A Primal-Dual Box-Constrained QP Pressure Poisson Solver with Topology-Aware Geometry-Inspired Aggregation AMG

Tetsuya Takahashi, Christopher Batty

Abstract— We propose a new barrier-based box-constrained convex QP solver based on a primal-dual interior point method to efficiently solve large-scale pressure Poisson problems with non-negative pressure constraints, which commonly arise in liquid animation. The performance of prior active-set-based approaches is limited by the need to repeatedly update the active set. Our solver eliminates this issue by entirely avoiding the use of an active set, which in turn makes the inner problems of our Newton iteration process fully unconstrained. For efficiency, exploiting the solution uniqueness of convex QPs and the fact that the pressure constraints are simple box constraints, we aggressively update solution candidates without performing any step selection procedure (such as line search) and instead directly clamp candidates back to the bounds wherever constraint violations occur. Additionally, to accelerate the inner linear solves, we present a topology-aware geometry-inspired aggregation algebraic multigrid preconditioner and describe in detail several key performance optimizations that we incorporate. We demonstrate the efficacy of our solver in various practical scenarios and show that it often surpasses various alternatives in terms of speed and memory usage.

Index Terms—Fluid simulation, quadratic program, multigrid, primal-dual interior point method

1 Introduction

The visual quality of Eulerian grid-based liquid animation benefits significantly from using inequality-constraint-based boundary conditions, which allow natural separation of liquid from solid surfaces [7, 13, 24, 30, 43]. However, this improvement over naive boundary treatments comes at a cost. Unlike the traditional pressure projection, which requires solving only a linear system, the required non-negative pressure constraints lead to a more numerically challenging problem: a linear complementarity problem (LCP) or equivalent box-constrained convex quadratic program (QP) [7, 34]. We seek to develop a highly efficient numerical solver for this problem.

While one can solve the LCP using variants of standard linear solvers, e.g., projected Gauss-Seidel (PGS) [19] or standalone nonlinear multigrid methods [13, 30], numerically minimizing a corresponding quadratic objective (e.g., using iterative Krylov solvers) can often be more robust, efficient, and flexible [33,39,47]. Therefore, a popular and effective approach has been to apply a Krylov solver with an active-set method, specifically *modified proportioning with reduced gradient projections* (MPRGP) [17, 18], to solve the box-constrained convex QPs. This formulation supports further acceleration with preconditioners, such as modified incomplete Cholesky (MIC) [22, 34] or algebraic multigrid (AMG) [43].

Despite its success, this MPRGP-based methodology suffers from a key remaining bottleneck. When one solves box-constrained QPs using MIC-MPRGP [22, 34], the number of MPRGP iterations required to achieve convergence is typically dictated by the (limited) effectiveness of MIC preconditioning. By contrast, since AMG preconditioning is much more effective, the necessary iteration count for AMG-MPRGP [43] instead becomes dictated by the need to repeatedly update the active set of constraints on the way to finding the optimal solution. In particular, this issue becomes more noticeable when more variables interact with lower/upper bounds [43] and can persist even with a fast active-set expansion technique [29, 42]. As such, MPRGP cannot take maximal advantage of AMG preconditioning, and further improvements to preconditioning effectiveness are unlikely to remove the existing performance bound in practice. Moreover, it is challenging to develop effective preconditioners that take the active sets into account [43].

- *Tetsuya Takahashi is with Tencent America. E-mail: ttakahashi@tencent.com,*
- *Christopher Batty is with the University of Waterloo. E-mail: christopher.batty@uwaterloo.ca.*

We therefore set MPRGP aside and propose a new box-constrained convex QP solver based on a primal-dual interior point method to efficiently handle large-scale sparse QP problems arising from fluid pressure projection with separating solid boundary conditions. Our new solver handles the constraints with the help of log-barriers [36] and is thus free of active sets, unlike MPRGP [17, 18]. This design decision removes the performance bound due to the active-set updates while taking full advantage of AMG preconditioning. In addition, we exploit the solution uniqueness of the convex QP and the box constraints to achieve greater efficiency by aggressively updating solution candidates without a step selection procedure and instead directly clamping them back to the bound where constraint violations occur. As our barrier-based solver leads to inner problems that are fully unconstrained linear systems with symmetric positive definite (SPD) M-matrices, we can efficiently solve the inner problems using AMG-preconditioned CG (AMGCG). To further accelerate solving the inner linear systems, we present a new topology-aware geometry-inspired aggregation AMG preconditioner (which can also be used independently of our barrier-based QP solver). Our AMG preconditioner simultaneously possesses five key features: 1) it ensures solvability at coarser levels due to the Galerkin principle, 2) it avoids merging algebraically disconnected components so coarser-level systems remain effective, 3) it preserves the regular Cartesian grid structure that permits repeated applications of the same coarsening scheme, 4) it enables red-black coloring for effective GS-type smoothers, and 5) it maintains relatively sparse restriction/prolongation operators and coarser-level systems for memory and computational efficiency. We also present further details for performance optimizations within AMG preconditioning, utilizing red-black coloring and traditional seven-point finite-difference stencils in 3D. Finally, while our overall QP solver was designed specifically with pressure Poisson problems in mind, its application is not limited to these problems; we therefore further demonstrate its effectiveness and versatility with an application to diffusion equations.

2 Related Work

2.1 Multigrid for Pressure Poisson Equations

Multigrid (MG) is among the most efficient approaches for solving systems of equations and is especially well-studied for Poisson equations, such as those which arise in the pressure projection step of fluid simulation. Background on MG methods can be found in various textbooks, e.g., [10, 47]. MG approaches can generally be classified into two categories based on how their multiresolution hierarchies are constructed: geometric MG (GMG) and algebraic MG (AMG).

GMG uses geometric coarsening and rediscretization of systems

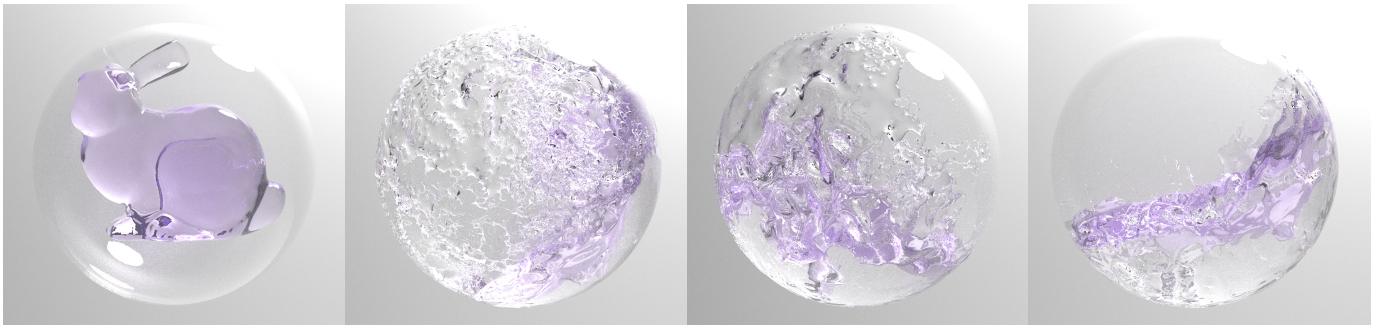


Fig. 1: A liquid bunny is dropped inside a moving solid sphere, simulated with non-negative pressure constraints. Our box-constrained convex QP solver yields results efficiently without exhibiting boundary suction artifacts.

based on geometric approximations at coarser levels [13, 33, 49]. While GMG is conceptually simpler and can be effective for simple domain shapes, it can diverge or stagnate with complex boundaries, which frequently appear in liquid simulations [33]. For improved robustness, one can perform extra steps of boundary smoothing [33], adopt smoother prolongation operators (via trilinear interpolation) coupled with red-black GS smoothing [16], employ topology-aware geometric coarsening [16, 20], or use GMG as a preconditioner for CG [33]. However, geometric discrepancies between finer and coarser levels (which are generally unavoidable to guarantee solvability at coarser levels) ultimately limit the effectiveness of GMG in many practical scenarios [43].

On the other hand, AMG uses only algebraic information to build coarser-level systems $\mathbf{A}_c$ via Galerkin coarsening applied to finer-level systems $\mathbf{A}_f$. Using prolongation operators $\mathbf{P}$ and restriction operators $\mathbf{R}(=\mathbf{P}^T)$, one defines $\mathbf{A}_c = \mathbf{R}\mathbf{A}_f\mathbf{P}$. AMG automatically ensures solvability at coarser levels, assuming a non-singular finest system and full-column-rank prolongation operators, due to the Galerkin principle [10,47]. In addition, AMG never merges algebraically disconnected components, thus offering effective coarser-level systems even with complex domain boundaries, unlike GMG [43]. While AMG can be an effective standalone solver or preconditioner, purely algebraic construction of prolongation and restriction operators (using e.g., plain unsmoothed aggregation, smoothed aggregation [45, 48], or the Ruge-Stüben method [40]) generally produces denser systems at the coarser levels, thereby sacrificing computational and memory efficiency, as compared to GMG [28, 50]. In addition, AMG typically destroys any underlying regular structure (such as that of Cartesian grids), which means that it cannot take advantage of convenient red-black coloring for embarrassingly parallel GS-type smoothers (which are typically more effective than other stationary iterative methods, such as weighted Jacobi [10, 47] or SPAI-0 [11]). Parallelizing instead via multicolor GS additionally requires an effective coloring scheme, increasing computational cost and implementation complexity.

To mitigate performance and memory issues due to the denser systems at the coarser levels in purely algebraic MG while avoiding the solvability and geometric discrepancy problems of purely geometric MG, geometry-inspired aggregation AMG (GIAAMG) approaches have recently been proposed [30, 38, 51]. GIAAMG constructs the prolongation and restriction operators in an algebraic way (similar to AMG), while also employing geometric information (e.g., regular grid structures for fluid simulations) as an additional guide when building these operators, e.g., using 8-to-1 averaging. Exploiting these geometric structures, one can form much sparser restriction/prolongation operators, which also leads to sparser coarser-level systems compared to purely algebraic MG, resulting in reduced memory usage and per-iteration costs [10]. In addition, GIAAMG can preserve regular grid structures across successive coarsening operations, potentially enabling one to employ red-black GS-type smoothers [38]. However, unfortunately, as the previous GIAAMG approaches are unaware of the liquid topology, i.e., topology-oblivious, they neglect algebraic connectivities in the operator construction, inappropriately merging algebraically disconnected components and thus degrading the effectiveness of the coarser-level systems, especially in topology-critical scenarios, such as maze structures [43].

To address this issue, we present a topology-aware GIAAMG scheme that merges only algebraically connected components (similar to purely algebraic MG) while simultaneously ensuring the solvability of the system, maintaining the regular grid structure, enabling red-black coloring, and achieving relatively sparse restriction/prolongation operators and coarser-level systems.

2.2 Non-Negative Pressure Constraints

The key to wall-separating liquid behavior is enforcement of non-negative pressures within the pressure projection [7, 13, 24, 30, 43]. Researchers have formulated LCPs or equivalent box-constrained QPs to simultaneously enforce the incompressibility (divergence-free) condition and non-negative pressure constraints. These formulations have been solved in various ways, e.g., using a PATH solver [7], PGS [31], nonlinear multigrid [13, 30], a non-smooth Newton method [5], and a variant of the alternating direction method of multipliers (ADMM) [24]. Alternatively, iterative Krylov solvers augmented with an active-set method have demonstrated their potential in practical scenarios for splashing liquids and granular flows [22, 34, 42, 43]. In particular, a CG-based box-constrained QP solver, MPRGP [17, 18] has become popular and is effective in combination with MIC preconditioning [22, 34, 42] and AMG preconditioning [43]. It has been shown that the performance of MPRGP can then become bounded by the efficiency of the active-set update, when sufficiently effective preconditioners (e.g., AMG) are employed [43]. Instead of using MPRGP, we present a new barrier-based box-constrained QP solver that eliminates the performance bound due to the active-set updates by eschewing active sets altogether, and can thereby take full advantage of AMG preconditioning.

