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 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 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.

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 proportionalizing 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].

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 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 multisresolution hierarchies are constructed: geometric MG (GMG) and algebraic MG (AMG).

GMG uses geometric coarsening and rediscretization of systems...
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 \(\mathbf{A}_j\) via Galerkin coarsening applied to finer-level systems \(\mathbf{A}_j\). Using prolongation operators \(\mathbf{P}\) and restriction operators \(\mathbf{R}(=\mathbf{P}^T)\), one defines \(\mathbf{A}_j = \mathbf{RA}_j\mathbf{P}\). 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-Stüber 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 splatty 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

\[
\mathbf{p} = \arg \min_{\mathbf{s}(\mathbf{p}) \geq 0} E(\mathbf{p}), \quad E(\mathbf{p}) = \frac{1}{2} \mathbf{p}^T \mathbf{A}_{\mathbf{p}} \mathbf{p} - \mathbf{p}^T \mathbf{q},
\]

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 splat liquid behavior [22] using

\[
\mathbf{s}(\mathbf{p}) = \mathbf{p} \geq 0.
\]
Alternatively, we can define \( s(p) \) for wall-separating but "non-splashy" liquid given the set of indices \( \mathcal{S} \) for only those pressure cells in contact with solid boundaries [30] using

\[
s(p) = p_i \geq 0, \text{ if } i \in \mathcal{S}.
\]

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 \( Ax = b \) 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 observed to be more efficient than a V-cycle with unsmoothed aggregation. 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 resolve the coarser-level systems [40] (equivalent to over-interpolation of error), defining \( A_c = \alpha R A \mathcal{P} \), where \( \alpha \) 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-vector 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 \( P \) from the finest system \( A_f \) and geometric information (specifically, three indices \( x, y, z \) for each cell location in the 3D Cartesian grid), define the restriction operator \( R \) as \( \mathcal{P} = P^T \), and then assemble the coarser-level system \( A_c = \alpha R A \mathcal{P} \). (While earlier geometry-inspired aggregation approaches [30, 38, 51] use \( R = (1/2)^d \mathcal{P}^T \), 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 coarser one based on the coarse grid ID \( d_{x,y,z} \), computed for each fine cell by

\[
d_{x,y,z} = ((x/2) \ll 20) |(y/2) \ll 10) |(z/2).
\]
Algorithm 2 Construction of Prolongation Operator $P$

1: Compute coarse grid ID $d_{i,y,z}$ for each fine cell using (4)
2: for coarse cells $c_j$ in red and then black
3: Set $\mathcal{F}$ as fine cells in $c_j$
4: while $\mathcal{F}$ is not empty
5: for fine cell $f_i$ in $\mathcal{F}$
6: if aggregatable neighbor $n_i$ of $f_i$
7: Relate $n_i$ to $c_j$ and add to a next set $\mathcal{G}$
8: $\mathcal{F} = \mathcal{G}$
9: Construct $P$ using (5)

Then, while scanning algebraically connected fine cells from another coarse cell, the coarse one needs to physically contain the fine ones (relation drawn as cyan arrows). (Middle) In our algorithm, when fine cells are related to a coarse cell, the coarse cell to the 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 $A\mathbf{x} = \mathbf{b}$ (and corresponding coarser-level systems) in block form as

$$
\begin{bmatrix}
A_{rr} & A_{rb} \\
A_{br} & A_{bb}
\end{bmatrix}
\begin{bmatrix}
x_r \\
x_b
\end{bmatrix} =
\begin{bmatrix}
\mathbf{b}_r \\
\mathbf{b}_b
\end{bmatrix},
$$

where subscripts $r$ and $b$ indicate red and black colors (for variables, cells, etc., respectively). We observe that $A_{rr} = A_{bb}$. This matrix structure enables sequential access to $x_r$, completely avoiding access to $x_b$ 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, $A_{rr}$ and $A_{bb}$ 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 $P$ construction, the result would be multiple coarser level DOFs coupled with each other, causing $A_{rr}$ and $A_{bb}$ to no longer be strictly diagonal and thus invalidating the seven-point stencils.) Given the diagonality of these matrices, we define $D_{rr} = A_{rr}$ and $D_{bb} = A_{bb}$ for convenience, and precompute and store $D_{rr}^{-1}$ and $D_{bb}^{-1}$ prior to performing CG iterations to enable efficient smoothing operations. In addition, we can directly access diagonal elements $A_{rr}$ and $A_{bb}$ as the first and last entries of $A$ at the corresponding row in the CSR format, respectively, since $A_{rr}$ and $A_{bb}$ have only one non-zero per row. Furthermore, due to this structure, we can exclusively access the non-zeros in the $A_{rr}$ and $A_{bb}$ blocks by skipping the first and last entries of $A$ 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

