in the stability of the linear solver. In our experience $k = 2$ works reliably for most problems. Note that all of this work is not computationally demanding and should be done in a separate step on a reduced amount of nodes compared to the following simulation step.

2. **Simulation**. Solve (1) as described in section 3 and write out the solution as vertex data on T_h.
3. **Postprocessing**. Use the generated solution to calculate and visualize flow distributions, shear stresses and so on, which is finally sent back to the Strömunsgraum® interface. Again, since this part is not computationally demanding, this step is done afterwards as a separated generalized task.

While this pipeline is able to generate results in about a day with a currently deployed pressure projection based solver [13], from an industry perspective the reduction of the overall time required for the pipeline to under one hour is the next large stepping stone. Accomplishing this speed-up was one of the main tasks of the "StroemungsRaum" project in which a large part of the developments we present here was done.

3 A filtering based fictitious boundary method

In accordance to a phase-field description of the Stokes-flow, also widely used in the context of topology optimization of fluid domains [14], [15], we split our domain into inside and outside regime, deriving a weak form of (1) on the extended domain Ω_b: Find $\mathbf{u} \in H_0^1(\Omega_b)^d$, $p \in L^2(\Omega_b)$ such that

$$\int_{\Omega_b} \mathbb{1}_\Omega \sigma(\mathbf{u}, p) : \nabla\varphi + (1 - \mathbb{1}_\Omega)\mathbf{u} \cdot \varphi = \int_{\Omega_b} \mathbb{1}_\Omega \mathbf{f} \cdot \varphi \qquad \forall \varphi \in H_0^1(\Omega_b)^d, \tag{6}$$

$$\int_{\Omega_b} \psi \nabla \cdot \mathbf{u} = 0 \; \forall \psi \in L^2(\Omega_b), \qquad \mathbf{u} = \mathbf{u}_{\text{in}} \text{ on } \Gamma_{\text{in}}, \; \mathbf{u} = 0 \text{ on } \partial\Omega \setminus (\Gamma_{\text{in}} \cup \Gamma_{\text{out}}). \tag{7}$$

Here $H_0^1(\Omega_b)^d$ is the vector-valued Sobolev space with zero trace on the Dirichlet boundary $\delta\Omega_b \setminus \Gamma_{\text{out}}$ and $\mathbb{1}_\Omega$ the indicator function marking Ω. Since our interface conditions are still exact, $u|_\Omega$ is a weak solution to (1).

While there are numerous approaches to capture the interface condition within a finite element approach, order preserving methods generally require either additional degrees of freedom for the interface problem or some form of jump-stabilization terms (or even both). In the view of automation, methods that rely on a good choice for the stabilization parameters are less preferable and in the context of multigrid solvers, large local modifications of the discretized system can lead to problems for cheap smoothers like Block-Jacobi type preconditioners.

By relaxing the constraint of exactly matching the no-slip boundary, we propose a fictitious boundary method which works with a standard finite element approach suitable for fast multigrid solvers based on [8]. While the base idea relies on the implicit distinction between the two domains through the fluid parameter, as shown in [6] it is preferable to implement the interface condition (7) in an explicit manner through an iterative filtering technique described in the following section.

3.1 Fictitious Boundary Finite Element Discretization

Let $T_h = \{T_{h,1}, \dots, T_{h,N_h}\}, N_h \in \mathbb{N}$ the hexahedral elements of the background mesh with $\bigcup_i T_{h,i} = \overline{\Omega_b}$. Define the set $T_h^o = \{T \in T_h | T \cap \Omega \neq \emptyset\}$ of all mesh cells intersecting the actual domain, prescribing the culled domain $\Omega_h := (\bigcup T_h^o)^{\text{o}}$. For $u, w, z \in H^1(\Omega_h)^d$ and $q \in L^2(\Omega_h)$ define the (non-linear) operators

$$A_h(v, w, z) := \int_{\Omega_h} 2\nu(\dot{\gamma}(v))\mathbf{D}(w) : \nabla z, \quad B_h(z, q) := \int_{\Omega_h} q\nabla \cdot z, \quad \langle f, z\rangle = \int_{\Omega_h} f \cdot z \tag{8}$$

Additionally denote F_h^o as the set of all inner and outer faces of T_h^o.