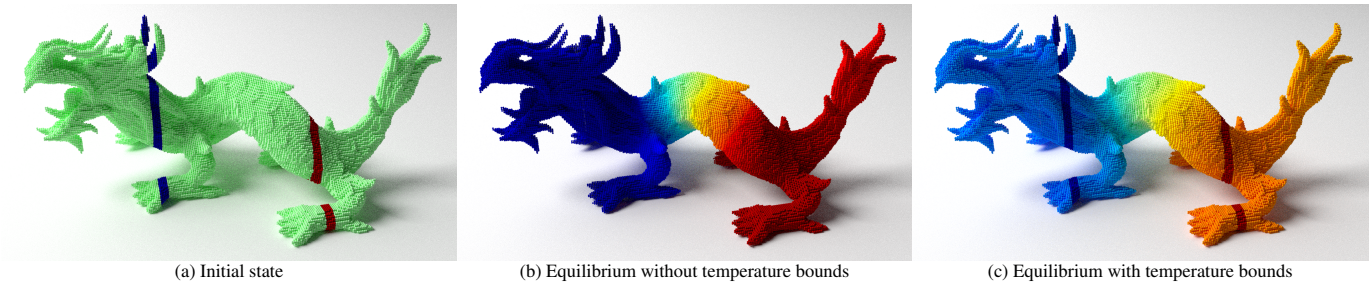
3 Liquid Simulation Formulation

To simulate splashy liquids with separating solid boundary conditions (but neither viscosity nor surface tension), we solve the incompressible Euler equations and adopt the affine particle-in-cell (APIC) framework [27] with the traditional finite-difference method on a staggered grid [9] and particle-level position correction for improved particle distributions [44]. We enforce fluid incompressibility with separating solid boundary conditions via minimization over pressure with non-negative pressure constraints [7, 34]. The resulting constrained minimization can be written as

$$\mathbf{p} = \underset{\mathbf{s}(\mathbf{p})\geq 0}{\arg\min} E(\mathbf{p}), \quad E(\mathbf{p}) = \frac{1}{2}\mathbf{p}^T\mathbf{A}\mathbf{p} - \mathbf{p}^T\mathbf{q}, \tag{1}$$

where $\mathbf{p}$ denotes the pressure, $\mathbf{A}$ and $\mathbf{q}$ the Laplacian matrix and the source term in the pressure Poisson equations, respectively, and $\mathbf{s}(\mathbf{p})$ the separating-boundary constraint function for pressure. We can define $\mathbf{s}(\mathbf{p})$ to uniformly enforce non-negative pressures throughout the fluid and thereby achieve splashy liquid behavior [22] using

$$\mathbf{s}(\mathbf{p}) = \mathbf{p} \geq 0. \tag{2}$$



(a) Initial state (b) Equilibrium without temperature bounds (c) Equilibrium with temperature bounds

Fig. 2: Constrained heat diffusion on a static dragon. Dark red/blue bands in (a) indicate Dirichlet values. Our QP solver correctly handles heat diffusion while respecting temperature bounds, achieving an equilibrium (c) that differs from the equilibrium without temperature bounds (b).

Alternatively, we can define $\mathbf{s}(\mathbf{p})$ for wall-separating but "non-splashy" liquid given the set of indices $\mathscr{S}$ for only those pressure cells in contact with solid boundaries [30] using

$$\mathbf{s}(\mathbf{p}) = \mathbf{p}_i \geq 0, \text{ if } i \in \mathscr{S}. \tag{3}$$

Our goal is to efficiently solve the box-constrained convex QP (1) using our barrier-based solver (§5), harnessing AMG preconditioning (§4) for inner CG solves.

4 Topology-Aware GIAAMG Preconditioning

To address various shortcomings of GMG, AMG, and topology-oblivious GIAAMG, we propose a topology-aware GIAAMG preconditioner that simultaneously achieves the following five features: 1) coarser-level solvability, 2) topology-aware algebraic aggregation, 3) regular grid structure preservation, 4) red-black coloring, and 5) sparse restriction/prolongation operators and coarser-level systems. In addition, we describe our performance optimizations, which take full advantage of the red-black coloring for GS-type smoothers and residual computations.

4.1 AMG Preconditioning

The goal of AMG preconditioning is to solve the preconditioning system $\mathbf{Ax} = \mathbf{b}$ derived from the quadratic objective in (1) as accurately and efficiently as possible. In practice, the accuracy and efficiency relationship exhibits a trade-off, and it is typically preferred to perform just one V- or W-cycle along with up to a few smoothing iterations to minimize the preconditioning cost [10, 38] because performing more CG iterations is generally more efficient than spending more time on preconditioning. Here, we use a single W-cycle (as we consistently observed it to be more efficient than a V-cycle with unsmoothed aggregation; see our supplementary materials). The algorithm for W-cycle preconditioning is given in Algorithm 1 (which is equivalent to a V-cycle if one skips lines 10 to 14).

Algorithm 1 W-cycle($\mathbf{A}_f$, $\mathbf{x}_f$, $\mathbf{b}_f$)

1: Initialize $\mathbf{x}_f = 0$
2: **if** Coarsest
3: Smooth $\mathbf{x}_f$ //§4.4
4: **else**
5: Presmooth $\mathbf{x}_f$ //§4.4
6: Compute residual $\mathbf{r}_f = \mathbf{b}_f - \mathbf{A}_f\mathbf{x}_f$ //§4.5
7: Restrict residual $\mathbf{b}_c = \mathbf{R}\mathbf{r}_f$
8: W-cycle($\mathbf{A}_c$, $\mathbf{x}_c$, $\mathbf{b}_c$)
9: Prolongate and correct $\mathbf{x}_f \mathrel{+}= \mathbf{P}\mathbf{x}_c$
10: Resmooth $\mathbf{x}_f$ //§4.6
11: Compute residual $\mathbf{r}_f = \mathbf{b}_f - \mathbf{A}_f\mathbf{x}_f$ //§4.7
12: Restrict residual $\mathbf{b}_c = \mathbf{R}\mathbf{r}_f$
13: W-cycle($\mathbf{A}_c$, $\mathbf{x}_c$, $\mathbf{b}_c$)
14: Prolongate and correct $\mathbf{x}_f \mathrel{+}= \mathbf{P}\mathbf{x}_c$
15: Postsmooth $\mathbf{x}_f$ //§4.8
16: **end if**

In AMG, if the prolongation and restriction operators are not sufficiently smooth, coarser-level errors are typically underestimated, delaying convergence. In such cases, there are a few possible treatments to apply: perform more or stronger (boundary) smoothing [2, 38, 52]; smooth the prolongation operator itself [48] (although such operator smoothing suffers from a non-negligible cost); use a W-cycle instead of a V-cycle; and/or rescale the coarser-level systems [40] (equivalent to over-interpolation of error), defining $\mathbf{A}_c = \alpha\mathbf{R}\mathbf{A}_f\mathbf{P}$, where α denotes the rescaling parameter (for unsmoothed prolongation/restriction operators, we use $\alpha = 0.55$ for V-cycles and $\alpha = 0.65$ for W-cycles).

The system matrices and prolongation/restriction operators can be stored in a memory-efficient way using the compressed sparse row (CSR) format, and parallel sparse matrix-matrix multiplication can be efficiently performed in the CSR format [37]. For implementation details of AMG, we refer the reader to available open source codes [15, 51].

4.2 Topology-Aware GIA Coarsening

To establish the MG hierarchy while ensuring the coarser level remains solvable, we use the following AMG-style procedure. We first build a prolongation operator $\mathbf{P}$ from the finest system $\mathbf{A}_f$ and geometric information (specifically, three indices x, y, z for each cell location in the 3D Cartesian grid), define the restriction operator $\mathbf{R}$ as $\mathbf{R} = \mathbf{P}^T$, and then assemble the coarser-level system $\mathbf{A}_c = \alpha\mathbf{R}\mathbf{A}_f\mathbf{P}$. (While earlier geometry-inspired aggregation approaches [30, 38, 51] use $\mathbf{R} = (1/2)^d\mathbf{P}^T$, where d is the number of spatial dimensions, we observed the constant scaling by $(1/2)^d$ to have no effect.) We repeatedly perform this process until only a single element is left for each algebraically connected group (i.e., we cannot reduce the system size any further). We give the details of this process below.

To completely avoid merging algebraically disconnected components, we base our approach on algebraic aggregation [15, 48], but without relying on geometric coarsening because the purely geometric perspective makes it more challenging to efficiently identify whether elements are algebraically connected or not (especially at the coarser levels) [16, 20]. In our algebraic aggregation approach, we additionally impose geometry-inspired constraints on aggregatable elements when forming coarser aggregation groups so as to maintain the regular grid structures, enable red-black coloring, and ensure sparsity of prolongation/restriction operators and coarser-level systems. Simultaneously, in this aggregation step, we allocate necessary degrees of freedoms (DOFs) for disconnected components, similar to cell duplication [16, 20]. In particular, we prefer unsmoothed aggregation because it lets us keep operators and systems sparse, and hierarchy construction is much faster than for smoothed aggregation approaches, which suffer from non-negligible costs of operator smoothing [48]. In addition, the slower convergence rates of unsmoothed aggregation (compared to smoothed aggregation) can be significantly improved via Galerkin scaling, W-cycles, and red-black GS-type smoothers. Our algorithm for the prolongation operator construction is given in Algorithm 2.

We identify whether a fine cell can be merged into a coarse one based on the coarse grid ID $\mathbf{d}_{x,y,z}$ computed for each fine cell by

$$\mathbf{d}_{x,y,z} = ((x/2) \ll 20) \,|\, ((y/2) \ll 10) \,|\, (z/2). \tag{4}$$

Algorithm 2 Construction of Prolongation Operator $\mathbf{P}$

```
Compute coarse grid ID d_{x,y,z} for each fine cell using (4)
for coarse cells c_j in red and then black
    Set F as fine cells in c_j
    while F is not empty
        for fine cell f_i in F
            for aggregatable neighbor n_i of f_i
                Relate n_i to c_j and add to a next set G
        F = G
Construct P using (5)
```

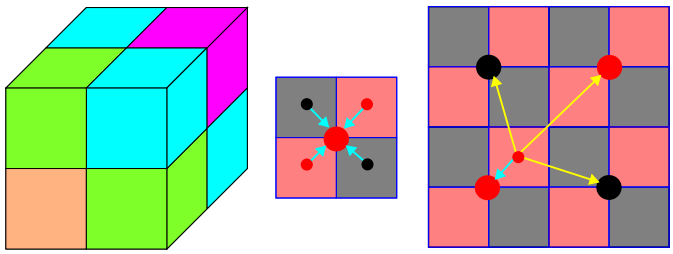


Fig. 3: (Left) 3D Illustration of the access order from a fine cell within a coarse cell in the scan-and-merge process. Starting from the orange cell, we reach green cells in the first pass, then cyan in the second pass, and finally magenta in the third pass. (Middle and right) 2D illustration of the configuration of fine and coarse cells with red/black dots representing red/black cells. (Middle) In our algorithm, when fine cells are related to a coarse cell, the coarse one needs to physically contain the fine ones (relation drawn as cyan arrows). (Right) Since linear interpolation relates the red fine cell to the coarser cells that do not physically contain the fine one (relation drawn as yellow arrows), the linear interpolation-based prolongation operator is denser.

Then, while scanning algebraically connected fine cells from another fine cell, if their coarse grid IDs $\mathbf{d}_{x,y,z}$ are the same, we permit the finer cells to merge (i.e., relate) to the coarse one. When fine cells are algebraically connected following the seven-point stencils, repeating this scan-and-merge process three times is sufficient because a search starting from a fine cell can reach the fine one diagonally located in the coarse cell (see Figure 3 (left)). However, in practice, it can happen that two neighboring fine cells are not directly connected (e.g., due to inactive velocity DOFs between two pressure cells [7]) but algebraically connected through other fine cells, requiring more than three iterations to reach all algebraically connected fine cells in the coarse cell. Furthermore, it is generally not possible to know how many iterations are necessary to reach all of the (algebraically connected) duplicated fine cells in the coarse cell. As such, we repeat this scan-and-merge process until no fine cells can further be merged into the coarse cell (see Algorithm 2), which can be considered as a variant of the flood fill algorithm based on the breadth-first search (BFS) for connected component analysis (CCA). Here, we note that our aggregation procedure eschews eliminating weakly connected elements with small off-diagonal values, unlike other AMG approaches [15, 28], to avoid cutting connections of elements in the grid structure, and assembles red coarse cells first and then black ones to ensure the special block matrix structure (see §4.3). Once we identify the relations between fine and coarse cells, we generate the prolongation operator as

$$\mathbf{P}_{ij} = \begin{cases} 1 & \text{if fine cell } i \text{ and coarse cell } j \text{ are related,} \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Since each fine cell is related to a single coarse cell, the resulting prolongation operator is guaranteed to be full column rank.