$$
x_r^{k+1} = x_r^k + \omega D_{rr}^{-1} (b_r - A_{rr} x_r^k - A_{rb} x_b^k),
$$

$$
x_b^{k+1} = x_b^k + \omega D_{bb}^{-1} (b_b - A_{br} x_r^k - A_{bb} x_b^k),
$$

where $k$ denotes the smoothing iteration index, and $\omega$ 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

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 cell to the 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). (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.
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

\[ x_r^{k+1} = c_r + \omega b, \quad c_r = A_r x_r^{k+1}, \tag{9} \]

\[ x_b^{k+1} = c_b + \omega D^{-1}_{bb} (b - c_b), \quad c_b = A_b x_b^{k+1}. \tag{10} \]

Here, \( c_b \) 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 \( Ax \) since most of the non-zeros are in \( A_r \) and \( A_b \) (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 \( 1 \times 1 \) systems.

### 4.5 First Residual Computation

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

\[ r_r = b_r - A_r x_r - A_{rb} x_b, \tag{11} \]

\[ r_b = b_b - A_b x_r - A_{bb} x_b = b_b - c_b - D_{bb} x_b. \tag{12} \]

Here, \( A_{rb} x_b \) in (12) is equivalent to \( c_b = A_b x_b^{k+1} \) in (10) as \( x_r \) is not updated in (10). As such, we can compute and store \( c_b \) during the presmoothing for \( x_b \) (10) and avoid recomputing \( c_b \) when computing \( r_b \) 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 [3,33,43], if V-cycle is used. In addition, we can directly access \( D_{bb} \) as the last element in the row due to the CSR format (or as precomputed values), completely skipping access to \( A_{bb} \) when computing \( A_{bb} x_b \) in (12). Furthermore, since \( r_{rb} \) (4th element of \( r_b \)) is dependent only on \( x_{rb} \) (4th element of \( x_b \)) among \( x_b \), we can compute \( r_{rb} \) once we obtain \( x_{rb} \) by (10). This fact enables us to unify (12) with (10) (i.e., computing \( x_{rb} \) and \( r_{rb} \) 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 ressmoothing, by performing forward and backward smoothing (without requiring other ressmoothing operations at the same level for symmetry). Specifically, we perform ressmoothing in the order of red-black-black-red; we have tested black-red-black-red and observed no particular differences. The ressmoothing operations can thus be written as

\[ x_r^{k+2} = x_r^k + \omega D_{rr}^{-1} (b_r - A_r x_r^{k+1} - A_{rb} x_b^k), \tag{13} \]

\[ x_b^{k+2} = x_b^k + \omega D_{bb}^{-1} (b_b - A_b x_r^{k+1} - A_{bb} x_b^{k+1}), \tag{14} \]

\[ x_r^{k+2} = (1 - \alpha) x_r^{k+1} + \omega D_{rr}^{-1} (b_r - A_r x_r^{k+1} - A_{rb} x_b^{k+1}). \tag{17} \]

In addition, since \( A_r = D_{rr} \), we can rewrite (16) with \( c_r \) to unify the ressmoothing operations (13) and (14) as

\[ x_r^{k+2} = (1 - \alpha) x_r^{k+1} + \omega D_{rr}^{-1} (b_r - c_r), \quad c_r = A_r x_r^{k+2}. \tag{18} \]

### 4.7 Second Residual Computation

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

\[ r_r = b_r - A_r x_r - A_{rb} x_b, \tag{19} \]

\[ r_b = b_b - A_b x_r - A_{bb} x_b. \tag{20} \]

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

\[ x_r^{k+1} = x_r^k + \omega D_{rr}^{-1} (b_r - A_r x_r^{k+1} - A_{rb} x_b^k), \tag{21} \]

\[ x_b^{k+2} = x_b^{k+1} + \omega D_{bb}^{-1} (b_b - A_b x_r^{k+1} - A_{bb} x_b^{k+1}). \tag{22} \]