The fictitious boundary is then defined as the union of all outer boundary facets of F_h^o having zero measure intersections with the inflow or outflow boundary

$$\Gamma_{h,\text{fict}} := \bigcup \{F \in F_h^o | F \cap \partial\Omega_h \neq \emptyset \wedge |F \cap (\Gamma_{\text{in}} \cup \Gamma_{\text{out}})| = 0\}. \tag{9}$$

Additionally we define the extended inflow and outflow boundaries as

$$\Gamma_{h,\text{in}} := \bigcup \{F \in F_h^o | |F \cap \partial\Gamma_{\text{in}}| > 0\}, \quad \Gamma_{h,\text{out}} := \bigcup \{F \in F_h^o | |F \cap \partial\Gamma_{\text{out}}| > 0\}. \tag{10}$$

To discretize our system with a standard mixed formulation, we use the well-known inf-sup stable finite element pair Q^2/P^1_{disc} on T_h^o of continuos piecewise tri-quadratic polynomials for the velocity resp. discontinuos piecewise linear polynomials for the pressure space, [16]. Further let $N_h^u, N_h^p \in \mathbb{N}$ the number of basis functions of the velocity resp. pressure space.

By using a zero extension $\tilde{\mathbf{u}}_{\text{in}}$ of $\mathbf{u}_0$ to $\Gamma_{h,\text{in}}$ we can now modify the boundary condition (7) and arrive at the approximate discrete problem on Ω_h:
Find $\mathbf{u_h} \in Q^2(T_h^o)$, $p_h \in P^1_{\text{disc}}(T_h^o)$ with

$$A_h(\mathbf{u_h}, \mathbf{u_h}, \varphi_\mathbf{h}) - B_h(p_h, \varphi_\mathbf{h}) = \langle \mathbf{f}, \varphi_\mathbf{h} \rangle \qquad \forall \varphi_\mathbf{h} \in Q^2(T_h^o), \tag{11}$$

$$-B_h(\psi_h, \mathbf{u_h}) \qquad = 0 \qquad \forall \psi_h \in P^1_{\text{disc}}(T_h^o), \tag{12}$$

$$\mathbf{u_h} = \tilde{\mathbf{u}}_{\text{in}} \text{ on } \Gamma_{h,\text{in}} \qquad \mathbf{u_h} = 0 \text{ on } \Gamma_{h,\text{fict}}. \tag{13}$$

By simply extending $\mathbf{u_h}$ and p_h with zero we can recover the approximation of (7) on the whole background mesh T_h. Note, since we only approximate our actual boundary with linear elements, we can only expect an approximation error of $||\mathbf{u} - \mathbf{u_h}||_{L^2(\Omega)} \sim \mathcal{O}(h)$.

While this approach already leads to sufficient resolution for laminar flows [6], we can further improve on it since we only capture Ω in the Q^1 degrees of freedom. Naively enforcing the no-slip condition on inner DoFs leads to an underdetermined pressure solution, so instead we enforce a divergence-free L2-projection to zero on all DoFs outside of Ω by substituting the respective row in the system matrix, mirroring the phase separation in (6). For this, let $\varphi_\mathbf{j} \in Q^2(T_h)$, $j = 1, ..., N_h^v$ the Lagrange-basis of the velocity space and $x_h^j \in \mathbb{R}^d$ the corresponding set of Lagrange-basis nodes. We explicitly substitute our system by modifying A_h and right hand side f_h depending on which discrete test function φ_j is used:

$$\tilde{A_h}(v, w, \varphi_j) := \begin{cases} A_h(v, w, \varphi_j) & \text{if } x_h^j \in \Omega \\ \int_{\Omega_h} w \cdot \varphi_j & \text{else} \end{cases}, \quad \tilde{f_h}(\varphi_j) := \begin{cases} \langle \mathbf{f}, \varphi_j \rangle & \text{if } x_h^j \in \Omega \\ 0 & \text{else} \end{cases}. \tag{14}$$