Our approach generates a so-called piecewise constant prolongation operator, and if all neighboring elements are algebraically connected in the regular grid structure, the resulting prolongation operator is exactly the same as the one generated by the previous topology-oblivious GIA approaches using 8-to-1 averaging [38, 51]. In addition, the cost for prolongation operator construction is comparable to algebraic aggregation [15, 48] and GIA [38, 51] (see our supplementary materials). Our algorithm enforces that finer cells are related to a coarser one only if these finer ones are physically included inside the coarser one (see 2D illustration in Figure 3 (middle)). This rule ensures that the resulting prolongation operator and thus coarser-level systems are sparse, and the coarser-level systems are structurally identical to the finer ones, enabling repeated applications of the same coarsening procedure and red-black coloring.

If we were instead to relate a finer cell to coarser cells that do not physically contain the finer cell (e.g., via trilinear interpolation [30], see Figure 3 (right)), the prolongation operators would be denser (though they also become smoother, similar to smoothed aggregation [48]), leading to denser and denser coarser-level systems and invalidating the red-black coloring with the seven-point stencil (although we would still be able to maintain the regular grid structures).

4.3 Block Matrix Structures

Given the red and black cells aggregated in this order, we can rewrite the preconditioning system $\mathbf{A}\mathbf{x} = \mathbf{b}$ (and corresponding coarser-level systems) in block form as

$$\begin{bmatrix} \mathbf{A}_{rr} & \mathbf{A}_{rb} \\ \mathbf{A}_{br} & \mathbf{A}_{bb} \end{bmatrix} \begin{bmatrix} \mathbf{x}_r \\ \mathbf{x}_b \end{bmatrix} = \begin{bmatrix} \mathbf{b}_r \\ \mathbf{b}_b \end{bmatrix}, \tag{6}$$

where subscripts r and b indicate red and black colors (for variables, cells, etc.), respectively. We observe that $\mathbf{A}_{rb} = \mathbf{A}_{br}^T$. This matrix structure enables sequential access to $\mathbf{x}_r$, completely avoiding access to $\mathbf{x}_b$ and improving the memory locality, and vice versa. In addition, because a red variable is algebraically connected exclusively to itself or black variables (and vice versa) due to the seven-point Cartesian finite-difference stencils in 3D (five-point stencils in 2D) at the finest level, $\mathbf{A}_{rr}$ and $\mathbf{A}_{bb}$ are guaranteed to be diagonal [37], and this diagonality also holds at the coarser levels due to our coarsening scheme. (If we did not aggregate all algebraically connected fine cells in a coarse cell during the $\mathbf{P}$ construction, the result would be multiple coarser level DOFs coupled with each other, causing $\mathbf{A}_{rr}$ and $\mathbf{A}_{bb}$ to no longer be strictly diagonal and thus invalidating the seven-point stencils.) Given the diagonality of these matrices, we define $\mathbf{D}_{rr} = \mathbf{A}_{rr}$ and $\mathbf{D}_{bb} = \mathbf{A}_{bb}$ for convenience, and precompute and store $\mathbf{D}_{rr}^{-1}$ and $\mathbf{D}_{bb}^{-1}$ prior to performing CG iterations to enable efficient smoothing operations. In addition, we can directly access diagonal elements $\mathbf{A}_{rr}$ and $\mathbf{A}_{bb}$ as the first and last entries of $\mathbf{A}$ at the corresponding row in the CSR format, respectively, since $\mathbf{A}_{rr}$ and $\mathbf{A}_{bb}$ have only one non-zero per row. Furthermore, due to this structure, we can exclusively access the non-zeros in the $\mathbf{A}_{rb}$ and $\mathbf{A}_{br}$ blocks by skipping the first and last entries of $\mathbf{A}$ at the corresponding row, respectively.

4.4 RBSSOR Presmoothing with Zero-Initialization

Considering the Cartesian grid structures and seven-point stencils that are strictly maintained from the finest to the coarsest levels, we prefer using a GS-type smoother parallelized via red-black coloring over weighted Jacobi [33] or SPAI-0 [11]. In addition, we can employ over-relaxation to improve the effectiveness of smoothing, and symmetrize the smoother to satisfy the symmetry requirement of CG preconditioning [37]. Specifically, we employ a red-black symmetric successive-over-relaxation (RBSSOR) smoother. Presmoothing operations that first process red and then black variables (i.e., in forward order) can be written as

$$\mathbf{x}_r^{k+1} = \mathbf{x}_r^k + \omega \mathbf{D}_{rr}^{-1}(\mathbf{b}_r - \mathbf{A}_{rr}\mathbf{x}_r^k - \mathbf{A}_{rb}\mathbf{x}_b^k), \tag{7}$$

$$\mathbf{x}_b^{k+1} = \mathbf{x}_b^k + \omega \mathbf{D}_{bb}^{-1}(\mathbf{b}_b - \mathbf{A}_{br}\mathbf{x}_r^{k+1} - \mathbf{A}_{bb}\mathbf{x}_b^k), \tag{8}$$

where k denotes the smoothing iteration index, and ω the over-relaxation parameter (we use $\omega = 1.2$). When $\omega = 1$, RBSSOR is equivalent to red-black symmetric GS (RBSGS).

As we perform only one W-cycle with one presmoothing (for efficient CG preconditioning [10, 33, 47]) along with the necessary zero-initialization for symmetric Krylov solvers [37, 46], we take advantage

of this zero-initialization to eliminate some unnecessary matrix-vector multiplications in (7) and (8) (as has also been done with damped Jacobi and SPAI-0 [3, 33, 43]), leading to

$$\mathbf{x}_r^{k+1} = \omega \mathbf{D}_{rr}^{-1} \mathbf{b}_r, \tag{9}$$

$$\mathbf{x}_b^{k+1} = \omega \mathbf{D}_{bb}^{-1} (\mathbf{b}_b - \mathbf{c}_b), \quad \mathbf{c}_b = \mathbf{A}_{br} \mathbf{x}_r^{k+1}. \tag{10}$$

Here, $\mathbf{c}_b$ can be reused for the residual computation (§4.5). The optimized RBSSOR presmoothing operations (9) and (10) remove essentially half of the matrix-vector multiplications needed to compute $\mathbf{Ax}$ since most of the non-zeros are in $\mathbf{A}_{rb}$ and $\mathbf{A}_{br}$ (whereas using damped Jacobi and SPAI-0, matrix-vector multiplication in presmoothing can be completely eliminated [3, 33, 43]).

While one can use any symmetric linear solver (e.g., Cholesky-based direct solver or MICCG [9]) at the coarsest level of MG preconditioning in general, we simply perform a smoothing operation, as done for presmoothing, because our coarsest system consists of a set of 1×1 systems.

4.5 First Residual Computation

To perform the W-cycle algorithm, we need to compute the residual $\mathbf{r}$ immediately after presmoothing (see Algorithm 1, line 6). The residual defined by $\mathbf{r} = \mathbf{b} - \mathbf{Ax}$ can be rewritten for red and black variables as

$$\mathbf{r}_r = \mathbf{b}_r - \mathbf{A}_{rr} \mathbf{x}_r - \mathbf{A}_{rb} \mathbf{x}_b, \tag{11}$$

$$\mathbf{r}_b = \mathbf{b}_b - \mathbf{A}_{br} \mathbf{x}_r - \mathbf{A}_{bb} \mathbf{x}_b = \mathbf{b}_b - \mathbf{c}_b - \mathbf{D}_{bb} \mathbf{x}_b. \tag{12}$$

Here, $\mathbf{A}_{br}\mathbf{x}_r$ in (12) is equivalent to $\mathbf{c}_b = \mathbf{A}_{br}\mathbf{x}_r^{k+1}$ in (10) as $\mathbf{x}_r$ is not updated in (10). As such, we can compute and store $\mathbf{c}_b$ during the presmoothing for $\mathbf{x}_b$ (10) and avoid recomputing $\mathbf{c}_b$ when computing $\mathbf{r}_b$ in (12). This optimization (which is not applicable to damped Jacobi or SPAI-0) allows us to further remove half of the matrix-vector multiplications, indicating that RBSSOR can essentially omit one full matrix-vector multiplication (in combination with the optimization within the presmoothing), equivalently to damped Jacobi and SPAI-0 [33, 43], if V-cycle is used. In addition, we can directly access $\mathbf{D}_{bb}$ as the last element in the row due to the CSR format (or as precomputed values), completely skipping access to $\mathbf{A}_{br}$ when computing $\mathbf{A}_{bb}\mathbf{x}_b$ in (12). Furthermore, since $\mathbf{r}_{b,i}$ (ith element of $\mathbf{r}_b$) is dependent only on $\mathbf{x}_{b,i}$ (ith element of $\mathbf{x}_b$) among $\mathbf{x}_b$, we can compute $\mathbf{r}_{b,i}$ once we obtain $\mathbf{x}_{b,i}$ by (10). This fact enables us to unify (12) with (10) (i.e., computing $\mathbf{x}_{b,i}$ and $\mathbf{r}_{b,i}$ together in this order) and to further remove the extra computation pass for (12).

4.6 RBSSOR Resmoothing

To perform the minimal amount of resmoothing while satisfying the symmetry requirement of CG preconditioning in the W-cycle [37], one possible approach is to perform a single forward or backward smoothing while switching the order every time at the coarser levels, and performing both forward and backward smoothing at the finest level. However, switching the smoothing order complicates the implementation and its optimization. Since the coarser-level smoothing is much less costly than that of the finest level, we prefer to ensure symmetry within each individual resmoothing, by performing forward and backward smoothing (without requiring other resmoothing operations at the same level for symmetry). Specifically, we perform resmoothing in the order of red-black black-red; we have tested black-red red-black and observed no particular differences. The resmoothing operations can thus be written as

$$\mathbf{x}_r^{k+1} = \mathbf{x}_r^k + \omega \mathbf{D}_{rr}^{-1} (\mathbf{b}_r - \mathbf{A}_{rr} \mathbf{x}_r^k - \mathbf{A}_{rb} \mathbf{x}_b^k), \tag{13}$$

$$\mathbf{x}_b^{k+1} = \mathbf{x}_b^k + \omega \mathbf{D}_{bb}^{-1} (\mathbf{b}_b - \mathbf{A}_{br} \mathbf{x}_r^{k+1} - \mathbf{A}_{bb} \mathbf{x}_b^k), \tag{14}$$

$$\mathbf{x}_b^{k+2} = \mathbf{x}_b^{k+1} + \omega \mathbf{D}_{bb}^{-1} (\mathbf{b}_b - \mathbf{A}_{br} \mathbf{x}_r^{k+1} - \mathbf{A}_{bb} \mathbf{x}_b^{k+1}), \tag{15}$$

$$\mathbf{x}_r^{k+2} = \mathbf{x}_r^{k+1} + \omega \mathbf{D}_{rr}^{-1} (\mathbf{b}_r - \mathbf{A}_{rr} \mathbf{x}_r^{k+1} - \mathbf{A}_{rb} \mathbf{x}_b^{k+2}). \tag{16}$$

Given the equivalence of $\mathbf{A}_{br}\mathbf{x}_r^{k+1}$ in (14) and (15), and independence of $\mathbf{x}_r$ and $\mathbf{x}_b$ from any other red and black variables except for itself, respectively, we can unify operations for $\mathbf{x}_b$ ((14) and (15)) to remove unnecessary operations as

$$\mathbf{x}_b^{k+2} = (1-\omega)^2 \mathbf{x}_b^k + (2-\omega)\omega \mathbf{D}_{bb}^{-1} (\mathbf{b}_b - \mathbf{A}_{br} \mathbf{x}_r^{k+1}). \tag{17}$$

In addition, since $\mathbf{A}_{rr} = \mathbf{D}_{rr}$, we can rewrite (16) with $\mathbf{c}_r$ to unify the resmoothing for $\mathbf{x}_r$ (16) and residual computation for $\mathbf{r}_r$ (19) (see §4.7) as

$$\mathbf{x}_r^{k+2} = (1-\omega)\mathbf{x}_r^{k+1} + \omega \mathbf{D}_{rr}^{-1} (\mathbf{b}_r - \mathbf{c}_r), \quad \mathbf{c}_r = \mathbf{A}_{rb} \mathbf{x}_b^{k+2}. \tag{18}$$

4.7 Second Residual Computation

The second residual computation (Algorithm 1, line 11) can be written as

$$\mathbf{r}_r = \mathbf{b}_r - \mathbf{A}_{rr} \mathbf{x}_r - \mathbf{A}_{rb} \mathbf{x}_b = \mathbf{b}_r - \mathbf{D}_{rr} \mathbf{x}_r - \mathbf{c}_r, \tag{19}$$

$$\mathbf{r}_b = \mathbf{b}_b - \mathbf{A}_{br} \mathbf{x}_r - \mathbf{A}_{bb} \mathbf{x}_b. \tag{20}$$

We unify (19) and (18) to eliminate unnecessary computations, as done for the first residual computation (12) and presmoothing (10).