### 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 precondoner for inner linear solves.

#### 5.1 Primal-Dual Interior Point Method for QP

We consider a box-constrained convex QP problem written as

\[ x = \arg \min_{x \geq 0} f(x), \quad f(x) = \frac{1}{2} x^T A x - b^T x, \tag{23} \]

with optimization variable \( x \), SPD M-matrix \( A \), vector \( b \), and constraint function vector \( c(x) \). The constraint function vector can be defined to support both lower and upper bounds as \( c(x) = [c_1(x), c_2(x)]^T \) using

\[ c_1(x) = x - x_l \geq 0, \quad c_2(x) = x_u - x \geq 0, \tag{24} \]

where \( x_l \) and \( x_u \) denote the lower and upper bounds, respectively. If the \( i \)-th variable \( x_i \) does not have lower/upper bounds, we do not define any corresponding constraints. Considering that variables (e.g., \( x \)) will be represented in single- or double-precision floating-point, the operators “\( > \)” and “\( \geq \)” can be used almost interchangeably because, when \( c(x) = 0 \), we can enforce \( c(x) + \varepsilon > 0 \) by introducing a tiny (physically negligible) positive offset value, \( \varepsilon \) (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 \( s = [s_1^T, s_2^T]^T \) introduced for the inequality constraints \( c(x) \), i.e.,

\[ c(x) - s = 0, \quad s \geq 0, \tag{25} \]

we can formulate an augmented objective \( L(x,s,z) \) with a Lagrange multiplier \( z = [z_1^T, z_2^T]^T \geq 0 \), a log-barrier function for \( s \), and a barrier parameter \( \mu \) as

\[ L(x,s,z) = f(x) - \mu \sum_i \log s_i - z^T (c(x) - s). \tag{26} \]
The associated Karush–Kuhn–Tucker (KKT) conditions are given as

\[ \frac{\partial L}{\partial x} = Ax - b - \nabla f(z) - \nabla g(u) = 0, \]

\[ \frac{\partial L}{\partial s_i} = -\mu S_i^{-1} \epsilon_i + z_i = 0, \]

\[ \frac{\partial L}{\partial z_i} = -(x_i - s_i - z_i) = 0, \]

where the matrix \( J_i \) is \( \frac{\partial c_i}{\partial x} \) is a constant diagonal matrix (i.e., \( J_i = J_i^T \)) with its ith diagonal element being 1 if \( x_i \) has a lower bound \( x_i \leq 0 \) otherwise, and similarly \( J_i = \frac{\partial c_i}{\partial x} \) is a constant diagonal matrix with its ith diagonal element being 1 if \( x_i \) has an upper bound \( x_i \geq 0 \) otherwise. The matrices \( S_i \) and \( S_u \) are diagonal with their ith diagonal elements equal to \( s_{ij} \) and \( s_{ij} \), respectively, and \( \epsilon_i \) and \( \epsilon_u \) are vectors of all ones with the same dimensions as \( s_i \) and \( s_u \), respectively. To improve numerical conditioning, we can left-multiply (29) by \( S_i \) and \( S_u \) [36], obtaining

\[ -\mu \epsilon_i + S_i z_i = 0, \quad -\mu \epsilon_u + S_u z_u = 0, \]

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 \( \Delta x \), \( \Delta s_i \), and \( \Delta s_u \):

\[ \begin{bmatrix} A & O & -J^T \\ O & W & I \\ -J & I & O \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta s_i \\ \Delta s_u \end{bmatrix} = \begin{bmatrix} -L \frac{\partial L}{\partial x} \\ -L \frac{\partial L}{\partial s_i} \\ -L \frac{\partial L}{\partial s_u} \end{bmatrix}, \]

where \( J = \frac{\partial c_i}{\partial x} \), \( O \) denotes the zero matrix, \( I \) the identity matrix, and \( W = ZS^{-1} \) (and \( Z \) are diagonal matrices with their ith diagonal elements set to \( s_i \) and \( s_u \), respectively). Solving this large symmetric indefinite system (31) (whose size can be up to \( 5N \times 5N \) when \( x \) 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 \( c(x) \). By first eliminating \( \Delta s \) from (31), we obtain