The rows to be substituted can easily be preprocessed and do not change the overall structure of our system matrix. The Dirichlet boundary condition are always assumed to be incorporated into the system, which can for example be realized with an iterative filtering technique or by explicitly substituting the rows with unit rows and modifying the right hand side, which will be omitted in the following sections.

Finally by representing the solution vector as the sum of the FE basis functions $\mathbf{u_h} = \sum_i \mathbf{c_{h,i}} \varphi_i$, with $\mathbf{c_h} \in \mathbb{R}^{N_h^u \times d}$ the vector-valued velocity coefficients as well as $p_h = \sum_l q_{h,l} \psi_\mathbf{l}$, with ψ_j the basis functions of P^1_{disc} and $q_h \in \mathbb{R}^{N_h^p}$ the pressure coefficients, we are able to define the fully discretized system. In the following the (velocity depended) discrete system matrices are defined as

$$\mathbf{A}(\mathbf{v}) \in \mathbb{R}^{d \cdot N_h^u \times d \cdot N_h^u}, \mathbf{A}_{i,j}(\mathbf{v}) := \tilde{A_h}(\mathbf{v}, \varphi_\mathbf{j}, \varphi_\mathbf{i}), \quad \mathbf{B} \in \mathbb{R}^{d \cdot N_h^u \times N_h^p}, \mathbf{B}_{i,j} := B_h(\psi_j, \varphi_i).$$

3.2 Monolithic Newton Method

To linearize (11) we directly apply Newton method to our continuos problem. While there are also a wide range of explicit and semi-implicit methods that rely on a pseudo time stepping approach and work well for highly viscous fluids [17], as long as the initial guess $(\mathbf{u_h}^0, p_h^0)^T$ is sufficient, we can expect a Newton type method to have a

superior super-linear convergence rate. To formulate the Newton iteration, we need the Jacobian of the operator A_h, which is in our case the sum of A_h and the operator

$$L_h(v, w, z) := \int_{\Omega_h} \frac{\nu'(\dot{\gamma}(v))}{\dot{\gamma}(v)} \mathbf{D}(w) : \nabla z. \tag{15}$$

Analogous to (14) we define the modified operator $\tilde{L_h}(v, w, z)$ by substituting a zero row instead for outside test functions as well as the finite element matrix $\mathbf{L}_{i,j}(\mathbf{v}) := \tilde{L_h}(\mathbf{v}, \varphi_\mathbf{j}, \varphi_\mathbf{i})$ and now are able to define a general fixpoint iteration for $k = 0, 1, \ldots$:

$$\begin{pmatrix} \mathbf{c_h}^{k+1} \\ q_h^{k+1} \end{pmatrix} = \begin{pmatrix} \mathbf{c_h}^{k} \\ q_h^{k} \end{pmatrix} + \begin{pmatrix} \mathbf{A}(\mathbf{u_h}^k) + \omega \mathbf{L}(\mathbf{u_h}^k) & \mathbf{B} \\ \mathbf{B}^T & 0 \end{pmatrix}^{-1} \left(\begin{pmatrix} f_h \\ 0 \end{pmatrix} - \begin{pmatrix} \mathbf{A}(\mathbf{u_h}^k) & \mathbf{B} \\ \mathbf{B}^T & 0 \end{pmatrix} \begin{pmatrix} \mathbf{c_h}^{k} \\ q_h^{k} \end{pmatrix} \right). \tag{16}$$