4.8 RBSSOR Postsmoothing

Given the resmoothing symmetrized by itself, we perform postsmoothing in the reverse order of presmoothing to satisfy the symmetry requirement [37], first processing black and then red variables as

$$\mathbf{x}_b^{k+1} = \mathbf{x}_b^k + \omega \mathbf{D}_{bb}^{-1} (\mathbf{b}_b - \mathbf{A}_{br} \mathbf{x}_r^k - \mathbf{A}_{bb} \mathbf{x}_b^k), \tag{21}$$

$$\mathbf{x}_r^{k+1} = \mathbf{x}_r^k + \omega \mathbf{D}_{rr}^{-1} (\mathbf{b}_r - \mathbf{A}_{rr} \mathbf{x}_r^k - \mathbf{A}_{rb} \mathbf{x}_b^{k+1}). \tag{22}$$

5 Primal-Dual Box-Constrained Convex QP Solver

To avoid the performance bound due to active-set updates [17, 18], we propose a new barrier-based box-constrained convex QP solver based on a primal-dual interior point method. This approach completely avoids the use of an active set while taking full advantage of our AMG preconditioner for inner linear solves.

5.1 Primal-Dual Interior Point Method for QP

We consider a box-constrained convex QP problem written as

$$\mathbf{x} = \underset{\mathbf{c}(\mathbf{x}) \geq 0,}{\arg\min} f(\mathbf{x}), \quad f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{A} \mathbf{x} - \mathbf{b}^T \mathbf{x}, \tag{23}$$

with optimization variable $\mathbf{x}$, SPD M-matrix $\mathbf{A}$, vector $\mathbf{b}$, and constraint function vector $\mathbf{c}(\mathbf{x})$. The constraint function vector can be defined to support both lower and upper bounds as $\mathbf{c}(\mathbf{x}) = [\mathbf{c}_l^T(\mathbf{x}), \mathbf{c}_u^T(\mathbf{x})]^T$ using

$$\mathbf{c}_l(\mathbf{x}) = \mathbf{x} - \mathbf{x}_l \geq 0, \quad \mathbf{c}_u(\mathbf{x}) = \mathbf{x}_u - \mathbf{x} \geq 0, \tag{24}$$

where $\mathbf{x}_l$ and $\mathbf{x}_u$ denote the lower and upper bounds, respectively. If the ith variable $\mathbf{x}_i$ does not have lower/upper bounds, we do not define any corresponding constraints. Considering that variables (e.g., $\mathbf{x}$) will be represented in single- or double-precision floating-point, the operators "$\geq$" and "$>$" can be used almost interchangeably because, when $\mathbf{c}(\mathbf{x}) = 0$, we can enforce $\mathbf{c}(\mathbf{x}) + \varepsilon > 0$ by introducing a tiny (physically negligible) positive offset value, ε (e.g., 10^{-16} for double-precision values). Hereafter, we consider that $\geq$ and $>$ are interchangeable for convenience, and thus $\log b$ and b^{-1} are valid for a scalar $b(\geq 0)$.

While the primal-dual interior point method can be formulated in multiple ways [36, §14, §16.6, §19], we derive its formulation via a barrier function for a slack variable. Given a slack variable $\mathbf{s} = [\mathbf{s}_l^T, \mathbf{s}_u^T]^T$ introduced for the inequality constraints $\mathbf{c}(\mathbf{x})$, i.e.,

$$\mathbf{c}(\mathbf{x}) - \mathbf{s} = 0, \ \mathbf{s} \geq 0, \tag{25}$$

we can formulate an augmented objective $L(\mathbf{x}, \mathbf{s}, \mathbf{z})$ with a Lagrange multiplier $\mathbf{z} = [\mathbf{z}_l^T, \mathbf{z}_u^T]^T (\geq 0)$, a log-barrier function for $\mathbf{s}$, and a barrier parameter μ as

$$L(\mathbf{x}, \mathbf{s}, \mathbf{z}) = f(\mathbf{x}) - \mu \sum_i \log \mathbf{s}_i - \mathbf{z}^T (\mathbf{c}(\mathbf{x}) - \mathbf{s}). \tag{26}$$

The associated Karush–Kuhn–Tucker (KKT) conditions are given as

$$\frac{\partial L}{\partial \mathbf{x}} = \mathbf{A}\mathbf{x} - \mathbf{b} - \mathbf{J}_l^T \mathbf{z}_l - \mathbf{J}_u^T \mathbf{z}_u = 0, \tag{27}$$

$$\frac{\partial L}{\partial \mathbf{s}_l} = -\mu \mathbf{S}_l^{-1} \mathbf{e}_l + \mathbf{z}_l = 0, \quad \frac{\partial L}{\partial \mathbf{s}_u} = -\mu \mathbf{S}_u^{-1} \mathbf{e}_u + \mathbf{z}_u = 0, \tag{28}$$

$$\frac{\partial L}{\partial \mathbf{z}_l} = -(\mathbf{x} - \mathbf{x}_l - \mathbf{s}_l) = 0, \quad \frac{\partial L}{\partial \mathbf{z}_u} = -(\mathbf{x}_u - \mathbf{x} - \mathbf{s}_u) = 0, \tag{29}$$

where the matrix $\mathbf{J}_l = \frac{\partial \mathbf{c}_l(\mathbf{x})}{\partial \mathbf{x}}$ is a constant diagonal matrix (i.e., $\mathbf{J}_l = \mathbf{J}_l^T$) with its ith diagonal element being 1 if $\mathbf{x}_i$ has a lower bound $\mathbf{x}_{l,i}$ and 0 otherwise, and similarly $\mathbf{J}_u = \frac{\partial \mathbf{c}_u(\mathbf{x})}{\partial \mathbf{x}}$ is a constant diagonal matrix with its ith diagonal element being -1 if $\mathbf{x}_i$ has an upper bound $\mathbf{x}_{u,i}$ and 0 otherwise. The matrices $\mathbf{S}_l$ and $\mathbf{S}_u$ are diagonal with their ith diagonal elements equal to $\mathbf{s}_{l,i}$ and $\mathbf{s}_{u,i}$, respectively, and $\mathbf{e}_l$ and $\mathbf{e}_u$ are vectors of all ones with the same dimensions as $\mathbf{s}_l$ and $\mathbf{s}_u$, respectively. To improve numerical conditioning, we can left-multiply (28) by $\mathbf{S}_l$ and $\mathbf{S}_u$ [36], obtaining

$$-\mu \mathbf{e}_l + \mathbf{S}_l \mathbf{z}_l = 0, \quad -\mu \mathbf{e}_u + \mathbf{S}_u \mathbf{z}_u = 0, \tag{30}$$

and arrive at a nonlinear system consisting of (27), (29), and (30).

5.2 Newton Step Direction

To find the optimal solution for the nonlinear system consisting of (27), (29), and (30), applying Newton's method (while symmetrizing its Jacobian) leads to the following unconstrained symmetric indefinite linear system (known as a primal-dual system [36]) to compute the Newton descent directions $\Delta\mathbf{x}$, $\Delta\mathbf{s}$, and $\Delta\mathbf{z}$:

$$\begin{bmatrix} \mathbf{A} & \mathbf{O} & -\mathbf{J}^T \\ \mathbf{O} & \mathbf{W} & \mathbf{I} \\ -\mathbf{J} & \mathbf{I} & \mathbf{O} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{s} \\ \Delta\mathbf{z} \end{bmatrix} = \begin{bmatrix} -\frac{\partial L}{\partial \mathbf{x}} \\ -\frac{\partial L}{\partial \mathbf{s}} \\ -\frac{\partial L}{\partial \mathbf{z}} \end{bmatrix}, \tag{31}$$

where $\mathbf{J} = \frac{\partial \mathbf{c}(\mathbf{x})}{\partial \mathbf{x}}$, $\mathbf{O}$ denotes the zero matrix, $\mathbf{I}$ the identity matrix, and $\mathbf{W} = \mathbf{Z}\mathbf{S}^{-1}$ ($\mathbf{S}$ and $\mathbf{Z}$ are diagonal matrices with their ith diagonal elements set to $\mathbf{s}_i$ and $\mathbf{z}_i$, respectively). Solving this large symmetric indefinite system (31) (whose size can be up to $5N \times 5N$ when $\mathbf{x}$ has length N) yields the correct descent direction; however, it is possible to derive a much smaller SPD system given that there are a large number of inequality constraints $\mathbf{c}(\mathbf{x})$. By first eliminating $\Delta\mathbf{s}$ from (31), we obtain

$$\begin{bmatrix} \mathbf{A} & -\mathbf{J}^T \\ -\mathbf{J} & -\mathbf{W}^{-1} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{z} \end{bmatrix} = \begin{bmatrix} -\frac{\partial L}{\partial \mathbf{x}} \\ \mathbf{W}^{-1} \frac{\partial L}{\partial \mathbf{s}} - \frac{\partial L}{\partial \mathbf{z}} \end{bmatrix}, \tag{32}$$

and then eliminating $\Delta\mathbf{z}$, we arrive at the smaller SPD system

$$\left(\mathbf{A} + \mathbf{J}^T \mathbf{W} \mathbf{J}\right) \Delta\mathbf{x} = -\frac{\partial L}{\partial \mathbf{x}} + \mathbf{J}^T \left(\mathbf{W} \frac{\partial L}{\partial \mathbf{z}} - \frac{\partial L}{\partial \mathbf{s}}\right). \tag{33}$$

Furthermore, by explicitly representing subscripts l and u, introducing Newton iteration index k, and further simplifying the system with (27) and (28), we finally obtain

$$\left(\mathbf{A} + \sum_{i=l,u} \mathbf{J}_i^T \mathbf{W}_i^k \mathbf{J}_i\right) \Delta\mathbf{x}^k = -\frac{\partial L^k}{\partial \mathbf{x}^k} + \sum_{i=l,u} \mathbf{J}_i^T \left(\mathbf{W}_i^k \frac{\partial L^k}{\partial \mathbf{z}_i^k} - \frac{\partial L^k}{\partial \mathbf{s}_i^k}\right) \tag{34}$$

$$= \mathbf{b} - \mathbf{A}\mathbf{x}^k + \sum_{i=l,u} \mathbf{J}_i^T \left(\mu [\mathbf{s}_i^k]^{-1} + \mathbf{W}_i^k \frac{\partial L^k}{\partial \mathbf{z}_i^k}\right), \tag{35}$$

where we define $\mathbf{W}_l = \mathbf{Z}_l \mathbf{S}_l^{-1}, \mathbf{W}_u = \mathbf{Z}_u \mathbf{S}_u^{-1}$, let $\mathbf{Z}_l$ and $\mathbf{Z}_u$ denote diagonal matrices with ith diagonal elements equal to $\mathbf{z}_{l,i}$ and $\mathbf{z}_{u,i}$, respectively, and let $[\mathbf{s}_l]^{-1}$ and $[\mathbf{s}_u]^{-1}$ denote elementwise inverses of $\mathbf{s}_l$ and $\mathbf{s}_u$, respectively. This system is SPD, and its size is $N \times N$. In addition, as $\sum_{i=l,u} \mathbf{J}_i^T \mathbf{W}_i^k \mathbf{J}_i$ in (34) is diagonal due to the special structures of box constraints, the resulting system matrix is an M-matrix since $\mathbf{A}$ is an M-matrix. This diagonality also lets us efficiently update the system matrix stored in the CSR format (if necessary, we can precompute indices to access diagonal elements in advance). In addition, we compute and store $\frac{\partial L_l^k}{\partial \mathbf{z}_l^k}$ and $\frac{\partial L_u^k}{\partial \mathbf{z}_u^k}$ here for later use in (37).