\[ \begin{bmatrix} A & -J^T \\ -J & -W^{-1} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta z \end{bmatrix} = \begin{bmatrix} W^{-1} \frac{\partial L}{\partial z} \\ \frac{\partial L}{\partial z} \end{bmatrix}, \]

and then eliminating \( \Delta z \), we arrive at the smaller SPD system

\[ \left( A + J^T W J \right) \Delta x = -\frac{\partial L}{\partial x} + J^T \left( W \frac{\partial L}{\partial z} - \frac{\partial L}{\partial s} \right). \]

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

\[ \left( A + \sum_{i=1,u} J_i^T W_i^2 J_i \right) \Delta x^k = -\frac{\partial L}{\partial x} + \sum_{i=1,u} J_i^T \left( W_i \frac{\partial L}{\partial z_i} - \frac{\partial L}{\partial s_i} \right), \]

\[ = b - A \Delta x^k + \sum_{i=1,u} J_i^T \left( \mu [s_{i}]^{-1} + W_i \frac{\partial L}{\partial z_i} \right), \]

where we define \( W_i = Z_i S_i^{-1}, W_u = Z_u S_u^{-1} \), let \( J_i \) and \( Z_u \) denote diagonal matrices with \( i \)th diagonal elements equal to \( s_{ij} \) and \( s_{ij} \), respectively, and let \( [s_i]^{-1} \) and \( [s_u]^{-1} \) denote elementwise inverses of \( s_i \) and \( s_u \), respectively. This system is SPD, and its size is \( N \times N \).

In addition, as \( \sum_{i=1,u} J_i^T W_i^2 J_i \) in (34) is diagonal due to the special structures of box constraints, the resulting system matrix is an M-matrix, which is symmetric indefinite. 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 \( \frac{\partial L}{\partial x} \) and \( \frac{\partial L}{\partial s_i} \) here for later use in (37).

After we obtain \( \Delta x^k \) 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 \( \Delta s_i^k \) and \( \Delta s_u^k \) by

\[ \Delta s_i^k = J_i \Delta x^k - \frac{\partial L}{\partial z_i}, \quad \Delta s_u^k = -W_u \Delta s_u^k + \mu [s_i]^{-1} - z_i, \]

and thus \( \Delta s_i^k, \Delta s_u^k, \Delta s_i^k, \) and \( \Delta s_u^k \) are given by

\[ \Delta s_i^k = J_i \Delta x^k - \frac{\partial L}{\partial z_i}, \quad \Delta s_u^k = -W_u \Delta s_u^k + \mu [s_i]^{-1} - z_i, \]

where \( i = l, u \).

Notably, applying Newton’s method directly to the augmented objective (26) instead of minimizing 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 \( s = e(x) \)) leads to a formulation exactly the same as the one derived from the log-barrier method only with primal variables \( x \).

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 \( x \) and the non-negative constraints for the slack variable \( s \geq 0 \) and Lagrange multiplier \( z \geq 0 \) (or to decrease a corresponding merit function) when updating \( x, s, \) and \( z \) [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 \( x, s, \) and \( z \) (without step selection) using

\[ x^{k+1} = \text{clamp}(x^k + \Delta x^k, x_i, x_u), \]

\[ s^{k+1} = \max(s^k + \Delta s^k, e), \]

\[ z^{k+1} = \max(z^k + \Delta z^k, e). \]

5.4 Implementation Details

The algorithm of our primal-dual box-constrained convex QP solver is given in Algorithm 3. We initialize \( x^0 = \text{clamp}(0, x_l, x_u) \) taking the box constraints into account. For the slack and Lagrange multiplier variables, we initialize with \( s^0 = 0 \) and \( z^0 = e \) to ensure \( s \geq 0 \) and \( z \geq 0 \), 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 \( \mu \) toward 0 as the solution candidates approach the correct solution. However, we found that if \( \mu \) 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