The iteration is repeated until a sufficiently small non linear defect is reached. With the damping parameter $\omega \in \mathbb{R}$ we can switch between a Picard fix point iteration ($\omega = 0$) and a Newton-Raphson method ($\omega = 1$). In any case we are now required to assemble and then solve a large linear sparse system of the form

$$\begin{pmatrix} \tilde{A} & \tilde{B} \\ B^T & 0 \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} g_1 \\ g_2 \end{pmatrix}. \tag{17}$$

multiple times. For this we employ a scalable geometric multigrid solver we describe in the next section. A few things to note

- By alternating between $\omega = 0$ and $\omega = 1$ we are able to vastly stabilize the fixpoint iteration even for bad initial values [18], which will be denoted as Alternating-Picard-Newton (AlPiNe) method. There is also a vast range of other stabilization and acceleration approaches, which in the end all require to efficiently solve systems of type (17). For extrusion die problems the AlPiNe method works sufficiently well, so that more expensive approaches like actual line searches are likely not worth it, even though more research should be invested here.
- The Picard iteration can be interpreted as the limit of an infinitely large time step size of a monolithic IMEX schema of a pseudo unsteady formulation of (1), which is unconditionally stable for shear-rate dependent viscosity flow problems, see [17]. For this reason, we can expect at least the Picard iteration to converge, even if very slowly, if a steady solution exists at all.

3.2.1 FBM Multigrid Solver

For the solution of the linear system we use a hierarchical V-cycle geometric multigrid method as preconditioner for a Krylov-subspace solver, for details refer to [19], which requires three main components: A prolongation operator, a smoother and a coarse grid solver. For a fine grid refinement $h > 0$ define the two level hierarchy with the discrete background mesh T_{2h}, T_h as well as the culled representation T_{2h}^o, T_h^o and their respective finite element spaces. The necessary system matrices can be directly assembled as long

as the restricted solution vector $u_h, u_{2h}, ...$ is available which we recover by sequentially applying the restriction operator $R^u_{\cdot,\cdot}$ defined below onto the fine grid solution u_h.

Since $Q^2(T^o_{2h}) \not\subseteq Q^2(T^o_h)$ nor $P^1_{\text{disc}}(T^o_{2h}) \not\subseteq P^1_{\text{disc}}(T^o_h)$ due to the culling of the mesh elements, we can not use standard transfer operators. Luckily, we can instead use the zero extension to the background domain to construct these. By extending the coefficient vectors with zeros we can define trivial extension operators $E^u_h : Q^2(T^o_h) \to Q^2(T_h)$ and $E^p_h : P^1_{\text{disc}}(T^o_h) \to P^1_{\text{disc}}(T_h)$. Since $Q^2(T_{2h}) \subseteq Q^2(T_h)$ (and analogous for P^1_{disc}) we can use standard prolongation operators $\tilde{P}^u_{2h,h} : Q^2(T_{2h}) \to Q^2(T_h)$, $\tilde{P}^p_{2h,h} : P^1_{\text{disc}}(T_{2h}) \to P^1_{\text{disc}}(T_h)$ defined by a local L2-projection for Q^2 and P^1_{disc} respectively. By composition we can thus construct prolongation operators on the culled spaces

$$\begin{aligned} P^u_{2h,h} : Q^2(T^o_{2h}) &\to Q^2(T^o_h) \\ u_{2h} &\mapsto E^{uT}_h \circ \tilde{P}^u_{2h,h} \circ E^u_h\,(u_{2h}) \end{aligned}, \qquad \begin{aligned} P^p_{2h,h} : P^1_{\text{disc}}(T^o_{2h}) &\to P^1_{\text{disc}}(T^o_h) \\ p_{2h} &\mapsto E^{pT}_h \circ \tilde{P}^p_{2h,h} \circ E^p_h\,(p_{2h}) \end{aligned}. \tag{18}$$

Further we use the canonical restriction operator, i.e. $R^x_{h,2h} := {P^x_{2h,h}}^T$.