After we obtain $\Delta\mathbf{x}^k$ by solving the reduced unconstrained linear system (34) using AMGCG (see §4 and §5.4) or any other linear solver, we can recover the eliminated descent directions $\Delta\mathbf{s}^k$ and $\Delta\mathbf{z}^k$ by

$$\Delta\mathbf{s}^k = \mathbf{J}\Delta\mathbf{x}^k - \frac{\partial L^k}{\partial \mathbf{z}^k}, \quad \Delta\mathbf{z}^k = -\mathbf{W}^k \Delta\mathbf{s}^k - \frac{\partial L^k}{\partial \mathbf{s}^k}, \tag{36}$$

and thus $\Delta\mathbf{s}_l^k, \Delta\mathbf{s}_u^k, \Delta\mathbf{z}_l^k$, and $\Delta\mathbf{z}_u^k$ are given by

$$\Delta\mathbf{s}_i^k = \mathbf{J}_i \Delta\mathbf{x}^k - \frac{\partial L^k}{\partial \mathbf{z}_i^k}, \quad \Delta\mathbf{z}_i^k = -\mathbf{W}_i \Delta\mathbf{s}_i^k + \mu [\mathbf{s}_i^k]^{-1} - \mathbf{z}_i^k, \tag{37}$$

where $i = l, u$.

Notably, applying Newton's method directly to the augmented objective (26) instead to minimize it without going through (30) leads to a so-called primal system, which is more ill-conditioned compared to the primal-dual system (31) [36]. Subsequently eliminating the dual and slack variables (also via $\mathbf{s} = \mathbf{c}(\mathbf{x})$) leads to a formulation exactly the same as the one derived from the log-barrier method only with primal variables $\mathbf{x}$.

5.3 Solution Update

In the traditional primal-dual interior point method for nonlinear functions, or even for QP with quadratic objectives, it is typically necessary to select an appropriate step length (e.g., via line search). Doing so avoids violating the imposed constraints for the primal variable $\mathbf{x}$ and the non-negative constraints for the slack variable $\mathbf{s} \geq 0$ and Lagrange multiplier $\mathbf{z} \geq 0$ (or to decrease a corresponding merit function) when updating $\mathbf{x}$, $\mathbf{s}$, and $\mathbf{z}$ [36], and is needed because violating these constraints can make the solver diverge or fail to converge to correct solutions. However, the step size chosen to satisfy these constraints can often be small, which significantly slows down the convergence of the solver. Therefore, we exploit two properties of our problem: the uniqueness of the solution for the convex QP and the fact that the constraints are specifically box constraints. Since a convex QP has a unique solution [36], it is still possible to converge to the correct solution even if solution candidates temporarily violate the constraints and are projected back into the valid solution domain. In addition, due to the box shape of the constraints, violated variables can be easily projected into the valid solution domain using simple clamping, unlike for general QPs with linear constraints (which would require solving another expensive constraint minimization problem just for the projection) [36]. Specifically exploiting these features, we simply update $\mathbf{x}$, $\mathbf{s}$, and $\mathbf{z}$ (without step selection) using

$$\mathbf{x}^{k+1} = \text{clamp}(\mathbf{x}^k + \Delta\mathbf{x}^k, \mathbf{x}_l, \mathbf{x}_u), \tag{38}$$

$$\mathbf{s}^{k+1} = \max(\mathbf{s}^k + \Delta\mathbf{s}^k, \varepsilon), \quad \mathbf{z}^{k+1} = \max(\mathbf{z}^k + \Delta\mathbf{z}^k, \varepsilon). \tag{39}$$

5.4 Implementation Details

The algorithm of our primal-dual box-constrained convex QP solver is given in Algorithm 3. We initialize $\mathbf{x}^0$ with $\mathbf{x}^0 = \text{clamp}(0, \mathbf{x}_l, \mathbf{x}_u)$ taking the box constraints into account. For the slack and Lagrange multiplier variables, we initialize with $\mathbf{s}^0 = \mathbf{e}$ and $\mathbf{z}^0 = \mathbf{e}$ to ensure $\mathbf{s} \geq 0$ and $\mathbf{z} \geq 0$, avoiding the too close proximity of the constraint boundaries [36].

Given the homotopy nature of the traditional interior point methods, they gradually reduce the barrier parameter μ toward 0 as the solution candidates approach the correct solution. However, we found that if μ is far from 0 then the nonlinear system due to the KKT conditions (27), (28), and (29) can generate local minima which differ from the correct solution; furthermore, clamping the solution candidates makes it more likely to erroneously converge to such local minima. As such,

Algorithm 3 Primal-Dual Box-Constrained Convex QP Solver

$k = 0, \mu = 10^{-20}$
Initialize $\mathbf{x}^k = \text{clamp}(0, \mathbf{x}_l, \mathbf{x}_u)$, $\mathbf{s}^k = \mathbf{e}$, and $\mathbf{z}^k = \mathbf{e}$
Compute $\mathbf{A}$
do
  Compute the right hand side using (35)
  Update the diagonals of the system matrix as (34)
  Obtain $\Delta\mathbf{x}^k$ by solving (34) with AMGCG (see §4)
  Recover $\Delta\mathbf{s}^k$ and $\Delta\mathbf{z}^k$ by (37)
  Update to $\mathbf{x}^{k+1}$ by (38) and $\mathbf{s}^{k+1}$ and $\mathbf{z}^{k+1}$ by (39)
  $k = k + 1$
while $\max(\|\Delta\mathbf{x}\|_2, \|\Delta\mathbf{s}\|_2, \|\Delta\mathbf{z}\|_2) > \varepsilon_N$

we initialize and fix the barrier parameter as $\mu = 10^{-20}$. This choice ensures that the correct and unique solution remains the same during Newton iterations.

Given the correct unique solution during the optimization procedure, we prefer using an inexact Newton's method because performing more Newton iterations is typically more efficient than spending more time to obtain more accurate Newton descent directions [36]. We therefore apply early termination of AMGCG iterations for (34) when obtaining the Newton step directions. While performing just one AMGCG iteration is minimal, we observed that the computed descent direction was typically not accurate enough, often failing to converge to correct solutions. Considering that CG is a *rougher* (as opposed to smoother) [39], it was necessary to perform at least a few AMGCG iterations to obtain sufficiently accurate Newton step directions in our experiments. We typically use up to three AMGCG iterations to balance efficiency and accuracy.

We terminate the Newton iterations based on the maximum L2 norm of the Newton step directions. Specifically, we evaluate whether $\max(\|\Delta\mathbf{x}\|_2, \|\Delta\mathbf{s}\|_2, \|\Delta\mathbf{z}\|_2) < \varepsilon_N$ holds or not (where ε_N denotes the maximum error threshold). This condition allowed us to reliably obtain converged visual results in our experiments.

6 Application to Diffusion Equations

As an additional application of our QP solver, we consider an implicit heat diffusion with box constraints, which can specifically be written as

$$(\mathbf{I} + \alpha\mathbf{L})\mathbf{q} = \mathbf{q}^t, \quad \text{s.t.} \quad \mathbf{q}_l \le \mathbf{q} \le \mathbf{q}_u, \tag{40}$$

where $\mathbf{I}$ and $\mathbf{L}$ denote the identity and Laplacian matrices, $\alpha(>0)$ a scaling parameter (including diffusion coefficient and time step size), $\mathbf{q}$ and $\mathbf{q}^t$ the next and current temperature, respectively, t time, $\mathbf{q}_l$ and $\mathbf{q}_u$ lower and upper bounds for $\mathbf{q}$, respectively. As both $\mathbf{I}$ and $\mathbf{L}$ are SPD M-matrices, $\mathbf{I} + \alpha\mathbf{L}$ is also an SPD M-matrix. Given the equivalence of the KKT conditions of the box-constrained QPs to this linear system with box constraints, we can reformulate (40) into a quadratic form as

$$\mathbf{q} = \underset{\mathbf{q}_l \le \mathbf{q} \le \mathbf{q}_u}{\arg\min} \frac{1}{2}\mathbf{q}^T\mathbf{A}\mathbf{q} - \mathbf{q}^T\mathbf{q}^t, \tag{41}$$

where $\mathbf{A} = \mathbf{I} + \alpha\mathbf{L}$. This box-constrained QP can efficiently be handled using our barrier-based QP solver.

7 Results and Discussions

We implemented our method in C++17 with double-precision floating-point for scalar values and parallelized it using OpenMP. All examples used adaptive timestepping with CFL numbers of 3.0 or larger (empirically chosen based on visual quality) with 60 frames per second. Unless otherwise mentioned, we use the non-negative pressure constraints defined by (2), termination norm of 10^2 (which is around 10^{-4} of the relative norm) for Newton iterations, and termination relative residual of 10^{-4} for CG. We executed two scenarios (Figure 4 and Figure 8) on a cluster with dual Intel Xeon Silver 4314 (each has 16 cores, and thus totaling to 32 cores) and 256GB RAM, and the other scenarios on a desktop machine with an Intel Core i7-9700 (8 cores) with 16GB RAM.

Table 1: Simulation settings and results for Figure 4, where the system size is 2.4M with 16.3M non-zeros. The time for preconditioning preparation is denoted by T_p, CG solve by T_s, the number of CG iterations per solve by N, and total time for the pressure solve phase (including level-set computations, valid cell evaluations, operator construction, system assembly, and system solve) by T. Timing is given in average seconds for a single frame. Z_{PR} and Z_A denote the sum of non-zeros over the MG hierarchy for $\mathbf{P}+\mathbf{R}$ and $\mathbf{A}$ (including the top level), respectively. Z_{IC} denotes the number of non-zeros in the lower triangular matrix computed by (modified) incomplete Cholesky.

Scheme	T_p	T_s	N	T	Z_{PR}	Z_A	Z_{IC}
MIC	0.68	22.10	505.4	26.85			9.3M
SAAMG	1.15	0.66	18.9	5.38	10.1M	27.3M	
GIAAMG	0.74	2.01	26.4	6.57	2.8M	18.6M	
TAGIAAMG	0.79	0.53	6.1	5.09	2.9M	19.0M	

7.1 Preconditioning Evaluation for Linear Solve

To evaluate the efficacy of our topology-aware GIAAMG approach, we compare various preconditioning schemes for CG. We experiment with fluids in a maze-like structure with purely Neumann boundaries and continuously added external forces [43], using a grid resolution of 160^3, as shown in Figure 4. We compare the following preconditioning schemes:

1. MIC: baseline [9];
2. SAAMG: smoothed aggregation AMG [48];
3. GIAAMG: [38, 51];
4. TAGIAAMG (ours): our topology-aware GIAAMG.

We use V-cycle and SPAI-0 for SAAMG and W-cycle and RBSSOR for GIAAMG and TAGIAAMG. We use a single simulation step per frame, and termination relative residual of 10^{-8} for CG. Figure 5 compares profiles of the computational cost for the linear system solve (i.e., preconditioning preparation and CG solve), and Table 1 summarizes the simulation settings and averaged results.