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

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 a 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 \( \max(\|\Delta \mathbf{x}\|_2, \|\Delta \mathbf{s}\|_2, \|\Delta \mathbf{z}\|_2) \leq \varepsilon_N \) holds or not (where \( \varepsilon_N \) 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

\[
(I + \alpha L)q = q^f, \quad \text{s.t.} \quad q_L \leq q \leq q_U, \tag{40}
\]

where \( I \) and \( L \) denote the identity and Laplacian matrices, \( \alpha > 0 \) a scaling parameter (including diffusion coefficient and time step size), \( q \) and \( q^f \) the next and current temperature, respectively, \( t \) time, \( q_L \) and \( q_U \) lower and upper bounds for \( q \), respectively. As both \( I \) and \( L \) are SPD M-matrices, \( I + \alpha L \) 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

\[
q = \arg \min_{q_L \leq q \leq q_U} \frac{1}{2}q^T Aq - q^T q^f, \tag{41}
\]

where \( A = I + \alpha L \). 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 \( 10^{-6} \) (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 Intel Xeon Silver 4134 (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 18.3M non-zeros. The time for preconditioning preparation is denoted by \( T_{pr} \), CG solve by \( T_c \), the number of CG iterations per solve by \( N_c \), 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 \( Z_c \) denote the sum of non-zeros over the MG hierarchy for \( P + R \) and \( A \) (including the top level), respectively. \( Z_{uc} \) denotes the number of non-zeros in the lower triangular matrix computed by (modified) incomplete Cholesky.

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 1603, as shown in Figure 4. We compare the following preconditioning schemes:

1. MIC: baseline [9];
2. SAAMG: baseline [9];
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: 643, 1283, and 2563. We compare the following schemes:

1. PLB: primal log-barrier method [36];
2. IPM-P: interior point 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 $10^{-12}$. (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.

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 gradient descent direction at an essential step in the projected gradient descent. 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 $144^2$ 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];
5. IPM-PD+SAAMG-CG;
6. IPM-PD+TAGIAAMG-CG;
7. IPM-PD+MIC-MPRGP.
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, we use 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× 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 TAGIAAMG-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× and 1.6× faster performance with around half the memory usage for the MG hierarchy as compared to SAAMG-MPRGP and IPM-PD+SAAMG-CG, respectively.

### 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);

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× as many non-negative constraints as the non-splashy one (3), and thus requires around 1.9× more iterations and 1.6× 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\(^2\) 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.

---

**Table 3**: 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.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>(M)</th>
<th>(N)</th>
<th>(s)</th>
<th>(T_s)</th>
<th>(T)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAGIAAMG-CG</td>
<td>3.5</td>
<td>1.9</td>
<td>0.80</td>
<td>9.56</td>
<td></td>
</tr>
<tr>
<td>SAAMG-MPRGP</td>
<td>273.4</td>
<td>2.4</td>
<td>23.87</td>
<td>35.84</td>
<td></td>
</tr>
<tr>
<td>IPM-PD+SAAMG-CG</td>
<td>18.1</td>
<td>2.0</td>
<td>2.4</td>
<td>10.00</td>
<td></td>
</tr>
<tr>
<td>IPM-PD+TAGIAAMG-CG</td>
<td>16.2</td>
<td>2.0</td>
<td>4.00</td>
<td>22.33</td>
<td></td>
</tr>
</tbody>
</table>

**Table 4**: Simulation setting and results for Figure 6. 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.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>(c)</th>
<th>(M)</th>
<th>(N)</th>
<th>(s)</th>
<th>(T_s)</th>
<th>(T)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unconstrained</td>
<td>0</td>
<td>3.3</td>
<td>5.2</td>
<td>4.06</td>
<td>14.63</td>
<td></td>
</tr>
<tr>
<td>Non-splashy</td>
<td>0.1M</td>
<td>6.9</td>
<td>2.0</td>
<td>5.1</td>
<td>15.94</td>
<td>25.89</td>
</tr>
<tr>
<td>Splashy</td>
<td>1.4M</td>
<td>13.2</td>
<td>2.0</td>
<td>5.2</td>
<td>26.20</td>
<td>36.64</td>
</tr>
</tbody>
</table>

---
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 various sets of initialization values, parameter values, and update rules [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 convexQP 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 multiplications [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 $\varepsilon$ and barrier parameter $\mu$, 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 expensiveQP solve [36]. Alternatively, one could sequentially project variables one by one, but this latter approach does not guarantee convergence despite the solution to the convexQP 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].

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.
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).

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