Since we use a discontinuous pressure space, a Vanka-type additive element-wise overlapping Schwarz domain-decomposition [20], [21] is well defined. Let $N = d \cdot N^u_h + N^p_h$ the number of velocity and pressure DoFs for the refinement level h and $N_l \in \mathbb{N}$ the number of DoFs on a single cell $T \in T^o_h$. For the system matrix $S \in \mathbb{R}^{N\times N}$ define an element-wise restriction matrix $R_T \in \mathbb{R}^{N_l \times N}$ extracting the local submatrix $\mathbb{R}^{N_l\times N_l} \ni S|_T := R_T \cdot S \cdot R^T_T$ of only the entries which corresponding basis function having support on element $T \in T_h$. Additionally define for each DoF an averaging weight $k_i, i = 1, ..., N$ counting the inverse of the number of elements in T_h on which again the corresponding basis function is not zero and define the diagonal weight matrix $K \in \mathbb{R}^{N\times N}, K_{i,j} = k_i\delta_{i,j}$. For a given damping parameter $a_V > 0$ we can define the application of the smoother on a sequence of solution vectors $t_k \in \mathbb{R}^N, k = 0, 1, \ldots$ and right-hand side $g \in \mathbb{R}^N$ as

$$t_{k+1} = t_k + a_V \left(\sum_{T\in T^o_h} K R^T_T (S|_T)^{-1} R_T \right) (g - St_k). \tag{19}$$

Importantly, by factorizing the local inverses beforehand, the application of the local block smoothers can be preprocessed into a single matrix $V \in \mathbb{R}^{N\times N}$, reducing the application of the Schwarz smoother to two consecutive global sparse matrix vector products and vector additions, which is cheap, scalable and well suited for GPU acceleration.

Finally, for the coarse grid solver we either use a direct sparse solver, if the coarse grid mesh size permits this, or we use above additive Schwarz as a preconditioner for a GMRES-Krylov subspace solver [22]. While practical, both options are suboptimal since on one hand it is a hard problem to construct coarse meshes, which are small enough for a direct solver while also capturing the geometry well enough without too anisotropic elements in an automation friendly way. On the other hand the convergence rate of the GMRES degenerates linearly with the coarse mesh size, leading to issues in the stability and efficiency of the overall multigrid solver.

For this reason, in the course of the ”StroemungsRaum” project, we added an interface to a monolithic FastAndRobustSchwarz (FROSch) solver in coorperation with TU Freiberg [23], [24], which remedy the scalability issues of the coarse-grid problem and appears to be a promising approach. We will further elaborate on this in an upcoming article.

With the components in place, the multigrid solver can be applied to the non-conformal FBM mesh hierarchy. Note that all components can be expressed as simple matrix vector multiplications as well as dot-products and vector additions, only relying on scalable neighbour to neighbour communication and global reductions. Additionally it is straightforward to dispatch these as GPU-native kernels, allowing us to execute the linear solver completely in device memory.

4 Benchmark results

To evaluate the improvements of our new approach, we compare the solution time (i.e. only phase 2 of the pipeline) for the large Scalexa benchmark problem from the ”StroemungsRaum” project, describing a typical case of a polymer melt flow through a fairly complicated thin-walled outflow geometry.

In figure 1 are the culled coarse and fine grid meshes generated from the Scalexa benchmark geometry represented. The problem size of the coarse grid is about 2 million DoFs, while the fine grid has about 105 million DoFs. For the GPU runs, we also used a more aggressive refinement strategy to generate a coarse mesh with only 20.000 DoFs, on which a direct solver can be deployed on a single GPU, which is then refined four times to reach the required resolution.

For our test hardware, we use four AMD EPYC 9354 each with 720 GB of DDR5-4800 memory for the CPU tests and for the GPU tests, we used the Helma cluster of NHR FAU, each node consisting of two AMD EPYC 9554 which support four NVIDIA H100 with 94 GB HBM2e on board memory. The Helma cluster provides up to 96 nodes, on which we were able to ran scaling tests.