As all the approaches fully converged, they generated comparable visual results while MIC was much slower than the others, and was not particularly memory efficient compared to TAGIAAMG (9.3M + 16.3M vs. 2.9M + 19.0M). Compared to the topology-oblivious GIAAMG approach, SAAMG and TAGIAAMG require fewer CG iterations because these schemes enable more effective preconditioning by respecting the fluid topology even at the coarser levels. Consequently, TAGIAAMG is faster than GIAAMG by a factor of around 3.8 in the CG solve while having a negligible additional cost for preconditioning preparation. As SAAMG requires smoothing the prolongation operator, the preconditioning cost is more expensive than that of TAGIAAMG. In addition, using W-cycle and RBSSOR reduces the necessary iterations more than V-cycle with SPAI-0 does, resulting in a slightly better performance for TAGIAAMG over SAAMG. In addition, since the restriction/prolongation operators and coarser-level systems are much sparser, TAGIAAMG used around half of memory compared to SAAMG.

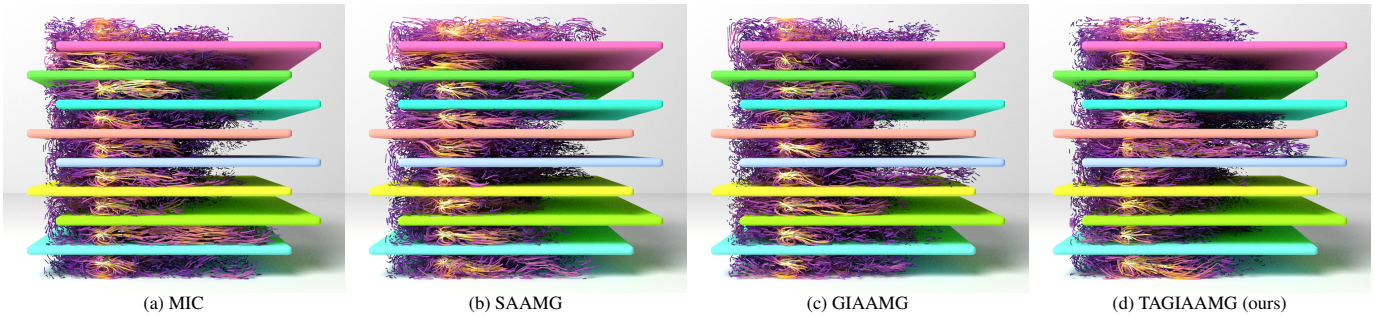
7.2 Box-Constrained QP Solver Evaluation

We conduct two evaluations. The first evaluation is to justify our QP solver design by comparing various variants of barrier-based QP solvers (§7.2.1). The second one is to evaluate the characteristics of MPRGP and our solver with different preconditioners (§7.2.2).

7.2.1 Barrier-based QP Solvers

We employ a 3D cubic domain, where a source term is placed on the domain center while the outermost boundaries are fixed with a zero pressure Dirichlet boundary condition and an upper bound imposed on the solution. We experiment with three different resolutions: $64^3, 128^3$, and 256^3. We compare the following schemes:

1. PLB: primal log-barrier method [36];
2. IPM-P: interior point method that solves the primal system [36];



(a) MIC (b) SAAMG (c) GIAAMG (d) TAGIAAMG (ours)

Fig. 4: Fluids in a maze-like domain with purely Neumann boundaries and continuously added external forces, simulated with different preconditioners for CG (see §7.1). Particles are color-coded based on the fluid velocities. All approaches generate comparable visual results.

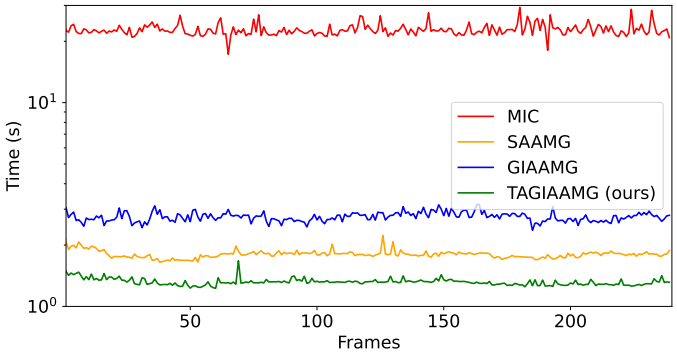


Fig. 5: Log-scale profiles of the preconditioning preparation + CG solve time per frame. Our TAGIAAMG achieves $17.3\times$, $1.4\times$, and $2.1\times$ faster performance than MIC, SAAMG, and GIAAMG, respectively.

Table 2: Evaluation of various box-constrained convex QP solver variants using resolutions of 64^3, 128^3, and 256^3. Numbers represent the solve time (Newton iteration count). Timing is shown in seconds.

Scheme \ Res	64^3	128^3	256^3
PLB	1.24 (16)	13.72 (24)	102.97 (23)
IPM-P	1.27 (17)	13.71 (25)	114.54 (25)
IPM-PD	0.74 (9)	5.52 (9)	45.63 (9)
IPM-PD+HS	4.00 (53)	33.36 (56)	281.59 (58)
IPM-PD+CAB+SS	2.02 (31)	21.06 (36)	203.02 (41)
Mehrotra	1.49 (15)	14.73 (15)	121.88 (16)

3. IPM-PD (ours): IPM that solves the primal-dual system;
4. IPM-PD+HS: IPM-PD with halved steps;
5. IPM-PD+CAB+SS: IMP-PD using complementarity-based adaptive barrier parameter tuning with step selection [36];
6. Mehrotra: Mehrotra predictor-corrector method [36].

With PLB, IPM-P, and IPM-PD (ours), we aggressively update solution candidates without using any step selection procedure and clamp them if constraint violation occurs. With IPM-PD+HS, we update the solution candidate in the same way but with halved step lengths. (Although we have tested step selection with IPM-PD, it stagnated due to the need for excessively small steps). With IPM-PD+CAB+SS and Mehrotra, we use step selection. With PLB, IPM-P, IPM-PD (ours), and IPM-PD+HS, we use a fixed barrier parameter, while with IPM-PD+CAB+SS and Mehrotra, the barrier parameters are adjusted following their methods. In this comparison, we uniformly set 10^{-7} as their termination Newton step norm since IPM-P and IPM-PD+CAB+SS did not converge to a norm value lower than that due to numerical issues while other approaches were able to reach a norm value lower than 10^{-10}. (In practice, due to better numerical conditioning, IPM-PD and Mehrotra methods lead to much smaller residual than PLB and IPM-P even with the same termination norm). Table 2 summarizes the evaluation settings and results.

Table 3: Simulation setting and results using the maze scenario, simulated with a grid resolution of 144^3 over five frames. The system size is around 1.4M. The number of outer Newton iterations is denoted by M, the number of CG/MPRGP iterations per solve by N, and total time (s) for the QP solve and entire pressure solve phase per frame by T_s and T, respectively. The top three methods are linear solvers included only for comparison.

Scheme	M	N	T_s	T
MICCG		434.0	13.44	17.54
SAAMG-CG		12.0	1.06	5.41
TAGIAAMG-CG		6.0	0.73	5.30
MIC-MPRGP		1,264.0	55.25	60.42
SAAMG-MPRGP		233.2	13.58	18.47
IPM-PD+MICCG	23.4	100.0	62.67	67.78
IPM-PD+SAAMG-CG	25.0	2.0	12.38	17.13
IPM-PD+TAGIAAMG-CG	22.2	2.0	7.78	12.94

Among the tested variants, our IPM-PD was fastest and most scalable, exhibiting the smallest Newton iteration counts. As PLB and IPM-P solve essentially the same primal system, their computational cost and Newton iteration counts were comparable. Their performance is worse than our IPM-PD due to their ill-conditioned primal system compared to the primal-dual system because the resulting Newton descent direction can neglect the constraints, making the projected variables significantly deviate from the correct solution [21]. On the other hand, using halved steps to mitigate the constraint violation significantly slows down convergence with IPM-PD+HS. Similarly, IPM-PD+CAB+SS was slow to converge due to small steps chosen during step selection even though the barrier parameter is adjusted with the complementarity-based approach [36]. The Mehrotra predictor-corrector method was able to reliably and scalably converge to the correct solution with relatively few Newton iterations although its performance was not fast enough because this approach needs two linear solves (despite it being the same system matrix) for prediction and correction [36].

7.2.2 Characteristics of MPRGP and IPM-PD

Next, we experiment with various preconditioners to evaluate characteristics of MPRGP and IPM-PD using the maze scenario (Figure 6). We use a grid resolution 144^3 and only the first five frames. We compare the following schemes:

1. MICCG: reference [9];
2. SAAMG-CG: reference [48];
3. TAGIAAMG-CG: reference;
4. MIC-MPRGP: [34];

Table 4: Simulation setting and results for Figure 6. The number of Newton iterations denoted by M, the number of CG/MPRGP iterations per solve by N, the number of substeps per frame by s, and total time (s) for the system solve and entire pressure solve phase per frame by T_s and T, respectively. The top pure linear solver method is included for comparison.

Scheme	M	N	s	T_s	T
TAGIAAMG-CG		3.5	1.9	0.80	9.56
SAAMG-MPRGP		273.4	2.4	23.87	35.84
IPM-PD+SAAMG-CG.	18.1	2.0	2.3	16.39	28.39
IPM-PD+TAGIAAMG-CG	16.2	2.0	2.4	10.00	22.33

5. SAAMG-MPRGP: [43];
6. IPM-PD+MICCG;
7. IPM-PD+SAAMG-CG;
8. IPM-PD+TAGIAAMG-CG (ours);

The first three methods (denoted 'reference') are linear solver methods included only for comparison, and thus do not actually solve the QP. We use a termination relative residual of 10^{-4} for CG/MPRGP and employ the fast active-set expansion technique [29, 42]. To compare MPRGP and IPM-PD in a fair condition, we terminate IPM-PD when its absolute residual (derived from the quadratic objective) is comparable to that of MPRGP. With IPM-PD, while we use up to two SAAMG-CG/TAGIAAMG-CG iterations, we use up to 100 MICCG iterations, as MICCG needed many iterations to achieve convergence of IPM-PD. Table 3 summarizes the evaluation results.

With MIC, because its preconditioning effectiveness is limited, the performance of MPRGP is not dictated by active-set updates, causing MIC-MPRGP to be more efficient than IPM-PD+MICCG. By contrast, with SAAMG, the performance of MPRGP is dictated by the active-set updates, making IPM-PD+SAAMG-CG more efficient than SAAMG-MPRGP. In addition, using TAGIAAMG-CG as an inner solver further improves the performance, and IPM-PD+TAGIAAMG-CG is around $1.7\times$ faster than SAAMG-MPRGP [43] in the QP solve.

7.3 Separating Solid Boundary

To evaluate the efficacy of our solvers, we compare our method with the state-of-the-art box-constrained QP solver, SAAMG-MPRGP [43], using a maze scenario, as shown in Figure 6. We use a grid resolution of 128^3 and 6.0M particles. We compare the following solvers:

1. TAGIAAMG-CG: reference;
2. SAAMG-MPRGP: [43];
3. IPM-PD+SAAMG-CG;
4. IPM-PD+TAGIAAMG-CG (ours).

The 'reference' TAGIAAMG-CG scheme is a linear solver approach included for comparison purposes. Although we have tested the quadratic penalty method [36] and augmented Lagrangian method (ALM) [35,36] (without using clamping to avoid solver stagnation [42]), these approaches did not converge to the correct solution with moderately large penalty parameters and they stagnated with very large penalty parameters due to numerical issues. Considering these problems and the difficulty of choosing appropriate penalty parameters for stable convergence, we did not consider them in this comparison. We also did not consider TAGIAAMG-MPRGP as it is nontrivial to support our optimized operations that utilize red-black coloring along with active sets. We use a termination relative residual of 10^{-3}. Figure 7 compares profiles of the computational cost for the system solve, and Table 4 summarizes the simulation settings and averaged results.