In figure 2 we see the speedup of our developed method in comparison to the original pseudo-unsteady pressure projection splitting approach. On the generic CPU resources and same background mesh, we are able to reach a speedup of factor 10.7 by combining our steady non-linear solver in combination with the FROSch coarse grid solver.

For the GPU clusters we are only able to reach a relative speedup of factor 3 for our large coarse grid GMRES AlPiNe variant, which is underwhelming for the resources expended. This is mainly due to scaling issues of the coarse grid GMRES due to the dispatch mechanism.

A fix to this issue is employing a smaller coarse grid, for which an efficient direct sparse solver can be applied. With this we can generate an additional factor 54 speedup to the original solver approach.

While the fine mesh is a bit smaller with about 70 million DoFs in comparison to the large coarse grid case, due to larger element anisotropies and destabilization due to the poor capturing of the geometry on the coarse grids, an increase in the required

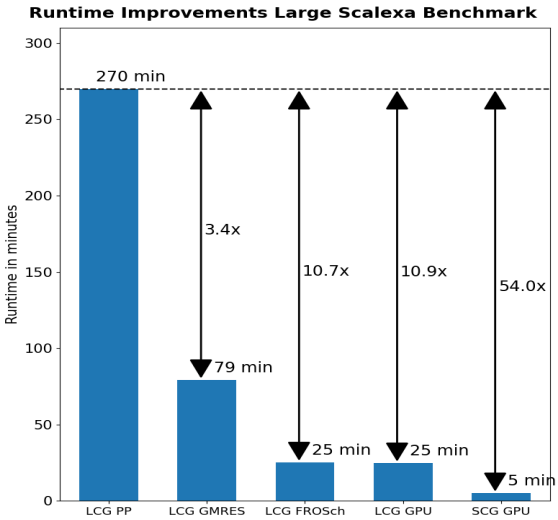

Fig. 2 Runtimes for the Large Scalexa Benchmark for different FBM solvers on 4 x AMD EPYC 9354 resp. 4 x Helma Nodes, the abbrevations mean PP: Original Large coarse grid Pressure Projection Solver, LCG GMRES: Monolithic AlPiNe with large coarse grid GMRES coarse solver, LCG FROSch: Monolithic AlPiNe with large coarse grid FROSch coarse solver, LCG GPU: Monolithic AlPiNe on GPU with large coarse grid GMRES, SCG GPU: Monolithic AlPiNe on GPU with small coarse grid direct sparse solver.

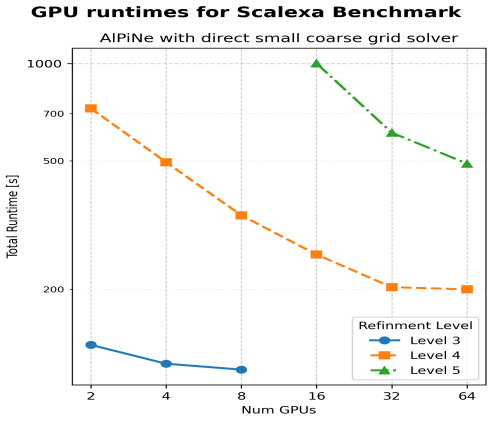

Fig. 3 Runtimes of the monolithic AlPiNe solver with small coarse grid solver for the Large Scalexa Benchmark for different number of refinement levels and number of GPUs.

multigrid iteration of factor ~ 3 makes this approach unfeasible on CPUs. Fixing these stability issues is still ongoing work but there is still room for additional improvements.

In figure 3 we see the scaling behavior of the small coarse grid AlPiNe solver for up to 16 nodes of the Helma cluster for 3 different target refinement levels. For a moderate amount of nodes (for level 4 up to 8 nodes) we reach a parallel efficiency larger 50%, while we reach nearly 100% parallel efficiency for even larger problem sizes. While the weak scalability from level 4 to 5, which matches a factor 8 in the increase of the problem size, is acceptable, the tests clearly show issues in form of a overhead per additional system level, which is visible in the difference in runtime between level 3 and level 4 as well as the stagnation in the strong scaling across all levels. The problems here are similar to the ones for the large coarse grid GMRES solver, albeit not as dominant, which can be traced back to the overhead of small kernel launches.

This issue will always inhibit the strong scaling on multi node GPU systems to a certain degree, but the effect is mitigable by fusing kernels and using graph executions. This will require some restructuring of our solver backends, but we are working on this at the moment and first tests show promising results.

5 Conclusion

We presented an efficient and robust solution approach for highly viscous fluids through implicitly captured domains within an automatic simulation pipeline for industrial relevant problems and showed the potential for recent HPC hardware.

There are still a lot of optimizations available for the GPU transition of our codebase, but we already see a significant speedup for our approach for conventional CPU systems as well as hybrid systems. We also want to test our approach on AMD-GPU systems in the near future. For traditional CPU-only systems especially the use of the FROSch solver is very promising and in this regard we are putting effort into improving the solver interface even further.

While the FBM approach is very robust and simplifies interfacing since it only relies on inside out information of the geometry, the reduced approximation order and non optimal resolution of the boundary could be improved on. Recent developments in the automatic aligned volume meshing [2] is very promising and can be easily adapted to by our method.

Furthermore, improving the convergence order and resolution of the implicit boundary, see [25], [26], seems promising but the stability in context of industrial relevant problems has to be evaluated.

Acknowledgements This work was supported by the Federal Ministry of Research, Technology and Space (BMFTR) through the project "StroemungsRaum" 16ME0706K, which is part of the initiative "Neue Methoden und Technologien für das Exascale-Höchstleistungsrechnen" (SCALEXA).

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 16ME0706K. NHR funding is provided by federal and Bavarian state authorities. NHR@FAU hardware is partially funded by the German Research Foundation (DFG) – 440719683.

References

1. Geuzaine, C., Remacle, J.-F.: Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. In: International Journal for Numerical Methods in Engineering **79** (11) 1309–1331 (2009)
2. Hu, Y., Zhou, Q., Gao, X., Jacobson, A., Zorin, D., Panozzo, D.: Tetrahedral Meshing in the Wild. In: ACM Trans. Graph. (2018) doi: 10.1145/3197517.3201353