Unlike TAIGAAMG-CG for reference, SAAMG-MPRGP, IPM-PD+SAAMG-CG, and IPM-PD+TAGIAAMG-CG correctly generated wall-separating liquid behaviors. Similar to the previous experiment, as the performance of SAAMG-MPRGP is dictated by the active-set updates, IPM-PD+SAAMG-CG was more efficient, enabling further acceleration and memory usage reduction with TAGIAAMG-CG. Consequently, IPM-PD+TAGIAAMG-CG achieved $2.4\times$ and $1.6\times$ faster performance with around half the memory usage for the MG hierarchy as compared to SAAMG-MPRGP and IPM-PD+SAAMG-CG, respectively.

Table 5: Simulation setting and results for Figure 8. The number of non-negative constraints denoted by c, the number of Newton iterations by M, the number of CG iterations per solve by N, the number of substeps per frame by s, and total time (s) for the system solve and entire pressure solve phase per frame by T_s and T, respectively.

Scheme	c	M	N	s	T_s	T
Unconstrained	0		3.3	5.2	4.06	14.63
Non-splashy	0.1M	6.9	2.0	5.1	15.94	25.89
Splashy	1.4M	13.2	2.0	5.2	26.20	36.64

7.4 Splashy vs. Non-Splashy

To evaluate the influence of the number of non-negative pressure constraints on the performance, we compare liquid simulations in the double dam break scenario, as shown in Figure 8. We used a grid resolution of 160^3 and 8.0M particles. We compare the following formulations:

1. Unconstrained: reference without any non-negative constraints;
2. Non-splashy: Non-negative constraints only at the solid boundaries with (3);
3. Splashy: Non-negative constraints everywhere with (2).

We use TAGIAAMG-CG for the unconstrained formulation and IPM-PD+TAGIAAMG-CG for the non-splashy and splashy constrained formulations. Figure 9 compares profiles of the computational cost for the system solve, and Table 5 summarizes the simulation settings and averaged results.

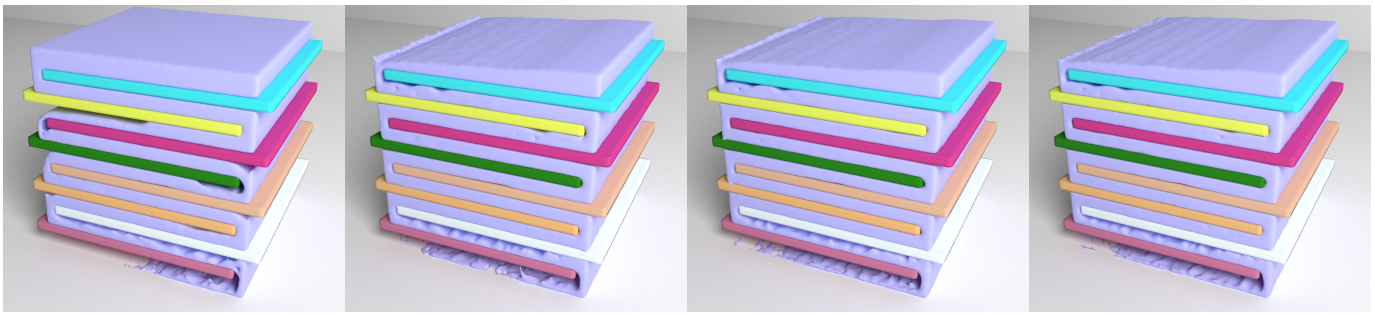
As expected, the unconstrained formulation does not support wall separation, while both non-splashy and splashy formulations generate plausible wall-separating liquid behaviors. The choice between non-splashy and splashy formulations depends on the particular application, but the splashy formulation (2) introduces $12.5\times$ as many non-negative constraints as the non-splashy one (3), and thus requires around $1.9\times$ more iterations and $1.6\times$ more time for the QP solve.

7.5 Dynamic Solid Boundaries

To demonstrate the ability of our solver to support dynamic solid boundaries, we simulate a liquid bunny in a moving solid sphere, as shown in Figure 1. We used a grid resolution of 192^3 and 1.0M particles. The entire pressure solve phase required 24.1 seconds per frame on average. As the contribution from the moving sphere can seamlessly be encoded into the quadratic objective (1) on a Cartesian grid structure [7], our QP solver can handle this scenario, in the same manner as static solid boundaries.

7.6 Box-Constrained Heat Diffusion

To demonstrate the versatility of our QP solver, we simulate heat diffusion with lower and upper temperature bounds on a static dragon, discretized on a Cartesian grid (using a resolution of 256^3 and 2.2M active cells) and rendered as color-coded particles, as shown in Figure 2. We initialize the temperature of the dragon to 50 (K), fix the middle left and right bands at 0 and 100 (K) as Dirichlet boundary conditions, respectively, specify outside of the dragon as Neumann boundary conditions, and set lower and upper temperature bounds as 20 and 80 (K), respectively. Once simulation begins, heat starts diffusing, and it reaches the equilibrium. Our QP solver correctly handles the diffusion process and enforces the temperature bounds throughout the simulation, reaching an equilibrium different from the one generated without the bounds.



(a) TAGIAAMG-CG (b) SAAMG-MPRGP (c) IPM-PD+SAAMG-CG (d) IPM-PD+TAGIAAMG-CG

Fig. 6: Liquids in a maze-like structure with separating solid-boundary conditions, simulated with different pressure solvers (see §7.3). TAGIAAMG-CG exhibits suction artifacts on the ceiling, while SAAMG-MPRGP, IPM-PD+SAAMG-CG, and IPM-PD+TAGIAAMG-CG generate wall-separating liquid behaviors and are visually comparable to each other.

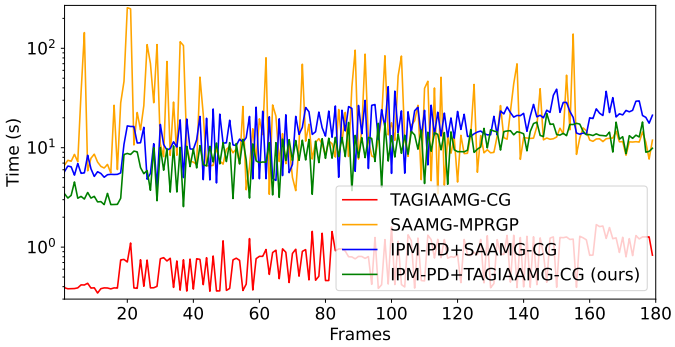


Fig. 7: Log-scale profiles of the system solve time per frame. Our IPM-PD+TAGIAAMG achieves $2.4\times$ and $1.6\times$ faster performance than SAAMG-MPRGP and IPM-PD+SAAMG-CG, respectively.

8 Conclusions and Future Work

We have proposed and evaluated an effective new solver strategy for pressure Poisson equations with non-negative pressure constraints that uses an active-set-free primal-dual interior point scheme combined with AMG preconditioning of the inner CG solves. We achieved additional speedups by developing and optimizing our topology-aware geometry-inspired aggregation AMG preconditioner. Below we discuss some tradeoffs inherent in our approach and highlight exciting directions for future work.

8.1 Multigrid

Our MG approach is designed to be effective in handling general and practical liquid simulation scenarios; however, other MG schemes may be more effective in certain special cases. For example, if boundary conditions are not complex, GMG or topology-oblivious GIAAMG approaches are conceptually simpler and can be implemented in a matrix-free way, performing the necessary operations on the fly and thus reducing memory usage and computational cost [33, 38]. If systems are unstructured or have no meaningful geometric structures to exploit, purely algebraic MG would be the only practical choice among the other MG schemes [12]. In addition, even when meaningful geometric information (e.g., regular grid structures) is available, using such data effectively in the solver does introduce additional complexity and require extra implementation effort, making the code maintenance and optimization more cumbersome. As such, computation and memory efficiency gains from exploiting geometric information may not necessarily be worthwhile in practice for some applications.

8.2 Unconstrained Inner Linear Systems

Our barrier-based box-constrained convex QP solver involves no active sets, making the inner linear systems fully unconstrained, and as such, various linear solvers and preconditioners can be used to solve the inner linear systems. By contrast, MPRGP [17, 18] needs specially designed preconditioning schemes for MIC and AMG to take active sets into account in the linear preconditioning system [43]. This property of our solver (i.e., unconstrainedness) is favorable and allows us to employ various approaches that would be nontrivial to use in the presence of constraints (or active sets), e.g., red-black GS-type solvers/smoothers (as used in our work), direct solvers based on Cholesky factorization and triangular solves, preconditioned CG [39], sequential multiplication [9, 41], or block GS decomposition with specialized solvers for each subsystem [1, 23]. In particular, as unconstrained problems are much easier to handle than constrained ones, reformulating constrained systems into unconstrained ones using our approach appears to be a promising avenue for addressing challenging systems featuring non-M-matrices.

8.3 Interior Point Method

Due to the tiny offset value ε and barrier parameter μ, barrier-based approaches are relatively sensitive to numerical issues even with double-precision floating-point, e.g., round-off errors and loss of significance (catastrophic cancellation), and might suffer from either erroneous early termination or changes in the optimization results that depend on parameter values and computation orders. While our primal-dual method was more robust than approaches based on the primal system, it might be recommended to use the more numerically stable Mehrotra method in some applications [36]. In addition, although our method was most efficient among tested variants in our experiment, it is possible that these approaches might outperform ours depending on scenarios and choices of algorithm-specific parameters. Holistically investigating various sets of initialization values, parameter values, and update rules would be beneficial but lies beyond the scope of our paper, and hence is left for future work.

Unlike MPRGP [17, 18], our method requires updating the system matrix in each outer Newton iteration although the update is limited only to its diagonal entries. As we prepare AMG preconditioning for each new system, the overhead for preconditioner preparation can be nonnegligible, especially with early termination. As such, it might be beneficial to reuse the previous MG hierarchies at the coarser levels, given that changes are limited to the diagonals.

To accelerate convergence, we exploited the fact that our application gives rise only to box constraints by aggressively updating solution candidates while projecting them back to the bounds when constraint violation occurs. However, for applications involving general linear inequality constraints, projecting the solution candidates to the nearest point in the valid solution domain would itself require another expensive QP solve [36]. Alternatively, one could sequentially project variables one by one, but this latter approach does not guarantee convergence despite the solution to the convex QP being unique. It would therefore be beneficial to respect the linear constraints (without violating them) within each Newton iteration, as done by step-selection approaches, e.g., IPM-PD+CAB+SS and Mehrotra methods [36].

(a) Unconstrained (b) Non-splashy (c) Splashy

Fig. 8: Double dam break simulated with different pressure formulations (see §7.4). The unconstrained formulation exhibits the usual suction artifact on the ceiling, but both non-splashy and splashy formulations correctly generate natural wall-separating liquid behaviors.

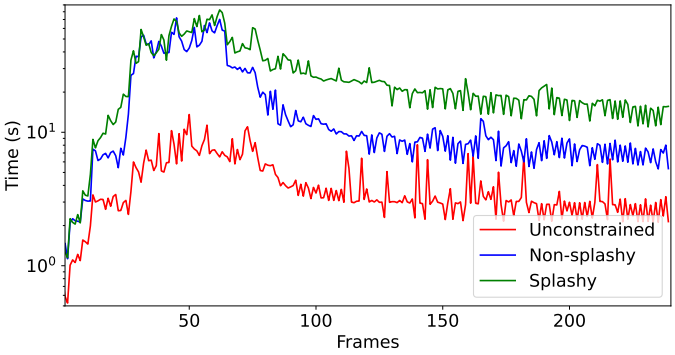


Fig. 9: Log-scale profiles of the system solve time per frame. As the number of non-negative constraints increases, our QP solver requires more iterations and thus becomes more expensive.