3. Burman, E., Hansbo, P.: Fictitious domain methods using cut elements: III. A stabilized Nitsche method for Stokes' problem. In: ESAIM: M2AN **48** (3) 859–874 (2014) doi: 10.1051/m2an/2013123
4. Wang, Q., Chen, J.: A new unfitted stabilized Nitsche's finite element method for Stokes interface problems. In: Computers & Mathematics with Applications **70** (5) 820–834 (2015) doi: 10.1016/j.camwa.2015.05.024
5. Girault, V., Glowinski, R., Pan, T.W.: A Fictitious-Domain Method with Distributed Multiplier for the Stokes Problem. In: Sequeira, A., da Veiga, H.B., Videman, J.H. (eds) Applied Nonlinear Analysis. Springer (2002) doi: 10.1007/0-306-47096-9_12
6. Turek, S., Wan, D., Rivkind, L.: The Fictitious Boundary Method for the Implicit Treatment of Dirichlet Boundary Conditions with Applications to Incompressible Flow Simulations. In: Lecture Notes in Computational Science and Engineering (2003) doi: 10.1007/978-3-642-19014-8_3
7. Wan, D., Turek, S.: An efficient multigrid-FEM method for the simulation of solid–liquid two phase flows. In: Journal of Computational and Applied Mathematics, **203** (2) 561–580 (2007)
8. Münster, R., Mierka, O. and Turek, S.: Finite element-fictitious boundary methods (FEM-FBM) for 3D particulate flow. In: Int. J. Numer. Meth. Fluids, **69** 294-313 (2012).
9. Carreau, P. J.: Rheological Equations from Molecular Network Theories. In: Transactions of The Society of Rheology **16** (1) 99–127 (1972) doi: 10.1122/1.549276
10. Frehse, J., M´alek, J., Steinhauer, M., An existence result for fluids with shear dependent viscosity-steady flows. In: Nonlinear Anal. TMA 30 (5) 3041-3049 (1997)
11. Schneiders, R., Schindler, R., Weiler, F.: Octree-based Generation of Hexahedral Element Meshes. In: Proceedings of the 5th International Meshing Roundtable (1999)
12. X. Gao, H. Shen, D. Panozzo: Feature preserving octree-based hexahedral meshing. In: Computer Graphics Forum, **38** 135–149 (2019)
13. Turek, S.: On discrete projection methods for the incompressible Navier–Stokes equations: An algorithmical approach In:Comput. Methods Appl. Mech. Engrg. **143** 271-288 (1997)
14. Borrvall, T., Petersson, J.: Topology optimization of fluids in Stokes flow. In: International Journal for Numerical Methods in Fluids **41** (1) 77–107 (2003)
15. Papadopoulos, I.P.A, Süli, E.: Numerical analysis of a topology optimization problem for Stokes flow. In: Journal of Computational and Applied Mathematics **412** (2022)
16. Boffi, D., Brezzi, F., Fortin, M.: Mixed Finite Element Methods and Applications. (2013) doi: 10.1007/978-3-642-36519-5.
17. Barrenechea, G., Castillo, E., Pacheco, D.: Implicit-explicit Schemes for Incompressible Flow Problems with Variable Viscosity. In: SIAM Journal on Scientific Computing **46** A2660-A2682 (2024)
18. Pollock, S., Rebholz, L.G., Tu, X. et al: Analysis of the Picard-Newton Iteration for the Navier-Stokes Equations: Global Stability and Quadratic Convergence. J Sci Comput **104** (25) (2025). doi: 10.1007/s10915-025-02946-6
19. Hackbusch, W.: Multi-grid methods and applications. In: Springer Science & Business Media, **4** (2013).
20. Cai X.-C., Sarkis M.: Restricted additive Schwarz preconditioner for general sparse linear systems. In: SIAM Journal of Scientific Computing, **21** (2) 792–797 (1999)
21. Saberi, S., Meschke, G., Vogel, A.: A restricted additive Vanka smoother for geometric multigrid. In: Journal of Computational Physics, **459** (2022) doi: 10.1016/j.jcp.2022.111123
22. Saad, Y. Schultz, M. H.: GMRES: A generalized minimal residual algorithm for solving non-symmetric linear systems. In: SIAM Journal on Scientific and Statistical Computing, **7** (1986), 856–869.
23. Heinlein, A., Klawonn, A., Rheinbach, O.: A parallel implementation of a two-level overlapping Schwarz method with energy-minimizing coarse space based on Trilinos. In: SIAM J. Sci. Comput. (2016) doi: 10.1137/16M1062843
24. Köhler, S., Rheinbach, O.: Monolithic Multi-level Overlapping Schwarz Solvers for Fluid Problems. In: Submitted to the PAMM Special Issue: 95th Annual Meeting of the International Association of Applied Mathematics and Mechanics (GAMM) (2025), https://arxiv.org/pdf/2508.04356
25. Duprez, M., Lozinski, A.: φ-FEM: A Finite Element Method on Domains Defined by Level-Sets. In: SIAM Journal on Numerical Analysis **58** (2) 1008–1028 (2020) doi: 10.1137/19M1248947
26. Kothari, H., Krause, R.: A Multigrid Method for a Nitsche-based Extended Finite Element Method. (2019) 10.48550/arXiv.1912.00496.