8.4 Box-Constrained Convex QP Applications and Beyond

Since our method is designed for box-constrained convex QPs with M-matrices, it seems promising to apply our solver to the componentwise frictional stress QP problems for granular flows [34, 42]. On the other hand, our solver may not offer ideal convergence rates for box-constrained convex QPs involving non-M-matrices, which arise in many settings, including frictional contact handling [6], skinning for non-negative least square [26] and bounded biharmonic weighting [25], cubature optimization [4], or level-set smoothing [8]. As such, it would be worthwhile to explore effective preconditioners and solvers for non-M-matrices (e.g., via better coarsening schemes, cycles, and smoothers), demonstrating their efficacy over preconditioned MPRGP [22, 34, 43]. In addition, considering that past authors have attempted to apply MPRGP (which was originally developed for box-constrained convex QPs) to general QPs [32], it could also be fruitful to apply our method to general QPs, given the versatility of primal-dual interior point methods, and to evaluate how it compares to existing QP solvers [14].

Acknowledgments

We would like to thank the anonymous reviewers for their valuable suggestions and comments. This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (Grant RGPIN-2021-02524).

References

[1] M. Aanjaneya. An efficient solver for two-way coupling rigid bodies with incompressible flow. *Computer Graphics Forum*, 37(8):59–68, 2018. 10

[2] M. Aanjaneya, C. Han, R. Goldade, and C. Batty. An efficient geometric multigrid solver for viscous liquids. *Proc. ACM Comput. Graph. Interact. Tech.*, 2(2), July 2019. 3

[3] M. Adams, M. Brezina, J. Hu, and R. Tuminaro. Parallel multigrid smoothing: Polynomial versus gauss–seidel. *J. Comput. Phys.*, 188(2):593–610, jul 2003. 5

[4] S. S. An, T. Kim, and D. L. James. Optimizing cubature for efficient integration of subspace deformations. *ACM Trans. Graph.*, 27(5), dec 2008. 11

[5] M. Andersen, S. Niebe, and K. Erleben. A fast linear complementarity problem (lcp) solver for separating fluid-solid wall boundary conditions. In *Proceedings of the 13th Workshop on Virtual Reality Interactions and Physical Simulations*, VRIPHYS '17, p. 39–48, 2017. 2

[6] S. Andrews, K. Erleben, and Z. Ferguson. Contact and friction simulation for computer graphics. In *ACM SIGGRAPH 2022 Courses*, SIGGRAPH '22, 2022. 11

[7] C. Batty, F. Bertails, and R. Bridson. A fast variational framework for accurate solid-fluid coupling. *ACM Trans. Graph.*, 26(3), 2007. 1, 2, 4, 9

[8] H. Bhattacharya, Y. Gao, and A. W. Bargteil. A level-set method for skinning animated particle data. *IEEE Transactions on Visualization and Computer Graphics*, 21(3):315–327, mar 2015. 11

[9] R. Bridson. *Fluid Simulation for Computer Graphics*. A K Peters/CRC Press, 2015. 2, 5, 7, 8, 10

[10] W. Briggs, V. Henson, and S. Mccormick. *A Multigrid Tutorial, Second Edition*. Society for Industrial and Applied Mathematics, 2000. 1, 2, 3, 4

[11] O. Bröker, M. J. Grote, C. Mayer, and A. Reusken. Robust parallel smoothing for multigrid via sparse approximate inverses. *SIAM Journal on Scientific Computing*, 23(4):1396–1417, 2001. 2, 4

[12] N. Chentanez, B. E. Feldman, F. Labelle, J. F. O'Brien, and J. R. Shewchuk. Liquid simulation on lattice-based tetrahedral meshes. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '07, p. 219–228, 2007. 10

[13] N. Chentanez and M. Müller. A multigrid fluid pressure solver handling separating solid boundary conditions. *IEEE Transactions on Visualization and Computer Graphics*, 18(8):1191–1201, 2012. 1, 2

[14] K. Cheshmi, D. M. Kaufman, S. Kamil, and M. M. Dehnavi. Nasoq: Numerically accurate sparsity-oriented qp solver. *ACM Trans. Graph.*, 39(4), 2020. 11

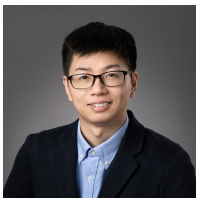
[15] D. Demidov. Amgcl: An efficient, flexible, and extensible algebraic multigrid implementation. *Lobachevskii Journal of Mathematics*, 40(5):535–546, May 2019. 3, 4

[16] C. Dick, M. Rogowsky, and R. Westermann. Solving the fluid pressure poisson equation using multigrid–evaluation and improvements. *Visualization and Computer Graphics, IEEE Transactions on*, 2016. 2, 3

[17] Z. Dostal and J. Schoberl. Minimizing quadratic functions subject to bound constraints with the rate of convergence and finite termination. *Computational Optimization and Applications*, 30(1):23–43, 2005. 1, 2, 5, 10

[18] Z. Dostl. *Optimal Quadratic Programming Algorithms: With Applications to Variational Inequalities*. Springer Publishing Company, Incorporated,

1st ed., 2009. 1, 2, 5, 10
[19] K. Erleben. Numerical methods for linear complementarity problems in physics-based animation. In *ACM SIGGRAPH 2013 Courses*, pp. 8:1–8:42, 2013. 1
[20] F. Ferstl, R. Westermann, and C. Dick. Large-scale liquid simulation on adaptive hexahedral grids. *Visualization and Computer Graphics, IEEE Transactions on*, 20(10):1405–1417, 2014. 2, 3
[21] A. Forsgren, P. E. Gill, and M. H. Wright. Interior methods for nonlinear optimization. *SIAM Rev.*, 44(4):525–597, 2002. 8
[22] D. Gerszewski and A. W. Bargteil. Physics-based animation of large-scale splashing liquids. *ACM Transactions on Graphics*, 32(6):185:1–185:6, 2013. 1, 2, 11
[23] R. Goldade, M. Aanjaneya, and C. Batty. Constraint bubbles and affine regions: Reduced fluid models for efficient immersed bubbles and flexible spatial coarsening. *ACM Trans. Graph.*, 39(4), aug 2020. 10
[24] T. Inglis, M.-L. Eckert, J. Gregson, and N. Thuerey. Primal-dual optimization for fluids. *Computer Graphics Forum*, 36(8):354–368, 2017. 1, 2
[25] A. Jacobson, I. Baran, J. Popović, and O. Sorkine. Bounded biharmonic weights for real-time deformation. *ACM Trans. Graph.*, 30(4), jul 2011. 11
[26] D. L. James and C. D. Twigg. Skinning mesh animations. *ACM Trans. Graph.*, 24(3):399–407, jul 2005. 11
[27] C. Jiang, C. Schroeder, A. Selle, J. Teran, and A. Stomakhin. The affine particle-in-cell method. *ACM Trans. Graph.*, 34(4):51:1–51:10, 2015. 2
[28] D. Krishnan, R. Fattal, and R. Szeliski. Efficient preconditioning of laplacian matrices for computer graphics. *ACM Trans. Graph.*, 32(4), jul 2013. 2, 4
[29] J. Kružík, D. Horák, M. Čermák, L. Pospíšil, and M. Pecha. Active set expansion strategies in mprgp algorithm. *Advances in Engineering Software*, 149:102895, 2020. 1, 9
[30] J. Lai, Y. Chen, Y. Gu, C. Batty, and J. W. Wan. Fast and scalable solvers for the fluid pressure equations with separating solid boundary conditions. *Computer Graphics Forum*, 39(2):23–33, 2020. 1, 2, 3, 4
[31] S. Lesser, A. Stomakhin, G. Daviet, J. Wretborn, J. Edholm, N.-H. Lee, E. Schweickart, X. Zhai, S. Flynn, and A. Moffat. Loki: A unified multiphysics simulation framework for production. *ACM Trans. Graph.*, 41(4), jul 2022. 2
[32] S. Li, Z. Pan, J. Huang, H. Bao, and X. Jin. Deformable objects collision handling with fast convergence. *Comput. Graph. Forum*, 34(7):269–278, oct 2015. 11
[33] A. McAdams, E. Sifakis, and J. Teran. A parallel multigrid poisson solver for fluids simulation on large grids. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 65–74, 2010. 1, 2, 4, 5, 10
[34] R. Narain, A. Golas, and M. C. Lin. Free-flowing granular materials with two-way solid coupling. *ACM Transactions on Graphics*, 29(6):173:1–173:10, 2010. 1, 2, 8, 11
[35] R. Narain, A. Samii, and J. F. O'Brien. Adaptive anisotropic remeshing for cloth simulation. *ACM Trans. Graph.*, 31(6):152:1–152:10, 2012. 9
[36] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, NY, USA, second ed., 2006. 1, 5, 6, 7, 8, 9, 10
[37] Y. Saad. Iterative Methods for Sparse Linear Systems. *Notes*, 3(2nd Edition):xviii+528, 2003. 3, 4, 5
[38] H. Shao, L. Huang, and D. L. Michels. A fast unsmoothed aggregation algebraic multigrid framework for the large-scale simulation of incompressible flow. *ACM Trans. Graph.*, 41(4), jul 2022. 2, 3, 4, 7, 10
[39] J. R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. August 1994. 1, 7, 10
[40] K. Stüben. A review of algebraic multigrid. *J. Comput. Appl. Math.*, 128(1–2):281–309, mar 2001. 2, 3
[41] T. Takahashi and C. Batty. Monolith: a monolithic pressure-viscosity-contact solver for strong two-way rigid-rigid rigid-fluid coupling. *ACM Transactions on Graphics (TOG)*, 39(6):1–16, 2020. 10
[42] T. Takahashi and C. Batty. Frictionalmonolith: A monolithic optimization-based approach for granular flow with contact-aware rigid-body coupling. *ACM Transactions on Graphics (TOG)*, 40(6):1–16, 2021. 1, 2, 9, 11
[43] T. Takahashi and C. Batty. A multilevel active-set preconditioner for box-constrained pressure poisson solvers. *Proc. ACM Comput. Graph. Interact. Tech.*, 6(3), aug 2023. 1, 2, 5, 7, 9, 10, 11
[44] T. Takahashi and M. C. Lin. A geometrically consistent viscous fluid solver with two-way fluid-solid coupling. *Computer Graphics Forum*, 38(2):49–58, 2019. 2
[45] R. Tamstorf, T. Jones, and S. F. McCormick. Smoothed aggregation multigrid for cloth simulation. *ACM Trans. Graph.*, 34(6):245:1–245:13, 2015. 2
[46] O. Tatebe. The multigrid preconditioned conjugate gradient method. In *Proceedings of the Sixth Copper Mountain Conference on Multigrid Methods (1993)*, pp. 621–634, 1993. 4
[47] U. Trottenberg, C. W. Oosterlee, and A. Schuller. *Multigrid*. Academic press, 2000. 1, 2, 4
[48] P. Vanek, J. Mandel, and M. Brezina. Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. *Computing*, 56:179–196, 1996. 2, 3, 4, 7, 8
[49] D. Weber, J. Mueller-Roemer, A. Stork, and D. Fellner. A cut-cell geometric multigrid poisson solver for fluid simulation. *Computer Graphics Forum*, 34(2):481–491, 2015. 2
[50] Z. Xian, X. Tong, and T. Liu. A scalable galerkin multigrid method for real-time simulation of deformable objects. *ACM Trans. Graph.*, 38(6), nov 2019. 2
[51] X. Zhang. tbb_liquid_amgpcg, 2015. 2, 3, 4, 7
[52] Y. Zhu, E. Sifakis, J. Teran, and A. Brandt. An efficient multigrid method for the simulation of high-resolution elastic solids. *ACM Trans. Graph.*, 29(2):16:1–16:18, 2010. 3



Tetsuya Takahashi is a senior researcher at Tencent America. Prior to that, he was a software engineer at Adobe. He earned his M.S. and Ph.D. from the University of North Carolina at Chapel Hill in 2017 and 2020, respectively, and B.S. and M.S. from Keio University in 2012 and 2014, respectively. His research interests include physically-based simulation, numerical optimization, and geometry processing.

Christopher Batty is an Associate Professor in the David R. Cheriton School of Computer Science at the University of Waterloo in Ontario, Canada. He received his PhD from the University of British Columbia in 2010 and was a Banting Postdoctoral Fellow at Columbia University from 2011 to 2013. His research is primarily focused on the development of novel physical simulation techniques for applications in computer graphics and computational physics, with an emphasis on the diverse behaviors of fluids.